Representing a set of reconciliations in a compact way

Celine Scornavacca, Wojciech Paprotny, Vincent Berry, Vincent Ranwez

▶ **To cite this version:**

Celine Scornavacca, Wojciech Paprotny, Vincent Berry, Vincent Ranwez. Representing a set of reconciliations in a compact way. Journal of Bioinformatics and Computational Biology, 2013, 11 (2), pp.1250025. 10.1142/S0219720012500254 . lirmm-00818801

HAL Id: lirmm-00818801

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00818801

Submitted on 22 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Journal of Bioinformatics and Computational Biology
© Imperial College Press

Representing a set of reconciliations in a compact way

CELINE SCORNAVACCA
ISEM, CNRS – Université Montpellier II, Place Eugène Bataillon
34095 Montpellier, France
celine.scornavacca@univ-montp2.fr

WOJCIECH PAPROTNY
Center for Bioinformatics (ZBIT), Tübingen University
Sand 14, 72076 Tübingen, Germany

VINCENT BERRY
LIRMM, CNRS – Université Montpellier II, 95 rue de la Galéra
34392 Montpellier, France

VINCENT RANWEZ
Montpellier SupAgro, UMR AGAP, 2, Place P. Viala
34060 Montpellier, France

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Comparative genomic studies are often conducted by reconciliation analyses comparing gene and species trees. One of the issues with reconciliation approaches is that an exponential number of optimal scenarios is possible. The resulting complexity is masked by the fact that a majority of reconciliation software pick up a random optimal solution that is returned to the end-user. However, the alternative solutions should not be ignored since they tell different stories that parsimony considers as viable as the output solution. In this paper we describe a polynomial space and time algorithm to build a minimum reconciliation graph – a graph that summarizes the set of all most parsimonious reconciliations. Amongst numerous applications, it is shown how this graph allows counting the number of non-equivalent most parsimonious reconciliations.

Keywords: Phylogenetics and Reconciliation and Graph representation

1. Introduction

Comparative genomic studies increase in accuracy due to the ever-growing number of fully sequenced genomes. In particular, complete annotated genomes are of importance for inferring past gene gains and losses in different species[5]. These studies are most often conducted by *reconciliation* analyses comparing gene and species trees (for a review see Ref. 9). A reconciliation method infers past events in the

species' history such as gene duplications, losses and transfers, by comparing the discrepancies between the topologies of the gene and species trees. Among other applications, this allows the inference of orthology relationships and the estimation of the content and evolution of gene repertoires in different branches of the living realms.[7]

Reconciliation methods have been designed according to both parsimonious (among others [14,11,10,4,12]) and probabilistic approaches.[3,1] One advantage of the latter approach is the integrated estimation of the event rates, while most parsimony methods require event costs (depending from event rates) to be given as input. However, the incomparable speed of parsimony makes it the tool of choice to analyze the dozens of thousands of gene families that flood genomic databanks. Available parsimony methods mainly differ in the set of evolutionary events they consider (duplication, loss, transfer, incomplete lineage sorting, etc) and in the efficiency of their algorithmic engine. Yet, they share the same principle: given a specific cost for each event type, they compute an optimal reconciliation, that is one that minimizes the sum of costs of all predicted events.

One of the issues with reconciliation approaches is that an exponential number of optimal scenarios is possible.[18] Indeed, we observed gene families on merely a hundred species where one million optimal scenarios are possible, despite the use of reasonable costs for the considered events.[16] To explain the discrepancy between the considered gene tree and species tree, alternative scenarios resort to locally different sets of events, even though they have the overall same optimal cost.

The fact that a majority of software pick up a random optimal solution which is returned to the end-user should not obliterate the informativeness of alternative reconciliations that encompass partly different evolutionary stories. For instance, alternative solutions could be used to assert the reliability of the whole reconciliation (depending on the number of optimal scenarios) and the reliability of each elementary event (using its frequency among optimal scenarios). Few reconciliation software, namely, CoRe-PA,[15] Mowgli,[10] and the new version of Jane,[6] can list all most parsimonious scenarios. Yet, since there can be an exponential number of those optimal scenarios, above mentioned support metrics cannot be obtained by a brute force approach, sequentially considering all optimal scenarios returned by those software packages.

In this paper, we introduce the concept of reconciliation graph, which represents a set of scenarios by factorizing their common parts. We show that, when the set of scenarios to be represented is the set of – potentially exponential – most parsimonious scenarios, this graph has polynomial size and can be computed in polynomial time. For a given set of reconciliations, several reconciliation graphs usually exist. We show that our algorithm outputs a reconciliation graph of minimum size. Our algorithm is thus the first one that can generate a compact representation of the – potentially exponential – set of parsimonious reconciliations in polynomial time.

This conceptual tool has numerous applications. Besides allowing to visually depict alternative solutions, one by one or collectively – so that biologists can then

use their expertise to select the most meaningful ones – it may be used to efficiently count the number of optimal scenarios in a very simple way, as we demonstrate at the end of this paper. Other applications are identifying bottleneck events, present in all optimal reconciliations, computing confidence values for events depending on the number of their alternatives, or investigating whether the gene content of ancestral species for an examined family is uncertain or well established according to parsimony. Most likely the results presented here can be applied on maximum likelihood reconciliations based on models that rely on dynamic programming such as the one presented in Ref. 17. A program computing reconciliation graphs is available at `http://celinescornavacca.wordpress.com/software/`. It outputs graphs in SIF format, which can be properly visualized by different graph visualization software such as Cytoscape.

2. Basic notations

Let $T = (V(T), E(T))$ be a rooted tree where only leaf nodes are labeled. We denote by $r(T)$, $L(T)$, and $\mathcal{L}(T)$ respectively the root node, the set of the leaf nodes, and the set of taxa labeling the leaves of T. If u is a leaf node, we denote by $s(u)$ the species that labels it and if u is not a root we denote by $p(u)$ its parent. Note that in a rooted tree, all edges are directed away from the root. In this paper we consider only binary trees, i.e. trees with nodes whose outdegree is at most two.

A *species tree* S is a rooted binary tree such that each element of $\mathcal{L}(S)$ represents an extant species and there is a bijection between $L(S)$ and $\mathcal{L}(S)$. A *gene tree* G is a rooted tree such that leaf nodes correspond to contemporary genes.

An internal node u of a binary tree T can have one or two children; we respectively denote by $\{u_l\}$ and $\{u_l, u_r\}$ the child set of u. Note that, because T is an unordered tree, u_l and u_r are interchangeable. For any two nodes u and v of T, we call u a (*strict*) ancestor of v, and v a (*strict*) descendant of u, if there exists a directed path from u to v (and $u \neq v$). For any node u of T, T_u denotes the subtree of T rooted at u. The *height* of a node u, denoted by $h(u)$, corresponds to the maximum number of edges along a direct path between u and any of the leaves of the subtree T_u. The *depth* of a node u of a tree T is defined as the number of edges along the direct path between $r(T)$ and u. A node of T is said to be *artificial* when its indegree and outdegree both equal one. To *suppress* an artificial node u of T means first to connect the two nodes adjacent to u by a new edge and then to delete u and its two adjacent edges. A tree T' is said to be a *subdivision* of a tree T if T can be obtained from T' by suppressing all artificial nodes of T'.

