ALGORITHMS FOR
MOLECULAR BIOLOGY

Gene tree correction for reconciliation and species tree inference

Krister M Swenson[1,2]*, Andrea Doroftei[1] and Nadia El-Mabrouk[1]*

Abstract

Background: Reconciliation is the commonly used method for inferring the evolutionary scenario for a gene family. It consists in "embedding" inferred gene trees into a known species tree, revealing the evolution of the gene family by duplications and losses. When a species tree is not known, a natural algorithmic problem is to infer a species tree from a set of gene trees, such that the corresponding reconciliation minimizes the number of duplications and/or losses. The main drawback of reconciliation is that the inferred evolutionary scenario is strongly dependent on the considered gene trees, as few misplaced leaves may lead to a completely different history, with significantly more duplications and losses.

Results: In this paper, we take advantage of certain gene trees' properties in order to preprocess them for reconciliation or species tree inference. We flag certain duplication vertices of a gene tree, the "non-apparent duplication" (NAD) vertices, as resulting from the misplacement of leaves. In the case of species tree inference, we develop a polynomial-time heuristic for removing the minimum number of species leading to a set of gene trees that exhibit no NAD vertices with respect to at least one species tree. In the case of reconciliation, we consider the optimization problem of removing the minimum number of leaves or species leading to a tree without any NAD vertex. We develop a polynomial-time algorithm that is exact for two special classes of gene trees, and show a good performance on simulated data sets in the general case.

Keywords: Gene tree, Species tree, Reconciliation, Error correction, Maximum agreement subtree (MAST)

Background

Almost all genomes which have been studied contain genes that are present in two or more copies. Duplicated genes account for about 15% of the protein coding genes in the human genome, for example [1]. In practise, homologous gene copies (*e.g.* copies in one genome or amongst different genomes that are descended from the same ancestral gene) are identified through sequence similarity; using a BLAST-like method, all gene copies with a similarity score above a certain threshold would be grouped into the same *gene family*. Using a classical phylogenetic method, a *gene tree*, representing the evolution of the gene family by local mutations, can then be constructed based on the similarity scores. However,

macroevolutionary events (duplications, losses, horizontal gene transfer) affecting the number and distribution of genes among genomes [2], are not explicitly reflected by this gene tree. Having a clear picture of the speciation, duplication and loss mechanisms that have shaped a gene family is however crucial to the study of gene function. Indeed, following a duplication, the most common occurrence is for only one of the two gene copies to maintain the parental function, while the other becomes nonfunctional (pseudogenization) or acquires a new function (neofunctionalization) [3].

The most commonly used methods to infer evolutionary scenarios for gene families are based on the *reconciliation* approach that compares the species tree S (describing the relationships among taxa) to the gene tree T. Assuming no sequencing errors and a "correct" gene tree, the incongruence between the two trees can be seen as a footprint of the evolution of the gene family through processes other than speciation, such as duplication and loss. The concept of reconciling a gene tree to a species tree under

*Correspondence: swensonk@iro.umontreal.ca; mabrouk@iro.umontreal.ca
[1]Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, CP 6128 succ Centre-Ville, Montréal, Québec, H3C 3J7, Canada
[2]Departement of Computer Science, McGill University, 3480 University Street, Montréal, Québec, H3A 2A7, Canada

the duplication-loss model was pioneered by Goodman [4] and then widely accepted, utilized and also generalized to models of other processes such as horizontal gene transfer [5-7]. Several definitions of reconciliation exist in the literature, one of them expressed in term of "tree extension" [8]. More precisely, a *reconciliation R* between *T* and *S* is an extension of *T* (obtained by grafting new subtrees onto existing edges of *T*) *consistent* with the species tree (*i.e.* reflecting the same phylogeny). A duplication and loss history for the gene family is then directly deduced from *R*. As many reconciliations exist, a natural approach is to select the one that optimizes a given criterion. Natural combinatorial criteria are the number of duplications (duplication cost), losses (loss cost) or both combined (mutation cost) [9,10]. The so called Lowest Common Ancestor (LCA) mapping between a gene tree and a species tree, formulated in [11,12] and widely used [2,10,12-16], defines a reconciliation that minimizes both the duplication and mutation costs. It has been shown in [8] that minimizing duplications follows from minimizing losses (*i.e.* a reconciliation minimizing losses also minimizes duplications, but the converse is false). When no preliminary knowledge on the species tree is given, a natural problem, known as the *species tree inference problem*, is to infer, from a set of gene trees, a species tree leading to a parsimonious evolution scenario, for a given cost. Similar to the case of a known species tree, methods have been developed for the duplication and mutation costs [9,10,17]. For both criteria, the problem of inferring an optimal species tree given a set of gene trees is NP-hard [10].

The main criticism of reconciliation is that the inferred duplication and loss history for a gene family is strongly dependent on the gene tree considered for this family. Indeed, a few misplaced leaves in the gene tree can lead to a completely different history, possibly with significantly more duplications and losses [18]. Reconciliation can therefore inspire confidence only in the case of a well-supported gene tree. Typically bootstrapping values are used as a measure of confidence in each edge of a phylogeny. How should the weak edges of a gene tree be handled? One reasonable answer is to transform the binary gene tree into an unresolved gene tree by removing each weak edge and collapsing its two incident vertices into one. Extensions of the duplication-loss model to non-binary gene trees have been considered [19,20]. Another strategy adopted in [9] is to explore the space of gene trees obtained from the original gene tree *T* by performing Nearest Neighbor Interchanges (NNIs) around weakly-supported edges. The problem is then to select, from this space, the tree giving rise to the minimum reconciliation cost.

In this paper, we explore a different strategy for "correcting" or preprocessing a gene tree *T*, prior to reconciliation

or species tree inference. Criteria for identifying potentially misplaced leaves were given in [8]. The duplication vertices of *T* with respect to a species tree *S* can be subdivided into apparent and non-apparent duplication (NAD) vertices, where the latter class has been flagged as potentially resulting from the misplacement of leaves in the gene tree. The reason is that each one of the NAD vertices reflects a phylogenetic contradiction with the species tree that is not due to the presence of duplicated gene copies. In the case of an unknown species tree, we showed in [8] that deciding whether *T* can be explained using only apparent duplications (we say that *T* is an MD-tree) can be done in polynomial time, as well as inferring an appropriate species tree. Here, we present algorithmic results for removing, for a given gene tree (or a forest of gene trees), the minimum number of leaves or leaf-labels (species) leading to a tree without any NAD vertex, in both cases of a known or an unknown species tree. The minimum leaf removal problem in case of a known species tree has been recently proved to be APX-hard [21].

In the next section, we begin by formally introducing our concepts. We then motivate and state our problems in Section "Motivations and problem statements". Section "Minimum species removal inference and reconciliation" gives a greedy heuristic for the minimum species removal problem in the case of an unknown species tree, and shows that any algorithm for this case can be applied to the case where the species tree is known. Section "Algorithms for the minimum removal reconciliation problems" is dedicated to the algorithmic developments in the case of a known species tree. We first describe two special classes of gene trees which lead to an exact polynomial-time algorithm. We then present a heuristic algorithm for the general case. In Section "Empirical results", we test the optimality of our algorithm for the minimum leaf-removal problem in the case of a known species tree, and the ability of the presented approach to identify misplaced genes. This paper is an extended version of [22].

