Removing Noise from Gene Trees

Andrea Doroftei[1] and Nadia El-Mabrouk[2]

[1] DIRO, Université de Montréal, H3C 3J7, Canada, andreea.doroftei@umontreal.ca
[2] DIRO, mabrouk@iro.umontreal.ca

Abstract. Reconciliation is the commonly used method for inferring the evolutionary scenario for a gene family. It consists in "embedding" an inferred gene tree into a known species tree, revealing the evolution of the gene family by duplications and losses. The main complaint about reconciliation is that the inferred evolutionary scenario is strongly dependant on the considered gene tree, as few misplaced leaves may lead to a completely different history, with significantly more duplications and losses. As using different phylogenetic methods with different parameters may lead to different gene trees, it is essential to have criteria to choose, among those, the appropriate one for reconciliation. In this paper, following the conclusion of a previous paper, we flag certain duplication vertices of a gene tree, the "non-apparent duplication" (NAD) vertices, as resulting from the misplacement of leaves, and consider the optimization problem of removing the minimum number of leaves 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.

1 Introduction

Almost all genomes which have been studied contain genes that are present in two or more copies. As an example, duplicated genes account for about 15% of the proteins genes in the human genome [19]. In operational 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. For example, 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.

From a functional point of view, grouping genes by sequence similarity is not sufficient to infer a common function for genes. Indeed, it is important to distinguish between two kinds of homologs: *orthologs* which are copies in different species related through speciation, and thus likely to have similar functions, and *paralogs*, which are copies that have evolved by duplication, and more likely to have acquired new functions. Duplication is, indeed, a major source of gene innovation and creation of new functions [21]. In addition, gene losses, arising through the pseudogenization of previously functional genes, also play a key role in the evolution of gene families [2, 8, 11, 17, 21]. Understanding the evolution of gene families through speciation, duplication and loss is thus a fundamental question in functional genomics, evolutionary biology and phylogenomics [25, 28].

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 the duplication-loss model was pioneered by Goodman [13] and then widely accepted, utilized and also generalized to models of other processes such as horizontal gene transfer [18, 9, 27]. Several definitions of reconciliation exist in the literature, one of them expressed in term of "tree extension" [5]. More precisely, a *reconciliation* R between T and S is an extension of T (obtained

by grafting new subtrees onto existing branches 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) [6, 20]. The so called Lowest Common Ancestor (LCA) mapping between a gene tree and a species tree, formulated in [15, 24] and widely used [3, 10, 14, 20, 22–24], defines a reconciliation that minimizes both the duplication and mutation costs.

The main complaint about reconciliation methods is that the inferred duplication and loss history for a gene family is strongly dependant 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 [16]. 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? A strategy adopted in [6] is to explore the space of gene trees obtained from the original gene tree T by performing Nearest Neighbour Interchanges (NNI's) 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 choosing an appropriate gene tree among a set of possible trees, that consists in identifying a number of "misplaced" gene copies in a given gene tree. Criteria for identifying potentially misplaced leaves were given in a previous paper [5], where "non-apparent duplication vertices", were flagged as potentially resulting from the misplacement of leaves in the gene tree. The reason is that each one of these vertices reflects a phylogenetic contradiction with the species tree that is not due to the presence of duplicated gene copies. We develop algorithmic methods for removing the minimum number of leaves resulting in a gene tree T without any non-apparent duplication vertex.

In the next section, we begin by formally introducing our concepts. We then motivate and state our problem in Section 3. Section 4 is dedicated to the algorithmic developments. 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 5, we test the optimality of our algorithm, and the ability of the presented approach to identify misplaced genes. We finally conclude in Section 6.