A *time function* $\theta_T : V(T) \to \mathbb{R}^+$ for a tree T associates any node of $V(T)$ with a non-negative value while respecting the two following constrains: first, for any two nodes $u, v \in V(T)$, if v is a strict descendant of u, then $\theta_T(v) < \theta_T(u)$. Second, $\forall u \in L(T), \theta_T(u) = 0$, i.e. all extant species are contemporary. T is said to be a *dated tree* when it is associated with a time function θ_T. We can derive a *dated subdivision* T' from a dated tree T with time function θ_T in the following way: first, initialize

T' as T; then, for each edge $(p(x), x)$ of T' and each time $t \in]\theta_{T'}(x), \theta_{T'}(p(x))[$ for which there exists a vertex y of T' such that $\theta_{T'}(y) = t$, add an artificial vertex w along the edge $(p(x), x)$ and set $\theta_{T'}(w) = t$.

2.1. *Reconciliations*

Among others, the authors of Ref. 10 propose a combinatorial model to reconcile a dated species tree S with a gene tree G such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$, considering gene duplication ($\mathbb{D}$), transfer ($\mathbb{T}$), loss ($\mathbb{L}$), and speciation ($\mathbb{S}$) events, along with $\varnothing$ (no event), $\mathbb{C}$, $\mathbb{TL}$ and $\mathbb{SL}$ events. A $\varnothing$ event is a fake event, just indicating that a gene evolving in a branch crosses a time boundary (called a *time slice*); a $\mathbb{C}$ event simply correctly associates each contemporary gene to the corresponding species. The $\mathbb{TL}$ and $\mathbb{SL}$ events result from the fact that losses are always considered in combination with other events: $\mathbb{SL}$ and $\mathbb{TL}$ are each considered as a single *event*, even though they are a combination of two elementary events ($\mathbb{S}$ + $\mathbb{L}$ and $\mathbb{T}$ +$\mathbb{L}$). Other works have referred to such combinations of events, see *e.g.* Ref. 15 where an $\mathbb{SL}$ event corresponds to a *sorting* event. Note that we do not consider here a combination of a duplication and a loss – that would be denoted $\mathbb{DL}$ – since it cannot be contained in a parsimonious reconciliation. The formal definition of a *reconciliation* is as follows:

Definition 1 (adapted from Ref. 10). Consider a gene tree G, a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$, and its subdivision S'. Let α be a function that maps each node u of G onto an ordered sequence of nodes of S', denoted $\alpha(u) = (\alpha_1(u), \alpha_2(u), \ldots, \alpha_\ell(u))$. The function α is said to be a *reconciliation* between G and S' if and only if exactly one of the following events occurs for each couple of nodes u of G and $\alpha_i(u)$ of S' (denoting $\alpha_i(u)$ by x' below):

a) if x' is the last node of $\alpha(u)$, one of the cases below is true:

1. $u \in L(G)$, $x' \in L(S')$ and $s(x') = s(u)$; ($\mathbb{C}$ event)
2. $\{\alpha_1(u_l), \alpha_1(u_r)\} = \{x'_l, x'_r\}$; ($\mathbb{S}$ event)
3. $\alpha_1(u_l) = x'$ and $\alpha_1(u_r) = x'$; ($\mathbb{D}$ event)
4. $\alpha_1(u_l) = x'$, and $\alpha_1(u_r)$ is any node other than x' having height $h(x')$ or $\alpha_1(u_r) = x'$, and $\alpha_1(u_l)$ is any node other than x' having height $h(x')$; ($\mathbb{T}$ event)

b) otherwise, one of the cases below is true:

5. x' is an artificial node and $\alpha_{i+1}(u)$ is its only child; ($\varnothing$ event)
6. x' is not artificial and $\alpha_{i+1}(u) \in \{x'_l, x'_r\}$; ($\mathbb{SL}$ event)
7. $\alpha_{i+1}(u)$ is any node other than x' having height $h(x')$. ($\mathbb{TL}$ event)

Several reconciliations can display very similar event sequences with events occurring at slightly different times in a same branch of the species tree (mainly by shifting the position of $\varnothing$ events in the sequences of α). It will be useful in Sec-

tion 4.2 to have a unique representative for such sets of reconciliations, hence we introduce the notion of *canonical* reconciliation.

Definition 2 (from Ref. 8). Consider a gene tree G, a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$, and its subdivision S'. A reconciliation α between G and S' is said to be *canonical* if and only if:

(1) for each node u of G and index $1 \leq i \leq |\alpha(u)|$, the node $\alpha_i(u)$ satisfies one of the following conditions:
   (a) $\alpha_i(u)$ is a $\mathbb{C}/\mathbb{S}/\varnothing/\mathbb{SL}$ event;
   (b) $\alpha_i(u)$ is a $\mathbb{D}/\mathbb{T}$ event such that at least one of $\alpha_1(u_l)$ and $\alpha_1(u_r)$ is not a $\varnothing$ event;
   (c) $\alpha_i(u)$ is a $\mathbb{TL}$ event such that $\alpha_i(u)$ is a non-artificial node of S' or $\alpha_{i+1}(u)$ is not a $\varnothing$ event.

(2) $\alpha_1(r(G))$ is not a $\varnothing$ event.

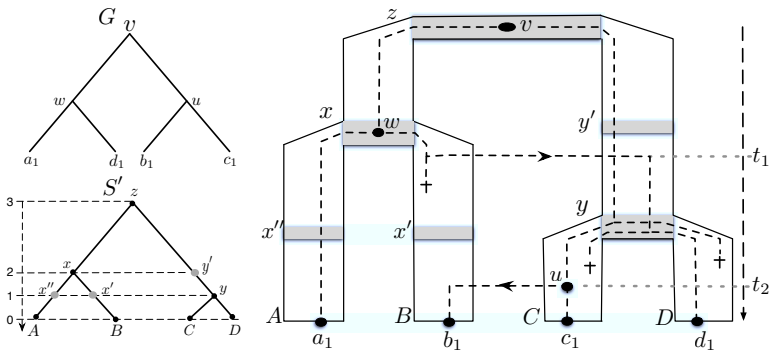

Fig. 1. (a) A gene tree G and a subdivided species tree S' (b) A reconciliation α between G and S', where α is defined as follows: $\alpha(v) = (z)$, $\alpha(u) = (y', y, C)$, $\alpha(w) = (x)$, $\alpha(a_1) = (x'', A)$, $\alpha(b_1) = (B)$, $\alpha(c_1) = (C)$ and $\alpha(d_1) = (x', y, D)$.

An example of reconciliation is given in Figure 1. Note that this reconciliation is canonical.

