JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 16, Number 10, 2009
© Mary Ann Liebert, Inc.
Pp. 1399–1418
DOI: 10.1089/cmb.2009.0095

Research Article

Space of Gene/Species Trees Reconciliations and Parsimonious Models

JEAN-PHILIPPE DOYON,[1] CEDRIC CHAUVE,[2] and SYLVIE HAMEL[1]

ABSTRACT

We describe algorithms to study the space of all possible reconciliations between a gene tree and a species tree, that is counting the size of this space, uniformly generate a random reconciliation, and exploring this space in optimal time using combinatorial operators. We also extend these algorithms for optimal and sub-optimal reconciliations according to the three usual combinatorial costs (duplication, loss, and mutation). Applying these algorithms to simulated and real gene family evolutionary scenarios, we observe that the LCA (Last Common Ancestor) based reconciliation is almost always identical to the real one.

Key words: algorithms, combinatorics, gene duplications and losses, gene families evolution.

1. INTRODUCTION

GENOMES OF CONTEMPORARY SPECIES, especially eukaryotes, are the result of an evolutionary history that started with a common ancestor from which new species evolved through evolutionary events called speciations. One of the main objectives of molecular biology is the reconstruction of this evolutionary history, that can be depicted with a rooted binary tree, called a *species tree*, where the root represents the common ancestor, the internal nodes the ancestral species and speciation events, and the leaves the extant species. Other events than speciation can happen, that do not result immediately in the creation of new species but are essential in eukaryotic genes evolution, such as gene duplication and loss (Graur and Li, 1999). Duplication is the genomic process where one or more genes of a single genome are copied, resulting in two copies of each duplicated gene. Gene duplication allows one copy to possibly develop a new biological function through point mutation, while the other copy often preserves its original role. A gene is considered to be lost when the corresponding sequence has been deleted by a genomic rearrangement or has completely lost any functional role (i.e., has become a pseudogene) (Graur and Li, 1999). Other genomic events such as lateral gene transfer, that occurs mostly in bacterial genomes, will not be considered here.

Genes of contemporary species that evolved from a common ancestor, through speciations and duplications, are said to be homologs (Fitch, 2000) and are grouped into a gene family. Such gene families are in

[1]DIRO, Université de Montréal, Montréal, Quebec, Canada.
[2]Department of Mathematics, Simon Fraser University, Burnaby, British Columbia, Canada.

general inferred using protein sequence comparison. The evolution of a gene family can be depicted with a rooted binary tree, called a *gene tree*, where the leaves represent the homologous contemporary genes, the root their common ancestral gene and the internal nodes represent ancestral genes that have evolved through speciations and duplications.

Given a gene tree G and the species tree S of the corresponding genomes, an important question is to locate in S the evolutionary events of speciations and duplications. A *reconciliation* between G and S is a mapping of the genes (extant and ancestral) of G onto the nodes of S that induces an evolutionary scenario, in terms of speciations, duplications and losses, for the gene family described by G. In this perspective, the notion of reconciliation was first introduced in the pioneering work of Goodman et al. (1979) and a first formal definition was given by Page (1994) to explain the discrepancies between genes and species trees. The LCA-mapping, that maps a gene u of G onto the most recent species of S that is ancestor of all genomes that contain a gene descendant of u, is the most widely used mapping, as it depicts a parsimonious evolutionary process according to the number of duplications or duplications and losses it induces. It is generally accepted that parsimony is a pertinent criterion in evolutionary biology, but that it does not always reflect the true evolutionary history. This leads to the definition of more general notions of reconciliations between G and S (Bonizzoni et al., 2005; Górecki and Tiuryn, 2006; Arvestad et al., 2004) and the natural problem of exploring all evolutionary scenarios of a given gene family. Arvestad et al. (2004) developed a Markov Chain Monte Carlo method that explores the possible reconciliations and approximates their posterior probabilities, but the efficient exploration of all reconciliations or only the most parsimonious ones has not been addressed until now.

Our theoretical contributions are the development of algorithms to study combinatorial aspects of the *space of the reconciliations* between G and S, and more specifically its exploration. These results allow us to give a first insight in the following question: is parsimony relevant to infer the true evolutionary scenario of a gene family? In Section 2, we introduce basic notations and a very general notion of reconciliation. In Section 3, we describe an algorithm that counts the total number of reconciliations or of sub-optimal ones (for the duplication cost) and an algorithm that generates a random reconciliation under the uniform distribution. In Section 4, we first define combinatorial operators that are sufficient to explore the complete space of reconciliations, and then develop an algorithm that exhaustively explores this space in optimal time. This allows us to compute the distribution of reconciliation scores in the duplication, loss, and mutation (duplication + loss) cost models. We also describe a variant of this algorithm that explores all and only all the sub-optimal reconciliations (according to an upper bound) for any of these models. There are several applications of our algorithms in functional and evolutionary genomics, such as inferring orthologs and paralogs (Fitch, 1970; Jensen, 2001), the gene content of an ancestral genome (Ma et al., 2007), or in the context of Markov Chain Monte Carlo analysis of gene families (Arvestad et al., 2004). In Section 5, we simulate several gene family evolutionary scenarios along two known species trees (Hahn et al. (2007b) and Hahn et al. (2007a)), with length (in time) and gene duplication and loss rates along each branch. We then study the shape of the reconciliation spaces according to the three usual cost models. Our main conclusion is that the less the cost of a reconciliation is, the more the reconciliation is similar to the real one.

2. PRELIMINARIES

Let T be a binary tree with vertices $V(T)$ and edges $E(T)$, and such that only its leaves are labeled. Let $r(T)$, $L(T)$, and $\Lambda(T)$ respectively denote its root, the set of its leaves, and the set of the labels of its leaves. We will adopt the convention that the root is at the top of the tree and the leaves at the bottom. A *species tree* S is a binary tree such that each element of $\Lambda(S)$ represents an extant species and labels exactly one leaf of S (there is a bijection between $L(S)$ and $\Lambda(S)$). A *gene tree* G is a binary tree. From now on, we consider a species tree S, with $|V(S)| = n$ and a gene tree G such that $\Lambda(G) \subseteq \Lambda(S)$ and $|V(G)| = m$. Let $\sigma : L(G) \to L(S)$ be the function that maps each leaf of G to the unique leaf of S with the same label.

For a vertex u of T, we denote by u_1 and u_2 its children and by T_u the subtree of T rooted at u. For a vertex $u \in V(T) \setminus \{r(T)\}$, we denote by $p(u)$ its parent. A *cell* of a tree T is either a vertex of T or an edge of T. Given two cells c and c' of T, $c' \leq_T c$ (resp. $c' <_T c$) if and only if c is on the unique path from c' to $r(T)$ (resp. and $c \neq c'$); in such a case, c' is said to be a *descendant* of c. The *LCA-mapping* $M : V(G) \to V(S)$ maps each vertex u of G to the unique vertex $M(u)$ of S such that $\Lambda(S_{M(u)})$ is the smallest cluster of S containing $\Lambda(G_u)$.

Definition 1. A reconciliation between a gene tree G and a species tree S is a mapping $\alpha : V(G) \rightarrow V(S) \cup E(S)$ such that

1. (Base constraint) $\forall u \in L(G), \alpha(u) = M(u) = \sigma(u)$.
2. (Tree Mapping Constraint) For any vertex $u \in V(G) \backslash L(G)$,
   a. if $\alpha(u) \in V(S)$, then $\alpha(u) = M(u)$.
   b. If $\alpha(u) \in E(S)$, then $M(u) <_S \alpha(u)$.
3. (Ancestor Consistency Constraint) For any two vertices $u, v \in V(G)$, such that $v <_G u$,
   a. if $\alpha(u), \alpha(v) \in E(S)$, then $\alpha(v) \leq_S \alpha(u)$,
   b. otherwise, $\alpha(v) <_S \alpha(u)$.

Remark 1. This definition of reconciliation differs slightly from the classical ones as vertices of G can be mapped onto edges of S, in order to represent duplication events. However, it is equivalent to the definitions given by Arvestad et al. (2004) and Górecki and Tiuryn (2006), that are the most complete ones known so far, and it is more general than the Inclusion-Preserving mapping of Bonizzoni et al. (2005).

