BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–12

Polytomy Refinement for the Correction of Dubious Duplications in Gene Trees

Manuel Lafond [1,*], Cedric Chauve [2,3], Riccardo Dondi [4] and Nadia El-Mabrouk [1,*]

[1]Department of Computer Science, Université de Montréal, Montréal (QC), Canada
[2]LaBRI, Université Bordeaux 1, Bordeaux, France
[3]Department of Mathematics, Simon Fraser University, Burnaby (BC), Canada
[4]Universitá degli Studi di Bergamo, Bergamo, Italy

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: Large scale methods for inferring gene trees are error-prone. Correcting gene trees for weakly supported features often results in non-binary trees, i.e., trees with polytomies, thus raising the natural question of refining such polytomies into binary trees. A feature pointing toward potential errors in gene trees are duplications that are not supported by the presence of multiple gene copies.

Results: We introduce the problem of refining polytomies in a gene tree while minimizing the number of created non-apparent duplications in the resulting tree. We show that this problem can be described as a graph-theoretical optimization problem. We provide a bounded heuristic with guaranteed optimality for well characterized instances. We apply our algorithm to a set of ray-finned fish gene trees from the *Ensembl* database to illustrate its ability to correct dubious duplications.

Availability: The C++ source code for the algorithms and simulations described in the paper are available at http://www-etud.iro.umontreal.ca/ lafonman/software.php.

Contact: lafonman@iro.umontreal.ca, mabrouk@iro.umontreal.ca

1 INTRODUCTION

With the increasing number of completely sequenced genomes, the task of identifying gene counterparts in different organisms becomes more and more important. This is usually done by clustering genes sharing significant sequence similarity, constructing gene trees and then inferring macro-evolutionary events such as duplications, losses or transfers through reconciliation with the phylogenetic tree of the considered taxa. The inference of accurate gene trees is an important step in this pipeline. While gene trees are traditionally constructed solely from sequence alignments [19, 28, 29], recent methods incorporate information from species phylogenies, gene order and other genomic footprints [1, 4, 13, 33, 35, 27, 38]. A large number of gene tree databases are now available [10, 16, 22, 26, 30].

But constructing accurate gene trees is still challenging; for example, a significant number of nodes in the *Ensembl* gene trees are labelled as "dubious" [16]. In a recent study, we have been able to show that about 30% of 6241 *Ensembl* gene trees for the genomes of the fishes Stickleback, Medaka, Tetraodon, and Zebrafish exhibit at least one gene order inconsistency, and thus are likely to be erroneous [24]. Moreover due to various reasons such as insufficient differentiation between gene sequences and alignment ambiguities, it is often difficult to support a single gene tree topology with high confidence. Several support measures, such as bootstrap values or bayesian posterior probabilities, have been proposed to detect weakly supported edges. Recently, intense efforts have been put towards developing tools for gene tree correction [3, 8, 17, 18, 6, 12, 32, 39, 34]. A natural approach is to remove a weakly supported edge and collapse its two incident vertices into one [2], or to remove "dubious" nodes and join resulting subtrees under a single root [23]. The resulting tree is non-binary with polytomies (multifurcating nodes) representing unresolved parts of the tree. A natural question is then to select a binary refinement of each polytomy based on appropriate criteria. This has been the purpose of a few theoretical and algorithmic studies conducted in the last years, most of them based on minimizing the mutation (i.e., duplication and loss) cost of reconciliation [5, 25, 36, 40].

In the present paper, we consider a different reconciliation criterion for refining a polytomy which consists in minimizing the number of *Non-Apparent Duplication* (NAD) nodes. A duplication node x of a gene tree (according to the reconciliation with a given species tree) is a NAD if the genome sets of its two subtrees are disjoint. In other words, the reason x is a duplication is not the presence of paralogs in the same genome, but rather an inconsistency with the species tree. Such nodes have been flagged as potential errors in different studies [7, 31, 16]. In particular, they correspond to the nodes flagged as "dubious" in *Ensembl* gene trees.

We introduce the polytomy refinement problem in Section 2, and we show in Section 3 how it reduces to a clique decomposition problem in a graph representing speciation and duplication relationships between the leaves of a polytomy. We develop a bounded heuristic in Section 4, with guaranteed optimality in well characterized instances. In Section 5 we exhibit a general methodology, using our polytomy refinement algorithm, for correcting NAD nodes of a gene tree. We then show in Section 6 that this approach is in agreement with the observed corrections of *Ensembl* gene trees from one release to another.

2 THE POLYTOMY REFINEMENT PROBLEM

Phylogenies and reconciliations. A *phylogeny* is a rooted tree which represents the evolutionary relationships of a set of elements (such as species, genes, ...) represented by its nodes: internal nodes are ancestors, leaves are extant elements, and edges represent direct descents between parents and children. We consider two kinds of phylogenies: species trees and gene trees. A species tree $\mathcal{S}$ describes the evolution of a set of related species, from a common ancestor (the root of the tree), through the mechanism of speciation. For our purpose, species are identified with *genomes*, and genomes are simply sets of genes. As for a gene tree, it describes the evolution of a set of genes, through the evolutionary mechanisms of speciation and duplication. Therefore, each gene g, extant or ancestral, belongs to a species denoted by $s(g)$. The set of genes in a gene tree is called a *gene family*. A leaf-label corresponds to a genome in a species tree, and to a gene belonging to a genome in a gene tree.

Given a a phylogeny T, we denote by $l(T)$ the leaf-set and by $V(T)$ the node-set of T. Given a node x of T, we denote by $l(x)$ and call the *clade of* x, the leaf-set of the subtree of T rooted at x. We call an *ancestor* of x any node y on the path from the root of T to the parent of x. In this case we write $y < x$. Two nodes x, y are unrelated if none is an ancestor of the other. For a leaf subset X of T, $lca_T(X)$, the *lowest common ancestor* (LCA) of X in T, denotes the farthest node from the root of T which is an ancestor of all the elements of X. In this paper, species trees are assumed to be binary: each internal node has two children, representing its direct descendants (see $\mathcal{S}$ in Figure 1). For an internal node x of a binary tree, we denote by x_ℓ and x_r the two children of x.

DEFINITION 1 (Reconciliation). *A* reconciliation *between a binary gene tree G and a species tree $\mathcal{S}$ consists in mapping each internal or leaf node x of G (representing respect. an ancestral or extant gene) to the species $s(x)$ corresponding to the LCA in $\mathcal{S}$ of the set $\{s(l),\ \text{for all } l \in l(x)\}$. Every internal node x of G is labelled by an* event *$E(x)$ verifying: $E(x)$ =*Speciation *(S) if $s(x)$ is different from $s(x_\ell)$ and $s(x_r)$, and $E(x)$ =*Duplication *otherwise.*

We define two types of duplication nodes of a gene tree G. A *Non-Apparent-Duplication* (NAD) is a duplication node x of G such that $\left(\cup_{x\in l(x_\ell)} x\right) \cap \left(\cup_{y\in l(x_r)} y\right) = \emptyset$. A duplication which is not a NAD is an *Apparent duplication (AD) node*, i.e., a node with the left and right subtrees sharing a common leaf-label. Therefore, any internal node x of G is of *type* S, AD or NAD.

The gene trees we consider might be non-binary. We call *polytomy* a gene tree with a non-binary root (see $\mathcal{F}$ in Figure 1).

DEFINITION 2 (Binary refinement). *A tree H_T is a* refinement *of a tree T if and only if the two trees have the same leaf-set and T can be obtained from H_T by contracting some edges. When H_T refines T, each node of T can be mapped to a unique node of H_T so that the ancestral relationship is preserved. H_T is a* binary refinement *of T if and only if H_T is binary and is a refinement of T.*

In this paper, as only binary refinements are considered, we omit the term binary from now.

Problem statement. The general problem we address is the following: Given a non-binary gene tree G and a species tree $\mathcal{S}$, find a refinement of G containing the minimum number of NADs with respect to $\mathcal{S}$. Such a refinement of G is called a *minimum refinement* of G w.r.t. $\mathcal{S}$.

Hence, we aim at refining each non-binary node of G. We first show that each such non-binary node of G can be refined independently of the other non-binary nodes.

THEOREM 1. *Let $\{G_i,\ \text{for } 1 \le i \le n\}$ be the set of subtrees of G rooted at the n children $\{x_i,\ \text{for } 1 \le i \le n\}$ of the root of G. Let $H_{min}(G_i, S)$ be a minimum refinement of G_i w.r.t. $\mathcal{S}$. Let G' be the tree obtained from G by replacing each G_i by $H_{min}(G_i, \mathcal{S})$. Then a minimum refinement of G is a minimum refinement of G'.*

It follows from Theorem 1 that a minimum refinement of G can be obtained by a depth-first procedure iteratively solving each polytomy G_x, for each internal node x of G.

In the rest of this paper, we consider G as a polytomy, and we denote by $\mathcal{F}$ the forest $\{G_1, G_2, \cdots G_n\}$ obtained from G by removing the root. For simplicity, we make no difference between a tree G_i of $\mathcal{F}$ and its root. In particular, $s(G_i)$ corresponds to $s(root(G_i))$, where $root(G_i)$ is the root of G_i (see Figure 1). We are now ready to define the main optimization problem we consider.

Minimum NAD Polytomy Refinement (MinNADref) Problem :
Input: A polytomy G and a species tree $\mathcal{S}$;
Output: In the set $\mathcal{H}(G)$ of all refinements of G, a refinement H with the minimum number of NAD nodes. Such a refinement is called a *solution to the MinNADref problem*.

3 A GRAPH-THEORETICAL CHARACTERIZATION

We show (Theorem 2) that the MinNADref Problem reduces to a clique decomposition problem on a graph that represents the impact, in terms of NAD creation, of joining pairs of trees from $\mathcal{F}$.