2 Definitions

2.1 Trees

In this paper, we only consider rooted trees. Let $\mathcal{G} = \{1, 2, \cdots, g\}$ be a set of integers representing g different species (genomes). A ***species tree*** on $\mathcal{G}$ is a rooted binary tree with exactly g leaves, where each $i \in \mathcal{G}$ is the label of a single leaf (Figure 1(a)). A ***gene tree*** on $\mathcal{G}$ is a rooted binary tree where each leaf is labelled by an integer from $\mathcal{G}$, with possibly repeated leaves (Figure 1(b)). 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 occurs more than once, we will make no difference between a leaf and its label.

Given a tree U, the ***size of*** U, denoted $|U|$, is the number of leaves of U, and the ***genome set of*** U, denoted by $\mathcal{L}(U)$, is the subset of $\mathcal{G}$ 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, and the ***genome set of*** x, denoted by $\mathcal{L}(x)$, is the subset of $\mathcal{G}$ defined by the labels of the leaves of U_x (for example, in the tree of Figure 1(a), $\mathcal{L}(B) = \{1, 2\}$). 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 on a path from x to the root is an *ancestor* of x.

Given a tree U, a ***leaf removal*** consists in removing a given leaf i from U, and suppress the resulting degree two vertex. A tree U' obtained from U through a sequence of leaf removals is said to be ***included in*** U.

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 iff U_x verifies property P and, for any vertex y that is an ancestor of x, U_y does not verify property P.

2.2 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, 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 [3, 10, 14, 15, 20, 22, 24]. Here we define reconciliation in terms of subtree insertions, following the notation used in [4, 14]. We begin by introducing some definitions:

- A *subtree insertion* in a tree T consists in grafting a new subtree onto an existing branch of T.
- A tree T' is said to be an *extension* of T if it can be obtained from T by subtree insertions on the branches 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 $\mathcal{G}$ is an extension $R(T, S)$ of T that is DS-consistent with S.*

For example, the tree of Figure 1(c) is a reconciliation between the gene tree T of Figure 1(b) and the species tree of Figure 1(a). 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 $R(T, S)$.

2.3 LCA Mapping

The LCA mapping between T and S, denoted by M, maps every vertex t of T towards 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 1(b)). We denote by $\mathbf{d}(\mathbf{T}, \mathbf{S})$ the number of duplication vertices of T with respect to S.

This mapping induces a reconciliation $M(T, S)$ between T and S, where an internal vertex t of T leads to a duplication vertex in $M(T, S)$ iff 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 [3, 20, 22] for more details on the construction of a reconciliation based on the LCA mapping). Moreover, $M(T, S)$ is a reconciliation that minimizes all of the duplication, loss and mutation costs [5, 14]. In particular, $d(T, S)$ is the minimum duplication cost of any reconciliation between T and S.

2.4 Duplication Vertices and MD-trees.

Let T be a gene tree. It is immediate to see that 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

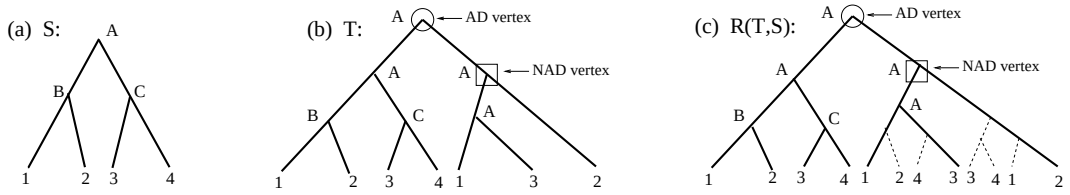

Fig. 1. (a) A species tree S for $\mathcal{G} = \{1, 2, 3, 4\}$. The three internal vertices of S are named A, B and C; (b) A gene tree T. A leaf label x indicates a gene copy in genome x. Internal vertices' labels are attributed according to the LCA mapping between T and S. Flagged vertices are duplication vertices of T with respect to S (Section 2.3); (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 vertices' 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.

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 1(b), the root is an AD vertex as its left and right subtree both contain a gene copy in genome 1. Following our notations in [5], given a species tree S, we say that T is a *Minimum-Duplication tree consistent with* S, or equivalently a ***tree that is MD-consistent with*** S, iff the duplication cost $d(T, S)$ is equal to the number of apparent duplications of T. In other words, all duplication vertices of T with respect to S are AD vertices.