The whole set of reconciliations between a gene tree G and a species tree S is denoted $\Psi(G, S)$. A reconciliation α of $\Psi(G, S)$ implies an evolutionary scenario for the genes of G in terms of gene duplications, gene losses, and speciations. A vertex u of G that is mapped onto an edge (x, y) of S (where $x = p(y)$) represents a gene of the ancestral species $p(y)$ that has been duplicated in y. If u is mapped onto an internal vertex x of S, then this represents a gene that will be present in a single copy in the two genomes x_1 and x_2 following a speciation event that happened to x. It is important to point out that the number of reconciliations is finite. Briefly, a reconciliation α between G and S represents any birth-and-death scenario along S such that the resulting gene tree is consistent with G and each duplication event that implies an internal vertex u of G is consistent with the mapping $\alpha(u)$ (Fig. 1).

We denote by $dup(\alpha)$ and $los(\alpha)$ respectively the number of duplications and losses induced by a reconciliation α. $dup(\alpha)$ is the number of vertices of G that are mapped onto an edge of S.[1] Given two cells $c, c' \in V(S) \cup E(S)$, where $c' <_S c$, $D(c, c')$ is the number of vertices $x \in V(S)$ such that $c' <_S x <_S c$. Also, if $c = c'$, then $D\ (c,\ c') = 0$. The number of losses associated to a vertex $u \in V(G) \backslash L(G)$ is noted l_u and equal to $D\ (\alpha(u), \alpha(u_1)) + D\ (\alpha(u), \alpha(u_2))$ (see Ma et al. (2001) for example). $los(\alpha)$ is then the sum of l_u over all internal vertices u. The third constraint of Definition 1 leads to the notion of *forced duplication*, that corresponds to vertices of G that can only be mapped onto an edge of S: an internal vertex $u \in V(G) \backslash L(G)$ is said to be a forced duplication if and only if $M(u) = M(u_1)$ or $M(u) = M(u_2)$.

For a vertex $u \in V(G)$, a cell of S *covers* it if u can be mapped onto this cell according to Definition 1. The set of cells that can cover it is denoted by $A(u)$ and is defined below.

$$A(u) = \begin{cases} \{M(u)\} & \text{if } u \in L(G) \text{ or } u = r(G) \\ \{c \in E(S) : M(u) <_S c\} & \text{if } u \text{ is a forced duplication} \\ \{c \in E(S) : M(u) <_S c\} \cup \{M(u)\} & \text{otherwise} \end{cases}$$

It is important to point out that there is three mappings that are considered here: $M(u)$, $\alpha(u)$, and $A(u)$. From now on, except when indicated, the term mapping will refer to the reconciliation mapping $\alpha(u)$ of Definition 1.

Finally, combinatorial and probabilistic criteria can be used to compare the different possible reconciliations and pick one that is supposed to reflect the most the true evolution of G according to S. Three parsimonious cost models, that aim to minimize the number of genomic events, have been proposed so far: duplication (Ma et al., 2001), loss (Chauve et al., 2008), and mutation (duplication + loss; Ma et al. (2001)). Arvestad et al. (2004) also introduced a notion of likelihood of a reconciliation in the framework of birth-and-death processes (Kendall, 1948).

3. COUNTING AND UNIFORM RANDOM GENERATION

In this section, we describe an efficient algorithm that computes a random reconciliation between G and S following the uniform distribution. This problem is important in the context of MCMC analysis for gene

[1]To consider duplication that precedes the first speciation event represented by $r\ (S)$, we can insert in S an "artificial" cell c such that $r(S) <_S c$. For the sake of clarity, and as handling such early duplications follows easily from our work, we assume here that no duplication occurs in the most ancestral species.

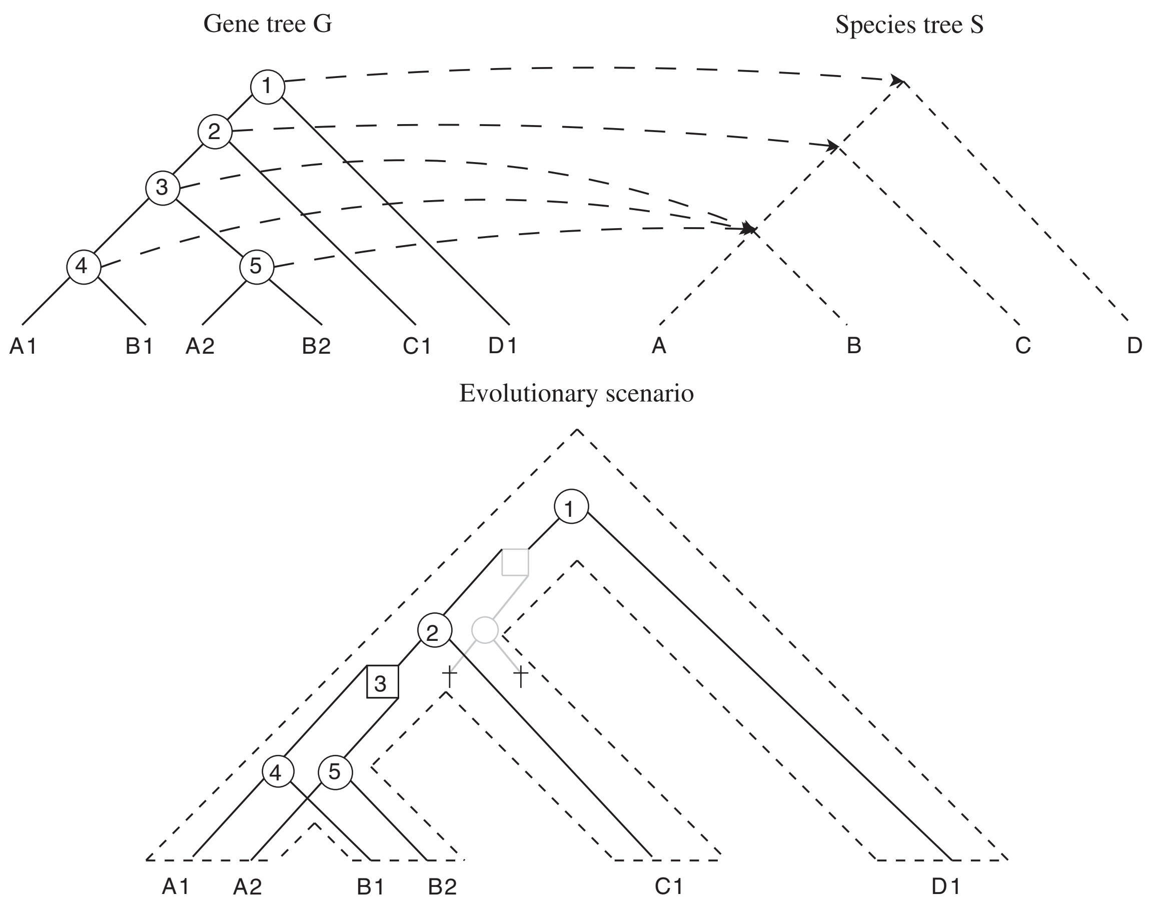

FIG. 1. (**Above**) The species tree S has four (extant) species (A, B, C, and D). The gene tree G has six (extant) genes, where each gene belongs to one of the four species (i.e., gene A1 belongs to species A). The arrows represent the LCA-mapping between G and S. (**Below**) A reconciliation between G and S. A circle (square) represents an internal vertex of G that is mapped on an internal vertex (resp. edge) of S, that is a speciation (resp. duplication) event. A cross represents a gene loss. The right lineage of the first duplication has no extant gene that descents from it, as opposite to its left lineage. We then say that this duplication is hypothetical, because it is not a useful information for the evolutionary scenario of the extant genes of G along S. Hence, such duplication is not depicted by the reconciliation.

families, as a major issue is to analyze if the Markov chain converges to the true posterior probabilities. One of the most popular and simple tests of convergence is to run several Markov chains, each starting at a different state in the space, which motivates our random generation algorithm.

As usual in uniform random generation, it is based on a preprocessing that computes the cardinality of $\Psi(G, S)$ (Denise and Zimmermann, 1997). We first address this problem, then describe the random generation algorithm.

Counting reconciliations. For every vertex $u \in V(G)$ and cell $c \in A(u)$, we denote by $Nb(u, c)$ the number of reconciliations of G_u and S_c for which u is mapped on c. It follows immediately that $|\Psi(G,S)| = Nb\ (r(G), r(S))$.