Definitions
Trees

In this paper, we only consider rooted trees. Let $G = \{1, 2, \cdots, g\}$ be a set of integers representing g different species (genomes). A ***species tree*** on G is a rooted binary tree with exactly g leaves, where each $i \in G$ is the label of a single leaf (Figure 1a). A ***gene tree*** on G is a rooted binary tree where each leaf is labelled by an integer from G, with possibly repeated leaves (Figure 1b). A gene tree represents a gene family, where each leaf labelled i represents a gene copy located on genome i. In the case of a species tree or a ***uniquely leaf-labelled gene tree*** (*i.e.* no leaf-label occurring more than once) we will make no distinction between a leaf and its label.
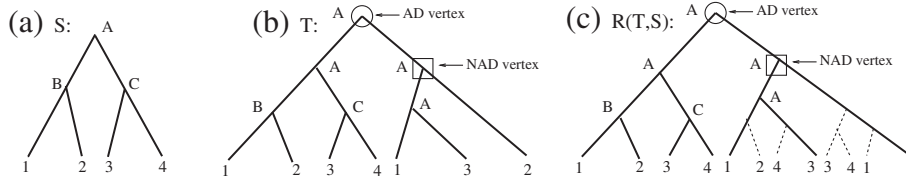
Figure 1 (a) A species tree *S* for *G* = {1, 2, 3, 4}. The three internal vertices of *S* are named *A*, *B* and *C*; **(b)** A gene tree *T*. A leaf label *x* indicates a gene copy in genome *x*. Internal vertices are labelled according to the LCA mapping between *T* and *S*. Flagged vertices are duplication vertices of *T* with respect to *S* (Section "LCA Mapping"); **(c)** A reconciliation *R(T, S)* of *T* and *S*. Dotted lines represent subtree insertions. The correspondence between vertices of *R(T, S)* and *S* is indicated by vertex labels. Flagged vertices are duplication vertices. All other internal vertices of *R(T, S)* are speciation vertices. This reconciliation reflects a history of the gene family with two gene duplications preceding the first speciation event, and 4 losses.

Given a tree U, the **size of** U, denoted by $|U|$, is the number of leaves of U, and the **genome set of** U, denoted by $\mathcal{L}(U)$, is the subset of G defined by the labels of the leaves of U. Given a vertex x of U, U_x is the subtree of U rooted at x. The genome set of U_x is denoted by $\mathcal{L}(x)$ (for example, in the tree of Figure 1a, $\mathcal{L}(B) = \{1, 2\}$). If x is not a leaf, we denote by x_l and x_r the two children of x. Finally, if x is not the root, any vertex $y \neq x$ on a path from x to the root is an **ancestor** of x.

Given a tree U on a genome set G, a **leaf removal** consists of removing a given leaf from U, along with its parental node x, and if x is not the root joining the parent of x and the remaining child by a new edge. A tree U' obtained from U through a sequence of leaf removals is said to be **included in** U. The tree U **restricted to a subset** G' of G is the tree U' obtained from U through a sequence of leaf removals that removes all the leaves with labels in $G \setminus G'$.

Finally, a subtree U_x of U, for a given vertex x, is said to be a **maximum subtree** of U verifying a given property P if and only if U_x verifies property P and, for any vertex y that is an ancestor of x, U_y does not verify property P.

Reconciliation

Applying a classical phylogenetic method to the gene sequences of a given gene family leads to a gene tree T that is different from the species tree S, mainly due to the presence of multiple gene copies in T, and that may reflect a divergence history different from S. The reconciliation approach consists in "embedding" the gene tree into the species tree, revealing the evolution of the gene family by duplications and losses.

There are several definitions of reconciliation between a gene tree and a species tree [2,10-15]. Here we define reconciliation in terms of subtree insertions, following the notation used in [14,23]. We begin with some definitions:

- A **subtree insertion** in a tree T grafts a new subtree onto an existing edge of T. Formally, inserting a subtree onto an edge linking two nodes x and y (y is a

child of x) consists in creating a new node z with parent x and two children being y and the root of the inserted subtree.

- A tree T' is said to be an **extension** of T if it can be obtained from T by subtree insertions on the edges of T.

- The gene tree T is said to be **DS-consistent with** S (DS standing for Duplication/Speciation) if T reflects a history with no loss, *i.e.* if for every vertex t of T such that $|\mathcal{L}(t)| \geq 2$, there exists a vertex s of S such that $\mathcal{L}(t) = \mathcal{L}(s)$ and one of the two following conditions holds:

   (D)   either $\mathcal{L}(t_r) = \mathcal{L}(t_\ell)$ (indicating a Duplication),

   (S)   or $\mathcal{L}(t_r) = \mathcal{L}(s_r)$ and $\mathcal{L}(t_\ell) = \mathcal{L}(s_\ell)$ (indicating a Speciation).

Definition 1. *A* **reconciliation** *between a gene tree T and a species tree S on G is an extension $R(T, S)$ of T that is DS-consistent with S.*

For example, the tree of Figure 1c is a reconciliation between the gene tree T of Figure 1b and the species tree of Figure 1a. Such a reconciliation between T and S implies an unambiguous evolution scenario for the gene family, where a vertex of $R(T, S)$ that satisfies property (D) represents a duplication (duplication vertex), a vertex that satisfies property (S) represents a speciation (speciation vertex), and an inserted subtree represents a gene loss. The number of duplication vertices of $R(T, S)$ is called the **duplication cost** of $R(T, S)$.

The notion of reconciliation can naturally be extended to the case of a set, or *forest*, of gene trees $\mathcal{F} = \{T_1, \ldots, T_f\}$: a reconciliation between $\mathcal{F}$ and S is a set $\mathcal{R}(\mathcal{F}, S) = \{R_1(T_1, S), \ldots, R_f(T_f, S)\}$ of reconciliations, respectively for $T_1, \ldots, T_f$, such that each $R_i(T_i, S)$ is DS-consistent with S.

LCA Mapping

The LCA mapping between a gene tree T and a species tree S, denoted by M, maps every vertex t of T to the Lowest Common Ancestor (LCA) of $\mathcal{L}(t)$ in S. A vertex t of T is called a ***duplication vertex*** of T with respect to S if and only if $M(t_\ell) = M(t)$ and/or $M(t_r) = M(t)$ (see Figure 1b). We denote by $\mathbf{d(T, S)}$ the number of duplication vertices of T with respect to S. Any vertex of T that is not a duplication vertex is a ***speciation vertex*** with respect to S.

The LCA mapping induces a reconciliation $M(T, S)$ between T and S, where an internal vertex t of T leads to a duplication vertex in $M(T, S)$ if and only if t is a duplication vertex of T with respect to S. In other words, the duplication cost of $M(T, S)$ is $d(T, S)$ (see for example [10,13,15] for more details on the construction of a reconciliation based on the LCA mapping). Moreover, $M(T, S)$ is a reconciliation that minimizes the duplication, loss, and mutation costs [8,14].

Duplication vertices and MD-trees

Let T be a gene tree. As noticed in [8], any vertex t of T such that $\mathcal{L}(t_\ell) \cap \mathcal{L}(t_r) \neq \emptyset$ (*i.e.* the left and right subtrees rooted at t contain a gene copy in the same genome) will be a duplication vertex in any reconciliation between T and any species tree S, in particular in $M(T, S)$. Such a vertex is called an ***Apparent Duplication vertex*** (***AD vertex*** for short) of T. In the tree of Figure 1b, the root is an AD vertex as its left and right subtree both contain a gene copy in genome 1. Following our notation in [8], a gene tree T is said to be a ***Minimum Duplication Tree*** (henceforth called an ***MD-tree***) if there exists a species tree S such that $d(T, S)$ is exactly the number of apparent duplications present in T. In which case, T is said to be ***MD-consistent*** with S.