The join graph of a polytomy. We first define a graph R based on the notion of *join*. A join is an unordered pair $\{G_1, G_2\}$ where $G_1, G_2 \in \mathcal{F}$. The *join operation* j on $\{G_1, G_2\}$ consists in joining the roots of G_1 and G_2 under a common parent; we denote by $G_{1,2}$ the resulting join tree. We call the *join type* of $j = \{G_1, G_2\}$, and denote by $jt(G_1, G_2)$, the reconciliation label of the node created by joining G_1 and G_2 (i.e., the root of $G_{1,2}$), where $jt(G_1, G_2) \in \{S, AD, NAD\}$, respectively for Speciation, Apparent Duplication and Non-Apparent Duplication, w.r.t. the species tree $\mathcal{S}$.

We denote by $R = (V, E)$ the join graph of $\mathcal{F}$, defined as the unoriented complete graph on the set of vertices $V = \mathcal{F}$, where each edge (join) is labelled by the corresponding join type (see Fig. 1). We denote by R_S and R_{AD} the subgraphs of R defined by the edges of type respectively S and AD. We call a connected component of R_{AD} an *AD-component*.

Let $\mathcal{F}'$ be the new forest obtained by replacing the two trees G_1 and G_2 of $\mathcal{F}$ by the join tree $G_{1,2}$. The rules given below, following directly from the definition of speciation and duplication in reconciliation, are used to update the join type $jt(G_{1,2}, T)$ for any $T \in \mathcal{F} \setminus \{G_1, G_2\}$.

RULESET 1.

1. *If $jt(G_1, T) = AD$ or $jt(G_2, T) = AD$, then $jt(G_{1,2}, T) = AD$;*

2. *Otherwise, if* $jt(G_1, T) = NAD$ *or* $jt(G_2, T) = NAD$*, then* $jt(G_{1,2}, T) = NAD$*;*
3. *Otherwise, if* $lca(T)$ *is not a descendant of* $lca(G_{1,2})$*, then* $jt(G_{1,2}, T) = S$*;*
4. *Otherwise,* $jt(G_{1,2}, T) = NAD$*.*

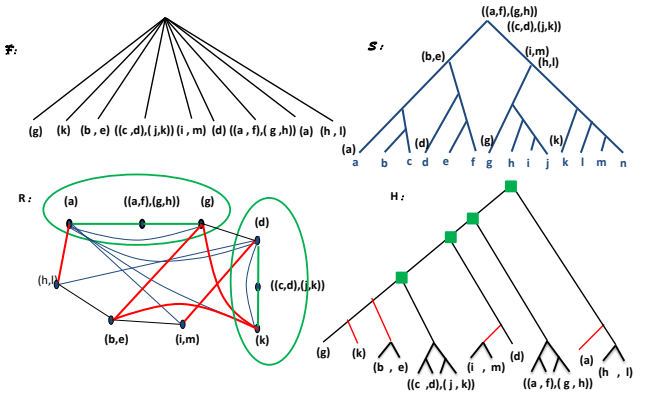

Fig. 1. A forest $\mathcal{F}$, a species tree $\mathcal{S}$ and the corresponding graph R. Each gene tree G of $\mathcal{F}$ is attached to its corresponding node $s(G)$ in $\mathcal{S}$. In R, joins of type AD are represented by green lines. All other lines are the joins of type S. Non-trivial AD-components (AD-components containing at least two nodes) are represented by green ovals. Red lines in R represent a vertex-disjoint clique W of R_S. Here, $R_{AD} \cup W$ has a single connected component, which leads to the binary refinement H of $\mathcal{F}$ with no NAD. After the joins of W are applied (red edges in H), the speciation-free forest can be joined with four joins AD (green vertices in H).

Clique Decomposition of the join graph. Let a *join sequence* $J = (J_1, J_2, \ldots, J_{|J|})$ be an ordered list of joins. We denote by $\mathcal{F}(J, i)$ the forest obtained after applying the first i joins of J, starting with $\mathcal{F}$. Note that $\mathcal{F}(J, 0) = \mathcal{F}$, and that $J_i \in J$ is a join on $\mathcal{F}(J, i-1)$. Let $\mathcal{J}$ denote the set of all possible join sequences of size $|\mathcal{F}| - 1$. Clearly, applying all joins of a sequence $J \in \mathcal{J}$ yields a single binary tree; and there exists a gene tree $H \in \mathcal{H}(G)$ with d NADs if and only if there exists a join sequence $J \in \mathcal{J}$ with d joins of type NAD. We refine this property by showing that there is a solution to the MinNADref Problem where all duplication nodes are ancestral to all speciation nodes (see the tree H of Figure 1 for an example). The proof (not shown) makes abundant use of Ruleset 1.

LEMMA 1. *There exists a binary refinement* $H \in \mathcal{H}(G)$ *with* d *NADs if and only if there exists a join sequence* $J \in \mathcal{J}$ *with* d *joins of type NAD such that, if* $J_i \in J$ *is the first join not of type* S *in* J*, then all following joins* J_j*, for* $j > i$*, are of type AD or NAD.*

We define a *speciation tree* as a gene tree in which every internal node is a Speciation node. We deduce from the previous lemma that we can obtain a solution H to the MinNADref Problem by creating a forest of speciation trees first, then successively joining them with joins of type AD or NAD. As the nodes of R corresponding to the leaves of a given speciation subtree of H are pairwise joined by speciation edges, they form a clique in R_S (in Fig. 1 the cliques in red are selected and the corresponding joins are applied to compute refinement H). The next theorem makes the link between the number of NADs of H and the cliques of R_S. For a set W of vertex-disjoint cliques of R_S, we denote by $R_{AD} \cup W$ the graph defined by the union of the edges of R_{AD} and W.

THEOREM 2. *A solution to the* MinNADref Problem *has* d *NADs if and only if, among all graphs* $R_{AD} \cup W$ *where* W *is a set of vertex-disjoint cliques of* R_S*, at least one has* $d+1$ *connected components and none has less than* $d+1$ *connected components.*

The proof of Theorem 2 is constructive. Given an optimal set W of vertex-disjoint cliques of R_S, it leads to an optimal refinement H. Unfortunately, it can be shown that, given an arbitrary graph with two edge colours AD and S, finding if there exists a set W yielding a given number of connected components is an NP-hard problem (proof not shown). However, R is constrained by the structure of a species tree, which restricts the space of possible join graphs. An arbitrary complete graph R with edges labelled on the alphabet $\{S, AD, NAD\}$ is said to be *valid* if there exists a species tree and a polytomy whose join graph is R. We characterize below the valid graphs in terms of forbidden induced subgraphs. The proof is based on an exploration of all possible sets of valid edges.

THEOREM 3. *A graph* R *is valid if and only if* R_S *is* $\{P_4, 2K_2\}$*-free, meaning that no four vertices of* R_S *induce a path of length 4, nor two vertex-disjoint edges.*

Although we have not been able to find an exact polynomial-time algorithm for the MinNADref Problem, this very constrained structure of the R graph yields a bounded heuristic for this problem with good theoretical properties described in the next section.

REMARK 1. *The* P_4*-free property, that was already introduced in relation with reconciliations in [21], is of special interest, as many NP-hard problems on graphs have been shown to admit polynomial time solutions when restricted to this class of graphs. Unfortunately we can prove that, given an arbitrary* P_4*-free graph on which we add AD edges, finding an optimal* W *is still NP-hard (proof not shown). However, the added* $2K_2$*-free restriction imposes a rigid structure on the graph at hand, and we conjecture that there exists a polynomial time algorithm to find an optimal* W*.*

4 A BOUNDED HEURISTIC

We first describe a general approach based on the notion of *useful speciations*, followed by a refinement of this approach with guaranteed optimality criteria.

DEFINITION 3. *Let* $J = (J_1, \ldots, J_{|J|})$ *be a join sequence. A join* $J_i = \{G_1, G_2\}$ *of* J *is a* useful speciation *if* $jt(G_1, G_2) = S$ *and* G_1, G_2 *are in two different AD-components of the* R *graph obtained after applying the* $J_1, \ldots J_{i-1}$ *joins.*

Hence, if R has c AD-components, finding a zero NAD solution becomes the problem of finding a join sequence with $c-1$ useful speciations. For example, the graph R in Figure 1 has 5 AD-components (3 trivial and 2 non-trivial), and thus the 4 useful speciations represented by the red lines lead to a 0 NAD solution (the binary tree H). In the general case, the problem we face is to select as many useful speciations as possible, as the resulting AD-components will have to be connected by NAD joins. If we define a *speciation-free* forest as a forest $\mathcal{F}$ such that no edge of its join graph

R is a speciation edge, following Lemma 1, we would like to first compute a set of useful speciations that results in a speciation-free forest whose join-graph has the least number of AD-components.

DEFINITION 4. *A* lowest useful speciation *is a useful speciation edge $\{G_1, G_2\}$ of R_S such that $s(G_{1,2})$ is not the ancestor of any $s(G_{i,j})$, for $\{G_i, G_j\}$ being another useful speciation edge of R_S.*

Lowest useful speciations fit naturally in the context of bottom-up algorithms where speciations edges that correspond to lower vertices of $\mathcal{S}$ are selected prior to speciations edges corresponding to ancestral species. The theorem below shows that proceeding along these lines ensures that the resulting join sequence contains at least half of the optimal number of useful speciation.

THEOREM 4. *Let s be the maximum number of useful speciations leading to a solution to the* MinNADref *problem. Then any algorithm that creates a speciation-free forest through lowest useful speciations makes at least $\lceil s/2 \rceil$ useful speciations.*

This theorem implicitly defines a heuristic with approximation ratio 2 on the number of useful speciations, that visits $\mathcal{S}$ in a bottom-up way, making useful speciations (that would thus be lowest useful speciations) whenever such an edge is available.

We now describe an improved version of this general heuristic principle. A detailed example is given in Figure 2. The main idea is to consider a bottom-up traversal of the species tree $\mathcal{S}$, and for each visited vertex s, to find a useful set of speciation edges by finding a matching in a bipartite graph. More precisely, for a node $s \in V(\mathcal{S})$, we consider the complete bipartite graph $\mathcal{B} = (X \cup Y, \{xy | x \in X, y \in Y\})$ such that the *left* (respec. *right*) subset X (respec. Y) contains all the trees G_i of $\mathcal{F}$ where $s(G_i)$ is on the left (respec. right) subtree of s. Consider the two partitions AD_X and AD_Y of X and Y respectively into AD-components. The key step of our heuristic is to find a matching $M \subseteq E(\mathcal{B})$ of useful speciations between AD_X and AD_Y, called a *useful matching*. For example, in Figure 2, the bipartite graph and matching illustrated for Step 3 correspond to node l and that of Step 4 to node m of S.