Lemma 1. *Let $u \in V(G)$ and $c \in A(u)$ be a cell that covers u. Then $Nb(u, c) = 1$ if $u \in L(G)$, and otherwise*

$$Nb(u, c) = \sum_{c_1 \in A(u_1), c_1 \leq_S c} Nb(u_1, c_1) \sum_{c_2 \in A(u_2), c_2 \leq_S c} Nb(u_2, c_2). \tag{1}$$

Proof. If $u \in L(G)$, it is obvious that $Nb(u, c) = 1$ is the number of reconciliations of G_u and S_c for which u is mapped on c. We prove equation (1). The case of pairs $c \in A(u)$ and $u \in L(G)$ are the base cases. Consider an internal vertex $u \in V(G) \backslash L(G)$, its children u_1 and u_2, and suppose that u is covered by a cell

$c \in A(u)$. There are two cases: c is either a vertex or an edge of S, and Definition 1 respectively implies that $c_1 <_S c$ and $c_1 \leq_S c$. These constraints are considered by the left term in the summation of equation (1), where $c \notin A(u_1)$ if $c \in V(S)$. Hence, this left term is the total number of reconciliations for G_{u_1} and S_c when u is mapped on c. By applying the same reasoning for u_2, we obtain the right term of the summation in equation (1). The mapping of u_1 and u_2 are independent of each other, we can then conclude that equation (1) is the number of reconciliations of G_u and S_c for which u is mapped on c. ■

Proposition 1. *$|\Psi(G, S)|$ can be computed in $O(mn)$ time and space.*

Proof. It follows from Lemma 1 and the obvious facts that A and M can be computed in $O(mn)$ worst-case time. ■

It was shown by Chauve and El-Mabrouk (2009) that there is a single optimal reconciliation for the loss and mutation costs, but that there can be several ones for the duplication cost. An important question is then to count the number of these optimal reconciliations, and a more general problem is to count the number of sub-optimal reconciliations. We consider here the case of the duplication cost, and $\Psi_{dup}(G, S, \delta) = \{\alpha \in \Psi(G, S) : dup(\alpha) \leq \delta\}$ is the set of sub-optimal reconciliations, for a given bound δ.

Let $K(G)$ be the set of vertices of G that are not forced duplications, that is $K(G) = \{u \in V(G) : M(u) \in A(u)\}$. For a vertex $u \in V(G)$ and a cell $c \in A(u)$, let $f(c)$ ($d(c)$) be the ancestor (resp. descendant) cell of c in $A(u)$, that is the cell of $A(u)$ that is the closest one to c and above (resp. below) it. The lowest (highest) cell of $A(u)$ is the one that has no descendant (resp. ancestor) cell in $A(u)$.

Definition 2. α_{min} (resp. α_{max}) is the unique reconciliation of $\Psi(G, S)$ where, for each vertex u of G, $\alpha_{min}(u)$ (resp. $\alpha_{max}(u)$) is the lowest (resp. highest) cell of $A(u)$.

Note that α_{min} corresponds to the classical LCA-mapping reconciliation as every vertex u that is not (resp. is) a forced duplication is mapped onto the LCA (resp. the edge preceding the LCA) of $\Lambda(G_u)$. Together with classical results on this LCA-mapping (Chauve and El-Mabrouk, (2009) and the definition of $dup(\alpha)$ and $K(G)$, we have the following properties.

Property 1.

1. α_{min} minimizes the duplication, loss, and mutation cost models.
2. For every reconciliation $\alpha \in \Psi(G, S)$, $dup(\alpha_{min}) \leq dup(\alpha) \leq dup(\alpha_{min}) + |K(G)|$.
3. For every $dup(\alpha_{min}) \leq \delta \leq dup(\alpha_{min}) + |K(G)|$, a reconciliation α is in $\Psi_{dup}(G, S, \delta)$ if and only if the number of vertex u of $K(G)$ such that $\alpha(u) \neq M(u)$ is at most $\delta - dup(\alpha_{min})$.

From these properties, we can then generalize Proposition 1 and describe a counting algorithm whose complexity is exponential in the worst-case time, but parameterized by the quantity $\delta - dup(\alpha_{min})$. In particular, Proposition 2 used with $\delta = dup(\alpha_{min})$ allows to know in polynomial time the exact number of optimal (for the duplication cost) reconciliations.

Proposition 2. *For a given $dup(\alpha_{min}) \leq \delta \leq dup(\alpha_{min}) + |K(G)|$, $|\Psi_{dup}(G, S, \delta)|$ can be computed in $O(qmn)$ time and $O(mn)$ space, where $q = \sum_{\ell=0}^{\gamma} \binom{|K(G)|}{\ell}$ and $\gamma = \delta - dup(\alpha_{min})$.*

Proof. For each ℓ in $\{0, 1, \ldots, \gamma\}$, we have to count the number of reconciliations $\alpha \in \Psi(G, S)$ such that $dup(\alpha) - dup(\alpha_{min}) = \ell$. According to Property 1.(3), we have to consider each of the $\binom{|K(G)|}{\ell}$ combinations of vertex u of $K(G)$ such that $\alpha(u) \neq M(u)$. For each of these combinations, we can adapt the equation 1 of Lemma 1 to compute its number of reconciliations. ■

We describe in Section 4.5 how to exhaustively explore the set of sub-optimal reconciliations for any of the three cost models.

Generating a random reconciliation. Algorithm 1 below computes a random reconciliation between G and S.

Theorem 1. *Given a reconciliation $\alpha \in \Psi(G,S)$, Algorithm 1 returns α with probability $\frac{1}{|\Psi(G,S)|}$. Given the table Nb and the sets $A(u)$ for every vertex u of G, it can be implemented to run in $O(mn)$ space and $\Theta(mn)$ time in the worst case and $\Theta(m)$ time in the best case.*

Proof. Let $Pr(\alpha)$ be the probability that Algorithm 1 returns α. It follows immediately from lines 5–12 of the algorithm that

$$Pr(\alpha) = \prod_{u \in V(G) \setminus \{r(G) \cup L(G)\}} \frac{Nb(u,\alpha(u))}{\sum_{c \in A(u),\, c \leq_S \alpha(p(u))} Nb(u,c)} \tag{2}$$

Algorithm 1 Uniform random generation in $\Psi(G,S)$.

```
1:  Let $\alpha$ be an empty reconciliation.
2:  Perform a prefix traversal of $G$, and let $u \in V(G)$ be the current vertex.
3:    **if** $u = r(G)$ or $u \in L(G)$ **then** $\alpha(u) \leftarrow M(u)$
4:    **else**
5:      Let $\hat{c} \leftarrow \alpha(p(u))$.
6:      {Choose randomly a cell $c \in A(u)$ such that $c \leq_S \hat{c}$}
7:      Let $k \leftarrow \sum_{c \in A(u), c \leq_S \hat{c}} Nb(u,c)$
8:      Generate randomly and uniformly an integer $n \in \{1, \ldots, k\}$.
9:      $c \leftarrow$ lowest cell in $A(u)$ {If $u$ is a forced duplication, then $M(u) \notin A(u)$}
10:     $l \leftarrow Nb(u,c)$
11:     **while** $l < n$ **do** $c \leftarrow f(c)$, $l \leftarrow l + Nb(u,c)$
12:     $\alpha(u) \leftarrow c$
13: **return** $\alpha$
```

By expanding the term $Nb(u,\alpha(u))$ in (2) according to Lemma 1, we obtain

$$Pr(\alpha) = \prod_{u \in V(G) \setminus \{r(G) \cup L(G)\}} \frac{\sum_{c_1 \in A(u_1),\, c_1 \leq_S \alpha(u)} Nb(u_1,c_1) \sum_{c_2 \in A(u_2),\, c_2 \leq_S \alpha(u)} Nb(u_2,c_2)}{\sum_{c \in A(u),\, c \leq_S \alpha(p(u))} Nb(u,c)}. \tag{3}$$

Cancellations leads to the following formula, where r_1 and r_2 are the two children of the root $r(G)$:

$$Pr(\alpha) = \frac{\prod_{u \in L(G)} \sum_{c \in A(u)} Nb(u,c)}{\sum_{c_1 \in A(r_1)} NB(r_1,c_1) \sum_{c_2 \in A(r_2)} NB(r_2,c_2)}, \tag{4}$$

that, together with Lemma 1, implies that $Pr(\alpha) = \frac{1}{|\Psi(G,S)|}$.
For the time complexity, suppose that for each vertex $u \in V(G)$, the size of $A(u)$ is in $\Theta(n)$. In the worst case, each vertex is mapped on the closest edge to $r(S)$ and the algorithm is in $\Theta(mn)$. In the best case, where each vertex is mapped onto the closest cell to $M(u)$ (lowest cell of $A(u)$), the time complexity is in $\Theta(m)$. The space complexity follows from the fact that there are $O(nm)$ pairs (u, c). ■

Hence, the preprocessing time of our algorithm (computing the table Nb and the sets $A(u)$) requires $O(mn)$ time and space. However, it needs to be done once and can be used for generating several random reconciliations.