However, this is not always true, in other words, a duplication vertex of T with respect to a species tree S is not necessarily an AD vertex. We call such a duplication vertex a ***Non-Apparent Duplication vertex***, or simply a ***NAD vertex***. For example, the tree of Figure 1b contains one NAD vertex, indicated by a square, and thus T is not MD-consistent with S.

Motivations and problem statements

Non-apparent duplication vertices point to disagreement between a gene tree T and a species tree S that are not due to the presence of repeated leaf labels (*i.e.* multiple copies in the same genome). More precisely, we say that a vertex x of T *splits* three species $\{a, b, c\}$ into $\{a, b; c\}$ if the genome set of one of its children contains a and b but not c, and the genome set of its other child contains c but neither a nor b. Then for any NAD x of T, there is a triplet of species $\{a, b, c\}$ that are split differently by x and by the LCA mapping of x in S (see proof of Theorem 5 in [8]). We will say that such a triplet *exhibits a wrong phylogeny*. For example, in Figure 1, $\{1, 2, 3\}$ is split into $\{1, 3; 2\}$ by the NAD vertex of T, and into $\{1, 2; 3\}$ by the vertex A in S. It has therefore been suggested that NAD vertices may point at gene copies that are erroneously placed in the gene tree.

Observations made in [8] tend to support this hypothesis. In particular, using simulated data-sets based on the species tree of 12 *Drosophila* species given in [24] and a birth-and-death process, starting from a single ancestral gene, and with different gene gain/loss rates, it has been found that 95% of gene duplications lead to an AD vertex.

Notice however that a misplaced gene in a gene tree T does not necessarily lead to a NAD vertex. In other words, NAD vertices can only point to a subset of misplaced leaves. However, in the context of reconciliation, the damage caused by a misplaced leaf leading to a NAD vertex is to significantly increases the real mutation-cost of the tree, as shown in Figure 2.

Following the later observations, we exploit the properties of NAD vertices for gene tree correction. For generality, we consider a forest of gene trees $\mathcal{F} = \{T_1, \ldots, T_f\}$. If $\mathcal{F}$ is not MD-consistent with a given species tree S (*i.e.* there is at least one tree in $\mathcal{F}$ that is not MD-consistent with S) then an MD-consistent forest can always be obtained from $\mathcal{F}$ by performing a certain number of leaf removals. Indeed, a gene tree with only two leaves is always MD-consistent with any species tree. Our first optimization problem is the following, where the size of $\mathcal{F}$ is just the sum of sizes of all the trees of $\mathcal{F}$.
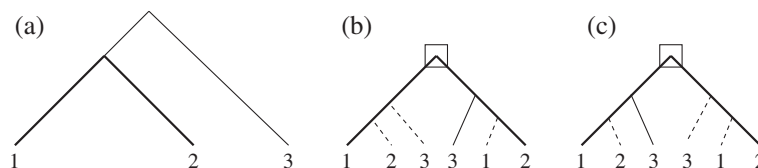

Figure 2 Let $S = ((1, 2), 3)$ (the tree in (a)) be the phylogenetic tree for the three species $\{1, 2, 3\}$. Let $T = (1, 2)$ be a gene tree. **(a)**, **(b)** and **(c)** are the three possibilities for T after a random insertion of a leaf labelled 3. **(a)** is the only case leading to a tree without any NAD vertex. It reflects a history of the three gene copies without any duplication or loss; **(b)** and **(c)** each contains a NAD vertex, and can be explained by a duplication-loss history of minimum mutation cost of 4: 1 duplication and 3 losses.

MINIMUM LEAF REMOVAL RECONCILIATION (MINLRR):

Input: A genome set G, a forest of gene trees $\mathcal{F} = \{T_1, \ldots, T_f\}$ on G, and a species tree S for G;
Output: A forest of gene trees $\mathcal{F}^{MAX} = \{T_1^{MAX}, \ldots, T_f^{MAX}\}$ of maximum size (*i.e.* obtained from $\mathcal{F}$ by a minimum number of leaf removals) which is MD-consistent with S, where each T_i^{MAX} is included in T_i.

In the case of an unknown species tree, we have shown in [8] that deciding whether a forest of gene trees $\mathcal{F}$ is an MD-forest (*i.e.* a set of MD-trees) can be done in polynomial time and space, as well as computing a parsimonious species tree. For a forest which is not an MD-forest, a natural generalization of the MINLRR problem is the following:

MINIMUM LEAF REMOVAL INFERENCE (MINLRI):

Input: A genome set G and a forest of gene trees $\mathcal{F} = \{T_1, \ldots, T_f\}$ on G;
Output: An MD-forest $\mathcal{F}^{MAX} = \{T_1^{MAX}, \ldots, T_f^{MAX}\}$ of maximum size (*i.e.* obtained from $\mathcal{F}$ by a minimum number of leaf removals), where each T_i^{MAX} is included in T_i.

A more conservative strategy that can be used to reduce the risk of inferring a wrong species tree, is to remove the minimum number of species from G such that the forest $\mathcal{F}$ restricted to the new genome set is an MD-forest. Removing the minimum number of species instead of leaves can also be considered in the case of reconciliation, a scenario that may be applicable when full confidence is not put in the species tree.

MINIMUM SPECIES REMOVAL RECONCILIATION (MINSRR):

Input: A genome set G, a forest of gene trees $\mathcal{F} = \{T_1, \ldots, T_f\}$ on G and a species tree S for G;
Output: A maximum subset G' of G such that forest $\mathcal{F}$ restricted to G' (*i.e.* the set of trees T_i restricted to G') is MD-consistent with the species tree S restricted to G'.

MINIMUM SPECIES REMOVAL INFERENCE(MINSRI):

Input: A genome set G and a forest of gene trees $\mathcal{F} = \{T_1, \ldots, T_f\}$ on G;
Output: A maximum subset G' of G such that the forest $\mathcal{F}$ restricted to G' is an MD-forest.

The latter two optimization problems (MINSRR and MINSRI) are the subject of the next section. Section "Algorithms for the minimum removal reconciliation problems" focuses on the two optimization problems related to reconciliation (MINLRR and MINSRR).

Minimum species removal inference and reconciliation

By linking the species tree inference problem to a supertree problem we have been able to prove that deciding whether a gene tree T is an MD-tree can be done in polynomial-time [8]. We used a constructive proof based on a min-cut strategy, which has been largely considered in the context of supertrees [25-27]. In this section, we develop a greedy heuristic for MINSRI based on a minimum vertex cut strategy.

Let $\mathcal{F} = \{T_1, T_2, \ldots, T_f\}$ be a forest of gene trees on a genome set G. Define $level_0(\mathcal{F})$ to be the set of highest (*i.e.* closest to the root) vertices of all T_is that are not AD-vertices. $level_j(\mathcal{F})$ is then the set of vertices of all T_is that are closest non-AD descendants of the vertices for $level_{j-1}(\mathcal{F})$. For a given level j, forest $\mathcal{F}$, and vertex $x \in level_j(\mathcal{F})$, consider the bipartition $B(x) = (\mathcal{L}(x_l), \mathcal{L}(x_r))$. Then $\mathcal{G}_j = (V, E)$ is the corresponding hypergraph [28] where $V = G$, and $\mathcal{L}(x_l), \mathcal{L}(x_r) \in E$ for $x \in level_j(\mathcal{F})$.