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

3 Motivation and Problem Statement

Non-apparent duplication vertices point to disagreements between a gene tree and a species tree 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. 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.

Different observations made in [5] tend to support this hypothesis. In particular, using simulated data-sets based on the species tree of 12 *Drosophila* species given in [17] 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, in other words, a gene randomly placed in 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 additional damage caused by a misplaced leaf leading to a NAD vertex is the fact that it 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. If T is not MD-consistent with S, then an MD-consistent tree can always be obtained from T by performing a certain number of leaf removals. Indeed, a gene tree with only two leaves is always MD-consistent with any species tree. The optimization problem considered in this paper is therefore:

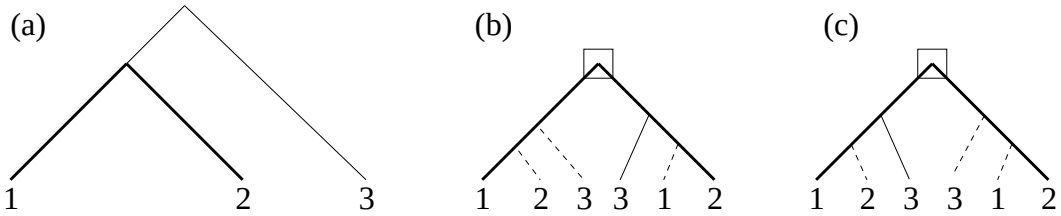

Fig. 2. Let $S = ((1,2),3)$ (the tree in (a)) be the phylogenetic tree for the three species $\{1,2,3\}$. Let $T = (1,2)$ be a gene tree. (a), (b) and (c) are the three possibilities for T after a random insertion of a leaf labelled 3. (a) is the only case leading to a tree without any NAD vertex. It 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 Problem:
Input: A gene tree T on $\mathcal{G}$ and a species tree S for $\mathcal{G}$;
Output: A tree T^{MAX} included in T and MD-consistent with S of maximum size (i.e. obtained form T by a minimum number of leaf removals).

4 Method

In the rest of this section, we assume that the set of genomes $\mathcal{G}$ and the species tree S for $\mathcal{G}$ are fixed. Let T be a gene tree for a gene family on $\mathcal{G}$. 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.

4.1 Uniquely leaf-labelled gene trees

When the considered gene family contains at most a unique gene copy per genome, the gene tree T is uniquely leaf-labelled. In this case, minimizing the number of leaves that should be removed from T to obtain an MD-tree consistent with S is equivalent to finding the maximum number of genes that lead to the same phylogeny in T and S. In other words, it is immediate to see that the Minimum Leaf Removal Problem reduces, in this case, to the Maximum Agreement Subtree Problem given below.

Maximum Agreement Subtree (MAST) Problem:
Input: A uniquely leaf-labelled gene tree T on $\mathcal{G}$ and a species tree S for $\mathcal{G}$;
Output: A tree T^{MAX} included in T and MD-consistent with S 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, respectively [7]. 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 [12], is one way to obtain such a consensus. Amir *et al.* [1] showed that computing a MAST of three trees with unbounded degree is NP-hard. However, in the case of two binary trees T and S (which is the case of interest in this paper), the problem is polynomial. The first polynomial-time algorithm for this problem was given by Steel and Warnow [26]. 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 size of the trees. Later, Cole *et al.* [7] developed an $O(n \log n)$ time algorithm, which, as far as we know, is the most efficient algorithm for solving the MAST problem on two binary trees.

4.2 No AD above NAD

In this section, we consider a tree T containing no AD vertex above a NAD vertex (Figure 3(a)). 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.