4. EXPLORING THE SPACE $\Psi(G,S)$

We present in this section an algorithm that visits the set of all possible reconciliations between a gene tree G (with $|V(G)| = m$) and a species tree S (with $|V(S)| = n$) in time $\Theta(|\Psi(G,S)|)$ (see Theorem 3), which gives a CAT (Constant Amortized Time) algorithm to generate $\Psi(G,S)$.

First, we define combinatorial operators used to explore the whole space of reconciliations $\Psi(G, S)$ (Section 4.1). Second, we define a tree that covers $\Psi(G, S)$ and that is used by our algorithm to explore this space (Section 4.2). Third, we give some theoretical preliminaries that are required for the algorithm (Section 4.3) and then formally describe the algorithm (Section 4.4). We conclude this section by a variant of this algorithm that explores all and only all sub-optimal reconciliations for any of the three cost models (duplication, loss, or mutation) (Section 4.5).

4.1. Space exploration operators

We present in this section a combinatorial operator, called *Nearest Mapping Change* (NMC), acting on a reconciliation between a gene tree G and a species tree S. A similar operator was described by Górecki and Tiuryn (2006), in the framework of DLS-trees, to show that every DLS-tree can be obtained from the most parsimonious one. We develop it here with our definition of reconciliation and studies its properties. We first show that this operator is sufficient to explore the space of all possible reconciliations.

Definition 3. Let $\alpha : V(G) \to V(S) \cup E(S)$ be a given reconciliation between G and S, and u a vertex of $V(G) \backslash L(G)$ such that $u \neq r\,(G)$. Let $\hat{c}, \mathrm{c}, \mathrm{c}_1$, and c_2 respectively denote $\alpha(p(u))$, $\alpha(u)$, $\alpha(u_1)$, and $\alpha(u_2)$.

1. An *upward* NMC (uNMC) can be applied to u if $c <_S \hat{c}$, and if $\hat{c} \in V(S)$ and $c \in E(S)$, then $D(\hat{c}, c) > 0$. It changes $\alpha(u)$ into its ancestor cell $f(\alpha(u))$ of $A(u)$.
2. A *downward* NMC (dNMC) can be applied to u if $c_1 <_S c$, $M(u) <_S c$, and if $c_1 \in V(S)$ and $c \in E(S)$, then $D(c,\ c_1) > 0$ (idem for c_2). It changes $\alpha(u)$ into its descendant cell $d(\alpha(u))$ of $A(u)$.

It follows immediately from the definition of NMC operators that, given $\alpha \in \Psi(G, S)$, applying a NMC operator to a vertex u of G results in a reconciliation α' between G and S. More precisely, it can induce the following changes in the evolutionary scenario for the gene family (Fig. 2).

- Changing a speciation by a duplication (uNMC, $\alpha(u) = M(u)$).
- Changing a duplication by a speciation (dNMC, $\alpha'(u) = M(u)$).
- Moving a duplication upward (uNMC, $\alpha(u) \neq M(u)$).
- Moving a duplication downward (dNMC, $\alpha'(u) \neq M(u)$).

For $u \in V(G)$, and $c, c' \in A(u), d_u(c, c')$ is the number of cells of $A(u)$ between c and c', where $du\ (c, c' = 0$ if and only if $c = c'$. For two reconciliations α and α', $D_{NMC}(\alpha, \alpha') = \sum_{u \in V(G)} d_u(\alpha(u), \alpha'(u))$. We call $D_{NMC}(\alpha,\ \alpha')$ the *NMC distance* between α and α'. A valid (according to Definition 3) NMC application to α can be encoded by a vertex $u \in V(G)$, that is the vertex being moved, and by a direction that is either downward or upward. We denote by $uNMC(\alpha)$ the subset of $V(G)$ such that an upward operator can be applied on any $u \in uNMC(\alpha)$ and by $uNMC(\alpha,\ \alpha')$ the set of vertex $u \in uNMC(\alpha)$ such that applying an upward operator on u results in a new reconciliation where the mapping of u is closer to its mapping in α'. Formally,

$$uNMC(\alpha, \alpha') = \{u \in uNMC(\alpha) : d_u(\alpha(u), \alpha'(u)) = 1 + d_u(f(\alpha(u)), \alpha'(u))\}.$$

For the downward operator, let $dNMC(\alpha)$ and $dNMC(\alpha, \alpha')$ be the sets defined similarly as for the upward operator. Observe that $uNMC(\alpha) \cup dNMC(\alpha)$ is the set of all possible operators for α.

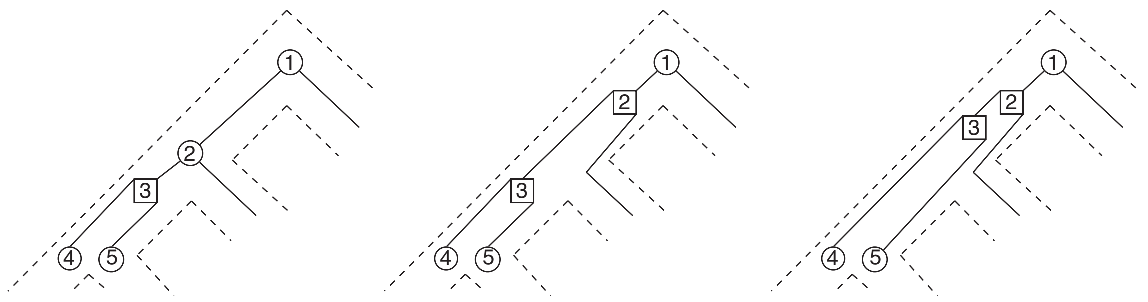

FIG. 2. (**Left**) A section of the reconciliation depicted in Figure 1. Here, the mapping of vertex 2 forbids to move up vertex 3. (**Center**) The vertex 2 changes from a speciation to a duplication by moving it up. (**Right**) Then, vertex 3 can be moved up and still is a duplication.

The lemma below is the first step toward the definition of a combinatorial structure with vertex set $\Psi(G, S)$ (Definition 4 below), where each two reconciliations α and α' are connected by a path of minimal length $D_{NMC}(\alpha, \alpha')$ (Theorem 2 below).

Lemma 2. *Let α and α' be two reconciliations of $\Psi(G, S)$. Then,*

1. *$uNMC(\alpha, \alpha') \cap dNMC(\alpha, \alpha') = \emptyset$;*
2. *$uNMC(\alpha, \alpha') = dNMC(\alpha', \alpha)$ and $dNMC(\alpha, \alpha') = uNMC(\alpha', \alpha)$;*
3. *for any two nodes $u, v \in V(G)$, where $u <_G v$ and $u \in uNMC(\alpha, \alpha')$, if $\alpha(u) \leq_S \alpha(v) \leq_S \alpha'(u)$, then $v \in uNMC(\alpha, \alpha')$.*

Proof. The first two statements are obvious consequences of Definitions 1 (reconciliation) and 3 (operators) and of $uNMC(\alpha, \alpha')$ and $dNMC(\alpha, \alpha')$. For the third statement, Definition 1 implies that $\alpha(u) \leq_S \alpha(v) \leq_S \alpha'(u) \leq_S \alpha'(v)$, and because $u \in uNMC(\alpha, \alpha')$, $\alpha(u) \neq \alpha'(u)$, $\alpha(v) <_S \alpha'(v)$, and then $v \in uNMC(\alpha, \alpha')$ by definition. ■

Theorem 2. *Let α and α' be two reconciliations of $\Psi(G, S)$. There exists a sequence of $D_{NMC}(\alpha, \alpha')$ operators that transforms α into α'. No shorter sequence of operators can transform α into α'.*

Proof. Assume that $\alpha \neq \alpha'$ (otherwise the statement obviously holds). In order to prove that there is a sequence of $D_{NMC}(\alpha, \alpha')$ operators that transforms α into α', we proceed in two steps. First, we prove that $uNMC(\alpha, \alpha') \cup dNMC(\alpha, \alpha') \neq \emptyset$, and Lemma 2.2 implies that it is sufficient to prove, without lost of generality, that $uNMC(\alpha, \alpha') \neq \emptyset$. Second, we prove that there is an "intermediate" reconciliation α'' such that i) there is sequence of upward operators of length $\sum_{u \in uNMC(\alpha, \alpha')} d_u(\alpha(u), \alpha'(u))$ that transforms α into α'', ii) $uNMC(\alpha'', \alpha') = \emptyset$ and $dNMC(\alpha'', \alpha') = dNMC(\alpha, \alpha')$, and iii) for any $u \in uNMC(\alpha, \alpha')(dNMC(\alpha, \alpha'))$, $\alpha''(u) = \alpha'(u)$ (resp. $= \alpha(u)$).