In order for $\mathcal{F}$ to be an MD-forest, all the vertices of $level_j(\mathcal{F})$, for any j, should represent speciation vertices with respect to some species tree S (as otherwise they would represent additional non-apparent duplication vertices, preventing the forest from being an MD-forest). In other words, the bipartitions $B(x)$ for all $x \in level_0(T)$ should reveal a first speciation event, which is possible if and only if the graph $\mathcal{G}_0$ contains at least two connected components. Indeed, in this case for any species tree S with a root r splitting G into two disconnected subsets, all the vertices of $level_0(\mathcal{F})$ would be speciation vertices. Conversely, if $\mathcal{G}_0$ contains a single connected component, then for any species tree S, at least one node of $level_0(T)$ would be a NAD node. The same reasoning applies to any $level_j(\mathcal{F})$ and $\mathcal{G}_j$.

On the other hand, if $\mathcal{G}_j$ is connected for some $level_j(\mathcal{F})$, there exists no species tree so that all $x \in level_j(\mathcal{F})$ represent speciation events. In this case, some number of species must be removed to make $\mathcal{G}_j$ disconnected. This corresponds exactly to a vertex cut in $\mathcal{G}_j$. These observations leads to the following heuristic for the MINSRI problem.

ALGORITHM MINIMUM SPECIES REMOVAL INFERENCE($\mathcal{F}$)

1. $G' = G$; $j = 0$; Compute $level_0(\mathcal{F})$;
2. WHILE $level_j(\mathcal{F})$ is not empty DO
3.   Construct the hypergraph $\mathcal{G}_j$;
4.   IF $\mathcal{G}_j$ is connected THEN
5. $G' = G' \setminus$ MINIMUM-VERTEX-CUT($\mathcal{G}_j$);
6.   Restrict $\mathcal{F}$ to G';
7.   END IF
8. $j = j + 1$;

9.   Compute $level_j(\mathcal{F})$;
10. END WHILE
11. RETURN (G')

MINIMUM-VERTEX-CUT in a hypergraph can be computed using the minimum vertex cut algorithm for simple graphs: each hyperedge corresponds to a clique in the simple graph. It is easy to confirm that a set of vertices disconnects the hypergraph if and only if it disconnects the corresponding simple graph. Vertex cut on a simple graph can be implemented with $2n - 2$ calls to the standard s-t vertex cut algorithm (based on minimum s-t edge cut). By reusing computation, Hao and Orlin [29] showed how to do all $2n - 2$ calls to the s-t cut algorithm in the same time it takes to do a single call. Thus, MINIMUM-VERTEX-CUT can be solved in $O(|V||E| \lg(|V|^2/|E|))$ time. Since we call MINIMUM-VERTEX-CUT $O(|V|)$ times in the worst case, ALGORITHM MINIMUM SPECIES REMOVAL INFERENCE runs in $O(|V|^2|E| \lg(|V|^2/|E|))$ time.

In the next section we give algorithms for MINSRR and MINLRR. We conclude this section by highlighting the relationship between MINSRR and MINSRI.

Remark 1. MINSRR *reduces to* MINSRI.

This is easy to see; take the species tree S given by the instance of MINSRR and add it to the forest $\mathcal{F}$ for the MINSRI problem. The solution to MINSRI gives a species tree that must be a subtree of S. Thus, any algorithm for MINSRI can be used to solve MINSRR.

Algorithms for the minimum removal reconciliation problems

In this section, we assume that a species tree S is known for the genome set G. For simplicity, we present the algorithms in the case of a single gene tree T, although it is straightforward to generalize them to the case of a forest of gene trees.

Let T be a gene tree for a gene family on G. We suppose that T is not an MD-tree consistent with S (*i.e.* there is at least one duplication vertex of T that is a NAD vertex). We begin by describing special classes of gene trees for which exact polynomial-time algorithms have been developed for the MINLRR and MINSRR problems.

Uniquely leaf-labelled gene trees

When the considered gene family contains at most one gene per genome, the gene tree T is uniquely leaf-labelled. In this case, minimizing the number of leaves, or equivalently species, that should be removed from T to obtain an MD-tree consistent with S is equivalent to finding the maximum number of genomes that lead to the same phylogeny in T and S. In other words, it is immediate to see

that the MINLRR problem reduces, in this case, to the MAST problem given below.

MAXIMUM AGREEMENT SUBTREE (MAST):

Input: A uniquely leaf-labelled gene tree T on G and a species tree S for G;
Output: A tree T^{MAX} included in T such that it is MD-consistent with S and of maximum size.

A more general definition is given in the literature, where the MAST problem is defined on a set of uniquely leaf-labelled trees as the largest tree included in each tree of the set. This definition is equivalent to ours in the case of a gene tree T and a species tree S.

The MAST problem arises naturally in biology and linguistics as a measure of consistency between two evolutionary trees over species or languages [30]. In the evolutionary study of genomes, different methods and different gene families are used to infer a phylogenetic tree for a set of species, usually yielding different trees. In such a context, one has to find a consensus of the various obtained trees. Considering the MAST problem, introduced by Finden and Gordon [31], is one way to obtain such a consensus. Amir *et al.* [32] showed that computing a MAST of three trees with unbounded degree is NP-hard. However, in the case of two binary trees, the problem is polynomial. The first polynomial-time algorithm for this problem was given by Steel and Warnow [33]. It is a dynamic programming algorithm considering the solution for all pairs of subtrees of T and S; it has a running time of $O(n^2)$, where n is the number of leaves in the trees. Later, Cole *et al.* [30] developed an $O(n \lg n)$ time algorithm, which, as far as we know, is the most efficient algorithm for solving the MAST problem on two binary trees. We use this result in the MINLRR version of our algorithms. In the case of k binary trees, the current fastest known algorithms run in $O(kn^3)$ time [34,35]. We use this result in the MINSRR version of our algorithms.
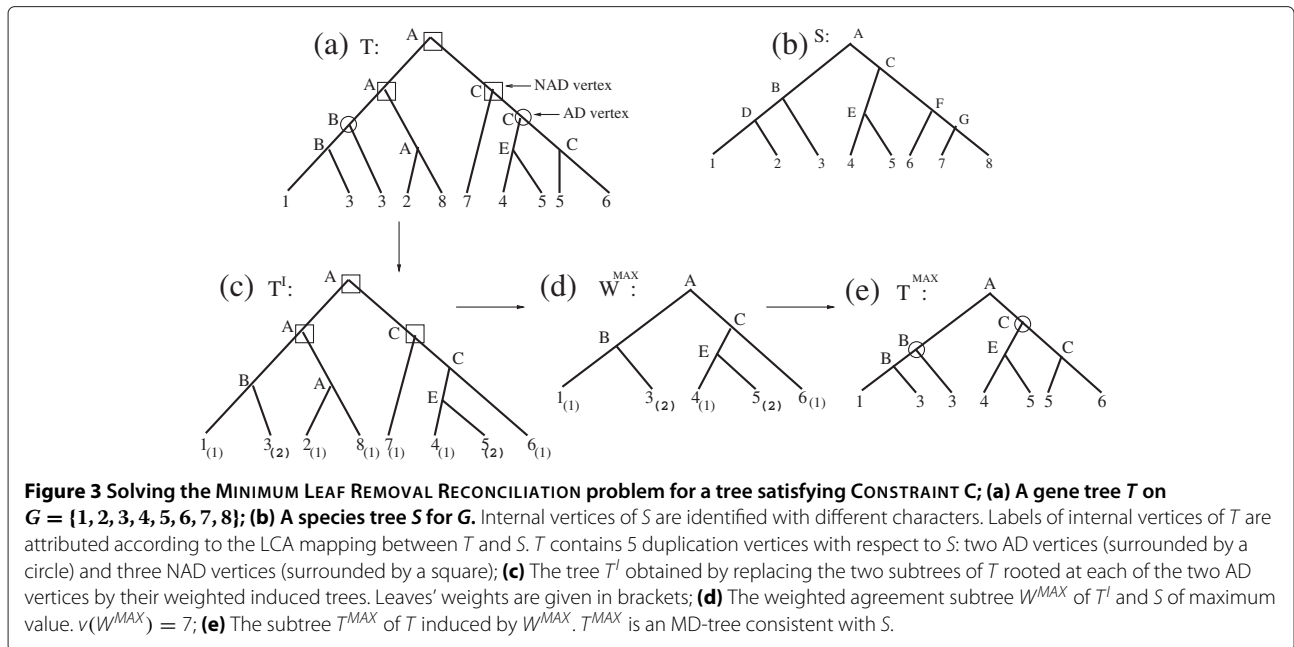
No AD above NAD