2.2. *Computing the cost of a most parsimonious reconciliation*

Given a fixed cost for individual $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events, denoted δ, τ and λ respectively, each possible reconciliation is assigned the cost $d\delta + t\tau + l\lambda$, where d, t and l denotes respectively the number of $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events induced by the reconciliation. In this model, the cost of $\mathbb{C}$, $\varnothing$ and $\mathbb{S}$ events is null, though the model could easily accommodate non-null costs. We now formally introduce the MPR problem:

Problem: Most Parsimonious Reconciliation (MPR)

Instance: A dated species tree S, a gene tree G such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$ and costs δ, τ, resp. λ for $\mathbb{D}$, $\mathbb{T}$ resp. $\mathbb{L}$ events.
Goal: A reconciliation of minimal cost between G and the subdivision of S.

An efficient dynamic programming algorithm to find a Most Parsimonious Reconciliation (MPR) has been presented in Algorithm 1 of Ref. 10 to which we refer the reader. Even though we do not recall here the algorithm due to lack of space, it is important in the following of the article as we will build upon it.

3. Depicting a set of reconciliations in a compact way

In this section we introduce the notions of *reconciliation tree* and *reconciliation graph* used respectively to encode a single reconciliation as a tree, and a set of reconciliations as a graph. These representations can be used to represent any kind of reconciliation, no matter whether it has been obtained by a parsimonious or a probabilistic method.

3.1. *Reconciliation tree*

Before detailing how a reconciliation α can be depicted as a rooted tree called *reconciliation tree*, we need to introduce the notion of *event-mapping graph*.

Definition 3. Given a gene tree G and the subdivision S' of a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$, an *event-mapping graph* R for G and S' is any acyclic digraph that contains two kinds of nodes – *event* nodes, denoted by $V_e(R)$, and *mapping* nodes, denoted by $V_m(R)$ – and whose edges never link two nodes of the same kind (bipartite graph). Each event node y is associated to a value $e(y) \in \{\mathbb{C}, \mathbb{S}, \mathbb{D}, \mathbb{T}, \varnothing, \mathbb{SL}, \mathbb{TL}\}$ (see Definition 1), whereas each mapping node z is associated to a pair $m(z) = (u, x)$, where $u \in V(G)$ and $x \in V(S')$. The two elements of $m(\cdot)$ are respectively denoted $m_G(\cdot)$ and $m_{S'}(\cdot)$. Finally, any root r of R is a mapping node such that $m_G(r) = r(G)$.

Given a reconciliation α, Algorithm 1 outputs a tree T_α depicting all the information contained in α. Roughly speaking, Algorithm 1 creates a mapping node per pair $(u, \alpha_i(u))$, with $u \in V(G)$ and $1 \leq i \leq |\alpha(u)|$, and an event node per each event implied by α (in the sense of Definition 1). Edges between nodes are then added to encode the information contained in α.

Definition 4. Given a gene tree G and the subdivision S' of a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$, an *event-mapping tree* T for G and S' is an event-mapping graph for G and S' whose nodes form a tree. Moreover, T is said to be a *reconciliation tree* for G and S' if there exists a reconciliation α for G and S' such that T is isomorphic to the tree returned by Algorithm 1 when inputted with α and G.

Note that the event-mapping graph returned by Algorithm 1 is a tree if and only, for any node $u \in V(G)$, there exists no pair (i, j) with $1 \leq i < j \leq |\alpha(u)|$ such that $\alpha_i(u) = \alpha_j(u)$. This means that no chain of $\mathbb{TL}$ events where the starting donor and the final recipient of the transfers coincide can exist in α. In this paper we will not consider this unnatural kind of reconciliations[a].

Algorithm 1 ReconciliationTree(α, G)

Input: A reconciliation α and a gene tree G.
Output: The reconciliation tree T_α.
$T_\alpha \leftarrow$ the graph composed of a single mapping node r_α such that $m(r_\alpha) = (r(G), \alpha_1(r(G)))$;
for each node $u \in V(G)$ in preorder **do**
    for i from 1 to $|\alpha(u)|$ **do**
        $z \leftarrow$ the mapping node of T_α such that $m(z) = (u, \alpha_i(u))$;
        Add to T_α an event node n_e as a child of z and with the event type associated to $\alpha_i(u)$ (see Def. 1);
        if one among conditions 2-4 of Definition 1 is satisfied **then**
            Add to T_α a mapping node having $(u_l, \alpha_1(u_l))$ as mapping and n_e as parent;
            Add to T_α a mapping node having $(u_r, \alpha_1(u_r))$ as mapping and n_e as parent;
        else if one among conditions 5-7 of Definition 1 is satisfied **then**
            Add to T_α a mapping node having $(u, \alpha_{i+1}(u))$ as mapping and n_e as parent;
return T_α.

As an example, the reconciliation tree corresponding to the reconciliation α depicted in Figure 1 is shown in Figure 2.

Algorithm 2 returns a mapping α given an event-mapping tree T_α. The algorithm simply consists in visiting T_α in preorder and, for each mapping node y encountered, adding $m_{S'}(y)$ at the end of the list $\alpha(m_G(y))$.

Remark 1. Note that for a reconciliation α there exists exactly one reconciliation tree (up to isomorphism), that we denote T_α. Conversely, the reconciliation associated to a reconciliation tree can be obtained as described in Algorithm 2.

Algorithm 2 Mapping(T_α)

Input: An event-mapping tree T_α.
Output: A mapping α.
for each node $u \in V(G)$ **do** $\alpha(u) \leftarrow ()$;
for each node y of T_α in preorder **do**
    if y is a mapping node **then**
        Add $m_{S'}(y)$ at the end of the list $\alpha(m_G(y))$;
return α.

[a]It will be proved that all parsimonious reconciliations cannot contain this aberrant kind of $\mathbb{TL}$ event chains in the proof of Theorem 1.

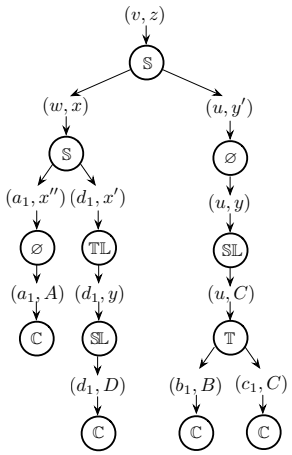

Fig. 2. The reconciliation tree T_α for the reconciliation α depicted in Figure 1.

The above results motivate the next one: if a single reconciliation can be depicted by a tree, is it possible to construct a graph depicting a whole set of reconciliations in a compact way?

3.2. *Reconciliation graph*

The following definition characterizes whether an event-mapping graph R *displays* an event-mapping tree T or not:

Definition 5. Given a gene tree G, the subdivision S' of a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$ and an event-mapping graph R for G and S', we say that an event-mapping tree T for G and S' is *displayed* by R if all following conditions are satisfied:

C_1. T is a connected subgraph of R;
C_2. $m_G(r(T)) = r(G)$;
C_3. all leaves of T are leaves in R;
C_4. for any event node z of T, all children of z in R are also children of z in T;
C_5. all mapping nodes of T have precisely one child.