Let u be a vertex of G such that $\alpha(u) \neq \alpha'(u)$. Let $c = \alpha(u)$ and $c' = \alpha'(u)$. Without loss of generality, we can assume that $c <_S c'$ and that $\alpha(p(u)) \neq c$. If $c = M(u)$, let $y = M(u)$ and (x, y) the edge of S such that x is the parent of y. Then, by definition of the upward operator, it is allowed to map u onto (x, y), and then $u \in uNMC(\alpha, \alpha'$. Now assume that $c \neq M(u)$, which implies that $c, c' \in E(S)$. Let $c = (x, y)$ and $c' = (s, t)$. As $c <_S c'$, we have that $x \leq_S t$. If $\alpha(p(u)) \neq x$, then $u \in uNMC(\alpha, \alpha')$. Otherwise, if $\alpha(p(u)) = x$, as $\alpha'(u) \leq_S \alpha'(p(u))$, then $p(u) \in uNMC(\alpha, \alpha')$. Hence, $uNMC(\alpha, \alpha') \neq \emptyset$. Now, let u be the vertex of $uNMC(\alpha, \alpha')$ with the smallest index $id(u)$. The Lemma 2.3 and the definition of this index imply that there is a reconciliation α'' obtained by applying d_u $(\alpha(u), \alpha'(u))$ times the upward operator on u, and only this one, such that $uNMC(\alpha'', \alpha') = uNMC(\alpha, \alpha') \setminus \{u\}$ and $dNMC(\alpha'', \alpha') = dNMC(\alpha, \alpha')$. Because $uNMC(\alpha, \alpha')$ and d_u $(\alpha(u), \alpha'(u))$ are both finite, repeating this process recursively with $\alpha \leftarrow \alpha''$ results in the "intermediate" reconciliation α'' described above.

We can use the same two steps described earlier with the two reconciliations α' and α'' instead of α and α', respectively. Because Lemma 2.2 implies that $uNMC(\alpha', \alpha'') = dNMC(\alpha'', \alpha') = dNMC(\alpha, \alpha')$, the resulting sequence that transforms α' into the "intermediate" reconciliation α'' has a length equals to $\sum_{u \in dNMC(\alpha, \alpha')} d_u$ $(\alpha(u), \alpha'(u))$.

Finally, the concatenation of the two sequences results in a sequence of $D_{NMC}(\alpha, \alpha')$ operators that transforms α into α'. The fact that no shorter sequence exists follows immediately from the definitions of $D_{NMC}(\alpha, \alpha')$ and $uNMC(\alpha, \alpha')$, and the fact that no operator can modify $D_{NMC}(\alpha, \alpha')$ by more than 1. ■

Definition 4. $\mathcal{G}(G, S)$ is the graph with vertex set $\Psi(G, S)$ and where two reconciliations are linked by an edge if and only if they differ by a single NMC.

The following results shows that, although $\Psi(G, S)$ can have an exponential size, NMC operators are sufficient to define a structure on this space of polynomial diameter.

Corollary 1. *The diameter of $\mathcal{G}(G, S)$ is equal to $D_{NMC}(\alpha_{min}, \alpha_{max})$ and is in $O(nm)$.*

Proof. By definition of α_{min} and α_{max} (Definition 2), for every vertex u of $V(G)$, the distance between the mapping of u in these two reconciliations is $|A(u)| - 1$, and is maximal for u as $A(u)$ is the set of all possible cells that can cover u. This implies immediately that the diameter of $\mathcal{G}(G, S)$ is $D_{NMC}(\alpha_{min}, \alpha_{max})$. The fact that this diameter is in $O(nm)$ follows immediately from the fact that for every m vertices u of G, $|A(u)| \in O(n)$. ■

Finally, as our NMC operators are intended to explore the space of reconciliations between a gene tree and a species tree, we address now the issue of updating the classical combinatorial criteria used to evaluate a reconciliation: the following observation implies that they can be easily updated in constant time.

Property 2. Let α and α' be two reconciliations of $\Psi(G, S)$ such that α' is obtained from α by a single upward operator on a vertex $u \in uNMC(\alpha)$. If $\alpha(u) = M(u)$, then $dup(\alpha') = dup(\alpha) + 1$ and $los(\alpha') = los(\alpha) + 2$. Otherwise, $dup(\alpha') = dup(\alpha)$ and $los(\alpha') = los(\alpha) + 1$.

4.2. A spanning tree of $\mathcal{G}(G, S)$

The exploration algorithm described next uses only upward operators and not downward operators, and from now on the term operator refers to the former.

For a vertex $u \in V(G)$, let $id(u)$ be the number of vertices that precede u according to the prefix traversal of G, where the left child u_1 of a vertex $u \in V(G)\backslash L(G)$ is visited before the right child u_2.

Definition 5. For a gene tree G and a species tree S, let $\mathcal{T}(G, S)$ be the ordered tree, with vertex set $\Psi(G, S)$, defined as follows (Fig. 3).

1. The root is the reconciliation α_{min}, and its children are the reconciliations that can be obtained from α_{min} by applying a single operator on a vertex from $uNMC(\alpha_{min})$.
2. Given a reconciliation α, that differs from its parent by an operator on a vertex $u_i \in V(G)$, its children are the reconciliations that can be obtained from α by applying a single operator on a vertex $u_j \in uNMC(\alpha)$ such that $id(u_i) \leq id(u_j)$.
3. Consider a reconciliation α and two of its children α_i and α_j, respectively, obtained by an operator on the vertices u_i and u_j from $uNMC(\alpha)$. α_i precedes α_j in the ordered children of α if and only if $id(u_i) < id(u_j)$.

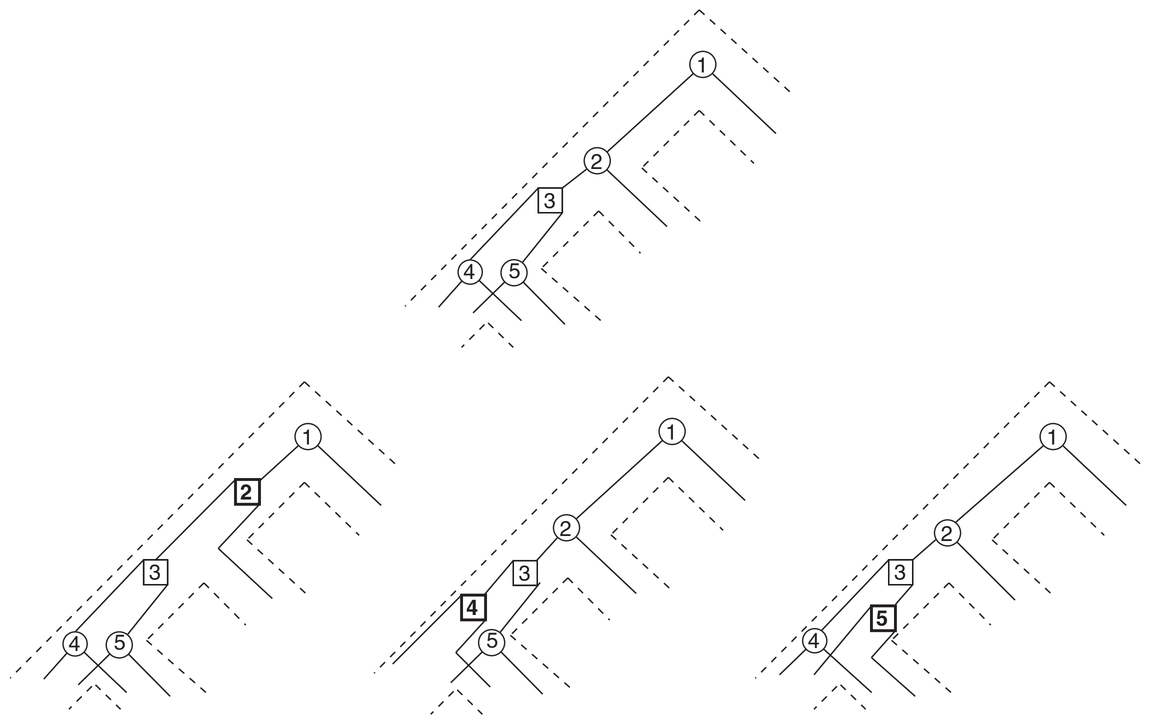

FIG. 3. The subtree of $\mathcal{T}(G, S)$ rooted at α_{min} for the trees G and S depicted in Figure 1. α_{min} and its children respectively are at the top and bottom of the figure. For each child, the vertex that has been moved upward is in boldface.