In this section, we consider a tree T containing no AD vertex above a NAD vertex (Figure 3a). More precisely, T satisfies CONSTRAINT C below:

CONSTRAINT C: For each NAD vertex x of T, if y is an ancestor of x that is a duplication vertex, then y is a NAD vertex.

We show that the MINSRR problem reduces, in this case, to the MAST problem, while the MINLRR problem reduces to a "generalization" of the MAST problem to weighted trees, where a *weighted tree* is a uniquely leaf-labelled tree with weighted leaves.

Definition 2. *Let U be a tree on G. The weighted tree induced by (U,S) is the tree included in S obtained from*

Figure 3 Solving the Minimum Leaf Removal Reconciliation problem for a tree satisfying Constraint C; (a) A gene tree *T* on
G = {1, 2, 3, 4, 5, 6, 7, 8}; (b) A species tree *S* for *G*. Internal vertices of *S* are identified with different characters. Labels of internal vertices of *T* are
attributed according to the LCA mapping between *T* and *S*. *T* contains 5 duplication vertices with respect to *S*: two AD vertices (surrounded by a
circle) and three NAD vertices (surrounded by a square); **(c)** The tree T^I obtained by replacing the two subtrees of *T* rooted at each of the two AD
vertices by their weighted induced trees. Leaves' weights are given in brackets; **(d)** The weighted agreement subtree W^{MAX} of T^I and *S* of maximum
value. $v(W^{MAX}) = 7$; **(e)** The subtree T^{MAX} of *T* induced by W^{MAX}. T^{MAX} is an MD-tree consistent with *S*.

S by removing all leaves that are not in $\mathcal{L}(U)$, with a
weight attributed to each leaf s, representing the number
of occurrences of s in U (i.e. the number of leaves of U
labelled s).

Let $T_1, T_2, \cdots T_m$ be the maximum subtrees of *T* rooted
at an AD vertex (*i.e.* subtrees of *T* rooted at the high-
est AD vertices). Then, the tree T^I obtained by replacing
each T_i, for $1 \leq i \leq m$, by the weighted tree induced by
(T_i, S), is a weighted uniquely leaf-labelled tree. An exam-
ple is given in Figure 3a,b and c. Let ρ_s be the operation
of removing the weighted leaf *s* from T^I. Then the cor-
responding removals in *T* consist of removing from *T* all
leaves labelled *s*.

Finally, we formulate the generalization of the MAST
problem to weighted trees as follows, where the value
$v(W)$ of a weighted tree *W* is the sum of its leaves' weights.

Weighted Maximum Agreement Subtree (WMAST):

Input: A weighted tree *W* on *G* and a species tree *S* for *G*;
Output: A weighted tree W^{MAX} included in *W* such that
it is MD-consistent with *S* and of maximum value.

We are now ready for the main theorem.

Theorem 1. *Let T be a gene tree satisfying Con-*
straint C. Let W^{MAX} be a solution of the WMAST
problem on T^I and S, and T^{MAX} be the subtree included in
T obtained by removing from T all the leaves that are not
leaves of W^{MAX}. Then T^{MAX} is a solution of the MinLRR
problem on T and S.

In other words, solving the MinLRR problem on *T*
is equivalent to solving the WMAST problem on T^I.
We show in the proof of Theorem 2 that WMAST can
be solved by the traditional MAST algorithms with no
change in the asymptotic running time.

A complete example of the algorithmic methodology
used for solving the MinLRR problem on *T* and *S* is given
in Figure 3. The algorithm will be detailed in the next
section.

We now provide a proof of Theorem 1, subdivided into
the two following lemmas.

Lemma 1. *The tree T^{MAX} is MD-consistent with S.*

Proof. We show, by contradiction, that T^{MAX} does not
contain any NAD vertex. Suppose that T^{MAX} contains a
NAD vertex *x*. Then *x* maps to the same vertex *s* of *S*
than one of its children, lets say the left child. Then there
exists two leaves of $T^{MAX}_{x_l}$, labelled *a* and *b*, and one leaf
of $T^{MAX}_{x_r}$ labelled *c* such that the triplet {*a*, *b*, *c*} exhibits a
wrong phylogeny. As a non-duplication vertex in *T* cannot
become a duplication vertex after leaf removals, we have
only two possibilities for *x* in *T*:

1. *x is a NAD vertex in T. Then the genome sets of T_{x_l}*
   and T_{x_r} are disjoint. Moreover, the genome set of
   $W^{MAX}_{x_l}$ (resp.. $W^{MAX}_{x_r}$) is a subset of the genome set
   of T_{x_l} (resp. T_{x_r}). On the other hand, as *x* is not a
   duplication vertex in W^{MAX}, one of the three genes
   a, *b* and *c* should be absent in W^{MAX}_x. And thus,
   {*a*, *b*, *c*} can not be a subset of the genome set of
   T^{MAX}_x, a contradiction.

2. *x is an AD vertex in T.* Then the subtree T_x of T rooted at x contains at least two leaves labelled with the same label d (different from a, b and c), one in T_{x_l} and one in T_{x_r}. Moreover the leaf labelled d in S should belong to the subtree of S rooted at s, and thus to the subtree S_i rooted at the left or right child of s. Such subtree S_i contains at least one leaf labelled a or b or c.

On the other hand, let y be the parent of x in T^I. As an optimal solution of the WMAST problem on T^I removes leaves from the subtree T_x^I, such an operation should result in removing the duplication vertex y. In other words, x and y should map to the same vertex s in S. Moreover the result of the leaf removal from T_x^I should result in a different LCA mapping for x and y. Indeed, removing leaves from the corresponding subtree in T^I does not contribute to eliminating any NAD from T^I. It follows that S should exhibit the phylogeny $((a, b, c), d)$, which is a contradiction with the result of the last paragraph. □

Lemma 2. *Let T' be a tree included in T that is MD-consistent with S. Then $|T'| \leq |T^{MAX}|$.*

Proof. We will show that, for any $s \in G$, if a leaf i labelled s is removed from T (*i.e.* i is not a leaf in T'), then all leaves of T labelled s are removed from T.