The set of event-mapping trees displayed by R is denoted by $\mathcal{T}(R)$.

Definition 6. Given a gene tree G, the subdivision S' of a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$ and a set A of reconciliations for G and S', we denote by $\mathcal{T}(A)$ the set of reconciliation trees for A. We define a *reconciliation graph* for A as an event-mapping graph R_A for G and S' such that $\mathcal{T}(R_A)$ coincides with $\mathcal{T}(A)$.

Note that the same notation $\mathcal{T}(\cdot)$ is used both for an event-mapping graph R and for a set of reconciliations as Definition 4 implies that event-mapping trees

$\mathcal{T}(R_A)$ in Definition 6 are indeed reconciliation trees.

Remark 2. Let G be a gene tree and S' the subdivision of a dated species tree S such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$. For each set of reconciliations A for G and S', there exists at least one reconciliation graph for A.

The previous remark follows from the fact that the set $\mathcal{T}(A)$ of all reconciliation trees for A is a trivial (disconnected) reconciliation graph for A. This extreme solution to obtain a reconciliation graph is quite unsatisfactory when A denotes the set of MPRs for a gene tree G and species tree S, as $|A|$ can be exponential in $|G|$ and $|S|$. Alternatively, the graph obtained from $\mathcal{T}(A)$ by merging only *mapping nodes* of identical label is a compact event mapping graph. Nonetheless, obtaining this graph by first computing $\mathcal{T}(A)$ would not be efficient, as this set can be of exponential size. In the following, we show how it can be computed in polynomial time without enumerating $\mathcal{T}(A)$ nor all MPRs.

4. *Minimum* all parsimonious reconciliation graph for the MPR problem

In this section we describe how to efficiently construct a reconciliation graph whose size is only polynomial in that of the gene and species trees. This graph yet displays the set of all canonical solutions of the MPR problem. Given a dated species tree S, a gene tree G such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$ and positive costs δ, τ and λ for $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events, we denote by $A(G,S)$ the set of most parsimonious reconciliations between G and S and by $C(G,S)$ the cost of one of such reconciliation. Moreover, we denote by $\mathbf{c} : V(G) \times V(S') \mapsto \mathbb{R}$ the cost matrix computed by Algorithm 1 of Ref. 10, when inputted with (S, G, δ, τ, λ). For each pair (u,x), with $u \in V(G)$ and $x \in V(S')$, the value $\mathbf{c}(u,x)$ is equal to the minimal cost over all reconciliations between G_u and subtrees of S' such that $\alpha_1(u) = x$. This implies[10] that $C(G,S) = min_{x \in V(S')}\mathbf{c}(r(G),x)$.

4.1. *Computing a minimum reconciliation graph*

Definition 7. A reconciliation graph R_A for a set of reconciliations A is said to be *minimum* if $V(R_A)$ is of minimum size among all reconciliation graphs of A.

Algorithm 3 – together with its subroutine Algorithm 4 – describes how to construct a minimum reconciliation graph for $A(G,S)$. The function *AllBestReceivers* used in Algorithm 3 is an extension of the function *BestReceiver* introduced in Ref. 10: $AllBestReceivers(u,x)$ returns the set of nodes $x' \neq x$ of S' such that $h(x') = h(x)$ and $\mathbf{c}(u,x')$ is minimum over all nodes at this height. This function is used to investigate horizontal transfers in S'. An example of application of Algorithm 3 is shown on Figure 3.

Remark 3. Note that, by construction, each event node of the graph R_A returned by Algorithm 3 can have at most two children and at most one parent. Moreover,

Algorithm 3 MinimumMPRReconciliationGraph(S, G, δ, τ, λ)

Input: A dated species tree S, a gene tree G such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$ and positive costs δ, τ, resp. λ for $\mathbb{D}$, $\mathbb{T}$ resp. $\mathbb{L}$ events.
Output: a minimum reconciliation graph R_A for $A = A(G, S)$.
Construct the dated subdivision S' of S;
Initialize q as an empty list and $\mathbf{c}$ as the cost matrix computed by Algorithm 1 of Ref. 10, when inputted with (S, G, δ, τ, λ);
$\mathcal{Y} \leftarrow$ all nodes $y' \in V(S')$ such that $\mathbf{c}(r(G), y') = C(G, S)$;
for each node $y' \in \mathcal{Y}$ **do**
    Create a mapping node z in R_A having $(r(G), y')$ as mapping;
    Add the pair $(r(G), y')$ at the head of the list q;
while q is not empty **do**
    Let (u, x) be the head element of q; Remove (u, x) from q;
    Let z be the node of $V_m(R_A)$ such that $m(z) = (u, x)$;
    if $u \in L(G)$, $x \in L(S')$, and $s(u) = s(x)$ **then**
        Update(R_A, z, q, null, null, $\mathbb{C}$); **{$\mathbb{C}$ event}**
    if u has two children **then**
        if x has two children in S' **then**
            if $\mathbf{c}(u, x) = \mathbf{c}(u_l, x_l) + \mathbf{c}(u_r, x_r)$ **then**
                Update(R_A, z, q, (u_l, x_l), (u_r, x_r), $\mathbb{S}$); **{$\mathbb{S}$ event}**
            if $\mathbf{c}(u, x) = \mathbf{c}(u_l, x_r) + \mathbf{c}(u_r, x_l)$ **then**
                Update(R_A, z, q, (u_l, x_r), (u_r, x_l), $\mathbb{S}$); **{$\mathbb{S}$ event}**
        if $\mathbf{c}(u, x) = \mathbf{c}(u_l, x) + \mathbf{c}(u_r, x) + \delta$ **then**
            Update(R_A, z, q, (u_l, x), (u_r, x), $\mathbb{D}$); **{$\mathbb{D}$ event}**
        $\mathcal{X}' \leftarrow AllBestReceivers(u_l, x)$
        $\mathcal{X}'' \leftarrow AllBestReceivers(u_r, x)$;
        for each node $x' \in \mathcal{X}'$ **do**
            if $\mathbf{c}(u, x) = \mathbf{c}(u_l, x') + \mathbf{c}(u_r, x) + \tau$ **then**
                Update(R_A, z, q, (u_l, x'), (u_r, x), $\mathbb{T}$); **{$\mathbb{T}$ event}**
        for each node $x'' \in \mathcal{X}''$ **do**
            if $\mathbf{c}(u, x) = \mathbf{c}(u_l, x) + \mathbf{c}(u_r, x'') + \tau$ **then**
                Update(R_A, z, q, (u_l, x), (u_r, x''), $\mathbb{T}$); **{$\mathbb{T}$ event}**
    if x has a single child x_l in S' **then**
        if $\mathbf{c}(u, x) = \mathbf{c}(u, x_l)$ **then**
            Update(R_A, z, q, (u, x_l), null, $\varnothing$); **{$\varnothing$ event}**
    if x has two children in S' **then**
        if $\mathbf{c}(u, x) = \mathbf{c}(u, x_l) + \lambda$ **then**
            Update(R_A, z, q, (u, x_l), null, $\mathbb{SL}$); **{$\mathbb{SL}$ event}**
        if $\mathbf{c}(u, x) = \mathbf{c}(u, x_r) + \lambda$ **then**
            Update(R_A, z, q, (u, x_r), null, $\mathbb{SL}$); **{$\mathbb{SL}$ event}**
    $\mathcal{X}' \leftarrow AllBestReceivers(u, x)$;
    for each node $x' \in \mathcal{X}'$ **do**
        if $\mathbf{c}(u, x) = \mathbf{c}(u, x') + \tau + \lambda$ **then**
            Update(R_A, z, q, (u, x'), null, $\mathbb{TL}$); **{$\mathbb{TL}$ event}**
end while
return R_A;