We now introduce properties that will be used to prove that $\mathcal{T}(G,S)$ is a spanning tree of $\mathcal{G}(G,S)$ (see Proposition 3 below). The first property (Property 3 below) follows immediately from Definitions 3 (operators) and 5 ($\mathcal{T}(G,S)$). The second one (Property 4) follows from Definition 5 and Property 3.

Property 3. Let α and α' be two reconciliations of $\mathcal{T}(G,S)$, where the latter is a child of the former obtained by an operator on a vertex $u \in uNMC(\alpha)$. Any reconciliation α'' in the subtree of $\mathcal{T}(G,S)$ rooted at α' is such that $\alpha'(u) \leq_S \alpha''(u)$.

Property 4. Let a vertex of $\mathcal{T}(G,S)$ labeled by the reconciliation α. Consider two children α_i and α_j of α, respectively, obtained by the operator on the vertices u_i and $u_j \in uNMC(\alpha)$, where $id(u_i) < id(u_j)$. Then,

1. for any reconciliation α' in the subtree of $\mathcal{T}(G,S)$ rooted at α_j, $\alpha(u_i) = \alpha'(u_i)$; and
2. the two subtrees of $\mathcal{T}(G,S)$, respectively rooted at α_i and α_j, are disjoints.

Proposition 3. *$\mathcal{T}(G,S)$ is a spanning tree of $\mathcal{G}(G,S)$.*

Proof. $\mathcal{T}(G,S)$ is a tree by definition. In order to prove it is a spanning tree of $\mathcal{G}(G,S)$, we only need to prove that every reconciliation $\alpha \in \Psi(G,S)$ appears once and exactly once as a vertex of $\mathcal{T}(G,S)$.

First, if $uNMC(\alpha_{min}, \alpha) = \emptyset$, $\alpha = \alpha_{min}$ and then α is in $\mathcal{T}(G,S)$ by definition. Otherwise, let $u \in uNMC(\alpha_{min}, \alpha)$ with the smallest index $id(u)$. By definition, there is a reconciliation α', obtained by applying d_u (α_{min} (u), $\alpha(u)$) times this operator, that is in the subtree of $\mathcal{T}(G,S)$ rooted at α_{min}, and such that $\alpha'(u) = \alpha(u)$. Then, $u \notin uNMC(\alpha', \alpha)$, and for any $u' \in uNMC(\alpha', \alpha), id(u') > id(u)$. Because both d_u (α_{min} (u), $\alpha(u)$) and $uNMC(\alpha_{min}, \alpha)$ are finite, repeating this recursion with $\alpha_{min} \leftarrow \alpha'$ ends at a reconciliation α' as a vertex of this subtree and then of $\mathcal{T}(G,S)$ such that $uNMC(\alpha', \alpha) = \emptyset$ and then $\alpha' = \alpha$.
Now assume that the reconciliation α labels two vertices of $\mathcal{T}(G,S)$. By definition, these vertices are not comparable in $\mathcal{T}(G,S)$, and this is in contradiction with Property 4.2. ■

4.3. Preliminaries to the algorithm

For a reconciliation α of $\mathcal{T}(G,S)$, we denote by $P(\alpha) \subseteq uNMC(\alpha)$ the list of allowed operators that can be applied to obtain the children of α, where the vertices are ordered according to the increasing value of their indexes *id*. Assuming that the children of α are visited in the order described in the Definition 5, it follows immediately that the efficient traversal of $\mathcal{T}(G,S)$ reduces to the following problem: For a reconciliation α' that is a child of α, how can the list $P(\alpha')$ be computed in constant time given the list $P(\alpha)$?

The solution to this problem is based on the two properties below (Properties 5 and 6), which both are obvious consequences of Definitions 3 (operators) and 5 $\mathcal{T}(G,S)$.

Property 5. Let α and α' be two reconciliations of $\mathcal{T}(G,S)$ such that α' is the first child of α and is obtained by an operator on the first vertex of $P(\alpha)$, noted u. Then, the difference between $P(\alpha)$ and $P(\alpha')$ implies a constant number of vertices and is such that

1. $P(\alpha') \backslash P(\alpha) \subseteq \{u_1, u_2\}$;
2. and $P(\alpha) \backslash P(\alpha') \subseteq \{u\}$.

Property 6. Let α be a reconciliation of $\mathcal{T}(G,S)$, and consider two of its children α' and α'' respectively obtained by an operator on the vertices u' and u'' from $P(\alpha)$. Suppose that α'' (u'') is the child (resp. vertex) of α (resp. $P(\alpha)$) immediately before α' (resp. u'). Then, the difference between $P(\alpha')$ and $P(\alpha'')$ implies a constant number of vertices and is such that

1. $P(\alpha') \backslash P(\alpha'') \subseteq \{u'_1, u'_2\}$;
2. $P(\alpha'') \backslash P(\alpha') \subseteq \{u', u'', u''_1, u''_2\}$ and $u'' \notin P(\alpha')$;
3. and u' may or may not be in $P(\alpha')$.

Let α be a reconciliation of $\mathcal{T}(G,S)$, and consider any vertex u of $P(\alpha)$. For the sake of clarity, we suppose in the following proposition that the insertion (or removal) of any of the three vertices u, u_1, or u_2 into (resp. from) the list can be done in constant time.

Proposition 4. *Let α be a reconciliation of $\mathcal{T}(G,S)$. Given the list $P(\alpha)$, if the children of α are visited in the order described in the Definition 5, the list $P(\alpha')$ can be computed in constant time for any reconciliation α' that is a child of α.*

Proof. The proof is by recurrence on the i-th child of α. If α' is the first child of α, the desired result follows directly from Property 5. Otherwise, assume that α'' is the child of α immediately before α' and that $P(\alpha'')$ is known. According to Property 6, $P(\alpha')$ can be computed in constant time given $P(\alpha'')$. ■

Let α be a reconciliation of $\mathcal{T}(G,S)$ and α' be one of its children that is obtained by an operator on a vertex $u \in P(\alpha)$. In the context of Proposition 4, the possible removal (insertion) of u (resp. u_1) from (resp. into) the list can obviously be done in constant time (resp. because $id(u_1) = id(u) + 1$). We now explain how to compute in constant time the position of u_2 in the list $P(\alpha')$ when u_2 is not in $P(\alpha)$. The main problem is that the number of vertices in $P(\alpha')$ that are between $id(u)$ and $id(u_2)$ is not constant. In this perspective, we have to implement the list in a particular way, which is formally described in the next section. Below, we formulate two lemmas that give a first intuition for the implementation and that are used to prove the completeness and the complexity of the algorithm that explores $\mathcal{T}(G,S)$.

Lemma 3. *Let α be a reconciliation of $\mathcal{T}(G,S)$, u be the first vertex of $P(\alpha)$, and suppose that $u_2 \notin P(\alpha)$. Then, for any $w \in P(\alpha)$ such that $w \notin P(\alpha_{min}, w \neq u$, and w is the second child of $f(w)$, $id(u_2) < id(w)$.*

Proof. Suppose that $id(u_2) > id(w)$. There is two possibilities. First, if $id(w) < id(u)$, this is in contradiction with the definition of $P(\alpha)$. Otherwise, $id(u) < id(u_1) < id(w) < id(u_2)$, and then w is in the subtree G_{u1}. However, because we assume that $w \notin P(\alpha_{min})$, there is a reconciliation α' that is along the path of $\mathcal{T}(G,S)$ that connects α_{min} and α such that $w \in P(\alpha')$. Then, $f(w)$ is used (at least one time) by an operator along this path, this operator precedes the one(s) on the vertex u, and this is in contradiction with the definition of $\mathcal{T}(G,S)$. ■

Lemma 4. *For any reconciliation α of $\mathcal{T}(G,S)$, there is a sequence of reconciliations that starts with α, ends with a reconciliation α_n that is in the subtree of $\mathcal{T}(G,S)$ rooted at α, and that respects the next three constraints. First, $P(\alpha) \cap P(\alpha_n) = \emptyset$. Second, each reconciliation of the sequence, except the first one (α), is the first child of the previous one. Third, each vertex of $P(\alpha)$ is the first vertex of $P(\alpha')$, for at least one reconciliation α' of this sequence.*

Proof. Let $u \in P(\alpha)$ be the first vertex of $P(\alpha)$ and α'_1 be the first child of α obtained by the operator on u. Because this operator can be repeated a limited number of times, each time defining the first child of the previous reconciliation, we obtain a finite sequence of reconciliations where the last one, noted α_1, is such that $P(\alpha) \backslash P(\alpha_1) = \{u\}$ (see Property 5). According to the definition of $\mathcal{T}(G,S)$, for any reconciliation α'' that is in the subtree of $\mathcal{T}(G,S)$ rooted at α_1, u is not in $P(\alpha'')$. Hence, because $V(G)$ is finite, by repeating recursively the two steps described above induces a sequence of reconciliations that ends at α_n and respects the desired constraints. ■

4.4. Algorithm for the prefix traversal of $\mathcal{T}(G,S)$

We now give a complete description of an algorithm that exhaustively explores $\Psi(G,S)$ in time $\Theta(|\Psi(G,S)|)$ by a prefix traversal of the spanning tree $\mathcal{T}(G,S)$.