Suppose this is not the case. Let y be the vertex of T representing the least common ancestor of all leaves labelled s in T. Then y is an AD node. As a leaf i labelled s is removed from T, such removal should contribute to resolving a NAD vertex x of T. From CONSTRAINT C, such a vertex should be outside the subtree of T rooted at y. Moreover, it should clearly be an ancestor of y (otherwise removing i will have no effect on x).

As x is a NAD vertex, it maps to the same vertex s of S as one of its children, say the left child. Then, there exist two leaves of T_{x_l} labelled a and b, and one leaf of T_{x_r} labelled c such that the triplet $\{a, b, c\}$ exhibits a wrong phylogeny. Moreover, as removing leaf i labelled s contributes to solving x, we can assume that $a = s$. However, from our assumption, there remains a leaf labelled s in T'. Thus: either (1) at least one leaf labelled b and one leaf labelled c remains in T', or (2) all leaves labelled b or all leaves labelled c are removed. In case (1), the wrong phylogeny exhibited by the triplet $\{a, b, c\}$ is still present, preventing vertex x from being a non-duplication vertex. In case (2), as all copies of b (or equivalently c) are removed, there is no need to remove leaf i labelled s to correct the wrong phylogeny exhibited by the triple $\{a, b, c\}$.

Therefore, the weighted tree W' induced by T' is obtained from T^I through a sequence of leaf removals. Now, as W^{MAX} is the solution of the WMAST problem on T^I, then $v(W^{MAX}) \geq v(W')$, and thus $|T^{MAX}| \geq |T'|$. □

Finally, the following corollary makes the link between the MinSRR problem and the MAST problem.

Corollary 1. *Let T be a gene tree satisfying* CONSTRAINT *C. Let W^{MAX} be a solution to the* MAST *problem on T^I and S (ignoring weights), and T^{MAX} be the subtree of T induced by W^{MAX}. Then T^{MAX} is a solution to the* MinSRR *problem on T and S.*

To apply the algorithm to MinSRR with the forest $\mathcal{F} = \{T_1, T_2, \ldots, T_k\}$, all trees T_i^I must simultaneously agree with S, so the $O(kn^3)$ MAST algorithm [34,35] must be used.

An Algorithm for the general case

In this section, we present a general algorithm, that is exact in the case of a uniquely leaf-labelled gene tree (Section "Uniquely leaf-labelled gene trees") or a gene tree satisfying CONSTRAINT C (Section "No AD above NAD"), and a heuristic in the general case. We first introduce preliminary definitions. For a given tree U (weighted or not), consider the two following properties on U:

Property ONLY-NAD: U has no AD vertices;
Property ONLY-AD: U is rooted at an AD vertex and contains no NAD vertex.

We define the ***NAD-border*** of U as the set of roots of the maximum subtrees of U verifying Property ONLY-NAD, and the ***AD-border*** of U as the set of roots of the maximum subtrees of U verifying Property ONLY-AD.

ALGORITHM CORRECT-TREE is a recursive algorithm that takes as input a gene tree T and a species tree S, and outputs a number of leaf removals transforming T into a tree that is MD-consistent with S. It proceeds as follows:

- Stop condition - Lines 2 to 4: If T is MD-consistent with S, then no leaf removal is performed, and the algorithm terminates.
- Recurrence Loop - Lines 6 to 13: Resolve all maximum subtrees of T verifying CONSTRAINT C as described in Section "No AD above NAD", that is:

  1. Construct the weighted tree T^I (Lines 6-8);
  2. For each root x of a maximum subtree T_x^I of T^I satisfying CONSTRAINT C (Line 9), solve the WMAST problem on T_x^I, which leads to the weighted tree W_x^{MAX} (Line 10), compute the induced tree T_x (Line 11) and store the number of performed leaf removals (Line 12).

Algorithm Correct-Tree (*T, S*)
1. LeafRemoval=0;
2. IF T is a tree MD-consistent with S THEN
3.   RETURN (LeafRemoval)

4. END IF
5. $T^I = T$;
6. FOR ALL $x \in$ AD-border(T) DO
7.   Replace T_x^I by its induced weighted tree;
8. END FOR
9. FOR ALL $x \in$ NAD-border(T^I) DO
10. $W_x^{MAX} = $ WMAST(T_x^I);
        Replace T_x by the subtree induced by W_x^{MAX};
        LeafRemoval = LeafRemoval +
        $(v(T_x^I) - v(W_x^{MAX}))$;
11. END FOR
12. RETURN (LeafRemoval+CORRECT-TREE(T, S))

If T is a uniquely leaf-labelled tree then $T^I = T$, NAD-border(T^I) is reduced to the root of the tree, and thus loop 9–13 is just executed once. Moreover, as T^I is unweighted (all labels are equal to 1), WMAST is reduced to MAST. The whole algorithm thus reduces to one resolution of the MAST problem.

If T satisfies CONSTRAINT C, then NAD-border(T^I) is also reduced to the root of the whole tree, and thus loop 9–13 is just executed once. In this case, the methodology is the one following Theorem 1, and illustrated in Example 3.

In the general case, NAD-border(T^I) is not restricted to a single vertex, and loop 9–13 can be executed many times. Moreover, at the end of loop 9–13, the resulting tree is not guaranteed to be MD-consistent with S, as NAD vertices higher than those in NAD-border(T^I) may exist. ALGORITHM CORRECT-TREE may therefore be applied many times.

Theorem 2. Algorithm Correct-tree *has time-complexity* $O(n^2 \lg n)$, *where n is the size of T.*

Proof. Let n be the size of T. Loop 2–4 requires the LCA mapping between T and S, and the identification of AD and NAD vertices. As the LCA mapping can be computed in linear time [8,36], testing whether a tree T is MD-consistent with S can be tested in time $O(n)$. Clearly, Loop 6–8 can be executed in time $O(n)$. As for Loop 9–13, it has the time complexity $O(C)$ of WMAST. Therefore, the complexity for one execution of the recursive ALGORITHM CORRECT-TREE is $O(C)$. As in the worst case the algorithm can be executed $\Omega(n)$ times, the total worst case running time is $O(nC)$.

Let us consider the complexity of WMAST. The $O(n^2)$ algorithm of Steel and Warnow [33] naturally generalizes to the case of weighted trees, and leads to the same complexity, $O(n^2)$. However, we show in the rest of this proof that an instance of WMAST can be transformed into an instance of MAST in linear-time, which allows us to consider C as being the best complexity found for MAST,
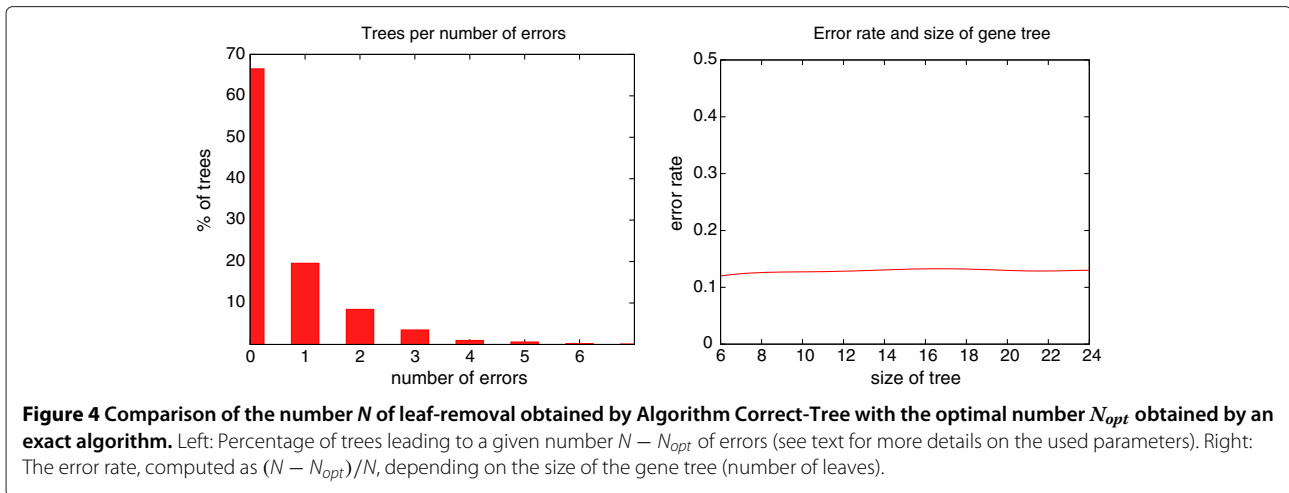
namely the $O(n \log(n))$ running-time of the algorithm given in [30].