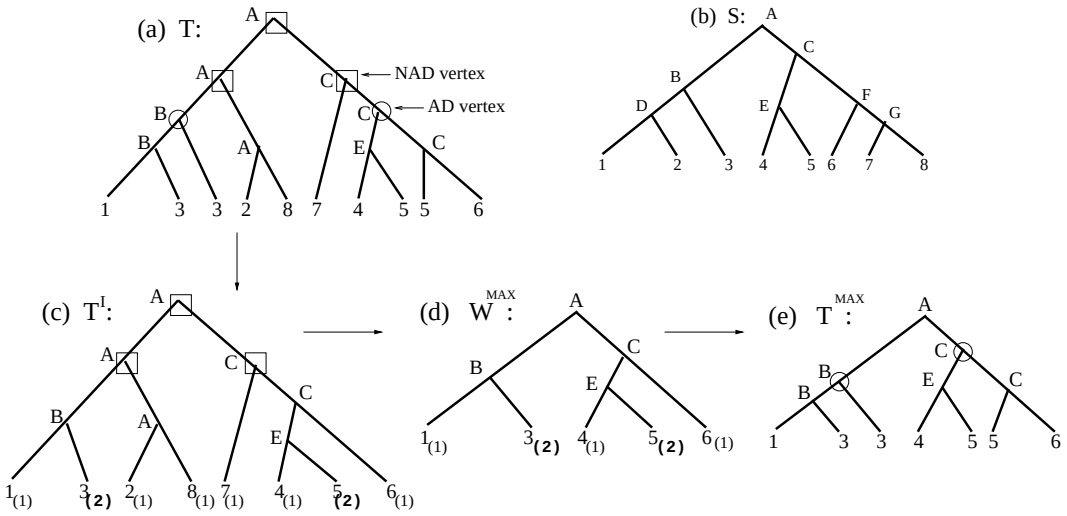

Fig. 3. Solving the MINIMUM LEAF REMOVAL PROBLEM for a tree satisfying CONSTRAINT C; (a) A gene tree T on $\mathcal{G} = \{1, 2, 3, 4, 5, 6, 7, 8\}$; (b) A species tree S for $\mathcal{G}$. Internal vertices of S are identified with different characters. Labels of internal vertices of T are attributed 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.

We show, in what follows, that the MINIMUM LEAF REMOVAL PROBLEM reduces, in this case, to a "generalization" of the MAXIMUM AGREEMENT SUBTREE 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 $\mathcal{G}$. The* weighted tree U^I induced by (U, S) *is the tree included in S obtained from 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 highest AD vertices). Then, the tree T^I obtained by replacing each T_i, for $1 \leq i \leq m$, by the weighted tree T_i^I induced by (T_i, S), is a weighted uniquely leaf-labelled tree. An example is

given in Figure 3(a), (b) and (c). Let ρ_s be the operation of removing the weighted leaf s from T^I. Then the *corresponding removals* in T consist in removing from T all leaves labelled s.

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

Weighted Maximum Agreement Subtree (WMAST) Problem:
Input: A weighted tree W on $\mathcal{G}$ and a species tree S for $\mathcal{G}$;
Output: A weighted tree W^{MAX} included in W and MD-consistent with S of maximum value.

We are now ready for the main theorem.

Theorem 1. *Let T be a gene tree satisfying* Constraint C. *Let W^{MAX} be a solution of the WMAST problem on T^I and S, and T^{MAX} be the subtree of T induced by W^{MAX}. Then T^{MAX} is a solution of the* Minimum Leaf Removal Problem *on T and S.*

In other words, Theorem 1 states that solving the Minimum Leaf Removal Problem on T is equivalent to solving the WMAST problem on T^I. We have developed an algorithm (not shown) for solving the WMAST problem, that is a direct generalization of the dynamic programming algorithm of Steel and Warnow [26] to weighted trees, and has the same quadratic running time complexity.