Notice that not all edges of $\mathcal{B}$ correspond to useful speciations. Indeed it is possible that for some $x \in X$ and some $y \in Y$, although $\{x, y\}$ is a speciation edge, x and y are in the same AD-component of R due to another tree z not in $\mathcal{B}$ such that $\{x, z\}$ and $\{z, y\}$ are AD-edges. For example in Figure 1, although $\{(a), (g)\}$ is a join of type S, the trees (a) and (g) are in the same AD-component of R due to the tree $((a, f), (g, h))$. For a vertex x of X (respec. y of Y), denote by $AD(x)$ (respec. $AD(y)$) the component of AD_X (respec. AD_Y) containing x (respec. y). We indicate the fact that $AD(x)$ and $AD(y)$ belong to the same AD-component in R by adding two dummy genes b_1 in $AD(x)$ and b_2 in $AD(y)$, and a *bridge* $\{b_1, b_2\}$ in $E(\mathcal{B})$. Such bridges will be included in every matching, preventing to include non-useful speciation edges.

An instance P of the problem associated with a vertex s of $\mathcal{S}$ is denoted by $P = (X, Y, AD_X, AD_Y, B)$ where X, Y, AD_X, AD_Y are defined as above and B is the set of bridges induced by R. The graph corresponding to P, i.e., the complete bipartite graph on sides X and Y to which we added the bridge edges B, is denoted by $\mathcal{B}(P)$. The whole method is summarized in Algorithm 1 MinNADref$(\mathcal{F}, \mathcal{S})$ and illustrated on a simple example in Figure 2.

Finding a useful matching of maximum size can be done in polynomial time by Algorithm 2. For an instance $P = (X, Y, AD_X, AD_Y, B)$, the algorithm progressively increments the set M of speciation edges, eventually leading to a useful matching of maximum size. At a given step, let $\mathcal{G}_{P,M}$ be the graph with vertices $X \cup Y$ and edges $E_{P,M} = E_{AD} \cup M$, where E_{AD} is the set of AD edges of R connecting vertices of $X \cup Y$. Components $AD_{X_i} \in AD_X$ and $AD_{Y_j} \in AD_Y$ are *linked* if there is a path in $\mathcal{G}_{P,M}$ linking a vertex of AD_{X_i} to a vertex of AD_{Y_j}, and *not linked* otherwise.

Algorithm 1 $MinNADref(\mathcal{F}, \mathcal{S})$

for each node s of $\mathcal{S}$ in a bottom-up traversal of $\mathcal{S}$ **do**
  Let $P = (X, Y, AD_X, AD_Y, B)$ be the problem instance corresponding to s;
  Find a useful matching M of $\mathcal{B}(P)$ of maximum size (Algorithm $MaxMatching$ below);
  Apply each speciation of M, and update $\mathcal{F}$
end for
For each connected component C of R_{AD}, join the trees of C under AD Nodes;
If there is more than one tree remaining, join them under NAD nodes.

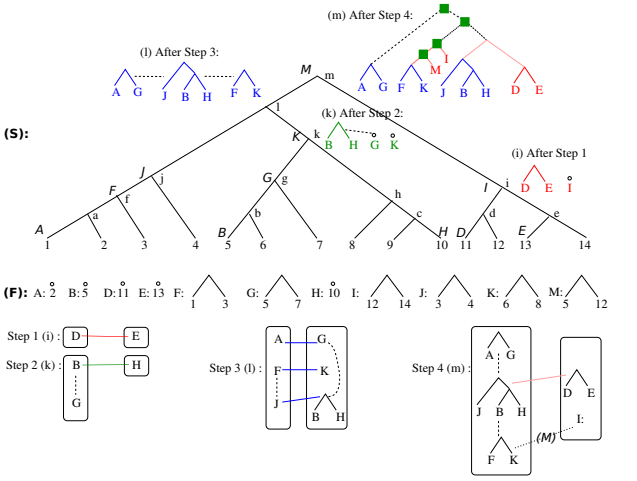

Fig. 2. A species tree $\mathcal{S}$ and a forest $\mathcal{F}$ of binary trees forming the polytomy. The trees of $\mathcal{F}$ are placed on $\mathcal{S}$ according to their LCA. The i, k, l and m nodes of $\mathcal{S}$ are annotated with the forest obtained after running Algorithm 2 on these nodes. Their corresponding complete bipartite matching instances are illustrated at the bottom. AD joins are represented by dotted lines, useful matching are represented by plain lines (we omit drawing all the other edges of the complete bipartite graphs). Note that there is a bridge induced by M between (F, K) and I at step 4. In the fourth step, we obtain a single connected component, which allows, in a final step, to connect all the subtrees by AD nodes (final tree is on the top of the figure).

THEOREM 5. *Given an instance $P = (X, Y, AD_X, AD_Y, B)$, Algorithm 1 finds a useful matching M of maximum size.*

Algorithm 1 is a heuristic as it may fail to give the optimal solution (refinement with minimum number of NADs), as in Figure 1 for example. In this example, a bottom-up approach would greedily speciate a and d, which cannot lead to the optimal solution. However, we prove in Theorem 6 that if transitivity holds for the duplication join type, then Algorithm 1 is an exact algorithm for

Algorithm 2 $MaxMatching(X, Y, X_X, AD_Y, B)$

$D = \emptyset$; $M = B$;
while $D \neq X \cup Y$ **do**
  Find $C \in AD_X \cup AD_Y$ of maximum cardinality with vertices not included in D, if any; assume w.l.o.g. $C = AD_{X_i} \in AD_X$;
  for each $x \in C$ that is not incident to an edge in M **do**
    if there is an $y \in Y$ such that $AD(y)$ is not linked to C **then**
      Find such y with $AD(y)$ of maximum cardinality;
      Add the vertices x and y to D and add the speciation edge $\{x, y\}$ to M;
    end if
  end for
  Add remaining vertices of C to D;
end while

the MinNADref problem. The example of Figure 1 does not satisfy this property, as $\{(a), ((a, f), (g, h))\}$ is a join of duplication type (AD), $\{((a, f), (g, h)), (g)\}$ is a join of duplication type but $\{(a), (g)\}$ is a join of speciation type.

THEOREM 6. *(1) Let s be the maximum number of useful speciations leading to a solution to the* MinNADref *problem. Then Algorithm 1 makes at least $\lceil s/2 \rceil$ useful speciations. (2) If, for every node s of $\mathcal{S}$ the instance P corresponding to s has no bridges, then Algorithm 1 outputs a refinement of the input polytomy with the maximum number of useful speciations.*

The following corollary provides an alternative formulation of the optimality result given by the above theorem.

COROLLARY 1. *Algorithm 1 exactly solves the MinNADref problem for an input $(\mathcal{F}, \mathcal{S})$ such that each AD-component of the corresponding graph R is free from S edges (i.e., there is no S edge between any two vertices of a given AD-component).*

5 GENE TREE CORRECTION

The polytomy refinement problem is motivated by the problem of correcting gene trees. Duplication nodes can be untrusted for many reasons, one of them being the fact that they are NADs, pointing to disagreements with the species tree that are not due to the presence of duplicated genes. Different observations tend to support the hypothesis that NAD nodes may point at erroneous parts of a gene tree [7, 32]. For example, the *Ensembl Compara* gene trees [37] have all their NAD nodes labelled as "dubious". In [7], using simulated data-sets based on the species tree of 12 *Drosophila* species given in [20] 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. Although suspected to be erroneous, some NAD nodes may still be correct, due to a high number of losses. However, in the context of reconciliation, the additional damage caused by an erroneous NAD node is the fact that it significantly increases the real rearrangement cost of the tree [32]. Therefore, tools for modifying gene trees according to NADs are required. We show now how Algorithm 1 can be used in this context.

In [23], a method for correcting untrusted duplication nodes has been developed. The correction of a duplication node x relies on *pushing x by multifurcation*, which transforms x into a speciation node with two children being the roots of two polytomies. Figure 3 recalls the *pushing by multifurcation* procedure. These polytomies are then refined by using an algorithm developed in [25], which optimizes the mutation cost of reconciliation.

In the context of correcting NADs, we use the same general methodology, but now using Algorithm MinNADref for refining polytomies. Removing all NADs of a gene tree can then be done by iteratively applying the above methodology on the highest NAD node of the tree (the closest to the root).

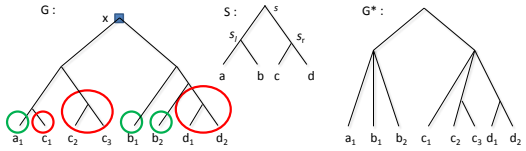

Fig. 3. A gene tree G and a species tree $\mathcal{S}$, from which we obtain G^* by pushing x by multifurcation. Here, x is a NAD, and is pushed by taking the forest of maximal subtrees of G that only have genes from species in the s_l subtree (green), then another forest for the s_r subtree (red) in the same manner. Both these forests are joined under a polytomy, which are then joined under a common parent, so the root of G^* is a speciation.

6 RESULTS

Simulated data. Simulations are performed as follows. For a given integer n, we generate a species tree $\mathcal{S}$ with a random number of leaves between $0.5n$ and $3n$. We then generate a forest $\mathcal{F} = (G_1, \ldots, G_n)$ of cherries by randomly picking, for each cherry $G_i \in \mathcal{F}$, one node $s_i \in S$ and two leaves one from each of the two subtrees rooted at s_i. Any leaf of $\mathcal{S}$ is used at most once (possibly by adding leafs to $\mathcal{S}$ if required), leading to a set of cherries related through joins of type S or NAD. Then, for each pair $\{G_i, G_j\}$ with join type NAD, we relate them through AD with probability $1/2$ (or do nothing with probability $1/2$), by adding a duplicated leaf.