each event node has its parent in $V_m(R_A)$, while each mapping node has its parent(s) in $V_e(R_A)$.

Remark 4. Note that, by construction, for each pair of mapping nodes z, z' of the

Algorithm 4 Update(R_A,z, q, m', m'', e_{type})

1: **Input:** An event-mapping tree (or graph) R_A, the mapping node $z \in V(R_A)$ currently considered, a list q of mapping nodes to process, two individual mappings m' and m'' and an event type e_{type}.
2: **Output:** updated versions of R_A and q.
3: Add to R_A an event node n_e of event type e_{type} and having z as parent;
4: **if** $m' \neq null$ **then**
5: **if** no node with mapping m' exists in R_A **then**
6: Add to R_A a mapping node having m' as mapping;
7: Add m' at the head of the list q;
8: Add to R_A an edge between n_e and the node having m' as mapping;
9: **if** $m'' \neq null$ **then**
10: **if** no node with mapping m'' exists in R_A **then**
11: Add to R_A a mapping node having m'' as mapping;
12: Add m'' at the head of the list q;
13: Add to R_A an edge between n_e and the node having m'' as mapping.

graph R_A returned by Algorithm 3 such that there exists a directed path from z to z', one of the following condition is satisfied:

(1) $m_{S'}(z)$ is an ancestor of $m_{S'}(z')$ and $m_G(z)$ is a strict ancestor of $m_G(z')$;
(2) $m_{S'}(z)$ is a strict ancestor of $m_{S'}(z')$ and $m_G(z)$ is an ancestor of $m_G(z')$;
(3) $m_{S'}(z)$ is in the same time slice as $m_{S'}(z')$ and $m_G(z)$ is an ancestor of $m_G(z')$.

Remark 4 follows from the fact that, at each iteration of Algorithm 3, we can only create mapping nodes (v, y) that are "grandchildren" of the currently analyzed node (u, x) and that respect one of the following (1) $y \in \{x, x_l, x_r\}$ and u is the parent of v (lines 17,19,21,26,29), (2) $v \in \{u, u_l, u_r\}$ and x is the parent of y (lines 17,19,32,35,37) or (3) $v \in \{u, u_l, u_r\}$ and x and y are on the same time slice (lines 26,29,41).

Theorem 1. *Given a dated species tree S, a gene tree G such that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$ and strictly positive costs δ, τ, resp. λ for $\mathbb{D}$, $\mathbb{T}$, resp. $\mathbb{L}$ events, Algorithm 3 computes a minimum reconciliation graph for $A(G, S)$.*

Proof. We first prove that the graph R_A constructed by Algorithm 3 is an event-mapping graph; then we show that it is a reconciliation graph for $A(G, S)$, that is $\mathcal{T}(R_A) = \mathcal{T}(A(G, S))$; last we prove that R_A is minimum.

R_A is an event-mapping graph To prove that the graph R_A is an event-mapping graph we need to prove that it is acyclic – the other conditions of Definition 3 are clearly satisfied by Algorithm 4. From Remarks 3 and 4, it is easy to see that the only possible cycles in R_A are composed of a series of mapping nodes sharing the same $m_G(\cdot)$ and such that all $m_{S'}(\cdot)$ are in the same time slice, linked by event nodes such that $e(\cdot) = \mathbb{TL}$. Since the costs τ and λ are greater than zero, this would imply that, for any mapping node z involved in the cycle, $\mathbf{c}(m_G(z), m_{S'}(z)) < \mathbf{c}(m_G(z), m_{S'}(z))$, a contradiction.

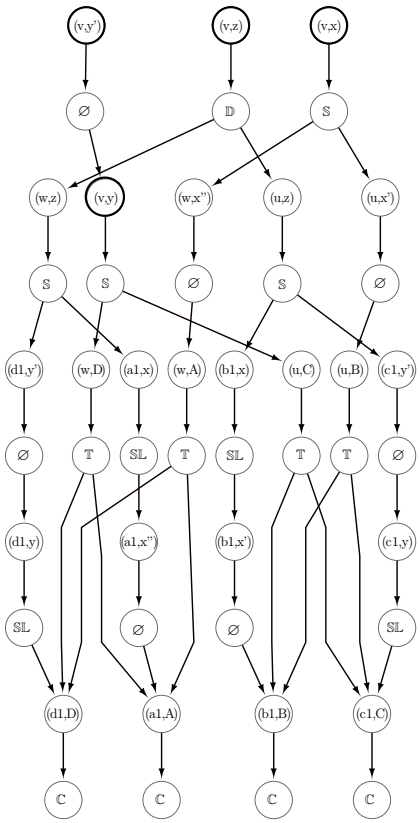

Fig. 3. The reconciliation graph for trees of Figure 1 that encodes the 4 MPRs obtained when using costs $\delta = 2$, $\tau = 3$ and $\lambda = 1$ (each bold node is the beginning of an MPR). The figure was drawn using *Cytoscape* (choosing the hierarchical mode).

$\boxed{\mathcal{T}(R_A) \subseteq \mathcal{T}(A(G,S))}$ To prove this, we will show that for all $T_\alpha \in \mathcal{T}(R_A)$, the mapping α associated to T_α is a reconciliation in A(G,S). Recall that the mapping α associated to T_α can be easily reconstructed from T_α using Algorithm 2.

We start by proving that α is a reconciliation. Note that any tree T_α displayed by R_A (see Definition 5) containing a mapping node z such that $m_G(z) = u$, with u internal node of G, also contains two mapping nodes z', z'' such that $m_G(z') = u_l$ and $m_G(z'') = u_r$. This follows from Remark 4 and from the fact that any time slice – as well as the subtree S'_x, where $m_{S'}(z) = x$ – can contain only a finite number of nodes. This implies that, after a finite number of iterations of the main loop of Algorithm 3 (line 9), we will encounter a $\mathbb{S}$, a $\mathbb{D}$ or a $\mathbb{T}$ event, initializing $\alpha(u_l)$ and $\alpha(u_r)$. Moreover, from Condition C_1 of Definition 5, we have that $m_G(r(T_\alpha)) = r(G)$. This implies by recursion that $\alpha(\cdot)$ is correctly defined for all nodes of $V(G)$.