A complete example of the algorithmic methodology used for solving the Minimum Leaf Removal Problem on T and S following Theorem 1 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 child, let say the left child. Then there exist 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 can not 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 $W^{MAX}_{x_l}$ (respec. $W^{MAX}_{x_r}$) is a subset of the genome set of T_{x_l} (respec. 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: 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^I_x, 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^I_x should result in a different LCA mapping for x and y. Indeed, otherwise removing leaves from the corresponding subtree in T^I does not contribute to eliminate 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 $s \in \mathcal{G}$, 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 in resolving a NAD vertex x of T. From CONSTRAINT C, such 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 in solving x, we can assume that $a = s$. However, from our assumption, it remains, in T', a leaf labelled s. Thus: (1) either it remains, in T', at least one leaf labelled b and one leaf labelled c, 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 of removing leaf i labelled s for correcting 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 $|T^{MAX}| \geq |T'|$ □

4.3 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 4.1) or a gene tree satisfying CONSTRAINT C (Section 4.2), 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 (Figure 4) 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 4.2, that is:

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

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.

```
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 ∈ AD-border(T) DO
7.      Replace T_x^I by its induced weighted tree;
8.  END FOR
9.  FOR ALL x ∈ NAD-border(T^I) DO
10.     W_x^MAX = WMAST(T_x^I);
11.     Replace T_x by the subtree induced by W_x^MAX;
12.     LeafRemoval = LeafRemoval + (v(T_x^I) − v(W_x^MAX));
13. END FOR
14. RETURN(LeafRemoval+Correct-Tree(T, S))
```

Fig. 4. An algorithm that takes as input a gene tree T and a species tree S, and outputs the number of leaf removals "LeafRemoval" performed to transform T into a tree that is MD-consistent with S. Here, WMAST points out to an algorithm for solving the WMAST 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.

Complexity: Let n be the size of T. Loop 2- 4 requires to perform the LCA mapping between T and S, and identify AD and NAD vertices. As the LCA mapping can be computed in linear time [29, 5], 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 of WMAST. As stated in the later section, the $O(n^2)$ algorithm of Steel and Warnow [26] naturally generalizes to the case of weighted trees, and leads to the same running time complexity $O(n^2)$. Therefore, the complexity for one execution of the recursive Algorithm Correct-Tree is $O(n^2)$. As in the worst case, the algorithm can be executed n times, the total worst case running time complexity is $\mathbf{O(n^3)}$.

Notice that a more efficient algorithm running in $O(n \log(n))$ time exists for the MAST problem [7]. If, as we conjecture, this algorithm can be generalized to the WMAST problem, then it will lead to a time complexity in $O(n^2 \log n)$ for Algorithm Correct-Tree. This is a short-term perspective for improvement.

5 Results

We only 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 4.1 or Section 4.2). We compared the number *NbObtained* of leaf-removal obtained by Algorithm Correct-Tree with the number *NbOptimal* obtained by an exact naive algorithm that tries all possible leaf-subset removals. More precisely, if the minimum number of leaf-removal output by

Algorithm Correct-Tree is r, then, we try all subsets of $r-1$, $r-2 \cdots r-i$ 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 5 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 $NbOptimal$ differs from $NbOptimal$, in mot cases the difference is 1. The right diagram of Figure 5 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 $(NbObtained - NbOptimal)/NbObtained$, is independent from the size of the tree, and did not exceed 0.15, based on our simulation settings. 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.