Let G be a genome set, W be a weighted tree on G and S be a species tree for G. Then consider the *expanded genome set G_{exp}* obtained from G by replacing each genome g with weight c in W by a set of genomes $\{g_1, \cdots g_c\}$, the *expanded gene tree W_{exp}* obtained from W by replacing each leaf g with weight $c > 1$ by an *expanded leaf, i.e.* a caterpillar tree of size c containing the leaves $g_1, \cdots g_c$ (i.e. the tree $(g_c, (\cdots g_3, (g_2, g_1) \cdots))$), and the *expanded species tree S_{exp}* obtained from S by replacing each leaf g with weight $c > 1$ in W by a caterpillar tree of size c containing the leaves $g_1, \cdots g_c$. Then W_{exp} and S_{exp} are uniquely leaf-labelled trees. It is easy to see that a solution W_{exp}^{MAX} of MAST will contain, for any $g \in G$, either c or 0 leaves labelled g (*i.e.* either 0 or all leaves labelled g removed from W_{exp}). Therefore the *compressed tree W_{exp}^{MAX}*, obtained by recovering a single weighted leaf from each *expanded leaf* of W_{exp} that was not removed by MAST, is a solution to WMAST. Further, since we add at most a constant number of genes per leaf, we will affect the running time of MAST by at most a constant factor. □

Empirical results

We test the optimality of Algorithm Correct-Tree in the case of a gene tree satisfying Property AD-above-NAD (*i.e.* containing at least one AD vertex above a NAD vertex). Indeed, the algorithm is guaranteed to give the optimal solution otherwise (*i.e.* for trees satisfying the constraints of Section "Uniquely leaf-labelled gene trees" or Section "No AD above NAD"). We compared the number N of leaf-removals obtained by Algorithm Correct-Tree with the number N_{opt} obtained by the exact algorithm that tries all possible leaf-subset removals. More precisely, if the minimum number of leaf-removals output by Algorithm Correct-Tree is r, we try all subsets of $r - 1$, $r-2, \ldots, r-i$ leaf removals, and stop as soon as a tree that is MD-consistent with S is obtained. As the naive algorithm has clearly an exponential-time complexity, tests are performed on trees of limited size.

We considered a genome set of fixed size 5, and gene trees with 6 to 24 leaves. For each size s (from 6 to 24, with steps of 2), we generated 500 random gene trees of s leaves, and kept only those satisfying Property AD-above-NAD. The left diagram of Figure 4 shows that Algorithm Correct-Tree gives an exact solution for more than 65% of the trees (among all of those satisfying Property AD-above-NAD). Moreover, when N_{opt} differs from N_{opt}, in most cases the difference is 1. The right diagram of Figure 4 is obtained by averaging, for each size s, the results obtained for all the gene trees of that size. We can see that the error-rate, computed as $(N - N_{opt}/N$, is independent from the size of the tree, and did not exceed 0.15, based on our

Figure 4 Comparison of the number *N* of leaf-removal obtained by Algorithm Correct-Tree with the optimal number N_{opt} obtained by an exact algorithm. Left: Percentage of trees leading to a given number $N - N_{opt}$ of errors (see text for more details on the used parameters). Right: The error rate, computed as $(N - N_{opt})/N$, depending on the size of the gene tree (number of leaves).

simulation parameters. After testing other dependency factors (non-shown results), it appears that the error-rate only depends on the number of times the loop 9–13 of Algorithm Correct-Tree is executed, which is not directly related to the number of NADs or ADs in the tree.

Finally, we tested the ability of the approach to identify misplaced genes. To do so, we considered a genome set of fixed size 10, and gene trees of size *s* varying from 10 to 100 (with a step of 10). For a random species tree *S* and a random tree *T* of size *s* that is MD-consistent with *S*, we incorporate randomly *NbAdded* = *s*/10 leaves with randomly chosen labels. We then test how many "misplaced" leaves our method is able to detect. For each size *s*, results are averaged over 100 trees. Figure 5 shows the detection percentage of ALGORITHM CORRECT-TREE, which is computed as $(N/NbAdded) \times 100$. This detection
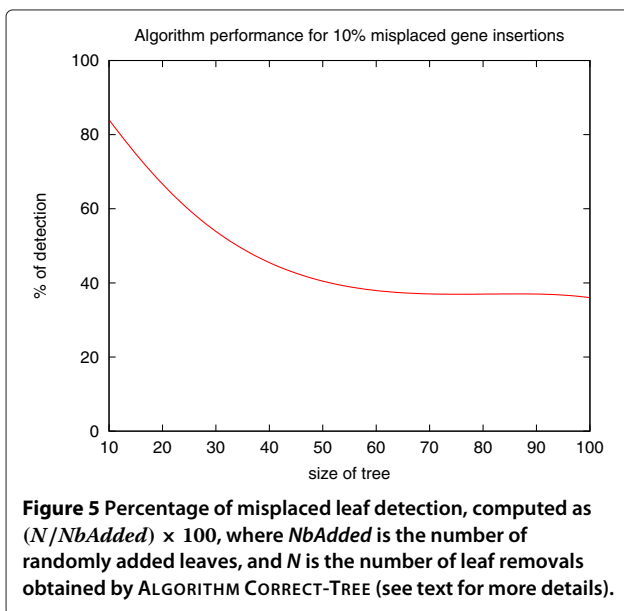
percentage decreases with increasing size of the gene tree. This is mainly due to the fact that as an MD-consistent tree needs no leaf removal, its detection percentage is always 100%, and that the more leaves we add (1 for a gene tree of size 10, but 10 for a gene tree of size 100) the less chance we have to end up with an MD-consistent tree. Removing the cases of MD-consistent trees lead to a detection percentage around 40%.

Conclusion

Based on observations pointing to NAD vertices of a gene tree as indicating potentially misplaced genes, we developed a polynomial-time algorithm for inferring the minimum number of leaf-removals required to transform a gene tree into an MD-tree, *i.e.* a tree with no NAD vertices. The algorithm is exact in the case of a uniquely leaf-labelled gene tree, or in the case of a gene tree that does not contain any AD vertex above a NAD vertex. In the general case, our algorithm exhibited near-optimal results under our simulation parameters. Unfortunately, NAD vertices can only reveal a subset of misplaced genes, as a randomly placed gene does not necessarily lead to a NAD vertex. Our experiments show that, on average, we are able to infer 40% of misplaced genes. However, the additional damage caused by a misplaced leaf leading to a NAD is an excessive increase of the real mutation-cost of the tree. Therefore, removing NADs can be seen as a preprocessing of the gene tree preceding a reconciliation approach, in order to obtain a better view of the duplication-loss history of the gene family.