We now show that when obtaining α from T_α with Algorithm 2, once $\alpha(u_l)$ and $\alpha(u_r)$ are initialized, the value of $\alpha(u)$ is never modified again. Let suppose that it is not the case. Then we can have two possibilities: T_α, and thus R_A from C_1 of Definition 5, contains two mapping nodes z, z' such that $m_G(z') = u$ and $m_G(z) \in \{u_l, u_r\}$, and 1) z is a strict ancestor of z' or 2) z, z' are not comparable. The first case is impossible from Remark 4. The second is not possible from the same remark and due to the fact that incomparable nodes of R_A have an event node with $e(\cdot) \in \{\mathbb{S}, \mathbb{D}, \mathbb{T}\}$ as last common ancestor, and thus, by construction, they have incomparable $m_G(\cdot)$. So, once $\alpha(u)$ has been modified by Algorithm 2 when considering nodes of T_α created on lines 13, 17, 19, 21, 26 and 29 of Algorithm 3, it is never modified again. With a similar reasoning it is possible to show that, apart from the root, the value of $\alpha_1(u)$ is set when considering nodes created on lines 17, 19, 21, 26 and 29, i.e when setting the value of $\alpha_\ell(u_p)$. The parameters given to Algorithm 4 at these respective lines and line 13 induce that α satisfies Condition (a) of Definition 1.

Moreover, α does also satisfy Condition (b) of Definition 1. This follows from the reasoning above and from the way α is updated when considering nodes created on lines 32, 35, 37 and 41. So α is a reconciliation.

We can now easily prove that the cost of α is actually $C(S, G)$. Indeed, the pairs $(r(G), y')$ defined on line 5 is such that $\mathbf{c}(r(G), y') = C(S, G)$. So the claim follows from Theorem 4 of Ref. 8, since Algorithm 3 backtracks through the cost matrix $\mathbf{c}(\cdot, \cdot)$.

$\boxed{\mathcal{T}(R_A) \supseteq \mathcal{T}(A(G, S))}$ By construction, the reconciliation tree T_α representing any MPR α is displayed by R_A if for any mapping node z' in T_α with mapping $(u, \alpha_i(u))$, there exists in R_A a mapping node z with $m(z) = m(z')$, and one of the following cases is true:

(1) $\alpha_i(u)$ satisfies Condition 1 of Definition 1 and Update(z, q, null , null , e_{type}) has been called by Algorithm 3;
(2) $\alpha_i(u)$ satisfies one among Conditions 2-4 of Definition 1 and Update(z, q, $\alpha_1(u_l)$, $\alpha_1(u_r)$, e_{type}) has been called by Algorithm 3;
(3) $\alpha_i(u)$ satisfies one among Conditions 5-7 of Definition 1 and Update(z, q, $\alpha_{i+1}(u)$, null, e_{type}) has been called by Algorithm 3.

Note that, since α is a most parsimonious reconciliation, then $\mathbf{c}(r(G), \alpha_1(r(G)) = C(G, S)$. This implies that there always exists in R_A a mapping node z such that $m(z) = m(r(T_\alpha)) = (r(G), \alpha_1(r(G))$ (such a node is created on line 7 of Algorithm 3).

Let z' be a mapping node of T_α with depth d. Let suppose that there exists a node z in R_A with $m(z) = m(z') = (u, \alpha_i(u))$. Let now suppose that $\alpha_i(u)$ satisfies one among Conditions 5-7 of Definition 1 and Condition (3) given above is not satisfied. Since a node z' with $m(z) = m(z') = (u, \alpha_i(u))$ exists in R_A, this means that z was in q at least once. But if Update(z, q, $\alpha_{i+1}(u)$, null, e_{type}) was not called by Al-

gorithm 3, this means that $\mathbf{c}(u, \alpha_i(u)) \neq \mathbf{c}(u, \alpha_{i+1}(u)) + \text{cost}(e_{type})$ which is impossible. Indeed $\mathbf{c}(u, \alpha_i(u))$ cannot be greater than $\mathbf{c}(u, \alpha_{i+1}(u)) + \text{cost}(e_{type})$ since, by definition,[10] $\mathbf{c}(u, \alpha_i(u))$ is the minimum of the costs of all possible events described in Definition 1, among whom $\mathbf{c}(u, \alpha_{i+1}(u)) + \text{cost}(e_{type})$. Moreover $\mathbf{c}(u, \alpha_i(u))$ cannot either be smaller than $\mathbf{c}(u, \alpha_{i+1}(u)) + \text{cost}(e_{type})$, otherwise for Theorem 4 of Ref. 8 α would not be parsimonious. In a similar way, we can prove that, if $\alpha_i(u)$ satisfied one among Conditions 1-4 of Definition 1 then either Condition (1) or Condition (2) given above is true. This also implies that all mapping nodes descendant from z' in T_α with depth $d+2$ are in R_A (nodes with depth $d+1$ are event nodes).

Since we proved that R_A contains the only mapping node z of T_α with depth zero (the root), this proves the claim by recursion on the depth of z.

We thus have that $\mathcal{T}(R_A) \supseteq \mathcal{T}(A(G,S))$ and hence that $\mathcal{T}(R_A) = \mathcal{T}(A(G,S))$.

Minimality We still need to prove the minimality of the graph, i.e. that $V(R_A)$ is of minimum size among all reconciliation graphs for $A(G,S)$. Let z be a mapping node of a reconciliation graph R of $A(G,S)$. Moreover, let $M(z)$ be the set of mapping nodes of R such that $m(\bar{z}) = m(z)$, $\forall \bar{z} \in M(z)$ and let $\mathcal{C}(z)$ be the child nodes of the nodes in $M(z)$. Let R' be another reconciliation graph of $A(G,S)$. Since R and R' display the same set of trees, there has to be a node z' in R' such that, $\forall j \in \mathcal{C}(z)$, there exists $j' \in \mathcal{C}(z')$ such that $e(j) = e(j')$, the outdegree of j and j' is the same and the child node(s) of j and j' has/have the same mapping. The same holds in the other direction, i.e. for $\mathcal{C}(z')$ w.r.t. $\mathcal{C}(z)$. The minimality of $V(R_A)$ follows from the fact that, for each mapping node z in $V(R_A)$, we have that $|M(z)| = 1$ (lines 5 and 10 of Algorithm 4) and that $\mathcal{C}(z)$ is, by construction, of minimum size. □

The next theorem establishes that the time and space complexities of Algorithm 3 are polynomial, hence that the potentially exponential number of most parsimonious reconciliations can be dealt with efficiently, due to many common parts.