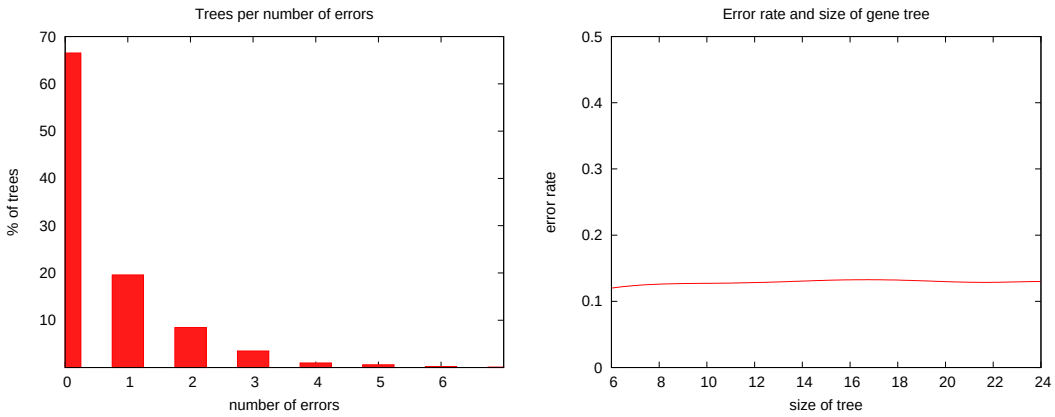

Fig. 5. Comparison of the number $NbObtained$ of leaf-removal obtained by Algorithm Correct-Tree with the optimal number $NbOptimal$ obtained by an exact algorithm. Left: Percentage of trees leading to a given number $NbObtained - NbOptimal$ of errors (see text for more details on the used parameters). Right: The error rate, computed as $(NbObtained - NbOptimal)/NbObtained$, depending on the size of the gene tree (number of leaves).

Finally, we tested the ability of the presented 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 6 shows the detection percentage of Algorithm Correct-Tree, which is computed as $(NbObtained/NbAdded) \times 100$. 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%.

6 Conclusion

Based on observations pointing to NAD vertices of a gene tree as indications of potentially misplaced genes, we developed a polynomial-time algorithm for inferring the minimum number of leaf-removals

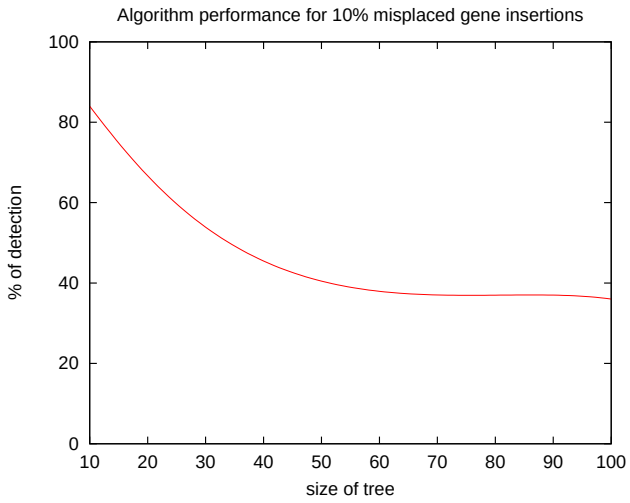

Fig. 6. Percentage of misplaced leaf detection, computed as $(NbObtained/NbAdded) \times 100$, where $NbAdded$ is the number of randomly added leaves, and $NbObtained$ is the number of leaf removals obtained by Algorithm Correct-Tree (see text for more details).

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 results very close to optimality under our simulation settings. 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 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.

Acknowledgments

We thank Sébastien Langevin for his help in the implementation of the algorithm, and for valuable discussion on the method.
Funding: Research supported by a grant to N.E.M. from the Natural Sciences and Engineering Research Council of Canada.