In the context of the prefix traversal of $\mathcal{T}(G,S)$, where a child α' of a reconciliation α is visited and is obtained by an operator on $u \in P(\alpha)$, recall that the difficulty to compute $P(\alpha')$ given $P(\alpha)$ comes from the (possible) insertion of u_2 into the list. We describe below how the list $P(\alpha)$ can be implemented to efficiently perform this insertion.

Definition 6. Let α be a reconciliation of $\mathcal{T}(G,S)$. The list $P(\alpha)$ is implemented on two sublists of $V(G)$ noted P and S and such that (i) $P \cap S = \emptyset$, (ii) any vertex w of S is the right child of $f(w)$ and is not in $P(\alpha_{min})$, (iii) $P(\alpha) = \{w \in P : id(w) \geq id(v)\} \cup S$, where v is the first vertex of $P(\alpha)$, and (iv) v is in P.

The recursion starts at the root of $\mathcal{T}(G,S)$ with the reconciliation $\alpha = \alpha_{min}$ and the ordered lists $P = P(\alpha_{min})$, that are computed during a preprocessing phase, and v as the first vertex of P and $S = \emptyset$. For a vertex $u \in V(G)$ and a cell $c \in A(u)$, recall that $f(c)$ is its ancestor cell in $A(u)$.

Theorem 3. *Algorithm 2 visits all reconciliations of $\Psi(G, S)$. Given α_{min} and $P = P(\alpha_{min})$, it can be implemented to run in time $\Theta(|\Psi(G, S)|)$ and space $O(nm)$.*

Proof. We first address the correctness of the algorithm. We prove that the algorithm visits all reconciliations of $\Psi(G, S)$ using Proposition 3, and then that it performs a prefix traversal of $\mathcal{T}(G, S)$. In this perspective, we need to prove that the children of a given reconciliation are visited according to the order described in the definition of $\mathcal{T}(G, S)$ (Definition 5). Formally, for each recursion of the algorithm, where α is the current reconciliation, we have to prove that $P(\alpha)$ is consistent (that is with the implementation of Definition 6).

Algorithm 2 Exhaustive exploration algorithm of the space $\psi(G, S)$.

```
1:  RecurExplore ($v$)
2:    $u \leftarrow v$
3:    while $u \neq end(P)$ do
4:      $\alpha(u) \leftarrow f(\alpha(u))$
5:      If $u_1 \notin P$ and $u_1 \notin L(G)$, then insert $u_1$ immediately after $u$ in $P$
6:      If $u_2 \notin P \cup S$ and $u_s \notin L(G)$, then insert $u_2$ at the front of $S$.
7:      if $u$ is a valid uNMC for $\alpha$ then
8:        RecurExplore ($u$)
9:      else
10:       Let $x$ be the vertex immediately after $u$ in $P$ and $y$ be the first one of $S$
11:       if $id(x) > id(y)$ then
12:         Insert $y$ immediately before $x$ in $P$ and remove $y$ from $S$
13:       Let $v'$ be the vertex immediately after $u$ in $P$
14:       RecurExplore ($v'$)
15:     Undo lines 4, 5, and 6. If line 6 is undone, the removal is done on $P$ instead of $S$.
16:     Let $u$ be the vertex immediately after $u$ in $P$
```

By construction, $P(\alpha_{min})$ is consistent, where $P = P(\alpha_{min})$ and $S = \emptyset$. Suppose that this is true for a recursion with $\alpha \in \mathcal{T}(G, S)$ as the current reconciliation and a vertex $v \in V(G)$ as the parameter. We then have to prove that the children of α are visited according to the usual order and that for any child α' of α, $P(\alpha')$ is consistent. Let u'' (u') be the vertex u considered at line 3 during the first (resp. second) pass of the loop and α'' (resp. α') be the new reconciliation defined at line 4.

Because of the hypothesis, $u'' = v$ is the first vertex of $P(\alpha)$ and then α'' is the first child of α according to the definition of $\mathcal{T}(G, S)$. According to Property 5, $P(\alpha)$ and $P(\alpha'')$ differ by at most three vertices, that are u'', u''_1 and u''_2. The required insertions of u''_1 and u''_2 are, respectively, done in lines 5 and 6. According to Lemma 3, any vertex w of S is such that $id(u''_2) < id(w)$, the usual order of the vertices of S is then conserved after the insertion of u''_2 at the front of this list. If u'' is a valid operator for the new reconciliation α'', u'' stays in $P(\alpha'')$. Otherwise, the removal of u'' from $P(\alpha'')$ is the consequence of line 13. In both case, $P(\alpha'')$ is consistent.

Moreover, by recursively applying the case of the first child described above, with α'' as the considered reconciliation instead of α, the Lemma 4 implies that each vertex $w \in P(\alpha'')$ is used at least one time as the parameter during this recursion. Then, if w was in S before the recursion on α'', w is moved from S into P at line 12 and is not moved back into S afterward. Hence, when all the children of α'' are visited, S is empty, all its vertices are in P, and the fact that $P(\alpha'')$ is consistent results from each time line 15 is performed.

Recall that α'' is the first child of α, and that $P(\alpha'')$ is consistent and S is empty immediately after the recursive call on α'' is completed (line 14). We now prove that the second child of α is α' and that $P(\alpha')$ is consistent. Remember the difference between the lists $P(\alpha'')$ and $P(\alpha')$ formally described in Property 6. First, the possible removals of the vertices u''_1 and u''_2 from the list are done in line 15. Afterward, $P(\alpha)$ is consistent and then, because u'' is the first vertex of $P(\alpha)$, the consequences of lines 16 and 4, respectively, are that u' is the second vertex of $P(\alpha)$ and that α' is the second child of α. The removal of u'' from the list follows immediately (see Property 6 and Definition 6). Second, the possible insertions of u'_1 and u'_2 are, respectively, done in lines 5 and 6, as with the previous child α'' of α, where S is in the usual order because of its emptiness before the insertion of u'_2. The possible removal of u' is similar to the one of u'' described above. Hence, α' is the second child of α and $P(\alpha')$ is consistent.

By recurrence on the i-th child of α, we can conclude that the children of α in $\mathcal{T}(G,S)$ are visited according to the usual order, that the algorithm performs a prefix traversal of $\mathcal{T}(G,S)$ and that all the reconciliations of $\Psi(G,S)$ are visited.

We now address the complexity of the algorithm. Because it performs a prefix traversal of $\mathcal{T}(G,S)$, its time complexity is in $\Theta(|\mathcal{T}(G,S)|q)$, where q is the time needed for any loop of RecurExplore. With a constant number of local variables for line 15, any loop can be done in constant time, that is without considering the recursive call. We can then conclude that the time complexity of the algorithm is in $\Theta(|\Psi(\mathrm{G, S})|)$.

The space complexity is first defined by the total size of both lists P and S, which is in $O(m)$, where $m=|V(G)|$. Second, for each recursive call, there is a constant number of local variables, and the maximal depth of $\mathcal{T}(G,S)$ is the diameter of $\mathcal{G}(G,S)$, which is in $O(nm)$ (see Corollary 1). We can then conclude that the space complexity of the algorithm is in $O(nm)$. ■

Together with Property 2, that implies that updating the number of duplications and/or losses after a single operator can be done in constant time, this algorithm allows to compute efficiently the exact distribution of the duplication, loss and mutation costs in optimal time $\Theta(|\Psi(G,S)|)$ (see Section 5).