For each pair $(S, \mathcal{F})$, we compared the number of NADs found by Algorithm MinNADref with the minimum number of NADs returned by an exact algorithm exploring all possible binary trees that can be constructed from $\mathcal{F}$. We generated a thousand random $\mathcal{S}$ and $\mathcal{F}$ for each $n \geq 4$. We stopped at $n = 14$, as the brute-force algorithm is too time costly beyond this point. Over all the explored datasets simulated as described above, Algorithm MinNADref was able to output an optimal solution, i.e., a refinement with the minimum number of NADs. Therefore, the examples on which the heuristic fails seem to be rare, and the algorithm performs well on polytomies of reasonable size.

We then wanted to assess how the NAD minimization criterion differs from the rearrangement cost minimization criterion. We generated 960 random instances with forests of sizes ranging between 5 and 100 (10 instances for each $5 \leq n \leq 100$). We compared the output of Algorithm MinNADref with that of Algorithm MinDLref, given in [25], which computes refinement minimizing the duplication+loss (DL) cost of reconciliation with the species tree. Both algorithms gave the exact same refinement

for only 12 instances (1.25%). As expected, Algorithm MinNADref always yielded a refined tree with a lower or equal number of NADs than the tree given by Algorithm MinDLref, but always had a higher or equal DL-cost. However in many cases, minimizing the DL-score did not minimize the number of NADs, as in 377 instances (39.3%), Algorithm MinNADref yielded strictly less NADs than Algorithm MinDLref.

Ensembl Gene Trees. Next we tested the relevance of the proposed gene tree correction methodology, by exploring how *Ensembl* gene trees are corrected from one release to another. As the *Ensembl* general protocol for reconstructing gene trees does not change between releases, the observed modifications on gene trees are more likely due to modifications on gene sequences.

We used the *Ensembl Genome Browser* to collect all available gene trees containing genes from the monophyletic group of ray-finned fishes (Actinopterygii), and filtered each tree to preserve only genes from the taxa of interest (ray-finned fish genomes). We selected from both Releases 74 (the present one) and 70 the 1096 gene trees that are present in both with exactly the same set of genes from the monophyletic group of fishes, and with less NAD nodes in Release 74. We wanted to see to what extent our general principle of correcting a NAD by transforming it to a speciation node is observed by comparing Rel.70 to Rel.74. Such a transformation requires to preserve the clade of the corrected NAD node x of the initial tree, meaning that $l(x)$ should also be the leaf-set of a subtree in the corrected tree. For more than 90% of these trees (993 trees), the highest NAD node clade was preserved in Rel.74. Moreover, among all such nodes that were corrected, i.e., were not NAD nodes in Rel.74 (641 trees) almost all were transformed into speciation nodes (630 trees), which strongly supports our correction paradigm.

In order to evaluate our methodology for correcting NADs, we applied it to the highest NAD node of each of the 1096 aforementioned trees of Rel.70. Figure 4 illustrates a comparison between the corrected trees (Rel.70C, C standing for "Corrected") obtained by our methodology and those of Rel. 74. Pairwise comparisons are based on the normalized Robinson-Foulds (RF) distance (number of identical clades divided by the total number of clades). The yellow curve shows a good correlation between Rel.70C and Rel.74, with about 65% exhibiting more than 80% similar clades between Rel.70C and Rel.74. If we reduce the set of trees to those for which the highest NAD node is also transformed to a speciation node in Rel.74 (630 trees), the correlation is even better (blue curve of Figure 4), with 44% of trees being identical (277 over 630 trees) and about 80% exhibiting more than 80% similar clades between Rel.70C and Rel.74. Now, in order to specifically evaluate Algorithm MinNADref, we further restricted the set of trees to those giving rise to a non-trivial polytomy (i.e., polytomy of degree > 2) after the *pushing by multifurcation*, which leads to a set of 117 trees. Overall the results for these trees (red curve in Figure 4) are very close to those observed for all trees (yellow curve) detailed above.

We then wanted to evaluate our correction of the 117 aforementioned trees compared to trees in Rel.74. Figure 5 provides an evaluation of the corrected trees (yellow curve) compared to those in Rel. 74 (blue curve) based on the normalized RF distance with the initial trees in Rel.70. Overall, the initial tree is closer to our correction than to the one of Rel.74. Therefore, even though gene trees of Rel.74 are likely to have stronger statistical support with respect to the gene sequences provided in Rel.74, our correction

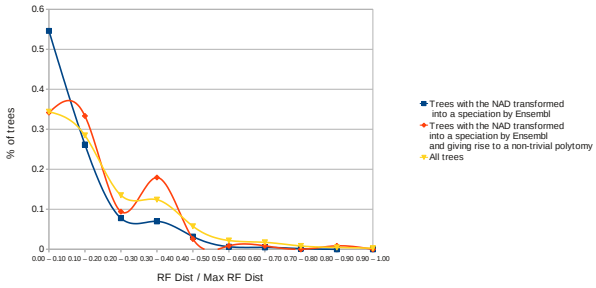

Fig. 4. Normalized RF-distance between corrected gene trees (by modification of the highest NAD) from Rel. 70 and corresponding gene trees in Rel. 74. Blue curve: transformation of the highest NAD into a speciation. Red curve: trees with a non-trivial polytomy after pushing by multifurcation. Yellow curve: all trees.

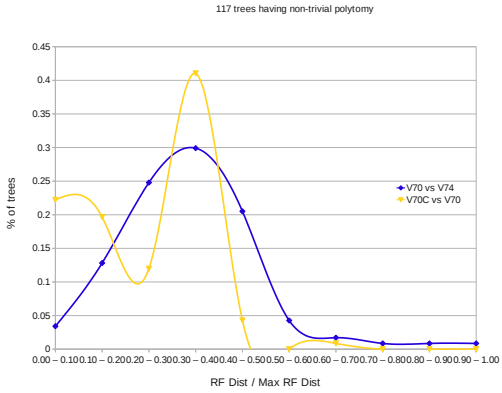

Fig. 5. Normalized RF-distance between corrected trees (yellow curve) and Rel. 74 trees (blue curve) and original Rel. 70 trees.

removes NADs whilst respecting as much as possible the given tree topology. Finally, we considered the reconciliation mutation cost as another evaluation criterion. Among the 117 trees of Rel.70C, 30 are identical to the corresponding trees in Rel. 74, and 60% have a lower mutation cost, which tend to support our correction compared to the tree in Rel.74. As for the 40% remaining trees, half of them have more NADs than the corresponding tree in Rel.74, which suggests that applying our correction to all NAD, instead of just the highest one, would help to obtain better results.

Finally, we evaluated the effect of NAD correction on the tree likelihood. For this purpose, we selected the 1891 *Ensembl Rel.74* gene trees of the considered monophyletic group containing at least one NAD, and we corrected each NAD individually. The sequences were aligned using ClustalW [14] and the likelihood values were computed with PhyML [19]. For a tree T and a NAD node x, denote by T_x the tree obtained after correcting x. For each T and each x, we computed the log-likelihood ratio $L(x) = logLH(T)/logLH(T_x)$. Among the 4454 NAD nodes found in the considered set of trees, 95.4% of the $L(x)$ ratios were between 0.98 and 1.02. Although the correction algorithm is not expected to outperform the *Ensembl* protocol in terms of likelihood as it ignores sequences, we found that the likelihood of the tree has been improved ($L(x) > 1$) after correction for 43.9% of the NAD

nodes. Moreover, a total of 1180 (62.4 %) trees contained at least one NAD node improving the likelihood.

7 CONCLUSION

The present work is dedicated to the polytomy refinement problem. While the mutation cost of reconciliation has been used previously as an optimization criterion for choosing an appropriate binary tree, here we use an alternative criterion which is the minimization of Non-Apparent Duplications (NADs). The tractability of the MinNADref Problem remains open, as is the problem to select, among all possible solutions, those leading to a minimum reconciliation cost. Although developing a gene tree correction tool is not the purpose of this paper, we show how our algorithm for polytomy refinement can be used in this context, by developing a simple algorithm allowing to correct a single NAD. This algorithm has been applied to trees of a previous *Ensembl* release, and the corrected trees have been compared to the trees of the current *Ensembl* release. A good correlation between the two sets of trees is observed, which tends to support our correction paradigm. While minimizing NADs cannot be a sufficient criterion for gene tree correction, it should rather be seen as one among others, such as statistical [39], syntenic [23] or based on reconciliation with the species tree [6, 24, 32], that can be integrated in a methodological framework for gene tree correction.

ACKNOWLEDGMENTS

N.El-Mabrouk and M. Lafond are supported by "Fonds de recherche du Québec - Nature et technologies" (FRQNT). C. Chauve and N. El-Mabrouk are supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). R.Dondi is supported by the MIUR PRIN 2010-2011 grant "Automi e Linguaggi Formali: Aspetti Matematici e Applicativi", code H41J12000190001.

REFERENCES

[1]O. Akerborg, B. Sennblad, L. Arvestad, and J. Lagergren. Simultaneous bayesian gene tree reconstruction and reconciliation analysis. *Proc. Natl. Acad. Sci. U.S.A.*, 106(14):5714–5719, 2009.

[2]R.G. Beiko and N. Hamilton. Phylogenetic identification of lateral genetic transfer events. *BMC Evol. Biol.*, 6(15), 2006.

[3]A.C. Berglund-Sonnhammer, P. Steffansson, M.J. Betts, and D.A. Liberles. Optimal gene trees from sequences and species trees using a soft interpretation of parsimony. *J. Mol. Evol.*, 63:240-250, 2006.

[4]B. Boussau, G. J Szöllosi, L. Duret, M. Gouy, E. Tannier, and V. Daubin. Genome-scale coestimation of species and gene trees. *Genome Res.*, 23:323-330, 2013.

[5]W.C. Chang and O. Eulenstein. Reconciling gene trees with apparent polytomies. In *COCOON 2006*, volume 4112 of *Lecture Notes in Computer Science*, pages 235–244, 2006.

[6]R. Chaudhary, J.G. Burleigh, and O. Eulenstein. Efficient error correction algorithms for gene tree reconciliation based on duplication, duplication and loss, and deep coalescence. *BMC Bioinformatics*, 13(Supp.10):S11, 2011.

[7]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 *Lecture Notes in Computer Science*, pages 46-58, 2009.