Theorem 2. *Algorithm 3 runs in $O(|S|^3 \cdot |G|)$ time and space.*

Proof. First consider the space complexity of the algorithm that handles the graph R_A, and data structures q, $\mathbf{c}(\cdot,\cdot)$, and $AllBestReceivers(\cdot,\cdot)$. The latter has size $O(|S|)$ as it contains nodes of S' of the same slice as some node of S' (this bound is obtained by noting that S' has the same breadth as S). The $\mathbf{c}(\cdot,\cdot)$ matrix stores one value for each of the $|V(G)| \cdot |V(S')|$ cells, hence occupies $O(|S'| \cdot |G|) = O(|S|^2 \cdot |G|)$ space. Due to lines 5 and 10 of Algorithm 4, each of the $V(S') \cdot V(G)$ possible mappings is added only once to the q list, which thus occupies $O(|S|^2 \cdot |G|)$ space.

The most space consuming data structure is the event-mapping graph R_A. Due to lines 5 and 10 of Algorithm 4, the size of $V_m(R_A)$ can be at most equal to the number of possible mappings between $V(S')$ and $V(G)$, that is $O(|S|^2 \cdot |G|)$. To bound the size of $V_e(R_A)$, remark that, starting from 0, the cardinality of $V_e(R_A)$ is increased by one each time the $Update(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ subroutine described in Algo-

rithm 4 is called. Thus, $|V_e(R_A)|$ equals the number of calls to this subroutine, which in turn directly depends on the number of iterations of the **while** loop (lines 9-42) of Algorithm 3. Each loop iteration considers a different mapping node z (line 10) of the q list, which we recall contains $O(|S|^2 \cdot |G|)$ nodes over the whole algorithm. When a mapping node z is considered, it generates a constant number of calls to $Update(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ at lines 13, 17, 19, 21, 32, 35, 37, and $O(\big|AllBestReceivers(\cdot,\cdot)\big|)$ calls at lines 26, 29 and 41. As we saw, the size of $AllBestReceivers(\cdot,\cdot)$ is bounded above by $O(|S|)$ which implies that $O(|S|^3 \cdot |G|)$ calls to the $Update(\cdots)$ subroutine are issued, hence that $|V_e(R_A)| = O(|S|^3 \cdot |G|)$. Finally, each node of $V_e(R_A)$ has one parent and at most 2 children (Remark 3). Since no edge links nodes of $V_m(R_A)$, this implies that the size of $E(R_A)$, as well as that of R_A, is $O(|S|^3 \cdot |G|)$ too. This proves the space complexity of the algorithm.

Now consider the running time of the algorithm. Computing the subdivision of S and the cost matrix at lines 3 and 4 costs $O(|S|^2 \cdot |G|)$, as proven in Ref. 10. Each computation of $AllBestReceivers(\cdot,\cdot)$ at line 38 (re-used in later loop iterations at lines 22 and 23) costs $O(|S|)$ by a simple modification of the $BestReceiver(\cdot,\cdot)$ subroutine introduced in Ref. 10. As the while loop is performed at most $O(|S|^2 \cdot |G|)$ times (see above), the set of calls to $AllBestReceivers(\cdot,\cdot)$ globally cost $O(|S|^3 \cdot |G|)$. Safe for this part, the most time consuming part of the while loop are the calls to the $Update(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ subroutine. We already showed above that $O(|S|^3 \cdot |G|)$ such calls are performed. Moreover, each run of $Update(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ only requires constant time as long as a $|V(G)| \times |V(S')|$ matrix is maintained over the whole algorithm to know in $O(1)$ if a mapping node has already been considered (tests of lines 5 and 10 of Algorithm 4). It is straightforward to see that maintaining such a matrix costs $O(|V(G)| \cdot |V(S')|)$. Thus, the calls to $Update(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot)$ overall require $O(|S|^3 \cdot |G|)$ time. In conclusion, the two most expensive steps of Algorithm 3 require $O(|S|^3 \cdot |G|)$, which is then the time complexity of this algorithm. □

4.2. *Scoring the number of* canonical *reconciliations*

Let $I = (S, G, \delta, \tau, \lambda)$ be an instance of the MRP problem. As evoked in Section 2, the authors of Ref. 8 introduced the concept of *canonical* reconciliations (see Definition 2). A canonical reconciliation acts as the standard member among reconciliations that differ on S' but are equivalent on S. Indeed, the subdivision S' is only used as a trick to solve the problem in polynomial time. Roughly speaking, two or more reconciliations are equivalent when each event occurs on (possibly different) vertices of S' that are all located on the same branch of S. Given an event node z of R_A, we say that z is non canonical if 1) $e(z) \in \{\mathbb{D}, \mathbb{T}\}$ and there exists two directed paths of length 2 from z to two distinct event nodes with $e(\cdot) = \varnothing$ or 2) $e(z) = \mathbb{TL}$, $m_S(z_p)$ has one child in S' and there exists a directed path of length 2 from z to an event node with $e(\cdot) = \varnothing$ or 3) $e(z) = \varnothing$ and z is a child of a root of R_A.

Algorithm 5 counts the number of (canonical) reconciliations contained in a reconciliation graph R_A. This process is made easier by adding a fake root f as

parent of the real roots of R_A and afterward calling Score(f):

Algorithm 5 Score(u)

1: **for** each node $z \in descendants(u)$ considered in post-order **do**
2: **if** z is a leaf **then**
3: $score(z) = 1$;
4: **else**
5: **if** z is a mapping node or the fake root **then**
6: $score(z) = \sum_{z' \in \text{children}(z)} score(z')$;
7: **if** z is a root of R_A **then** // remove non canonical reconciliations
8: $score(z) = score(z) - \sum_{\substack{z' \in \text{children}(z) \\ n_e(z')=\varnothing}} score(z')$;
9: **else**
10: $score(z) = \prod_{z' \in \text{children}(z)} score(z')$;
11: **if** z is a $\mathbb{D}$, $\mathbb{T}$ or $\mathbb{TL}$ non canonical event **then** // remove non canonical reconciliations
12: $score(z) = score(z) - \prod_{\substack{z'' \in \text{grandChildren}(z) \\ n_e(z'')=\varnothing}} score(z'')$;
13: **return** $score(u)$.

Theorem 3. *Let $I = (S, G, \delta, \tau, \lambda)$ be an instance of the MPR problem and let R_A be the reconciliation graph computed by Algorithm 3 for this instance. Additionally, let R'_A be the event-mapping graph obtained from R_A by adding a fake root f as parent of the real roots of R_A. Then Score(f) equals the number of (canonical) MPRs for I.*

Proof. Note that, from Definition 5, each (canonical) reconciliation tree containing a mapping node z contains exactly *one* outgoing edge of z. This implies that the number of solutions containing z is equal to the number of solutions containing exactly *one* of the children of z – the logical OR explains the *summation* in Algorithm 5. On the contrary, if z is event node, each (canonical) reconciliation tree containing z contains *all* outgoing edges of z, and thus all children of z – the logical AND explains the *product*. The correctness of this algorithm follows from this observation and from the fact that each reconciliation tree displayed by R_A has to satisfy also C_2 and C_3 of Definition 5. The subtractions on lines 8 and 12 allow to eliminate from the counting the non canonical MPRs. This works because, by construction, a mapping node can have only one child node with $e(\cdot) = \varnothing$.