Another use of our method would be to choose, among a set of equally supported gene trees output by a given phylogenetic method, the one that can be transformed to an MD-consistent tree by a minimum number of leaf removals.

A limitation of our approach is that a NAD resulting from a wrong bipartition {*a*, *b*; *c*} can be, *a priori*, solved

Figure 5 Percentage of misplaced leaf detection, computed as $(N/NbAdded) \times 100$, where *NbAdded* is the number of randomly added leaves, and *N* is the number of leaf removals obtained by ALGORITHM CORRECT-TREE (see text for more details).

by removing any gene from this bipartition. Our present approach is able to detect a number of misplaced genes but, in general, it is insufficient to detect precisely the genes that have been erroneously added in the tree. An extension would be to infer all optimal subsets of leaf removals, and to use bootstrapping values on the edges of the tree for a judicious choice of the genes to be removed.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

All authors read and approved the final manuscript.

References

1. Li WH, Gu Z, Wang H, Nekrutenko A: **Evolutionary analysis of the human genome.** *Nature* 2001, **409**:847–849.
2. Durand D, Haldórsson BV, Vernot B: **A hybrid micro-macro-evolutionary approach to gene tree reconstruction.** *J Comput Biol* 2006, **13**:320–335.
3. Zhang J: **Evolution by gene duplication: an update.** *TRENDS Ecol Evol* 2003, **18**(6):292–298.
4. Goodman M, Czelusniak J, Moore GW, Romero-Herrera AE, Matsuda G: **Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences.** *Syst Zool* 1979, **28**:132–163.
5. Doyon J-P, Scornavacca C, Gorbunov K, Szolloso G, Ranwez V, Berry V: **An effi. algo. for gene/species trees parsim. reconc. with losses, dup. and transf.** *J Comp Biol* 2010, **6398**:93– 108.
6. Hallett M, Lagergren J, Tofigh A: **Simultaneous identification of duplications and lateral transfers.** In *Proceedings of the Eight Annual International Conference on Computational Molecular Biology, RECOMB.* Edited by Bourne PE, Gusfield D. New York: ACM; 2004:347–356.
7. Tofigh A, Hallett M, Lagergren J: **Simultaneous identification of duplications and lateral gene transfers.** *IEEE/ACM Trans Comput Biol Bioinf* 2011, **8**:517–535.
8. Chauve C, El-Mabrouk N: **New perspectives on gene family evolution: losses in reconciliation and a link with supertrees.** In *Proceedings of the Thirteenth Annual International Conference on Computational Molecular Biology, RECOMB.* Edited by Batzoglou S: Springer; 2009:46–58. *volume 5541 of LNCS.*
9. Chen K, Durand D, Farach-Colton M: **Notung: Dating gene duplications using gene family trees.** *J Comput Biol* 2000, **7**:429–447.
10. Ma B, Li M, Zhang L: **From gene trees to species trees.** *SIAM J Comput* 2000, **30**:729–752.
11. Guigó R, Muchnik I, Smith TF: **Reconstruction of ancient molecular phylogeny.** *Mol Phylogenet Evol* 1996, **6**:189–213.
12. Page RDM, Charleston MA: **Reconciled trees and incongruent gene and species trees.** *DIMACS Ser Discrete Mathematics and Theor Comput Sci* 1997, **37**:57–70.
13. Bonizzoni P, Vedova Della G, Dondi R: **Reconciling a gene tree to a species tree under the duplication cost model.** *Theor Comput Sci* 2005, **347**:36–53.
14. Gorecki P, Tiuryn J: **DLS-trees: a model of evolutionary scenarios.** *Theor Comput Sci* 2006, **359**:378–399.
15. Page RDM: **Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas.** *Syst Biol* 1994, **43**:58–77.
16. Page RDM: **Genetree: comparing gene and species phylogenies using reconciled trees.** *Bioinformatics* 1998, **14**:819–820.
17. Hallett MT, Lagergren J: **New algorithms for the duplication-loss model.** In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, RECOMB.* Edited by Shamir R, Miyano S, Istrail S, Pevzner P, Waterman MS. New York: ACM; 2000:138–146.
18. Hahn MW: **Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution.** *Genome Biol* 2007, **8**:R141.
19. Chang WC, Eulenstein O: **Reconciling gene trees with apparent polytomies.** In *Proceedings of the 12th Conference on Computing and Combinatorics (COCOON), volume 4112 of Lecture Notes in Computer Science.* Edited by Chen DZ, Leepages DT. Taipei, Taiwan; 2006:235–244 .
20. Lafond M, Swenson KM, El-Mabrouk N: **An optimal reconciliation algorithm for gene trees with polytomies.** In *WABI, volume 7534 of LNBI/LNBI*; 2012:106–122.
21. Dondi R, El-Mabrouk N: **Minimum leaf removal for reconciliation: complexity and algorithms.** *Combinatorial Pattern Matching (CPM)* 2012. accepted.
22. Doroftei A, El-Mabrouk N: **Removing noise from gene trees.** In *WABI, volume 6833 of LNBI/LNBI*; 2011:76–91.
23. Chauve C, Doyon J-P, El-Mabrouk N: **Gene family evolution by duplication, speciation and loss.** *J Comput Biol* 2008, **15**:1043–1062.
24. Hahn MW, Han MV, Han S-G: **Gene family evolution across 12 drosophilia genomes.** *PLoS Genet* 2007, **3**:e197.
25. Page R D M: **Modified mincut supertrees.** In *LNCS, volume 2452 of WABI*; 2002:537–551.
26. Semple C, Steel M: **A supertree method for rooted trees.** *Discrete Appl Math* 2000, **105**:147–158.
27. Snir S, Rao S: *Using max cut to enhance rooted trees consistency*, Vol. 3; 2006.
28. Berge C: *Hypergraphs:Combinatorics of Finite Sets*, volume 45. Amsterdam, North-Holland; 1989.
29. Hao J, Orlin JB: **A faster algorithm for finding the minimum cut in a graph.** In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*; 1992:165–174.
30. Cole R, Farach-Colton M, Hariharan R, Przytycka T, Thorup M: **An $o(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees.** *SIAM J Comput* 2000, **30**(5):1385–1404.
31. Finden CR, Gordon AD: **Obtaining common pruned trees.** *J Classif* 1995, **2**:255–276.
32. Amir A, Keselman D: **Maximum agreement subtree in a set of evolutionary trees: matrics and efficient algorithms.** *SIAM J Comput* 1997, **26**:1656–1669.
33. Steel M, Warnow T: **Kaikoura tree theorems:computing the maximum agreement subtree.** *Inform Process Lett* 1993, **48**:77–82.
34. Bryant D: **Building trees, hunting for trees and comparing trees: Theory and method in phylogenetic analysis.** *PhD dissertation*, Department of Mathematics, University of Canterbury, UK; 1997.
35. Farach M, Przytycka TM, Thorup M: **On the agreement of many trees.** *Inf Process Lett* 1995, **55**(6):297–301.
36. Zhang LX: **On Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies.** *J Comput Biol* 1997, **4**:177–188.