[8]K. Chen, D. Durand, and M. Farach-Colton. Notung: Dating gene duplications using gene family trees. *J. Comp. Biol.*, 7:429–447, 2000.

[9]D. G. Corneil, Y. Perl, and L. K Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 14(4):926-934, 1985.

[10]R.S. Datta, C. Meacham, B. Samad, C. Neyer, and K. Sjölander. Berkeley phog: Phylofacts orthology group prediction web server. *Nucleic Acids Res.*, 37:W84-W89, 2009.

[11]P.S. Dehal and J.L. Boore. A phylogenomic gene cluster resource: the phylogenetically inferred groups (phigs) database. *BMC Bioinformatics*, 7(201), 2006.

[12]A. Doroftei and N. El-Mabrouk. Removing noise from gene trees. In *WABI 2011*, volume 6833 of *Lecture Notes in Bioinformatics*, pages 76-91, 2011.

[13]D. Durand, B.V. Haldórsson, and B. Vernot. A hybrid micro-macroevolutionary approach to gene tree reconstruction. *J. Comput. Biol.*, 13:320–335, 2006.

[14]M.A. Larkin *et al.* Clustalw and clustalx version 2. *Bioinformatics*, 23:2947-2948, 2007.

[15]J. Felsenstein. Evolurionary trees from DNA sequences: a maximum likelihood approach. *J. Mol. Evol.*, 17:368-376, 1981.

[16]P. Flicek and al. Ensembl 2012. *Nucleic Acids Res.*, 40:D84- D90, 2012.

[17]P. Gorecki and O. Eulenstein. Algorithms: simultaneous error-correction and rooting for gene tree reconciliation and the gene duplication problem. *BMC Bioinformatics*, 13(Supp 10):S14, 2011.

[18]P. Gorecki and O. Eulenstein. A linear-time algorithm for error-corrected reconciliation of unrooted gene trees. In *ISBRA 2011*, volume 6674 of *Lecture Notes in Bioinformatics*, pages 148-159, 2011.

[19]S. Guidon and O. Gascuel. A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst. Biol.*, 52(5):696- 704, 2003.

[20]M.W. Hahn, M.V. Han, and S.-G. Han. Gene family evolution across 12 *drosophilia* genomes. *PLoS Genetics*, 3:e197, 2007.

[21]M. Hellmuth, M. Hernandez-Rosales, K. Huber, V. Moulton, P. Stadler, and N. Wieseke. Orthology relations, symbolic ultrametrics, and cographs. *J. Math. Biol.*, 66(1–2):399–420, 2013.

[22]J. Huerta-Cepas J, S. Capella-Gutierrez S, L.P. Pryszcz LP, I. Denisov, D. Kormes, M. Marcet-Houben, and T. Gabald'on. Phylomedb v3.0: an expanding repository of genome-wide collections of trees, alignments and phylogeny-based orthology and paralogy predictions. *Nucleic Acids Res.*, 39:D556-D560, 2011.

[23]M. Lafond, M. Semeria, K.M. Swenson, E. Tannier, and N. El-Mabrouk. Gene tree correction guided by orthology. *BMC Bioinformatics*, 14 (supp 15)(S5), 2013.

[24]M. Lafond, K. Swenson, and N. El-Mabrouk. *Models and algorithms for genome evolution*, chapter Error detection and correction of gene trees. Springer, 2013.

[25]M. Lafond, K.M. Swenson, and N. El-Mabrouk. An optimal reconciliation algorithm for gene trees with polytomies. In *WABI 2012*, volume 7534 of *Lecture Notes in Computer Science*, pages 106-122, 2012.

[26]H. Mi, A. Muruganujan, and P.D. Thomas. Panther in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees. *Nucleic Acids Res.*, 41:D377-D386, 2012.

[27]M. D. Rasmussen and M. Kellis. A bayesian approach for fast and accurate gene tree reconstruction. *Mol. Biol. Evol.*, 28(1):273–290, 2011.

[28]F. Ronquist and J.P. Huelsenbeck. MrBayes3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19:1572- 1574, 2003.

[29]N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406-425, 1987.

[30]F. Schreiber, M. Patricio, M. Muffato, M. Pignatelli, and A. Bateman. Treefam v9: a new website, more species and orthology-on-the-fly. *Nucleic Acids Res*, 42:D922-D925, 2013.

[31]C. Scornavacca, V. Berry, and V. Ranwez. From gene trees to species trees through a supertree approach. In *LATA 2009*, volume 5457 of *Lecture Notes in Computer Science*, pages 702-714, 2009.

[32]K. M. Swenson, A. Doroftei, and N. El-Mabrouk. Gene tree correction for reconciliation and species tree inference. *Algorithms Mol. Biol.*, 7(31), 2012.

[33]G.J. Szöllosi, W. Rosikiewicz, B. Bousseau, E. Tannier, and V. Daubin. Efficient exploration of the space of reconciled gene trees. *Syst. Biol.*, 62(6):901-912, 2013.

[34]T.H. Nguyen TH, V. Ranwez, S. Pointet S, A.M. Chifolleau, J.P. Doyon, and V. Berry. Reconciliation and local gene tree rearrangement can be of mutual profit. *Algorithms Mol. Biol.*, 8(8):12, 2013.

[35]P.D. Thomas. GIGA: a simple, efficient algorithm for gene tree inference in the genomic age. *BMC Bioinformatics*, 11:312, 2010.

[36]B. Vernot, M. Stolzer, A. Goldman, and D. Durand. Reconciliation with non-binary species trees. *J. Comput. Biol.*, 15:981–1006, 2008.

[37]A.J. Vilella, J. Severin, A. Ureta-Vidal, L. Heng, and E. Birney. EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research*, 19(2):327–335, 2009.

[38]I. Wapinski, A. Pfeffer, N. Friedman, and A. Regev. Automatic genome-wide reconstruction of phylogenetic gene trees. *Bioinformatics*, 23(13):i549-i558, 2007.

[39]Y-C. Wu, M.D. Rasmussen, M.S. Bansal, and M. Kellis. Treefix: Statistically informed gene tree error correction using species trees. *Syst. Biol.*, 62(1):110-120, 2012.

[40]Y. Zheng, T. Wu, and L. Zhang. Reconciliation of gene and species trees with polytomies. eprint arXiv:1201.3995, 2012.

8 APPENDIX

Proof of Theorem 1: Let H be a minimum refinement of G. Then H is a binary arrangement of the n subtrees H_i for $1 \leq i \leq n$, where each H_i is a refinement of G_i. Suppose that for a given i, H_i is not a minimum refinement of G_i. Then replacing H_i by $H_{min}(G_i, S)$ lowers the number of NAD nodes in the subtree rooted at x_i, but has no effect on the type of nodes outside this subtree. It follows that a minimum refinement of G is a minimum refinement of G'. □

Proof of Lemma 1: Sufficiency is clearly true. As for necessity, let J be a join sequence with d NADs, and $J_{i-1} = \{G_1, G_3\}$ be the first join of type AD or NAD in J followed by a join $J_i = \{G_2, G_4\}$ of type S. We show that we can swap the join types of J_{i-1} and J_i. In other words, we create a new sequence J' where $J'_k = J_k$ for $k < i-1$, J'_{i-1} is of type S, J'_i of the same type as J_{i-1}, and all subsequent joins types are the same as in J. We can then apply this swapping procedure until all S joins are in the beginning of J'.

Let $G_{1,3}$ denotes the subtree created after applying J_{i-1}. If neither G_2 nor G_4 are equal to $G_{1,3}$, then we can safely swap J_{i-1} and J_i since they create two independent subtrees, which does not affect subsequent joins. So suppose w.l.o.g. that $G_4 = G_{1,3}$, and therefore $J_i = \{G_{1,3}, G_2\}$. Since J_i is of type S, then G_1 and G_3 both shared an S edge with G_2 in $\mathcal{F}(J, i-2)$. Let $J'_{i-1} = \{G_1, G_2\}$ be the join of type S, which creates a subtree denoted $G_{1,2}$, and let $J'_i = \{G_{1,2}, G_3\}$. If J_{i-1} is of type AD, then by applying Ruleset 1.1 (replacing T by G_3), it follows that $J'_i = \{G_{1,2}, G_3\}$ is of type AD. Conversaly, if J_{i-1} is of type NAD, then by applying Ruleset 1.2, it follows that J'_i if of type NAD. In other words, J'_i and J_{i-1} are of the same type. Since the subtrees created by applying J_i and J'_i share the same leafset, and that join types are defined by the leaves, all subsequent joins in J can be applied in J'. □

Proof of Theorem 2: We first prove a simple claim.

Claim i. Consider $H \in \mathcal{H}(G)$ with exactly d NADs. Then there exists a set W of vertex-disjoint cliques in R_S such that $R_{AD} \cup W$ has $d+1$ connected components.

Recall that d is the minimum number of NADs attainable. Let J be a join sequence realizing H. By Lemma 1, we can assume that all speciations precede all duplications. Let k be the number of maximum speciation subtrees of H. As stated before (statement just preceding the theorem), the set of leaves of each speciation subtree of H forms a clique in R_S. Moreover, as the k maximum speciation subtrees of H are disjoint (do not share a common node), the corresponding set W of cliques in R_S are vertex-disjoint. Let R^J be the graph obtained after applying all the speciations in J. If R^J has more than $d+1$ AD-components, then J cannot lead to a solution with d NADs. Indeed, we have exhausted all speciations used by J, which implies only NAD edges are used to join AD-components together - requiring more than d of them if there are more than $d+1$ AD-components. On the other hand, if R^J has less than $d+1$ AD-components, then there exists a solution with less than d NADs, contradicting the fact that d is the minimum number of NADs of a solution to the MinNADref Problem . It follows that R^J has exacly $d+1$ AD-components, which completes the proof of the claim.