Then $Score(f)$ equals the number of (canonical) reconciliation trees displayed by R_A and the claim follows from Theorem 1. □

For example, the number of reconciliations and the number of canonical reconciliations displayed by the minimum reconciliation graph in Figure 3 are respectively 4 and 3, since the reconciliation starting with the node whose mapping is (v, y') is

non canonical. Note that the time complexity for this function is obviously $O(|R_A|)$ as there is at most one elementary operation (sum or product) for each edge of the graph. Score(f) also computes, as a byproduct, for each node $z \in R_A$, the number of partial (canonical) MPRs between G_u and subtrees of S' such that $\alpha_1(u) = v$ containing the node z and displayed by R_A, where $u = m_G(z)$ and $v = m_S(z)$ if z is a mapping node and $u = m_G(z_p)$ and $v = m_S(z_p)$ otherwise. It is easy to see that these scores can be used to compute, with a simple walk on the graph visiting a node if only if all his parents have already been visited, the number of (canonical) MPRs containing z displayed by R_A, for each node $z \in R_A$. A normalized score can be obtained by dividing each node score by the score of the fake root, the resulting value corresponding for each node to the percentage of (canonical) MPRs displayed by R_A containing it. This can be used to assert the robustness of each node: the higher the value the more reliable the node.

5. Discussion

The introduction of reconciliation graphs allows us to efficiently count the number of optimal scenarios, to detect events present in all of them and to depict alternative solutions that can be analyzed by expert users to sort out the most meaningful ones.

An implementation of the algorithm to construct the minimum reconciliation graph depicting all most parsimonious optimal scenarios (pseudocode in Algorithm 3) is available from: `http://celinescornavacca.wordpress.com/software/`.

This work opens up new perspectives to infer confidence values for predicted gene duplications and lateral gene transfers, as well as to build up consensus of reconciliations. Though such useful tools exist in the field of molecular phylogenetic inference, they are still absent from the reconciliation field. Yet they are dearly needed there, not only to deal with equally likely (or equally parsimonious) reconciliations that may exist for a single gene family, but also to assert reconciliation robustness with respect to gene tree uncertainty, species tree dating or method parameters. Indeed, reconciliation methods can be misled by inaccurate gene and/or species tree[13,2] and predicted events vary depending on method parameters (as for phylogeny inference or sequence alignments). To confidently infer individual past events in the history of genes – and hence gene content of genomes through phylogenetic methods – we will likely need to consider not one gene tree, but several optimum or sub-optimum gene trees (and different event cost sets alike) and be able to summarize the corresponding reconciliations in a single one.

Acknowledgements This work has been partially funded by the Phylariane project ANR-08-EMER-011 and the ANCESTROME project ANR-10-IABI-0-01. This publication is the contribution no. 2012-147 of the Institut des Sciences de l'Evolution de Montpellier (ISE-M, UMR 5554). The authors would liked to thank Luay Nakhleh and an anonymous reviewer for their constructive comments.

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.*, 106:5714–5719, 2009.
2. A.M. Altenhoff and C. Dessimoz. Phylogenetic and functional assessment of orthologs inference projects and methods. *PLoS Comput. Biol.*, 5:e1000262, 2009.
3. L. Arvestad, A.-C. Berglund, J. Lagergren, and B. Sennblad. Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In *RECOMB'04*, pages 326–335, 2004.
4. M. S. Bansal, E. J. Alm, and M. Kellis. Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. *Bioinformatics*, 28(12):i283–i291, Jun 2012.
5. T Blomme, K. Vandepoele, S. De Bodt, C. Simillion, S. Maere, and Y. Van de Peer. The gain and loss of genes during 600 million years of vertebrate evolution. *Genome Biol.*, page 7:R43, 2006.
6. C. Conow, D. Fielder, Y. Ovadia, and R. Libeskind-Hadas. Jane: a new tool for the cophylogeny reconstruction problem. *Algorithms Mol Biol*, 5:16, 2010. Software downloadable at http://www.cs.hmc.edu/ hadas/jane/.
7. L. A. David and E. J. Alm. Rapid evolutionary innovation during an Archaean genetic expansion. *Nature*, 469:93–96, Jan 2011.
8. J.P. Doyon, V. Berry, C Scornavacca, and V. Ranwez. DTLS-trees: proving the parsimony model for reconciling gene and species trees. In preparation, 2012.
9. J.P. Doyon, V. Ranwez, V. Daubin, and V. Berry. Models, algorithms and programs for phylogeny reconciliation. *Brief. Bioinformatics*, 12:392–400, 2011.
10. J.P. Doyon, C. Scornavacca, K. Yu. Gorbunov, G.J. Szöllősi, V. Ranwez, and V. Berry. An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. In *Research in Computational Molecular Biology: Proceedings of the 14th International Conference on Research in Computational Molecular Biology (RECOMB)*, volume 6398 of *LNCS*, pages 93–108, Berlin/Heidelberg, Germany, 2010. Springer-Verlag. Software downloadable at http://www.atgc-montpellier.fr/Mowgli/.
11. D. Durand, B.V. Halldorsson, and B. Vernot. A hybrid micro-macroevolutionary approach to gene tree reconstruction. *J. Comput. Biol.*, 13:320–335, 2006.
12. Pawel Górecki and Jerzy Tiuryn. Inferring evolutionary scenarios in the duplication, loss and horizontal gene transfer model. In *Logic and Program Semantics*, pages 83–105, 2012.
13. M.W. Hahn. Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution. *Genome Biol.*, 8:R141, 2007.
14. Bin Ma, Ming Li, and Louxin Zhang. From gene trees to species trees. *SIAM J. Comput.*, 30(3):729–752, 2000.
15. D. Merkle, M. Middendorf, and N. Wieseke. A parameter-adaptive dynamic programming approach for inferring cophylogenies. *BMC Bioinformatics*, 11(Suppl 1):S60, 2010.
16. T.H. Nguyen, J.-P. Doyon, S. Pointet, A.-M. Arigon Chifolleau, V. Ranwez, and V. Berry. Accounting for gene tree uncertainty improves gene tree accuracy as well as duplications, transfers and losses predictions. Submitted, 2012.
17. Gergely J. Szöllősi, Bastien Boussau, Sophie S. Abby, Eric Tannier, and Vincent Daubin. Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. *Proceedings of the National Academy of Sciences*, 2012. In press.

18. C. Than, D. Ruths, H. Innan, and L. Nakhleh. Confounding factors in HGT detection: statistical error, coalescent effects, and multiple solutions. *J. Comput. Biol.*, 14:517–535, 2007.