References

1. A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees: matrics and efficient algorithms. *SIAM J. Computing*, 26:1656- 1669, 1997.
2. T. Blomme, K. Vandepoele, S. De Bodt, C. Silmillion, S. Maere, and Y. van de Peer. The gain and loss of genes during 600 millions years of vertebrate evolution. *Genome Biology*, 7:R43, 2006.
3. P. Bonizzoni, G. Della Vedova, and R. Dondi. Reconciling a gene tree to a species tree under the duplication cost model. *Theoretical Computer Science*, 347:36–53, 2005.
4. C. Chauve, J.-P. Doyon, and N. El-Mabrouk. Gene family evolution by duplication, speciation and loss. *J. Comput. Biol.*, 15:1043-1062, 2008.
5. C. Chauve and N. El-Mabrouk. New perspectives on gene family evolution: losses in reconciliation and a link with supertrees. In *RECOMB 2009*, volume 5541 of *LNCS*, pages 46-58. Springer, 2009.
6. K. Chen, D. Durand, and M. Farach-Colton. Notung: Dating gene duplications using gene family trees. *Journal of Computational Biology*, 7:429–447, 2000.
7. R. Cole, M. Farach-Colton, R. Hariharan, T. Przytycka, and M. Thorup. An $o(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. *SIAM J. Computing*, 30(5):1385-1404, 2000.
8. J.A. Cotton and R.D.M. Page. Rates and patterns of gene duplication and loss in the human genome. *Proceedings of the Royal Society of London. Series B*, 272:277–283, 2005.
9. J.-P. Doyon, C. Scornavacca, K. Gorbunov, G. Szolloso, V. Ranwez, and V. Berry. An effi. algo. for gene/species trees parsim. reconc. with losses, dup. and transf. *J. Comp. Biol.*, 6398:93- 108, 2010.
10. D. Durand, B.V. Haldórsson, and B. Vernot. A hybrid micro-macroevolutionary approach to gene tree reconstruction. *Journal of Computational Biology*, 13:320–335, 2006.
11. E.E. Eichler and D. Sankoff. Structural dynamics of eukaryotic chromosome evolution. *Science*, 301:793–797, 2003.
12. C.R. Finden and A.D. Gordon. Obtaining common pruned trees. *J. Classification*, 2:255- 276, 1985.
13. M. Goodman, J. Czelusniak, G.W. Moore, A.E. Romero-Herrera, and G. Matsuda. Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Zoology*, 28:132–163, 1979.
14. P. Gorecki and J. Tiuryn. DLS-trees: a model of evolutionary scenarios. *Theoretical Computer Science*, 359:378–399, 2006.
15. R. Guigó, I. Muchnik, and T.F. Smith. Reconstruction of ancient molecular phylogeny. *Molecular Phylogenetics and Evolution*, 6:189–213, 1996.
16. M.W. Hahn. Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution. *Genome Biology*, 8(R141), 2007.
17. M.W. Hahn, M.V. Han, and S.-G. Han. Gene family evolution across 12 *drosophilia* genomes. *PLoS Genetics*, 3:e197, 2007.
18. M. Hallett, J. Lagergren, and A. Tofigh. Simultaneous identification of duplications and lateral transfers. In *RECOMB*. ACM, 2004.
19. W.H. Li, Z. Gu, H. Wang, and A. Nekrutenko. Evolutionary analysis of the human genome. *Nature*, 409:847–849, 2001.
20. B. Ma, M. Li, and L. Zhang. From gene trees to species trees. *SIAM J. on Comput.*, 30:729–752, 2000.
21. S. Ohno. *Evolution by gene duplication*. Springer, Berlin, 1970.
22. R.D.M. Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Systematic Biology*, 43:58–77, 1994.
23. R.D.M. Page. Genetree: comparing gene and species phylogenies using reconciled trees. *Bioinformatics*, 14:819–820, 1998.
24. R.D.M Page and M.A. Charleston. Reconciled trees and incongruent gene and species trees. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 37:57–70, 1997.
25. M.J. Sanderson and M.M. McMahon. Inferring angiosperm phylogeny from EST data with widespread gene duplication. *BMC Evolutionary Biology*, 7:S3, 2007.
26. M. Steel and T. Warnow. Kaikoura tree theorems:computing the maximum agreement subtree. *Inform. Process. Lett.*, 48:77-82, 1993.
27. A. Tofigh, M. Hallett, and J. Lagergren. Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 8:517-535, 2011.
28. I. Wapinski, A. Pfeffer, N. Friedman, and A. Regev. Natural history and evolutionary principles of gene duplication in fungi. *Nature*, 449:54–61, 2007.
29. L.X. Zhang. On Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology*, 4:177–188., 1997.