"⇐" Let $d+1$ be the minimum number of connected components formed by the edges of R_{AD} augmented with the edges of a set W of vertex-disjoint cliques of R_S. Then all nodes of each connected component can be joined under a single subtree by applying joins of type AD and S. These $d+1$ subtrees can then be joined with exactly d NADs, yielding a refinement H with exactly d NADs. Then H is a solution to the MinNADref Problem as otherwise there is a refinement H^* with $d^* < d$ NADs, leading (Claim i) to a W^* such that $R_{AD} \cup W$ has $d* + 1 < d+1$ connected components, which contradicts the fact that $d+1$ is the minimum number of connected components formed by the edges of R_{AD} augmented with the edges of a set of vertex-disjoint cliques in R_S.

"⇒" Let H be a solution to the MinNADref Problem with d NADs. Then, by Claim i, there is a set W of vertex-disjoint cliques in R_S such that $R_{AD} \cup W$ has $d+1$ connected components. Now suppose that the minimum number of connected components induced by a set of vertex-disjoint cliques is $d^* + 1 < d+1$. By the sufficient proof above, it follows that d^* is the minimum number of NAD nodes of a resolution, contradicting the minimality of d.

□

Proof of Theorem 3: In this proof, for two vertices G_i, G_j of R, we denote $s_{i,j} = lca_{\mathcal{S}}(s(G_i), s(G_j))$.

"⇒"Suppose R_S is not $\{P_4, 2K_2\}$-free. Let G_1, G_2, G_3, G_4 be four vertices, with $\{G_1, G_2\}$ and $\{G_3, G_4\}$ being two edges in R_S, that form an induced P_4 or $2K_2$. This implies that at least one of the two edges $\{G_1, G_3\}$ and $\{G_2, G_4\}$ should be absent from R_S.

Assume w.l.o.g. that $\{G_1, G_3\}$ is the missing edge. The edge between G_1, G_2 means that $s(G_1)$ and $s(G_2)$ are unrelated in $\mathcal{S}$. Suppose w.l.o.g. that $s(G_1)$ is in the left subtree of $s_{1,2}$, and $s(G_2)$ in the right subtree. The missing edge between G_1 and G_3 then implies that $s(G_1)$ and $s(G_3)$ are related, in other words that $s(G_3)$ is on the left subtree of $s_{1,2}$ or is an ancestor of $s_{1,2}$.

Suppose that $\{G_2, G_3\}$ is not an edge in R_S. Then, by a similar reasoning as before, it follows that $s(G_3)$ is either on the right subtree of $s_{1,2}$ or is an ancestor of $s_{1,2}$. It follows from the two arguments that $s(G_3)$ is an ancestor of $s_{1,2}$. Now, the edge between G_3, G_4 means that $s(G_3)$ and $s(G_4)$ are unrelated in $\mathcal{S}$, and thus $s(G_4)$ is unrelated to $s(G_1)$ and $s(G_2)$ as well. But in this case $\{G_1, G_4\}$ and $\{G_2, G_4\}$ should be edges in R_S, and thus G_1, G_2, G_3, G_4 can neither form a $2K_2$ nor a P_4 struture.

Now suppose that $\{G_2, G_3\}$ is an edge in R_S. Then G_1, G_2, G_3, G_4 cannot form a $2K_2$ structure, and for the four edges to form a P_4 structure, $\{G_2, G_4\}$ should not be an edge in R_S. By taking the same proof as above (switching G_3 and G_4), we have that $s(G_4)$ is an ancestor of $s_{1,2}$. Now, the edge $\{G_3, G_4\}$ means that $s(G_3)$ and $s(G_4)$ are unrelated in $\mathcal{S}$, and thus $s(G_3)$ is unrelated to $s(G_1)$ and $s(G_2)$ as well. But in this case $\{G_1, G_3\}$ should be an edge in R_S, and thus G_1, G_2, G_3, G_4 can neither form a $2K_2$ nor a P_4 structure.

In both cases, the evolutionary constraints on S edges in a valid graph lead to a contradiction with the assumption that R_S contains a P_4 or a $2K_2$. So if R_S contains a P_4 or a $2K_2$, R is not valid.

"⇐"The other direction of the proof uses the notion of cotrees, related to P_4-free graphs. A *cotree* T is a rooted tree in which the internal nodes are labelled 0 or 1 and have at least two children. We say T is an *alternating cotree* if the labels of any root-leaf path alternate between 0 and 1. A cotree T *represents* a given graph H if $l(T) = V(H)$, and $xy \in E(H)$ if and only if $lca_T(x, y)$ is

labelled by 1. It is well-known that for any P_4-free graph H, there is a unique alternating cotree that represents H [9]. Let T denote the unique alternating cotree representing R_S. Note that $l(T) = V(R)$. The fact that R_S is also $2K_2$-free implies the following: any internal node x of T labelled 0 has at most one non-leaf child. If not, then x has two children x_1, x_2 labelled 1, which implies we can find $a, b \in l(x_1)$, $c, d \in l(x_2)$ such that $lca_T(a, b) = x_1$ and $lca_T(c, d) = x_2$. Since the lca of each pair $(a, c), (a, d), (b, c)$ and (b, d) is x, labelled 0, then a, b, c, d induces a $2K_2$ in which the edges are ab and cd.

Now, given R and T, we can construct a forest $\mathcal{F}$ and a species tree $\mathcal{S}$ that make R valid. Note that since T is constructed from R_S, for two leaves x, y of T, $lca_T(x, y)$ is labelled 1 if $jt(x, y) = S$, and labelled 0 if $jt(x, y) \in \{AD, NAD\}$. The reader may refer to Figure 6 for an example of the whole construction for a given R. The species tree is found by a transformation of T. Note that T is not necessarily binary, but the reader can verify that any binary refinement of the constructed species tree will result in a valid instance. Let $x \in l(T)$. We transform x into a bigger tree $\beta(x) = (\beta_{AD}(x), (x^*, \beta_{NAD}(x)))$, where x^* is a single leaf and $\beta_{AD}(x)$ and $\beta_{NAD}(x)$ are two copies of T. For some $y \in l(T)$, denote by $\beta_{AD}(x, y)$ (resp. $\beta_{NAD}(x, y)$) the unique leaf of $\beta_{AD}(x)$ (resp. $\beta_{NAD}(x)$) that corresponds to y in the copy.

The species tree $\mathcal{S}$ is obtained by replacing each leaf $x \in l(T)$ by $\beta(x)$. The point of $\beta(x)$ is to reserve the $\beta_{AD}(x)$ subtree for the vertices of R that x shares an AD relationship with, and the $\beta_{NAD}(x)$ subtree for the NAD relationship. Hence in $\mathcal{F}$, both trees corresponding to x and y will have a gene mapped to $\beta_{AD}(x, y)$ or to $\beta_{AD}(y, x)$ when $jt(x, y) = AD$. If $jt(x, y) = NAD$, then either y but not x will have a gene mapped to $\beta_{NAD}(x, y)$, or x but not y will have a gene mapped to $\beta_{NAD}(y, x)$

Denote by β_x the root of $\beta(x)$. Now on to the construction of $\mathcal{F}$. Let $x \in l(T)$. Let $\gamma(x)$ be a copy of $\beta(x)$ from which we remove the $\beta_{NAD}(x)$ subtree (hence $\gamma(x)$ is a copy of $(\beta_{AD}(x), x^*)$). The species that each gene of $\gamma(x)$ is mapped to is its corresponding leaf in $\beta(x)$. It follows from this that $s(\gamma(x)) = \beta_x$.

We finally construct $\mathcal{F}$ by adding a subtree for each $x \in l(T)$ as such :

- If the parent of x in T is labelled 1, add $\gamma(x)$ to $\mathcal{F}$.
- If the parent of x in T is labelled 0, start from $\gamma(x)$ and for each leaf y of T such that $lca_T(x, y)$ is the parent of x,
  - if $jt(x, y) = AD$, let $\gamma(x) \leftarrow (y', \gamma(x))$, where y' is a new gene such that $s(y') = \beta_{AD}(y, x)$.
  - if $jt(x, y) = NAD$, let $\gamma(x) \leftarrow (y', \gamma(x))$, where y' is a new gene such that $s(y') = \beta_{NAD}(y, x)$.

  then add the resulting $\gamma(x)$ to $\mathcal{F}$.

Note that from this, for $x \in l(T)$, if the parent of x is labelled 1 then $s(\gamma(x)) = \beta_x$, and if the parent of x is labelled 0, then $s(\gamma(x))$ is the parent of β_x, which is labelled 0. To see this, denote by $p(x)$ the parent of x in T, labelled 0. Observe that $p(x)$ remains unchanged in $\mathcal{S}$. Now, for each $y \in l(t)$ such that $lca_T(x, y) = p(x)$, $\gamma(x)$ has genes mapped to species of $\beta(y)$. Thus $\gamma(x)$ has only genes mapped to species that are descendants of $p(x)$ in $\mathcal{S}$, and thus $s(\gamma(x)) = p(x)$ in $\mathcal{S}$.

In both cases, $s(\gamma(x))$ is a descendant of its lowest ancestor labelled 1, if any. From this, we get that if x, y share an S edge in R, then they are left and right descendants of $lca_T(x, y)$ labelled 1, implying that $s(\gamma(x))$ and $s(\gamma(y))$ are left and right descendants of this same node labelled 1 in $\mathcal{S}$. They are therefore related by speciation as prescribed. Now, suppose that x, y are related by a NAD edge. If $lca_T(x, y)$ is $p(x)$, then $\gamma(x)$ contains y' mapped to $\beta_{NAD}(y, x)$. This forces $\gamma(x)$ and $\gamma(y)$ to be related by duplication, which is of type NAD since by construction $\gamma(x)$ and $\gamma(y)$ contain no gene mapped to the same species. The same argument holds when $lca_T(x, y)$ is $p(y)$. So suppose that $lca_T(x, y)$ is not $p(x)$ nor $p(y)$. Then $lca_T(x, y) = lca_T(p(x), p(y))$ and is labelled 0. But this implies that $lca_T(p(x), p(y))$ has at least two non-leaf children, one containing $p(x)$ and the other containing $p(y)$, contradicting the $2K_2$-free assumption as stated above. We observe that the same applies to vertices x, y of R related by an AD edge, except that they must share a gene mapped to $\beta_{AD}(y, x)$ or $\beta_{AD}(x, y)$, making them related by apparent duplication. We finally note that no other tree of $\mathcal{F}$ has genes mapped to $\beta_{AD}(y, x)$ or $\beta_{AD}(x, y)$, thereby removing the possibility of an unwanted apparent duplication. □

Before being able to prove Theorem 4, we need the following general property on P_4-free graphs.

LEMMA 2. *Let $\{x, y\}$ be an edge of a P_4-free graph $\mathcal{G}$, and let W_x and W_y be two vertex-disjoint cliques of $\mathcal{G}$ respectively containing x and y. Then we can partition the vertices $V(W_x) \cup V(W_y)$ into at most two other cliques, with one containing $\{x, y\}$.*

PROOF. If the set $V(W_x) \cup V(W_y)$ induces a single clique, then we are done. Otherwise, let $Y(x) \subseteq V(W_y)$ denote the set of vertices in W_y that share an edge with x (including y), and let $X(Y) \subseteq V(W_x)$ be the vertices of W_x that share an edge with every vertex of $Y(x)$. The set $V_1 = \{x\} \cup Y(x) \cup X(Y)$ induces a clique containing $\{x, y\}$. Now, let a and b be two vertices sharing an edge of W_x and W_y respectively with x and y, such that $a, b \notin V_1$. If a, b are both in W_x, or both in W_y, then they obviously share an edge. Otherwise, suppose w.l.o.g. that a is in W_x and b in W_y. Because $a \notin X(Y)$, there is some $b_i \in Y(x)$ such that $\{a, b_i\} \notin E(\mathcal{G})$. And because $b \notin Y(x)$, $\{x, b\} \notin E(\mathcal{G})$. But $\{a, x, b_i, b\}$ induces a P_4, unless $ab \in E(G)$. Therefore, every pair of vertices in W_x or W_y but not in V_1 share an edge, forming our second clique.

If $Y(x)$ is empty, we can apply the same argument by symmetry using $X(y)$ and $Y(X)$ if $X(y)$ is not empty. If both $Y(x)$ and $X(y)$ are empty, then let $V_1 = \{x, y\}$ induce the first clique. Let a, b be vertices sharing edges with x and y respectively. Now, a, x, y, b induce a P_4 unless $\{a, b\} \in E(G)$, and thus second clique is formed by the vertices sharing an edge with x, y. □

Let c_{AD} be the number of AD-components of R before applying any join. Suppose we have a join sequence with s useful speciations, all applied before any AD or NAD join. It follows that applying a useful speciation connects two AD-components together, and applying s of them results in a graph with $AD_{AD} - s$ AD-components, from which we can obtain a tree with $d = AD_{AD} - s - 1$ NADs. It is then clear that there exists a solution with d NADs iff there exists a join sequence with $s = AD_{AD} - d - 1$ useful speciations. Hence we can minimize the number of NADs by maximizing the number of useful speciations we can make. Our

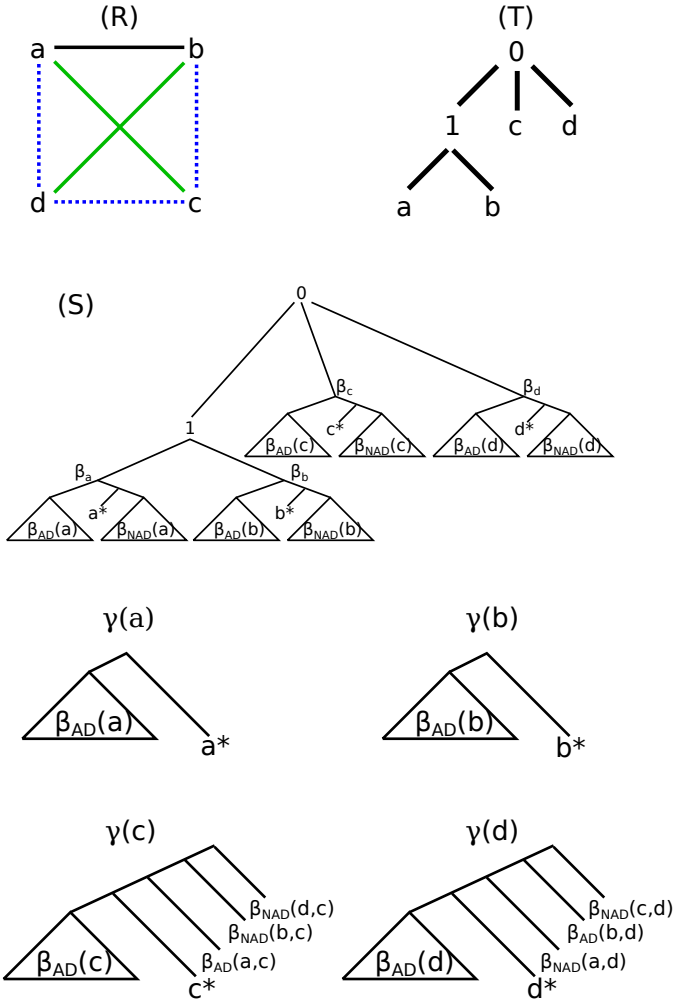

Fig. 6. A construction of $\mathcal{F}$ and $\mathcal{S}$ given R. The solid black edge of R is an S-edge, the green edges are AD-edges and the blue dotted edges are NAD-edges. T is the cotree corresponding to R_S, where $V(R) = l(T)$. The species tree $\mathcal{S}$ is build by replacing each leaf x of T by $\beta(x)$. For instance here, $\beta_{AD}(a)$ contains the leaves $\{\beta_{AD}(a,a), \beta_{AD}(a,b), \beta_{AD}(a,c), \beta_{AD}(a,d)\}$. The gene tree forest $\mathcal{F}$ consists of $\{\gamma(a), \gamma(b), \gamma(c), \gamma(d)\}$, in which we labelled the genes to their corresponding species, built from the construction given in the proof of Theorem 3.

heuristic consists in constructing a join sequence by always picking the lowest available speciation, which is shown to find at least half the number of useful speciations as the optimal solution. We first need the following property.

LEMMA 3. *Let $\{x, y\}$ be an S edge of R corresponding to a lowest available speciation, and let d be number of NADs of a solution to the* MinNADref *problem. Then there exists a solution which makes the $\{x, y\}$ speciation that has at most $d + 1$ NADs.*

PROOF. Let W be a set of vertex-disjoint cliques of R_S, and let R_W be the R graph restricted to the set of edges $W \cup R_{AD}$ (W must exist by Theorem 3). R_W has $d + 1$ connected components. Let W_x (resp. W_y) be the clique of W that contains x (resp. y). If $W_x = W_y$, then we are done. Otherwise, by Lemma 2, we can partition the vertices of W_x and W_y into two other cliques, namely W_1 containing the xy edge and the other clique W_2. Let $W' = W \setminus \{W_x, W_y\} \cup \{W_1, W_2\}$. Now, W' is another set of vertex-disjoint cliques. Denote by $R_{W'}$ the graph R restricted to $W' \cup R_{AD}$. Denote by Z_x, Z_y the vertices in $V(R) \setminus \{W_x, W_y\}$ in the same R_W component as x and y respectively. Similarly, let Z_1, Z_1 be the vertices in $V(R) \setminus \{W_1, W_2\}$ in the same $R_{W'}$ component as a vertex of W_1 and a vertex of W_2 respectively. We have that $Z_x \cup Z_y = Z_1 \cup Z_2$. If x, y were in two distinct components $W_x \cup Z_x$ and $W_y \cup Z_y$ in R_W, then $R_{W'}$ also has $d+1$ components, as these two components got replaced by $W_1 \cup Z_1$ and $W_2 \cup Z_2$. If x, y were in the same component, at worst $R_{W'}$ has $d+2$ components, having the x, y component replaced by $W_1 \cup Z_1$ and $W_2 \cup Z_2$. □

We are now ready to prove Theorem 4:

Proof of Theorem 4: Let $d = AD_{AD} - s - 1$ be the minimum number of NADs in an optimal solution, and let xy be the lowest useful speciation available in R. Note that $s = AD_{AD} - d - 1$. By Lemma 3, there exists a solution with $d + 1$ NADs that contains the xy speciation. Let R' be the graph obtained after applying the $\{x, y\}$ join, thus contracting x and y and applying Ruleset 1. Since xy is the lowest speciation, any common neighbor of x and y in R_S is a neighbor of the xy vertex in R'_S. Therefore, R'_S has $AD_{AD} - 1$ AD-components and admits an optimal solution with at most d NADs. Hence, the number of useful speciations we can make given R' is at least $s' = AD_{AD} - 1 - d - 1 = s - 2$. It then follows that after applying the first k lowest speciations, we have a solution with at least $s - 2k$ more useful speciations, which implies that k can be at least as big as $s/2$ if s is even. If s is odd, k can be as high as $(s - 1)/2$, and there is at least one useful speciation available, hence the lower bound of $\lceil s/2 \rceil$.

□

Proof of Theorem 5: First, we can notice that by including the bridges into M, we ensure that all other added edges are useful speciation edges.

Now, we prove the maximality of the useful matching by induction on $|X \cup Y|$. Given $P = (X, Y, AD_X, AD_Y, B)$, denote by M_P the solution returned by Algorithm 2, and by OPT_P a useful matching of maximum size over instance P.

If $|X \cup Y| = 1$, then the theorem trivially holds, since each useful matching of P contains no edge. Assume the theorem holds for $|X \cup Y| = k$, we show that it holds for $|X \cup Y| = k + 1$.

Let $\alpha \in X \cup Y$ be the last vertex added to D by Algorithm 2, and assume w.l.o.g that $\alpha \in X$. Write $X' = X \setminus \{\alpha\}$, and P' the instance obtained from P by removing α. By induction, since $|X' \cup Y| = k$, $|M_{P'}| = |OPT_{P'}|$. Moreover, by construction, $M_{P'}$ is exactly M_P minus the edge of M_P incident to α, if any.

Assume that α is incident to an edge of M_P. It holds that $|M_P| = |M_{P'}| + 1 = |OPT_{P'}| + 1$. On the other hand, remove from OPT_P the edge incident to α, if any. Then the edges left in OPT_P form a useful matching of P', and thus $|OPT_{P'}| \geq |OPT_P| - 1$. As it has been shown that $|M_P| = |OPT_{P'}| + 1$, it follows that $|M_P| \geq |OPT_P|$, and thus M_P is a useful matching of P of maximum size.

Now, assume that α is not incident to an edge of M_P. Denote by $c(\alpha)$ the connected component of $\mathcal{G}_{P,M_P}$ that contains α.

Claim i. Each vertex β in $Y \setminus c(\alpha)$ is incident to an edge in M_P.

If the claim was wrong, the algorithm would have added an edge between α and β. If, in addition, each vertex of $Y \cap c(\alpha)$ is incident

to an edge of M_P, then each vertex of Y is incident to an edge of M_P, implying that M_P is of maximum size, which completes the proof. Hence assume that there exists at least one vertex β of $Y \cap c(\alpha)$ such that β is not incident to any edge of M_P.

Claim ii. Each vertex γ in $X \setminus c(\alpha)$ must be incident to an edge of M_P(*statement ii*).

Again, the proof is immediate: if the claim was wrong, the algorithm would have defined an edge from β to γ.

Now, consider the set $AD_{\setminus\alpha} = AD_{X\setminus c(\alpha)} \cup AD_{Y\setminus c(\alpha)}$ of AD-components on the sets of vertices $(X \setminus c(\alpha)) \cup (Y \setminus c(\alpha))$. By definition of useful speciation edges, the graph defined by the vertex set $AD_{\setminus\alpha}$ and the edge set containing one edge for each pair $(AD_{X_i} \in AD_{X\setminus c(\alpha)}, AD_{Y_j} \in AD_{Y\setminus c(\alpha)})$ of linked components has no cycles, and thus at least one vertex (AD-component) of degree less than 2. Each such AD-component reduces to a single vertex as otherwise there would be a vertex of this AD-component not incident to any edge of M_P, which is in contradiction with Claim ii. Hence, as α is the last vertex added to D and the algorithm proceeds in decreasing order of AD-component cardinality, the AD-component containing α in X should be of cardinality one, meaning that x is an isolated vertex. Hence $Y \cap c(\alpha) = \emptyset$, and with Claim i it follows that each vertex of Y is adjacent to an edge of M_P, and thus M_P has maximum size. □

LEMMA 4. *Let $P = (X, Y, AD_X, AD_Y, B)$ and $P' = (X', Y', AD_{X'}, AD_{Y'}, B')$ be two instances such that $|X'| = |X|, |Y'| = |Y|, |AD_{X'}| = |AD_X|, |AD_{Y'}| = |AD_Y|$ and $|B| = |B'|$. Then P and P' admit maximum useful matchings of the same size.*

Proof of Lemma 4: Consider two maximum useful matchings M, M' of P, P' respectively and the induced graphs $\mathcal{G}_{P,M}, \mathcal{G}_{P',M'}$. Assume w.l.o.g. that $|M| > |M'|$.

• *Claim (i)*: Since $|X'| = |X|$, $|Y'| = |Y|$ and $|AD_{X'}| = |AD_X|$, it follows that $\mathcal{G}_{P,M}$ contains strictly less connected components than $\mathcal{G}_{P',M'}$.

• *Claim (ii)* Since $|M| > |M'|$, it follows that there exists a node x of X' and a node y of Y' that are not incident to an edge of M'. Then one of the two following cases hold.

Case 1.: x and y belong to different components of $\mathcal{G}_{P',M'}$. Then it holds that M' is not a maximum useful matching, since we can add edge $\{x, y\}$ to M', thus contradicting the assumption that M' is a maximum useful matching of P'.

Case 2.: x and y belong to the same connected component $c(x)$ of $\mathcal{G}_{P',M'}$. We show that we can compute a useful matching M^* of P', such that$|M^*(P')| > |M(P')|$. First, we show that there exist two nodes $x_1 \in X'$ and $y_1 \in Y'$ that belong to a connected component of $\mathcal{G}_{P',M'}$ different from $c(x)$ such that $\{x_1, y_1\}$ is an edge of M'. Notice that if x_1 and y_1 do not exist, then one of the following two cases holds: (2.1) There exists a single connected component in $\mathcal{G}_{P',M'}$, but this violates *Claim (i)*; (2) Each connected component of $\mathcal{G}_{P',M'}$ different from $c(x)$ contains only bridges, which implies that there exist two nodes of $\mathcal{G}_{P',M'}$ (one of x, y and a node that belongs to $(AD_{X'} \cup AD_{Y'}) \setminus c(x))$ not incident to an edge of M' and belonging to different components of $\mathcal{G}_{P',M'}$. But then we fall in *Case 1.* and M' is not a maximum useful matching of P'. Thus nodes x_1 and y_1 exist, so we can compute a useful matching M^* of P' starting from M' as follows: remove $\{x_1, y_1\}$ from M' and add edges $\{x, y_1\}$, $\{x_1, y\}$ to M^*. It follows that M^* is a useful matching for P' with $|M^*| > |M'|$, contradicting the assumption that M' is a maximum useful matching of P'. □

LEMMA 5. *Let $P = (X, Y, AD_X, AD_Y, B = \emptyset)$ and $P' = (X', Y, AD'_X, AD_Y, B' = \emptyset)$ be two instances such that $|X'| - |X| = |AD'_X| - |AD_X|$ with $|X'| \geq |X|$. If P' admits a useful matching M', then P admits a useful matching M such that $|M| \geq |M'| - (|X'| - |X|)$.*

Proof of Lemma 5: Let x_1, x_2 be two nodes of X' in two distinct components of AD'_X. If we join the trees corresponding to x_1 and x_2, leading to a single node $x_{1,2}$, we create a new instance $P^* = (X^*, Y, AD^*_X, AD_Y, \emptyset)$, in which $|X^*| = |X'| - 1$ and $|AD^*_X| = |AD'_X| - 1$. If x_1 and x_2 are incident to edges in M', say $\{x_1, y\}$ and $\{x_2, z\}$, then $M^* = M' \setminus \{\{x_1, y\}, \{x_2, z\}\} \cup \{x_{1,2}, z\}$ is a useful matching for P^*. Otherwise, if x_1 or x_2 is not incident to an edge of M', then construct a matching M^* of P^* from M' by removing the edge incident to x_1 or x_2, if any. In all cases, $|M^*| \geq |M'| - 1$. By applying such join operation $|X'| - |X|$ times, we obtain an instance P^* with $|X|$ nodes and $|AD_X|$ components and a useful matching M^* verifying $|M^*| \geq |M'| - (|X'| - |X|)$. By Lemma 4, it follows that P admits a useful matching M of the same size, which concludes the proof. □

Proof of Theorem 6: Let $\mathcal{F}$ be the input forest of Algorithm 1. Let $\mathcal{F}(s)$ be the subset of $\mathcal{F}$ containing the trees G such that $s(G)$ is s or one of its descendants. Let n_s be the total number of useful speciations performed on trees of $\mathcal{F}(s)$ at step s of the algorithm, i.e., after considering node s. We show by induction on the height of s that n_s is the maximum number of useful speciations that can be chosen on the trees of $\mathcal{F}(s)$, which proves the theorem as s can be the root of $\mathcal{S}$. This is trivially true if s is a leaf. So let s be an internal node of $\mathcal{S}$ with children x and y. Let $P = (X, Y, AD_X, AD_Y, B = \emptyset)$ be the instance corresponding to s. Let $|M_P|$ be the number of useful speciations performed by the algorithm for P. Then $n_s = |M_P| + n_x + n_y$.

Suppose we can make another choice of n'_x and n'_y useful speciations on $\mathcal{F}(x)$ and $\mathcal{F}(y)$ respectively, yielding a different instance $P' = (X', Y', AD'_X, AD'_Y, B' = \emptyset)$ for s. Suppose also that P' admits $|M_{P'}|$ useful speciations such that $n'_s = |M_{P'}| + n'_x + n'_y > |M_P| + n_x + n_y = n_s$. Note that by induction, $n_x \geq n'_x$ and $n_y \geq n'_y$, and thus we should have $|M_{P'}| > |M_P|$. Any of the n'_x speciations has the effect of merging two nodes potentially in X, and merging two components potentially in AD_X. Now $|X| = |\mathcal{F}(x)| - n_x$. If $|AD_R|$ is the number of AD-components of R before any speciation, then $|AD_X| = |AD_R| - n_x$. Similarly $|X'| = |\mathcal{F}(x)| - n'_x$ and $|AD'_X| = |AD_R| - n'_x$. This leads to $n_x - n'_x = |X'| - |X| = |AD'_X| - |AD_X|$. In the same manner, $n_y - n'_y = |Y'| - |Y| = |AD'_Y| - |AD_Y|$. From this, $n'_s > n_s \Rightarrow n'_s - n_s > n_x - n'_x + n_y - n'_y = |X'| - |X| + |Y'| - |Y|$. But we cal also deduce from Lemma 5 that $n'_s - n_s \leq |X'| - |X| + |Y'| - |Y|$: a contradiction. □

Proof of Corollary 1: A bridge is created between two AD-components AD_X, AD_Y if and only if there exist two vertices $x \in AD_X$ and $y \in AD_Y$ such that $\{x, y\}$ is an S edge in R and x and y belong to the same AD-component in R. It follows that for a pair

$(\mathcal{F}, \mathcal{S})$ leading to a graph R where AD-components are free from S edges, we are guaranteed that for every node s of S the instance P corresponding to s has no bridges. It follows from Theorem 6 that Algorithm 1 finds a maximum set M of useful speciations, i.e., a set of useful speciations leading to the minimum number ad of AD-components, and to a refinement H with $ad - 1$ NADs. Suppose H is not optimal, i.e., there is an H' with $ad' < ad - 1$ NADs. By Lemma 1, we can assume that the join sequences J' leading to H' has all M' joins of type S first, followed by AD and NAD joins. As M is a maximum set of useful speciations, we have $|M'| \leq M$. If $M' < M$, then after applying the M' speciations, the graph is left with $ad' > ad - 1$ AD-components, requiring more than $ad - 1$ NADs, contradicting the hypothesis. Therefore H is a solution to theMinNADref problem. □