4.5. Exhaustive exploration of sub-optimal reconciliations

It is often interesting to consider not all reconciliations between a gene tree and a species tree, but only a subset, whose cost is close to the optimal cost, or more generally bounded. In Proposition 2, we described how to count the number of such reconciliations, for the duplication cost, that is $|\Psi_{dup}(G,S,\delta)|$. Here, we rely on a monotony property (Lemma 5 below) of path in $\mathcal{T}(G,S)$ to adapt Algorithm 2 in order to explore the space of sub-optimal reconciliations for any of the three cost models. Let $\Psi_{cost}(G,S,\delta)=\{\alpha\in\Psi(G,S):cost(\alpha)\leq\delta\}$ be the considered set of reconciliations, where δ is the given bound and *cost* is one of the three usual cost models. Observe that the set $\Psi_{dup}(G,S,\delta)$ is a special case of this problem.

Lemma 5. *Let α and α' be two reconciliations of $\mathcal{T}(G,S)$, where the latter is a child of the former obtained by an operator on a vertex $u\in uNMC(\alpha)$. Any reconciliation α'' in the subtree of $\mathcal{T}(G,S)$ rooted at α' is such that $cost(\alpha'')\geq cost(\alpha)$, for any of the three cost models.*

Proof. Obvious consequence of the definition of $\mathcal{T}(G,S)$ (Definition 5), Property 2, and Property 3. ■

Because Property 1.1 says that α_{min} minimizes the three cost models, the considered bound is such that $\delta\geq cost(\alpha_{min})$. Let $\mathcal{G}_{cost}(G,S,\delta)$ be the subgraph of $\mathcal{G}(G,S)$ where each of its reconciliation is from $\Psi_{cost}(G,S,\delta)$. Because Algorithm 2 is based on the spanning tree $\mathcal{T}(G,S)$ to perform the exploration of $\Psi(G,S)$, we define below a similar combinatorial structure to explore $\Psi_{cost}(G,S,\delta)$.

Definition 7. *For a gene tree G, a species tree S, a cost model, and a bound $\delta\geq cost(\alpha_{min})$, let $\mathcal{T}_{cost}(G,S,\delta)$ be the subtree of $\mathcal{T}(G,S)$ rooted at α_{min} and defined as follows: any child α' of α_{min} in $\mathcal{T}(G,S)$ is also a child of α_{min} in $\mathcal{T}_{cost}(G,S,\delta)$ if and only if $cost(\alpha')\leq\delta$. The same rule is recursively applied to define the children of these vertices and then the whole structure of the tree.*

Proposition 5. *$\mathcal{T}_{cost}(G,S,\delta)$ is a spanning tree of $\mathcal{G}_{cost}(G,S,\delta)$.*

Proof. Obvious consequence of Property 1.1, Definition 7, and Lemma 5. ■

For a reconciliation $\alpha\in\mathcal{T}_{cost}(G,S,\delta)$, we denote by $P_{cost}(\alpha)$ the sublist of $P(\alpha)$ (see Section 4.3) that contains the allowed operators that can be applied to obtain the children of α in $\mathcal{T}_{cost}(G,S,\delta)$. We then use this list, instead of $P(\alpha)$, in the Algorithm 2 so that it explores the whole space $\Psi_{cost}(G,S,\delta)$.

Theorem 4. *Algorithm 2 can be adapted to visit all reconciliations of $\Psi_{cost}(G,S,\delta)$. Given α_{min} and $P=P_{cost}(\alpha_{min})$, it can be implemented to run in time $\Theta(|\Psi_{cost}(G,S,\delta)|)$ and space $O(nm)$.*

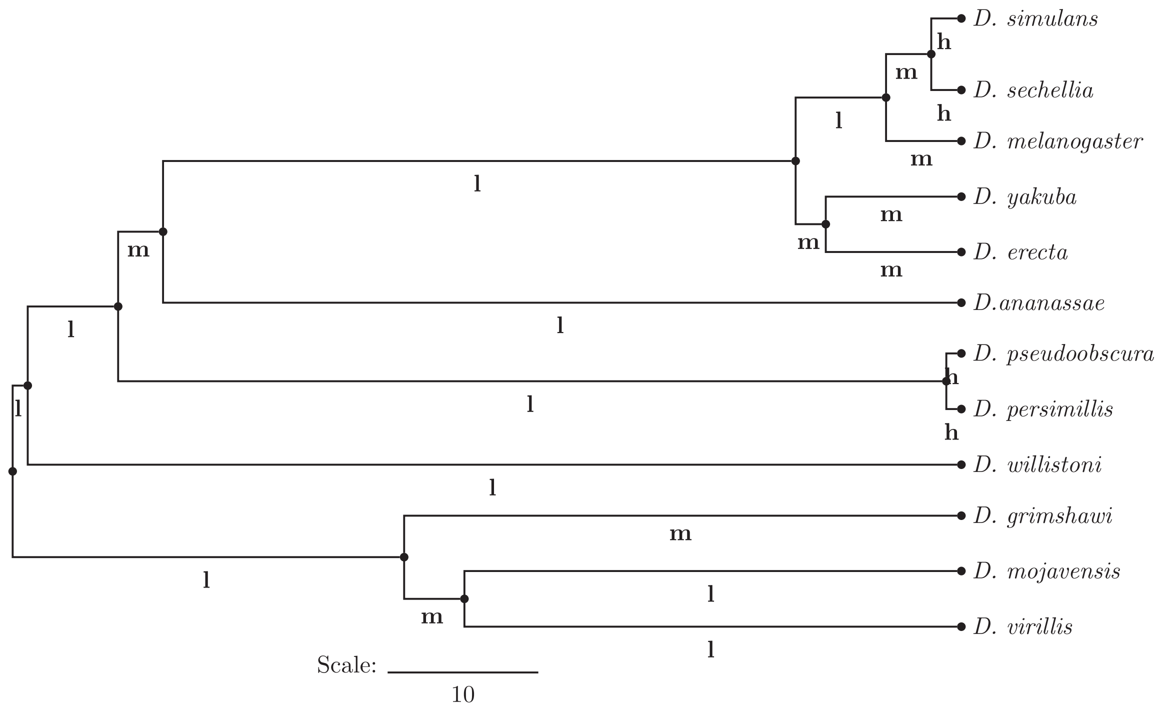

FIG. 4. Species tree for the Drosophila group (Figure 1 in Hahn et al., 2007b), where divergence time is in Million Years and the gene duplication/loss rate for each branch is as follows: h, high rate (0.0193 gene/MY); m, medium rate (0.0022 gene/MY); and l, low rate (0.0006 gene/MY).

Proof. From the definition of $\mathcal{G}_{cost}(G, S, \delta)$ and the Proposition 5, the set of reconciliations of $\Psi_{cost}(G, S, \delta)$ is equal to the one of $\mathcal{T}_{cost}(G, S, \delta)$. Because $\mathcal{T}_{cost}(G, S, \delta)$ is a prefix tree of $\mathcal{T}(G, S)$, the Algorithm 2 can be adapted to explore $\mathcal{T}_{cost}(G, S, \delta)$ with the modifications described below.

In the context of the Algorithm 2, let α, α', and α'', respectively, be the current reconciliation, the one defined by the operator on the vertex $u \in P_{cost}(\alpha)$ on line 4, and the one obtained by a second operator on the vertex u_1. Recall that because of Property 2, $cost(\alpha')$ and $cost(\alpha'')$ can be computed in constant time given $cost(\alpha)$. In line 5, if u_1 is inserted into P, then $cost(\alpha'') \leq \delta$. Moreover, if u_1 is already in P and $cost(\alpha'') > \delta$, then u_1 is removed from P. For the case of the vertex u_2 and the lists P and S, the conditions on its insertion or removal are similar as for u_1.
The correctness and the time and space complexities come from the Theorem 3 and the fact that $\mathcal{T}_{cost}(G, S, \delta)$ is a prefix tree of $\mathcal{T}(G, S)$. ■

5. EXPERIMENTAL RESULTS

Based on two known species trees, we simulated gene family evolutionary scenarios,[2] which resulted in realistic gene trees, and studied the reconciliation spaces and the effectiveness of parsimonious models to retrieve the true reconciliation (the one that corresponds to the real evolutionary scenario). The first phylogeny (Fig. 4) has 12 *Drosophila* species (Figure 1 in Hahn et al., 2007b), and the second one (Fig. 5) includes only 6 mammalian species (Figure 1 in Hahn et al., 2007a). In the smallest phylogeny, the clade of the 3 primates has high duplication and loss rates according to the rest of the tree, and we suspected that

[2]This is done using the birth-and-death process (Kendall, 1948) along the considered species tree, where each branch has its own length (in time) and gene duplication and loss rates.

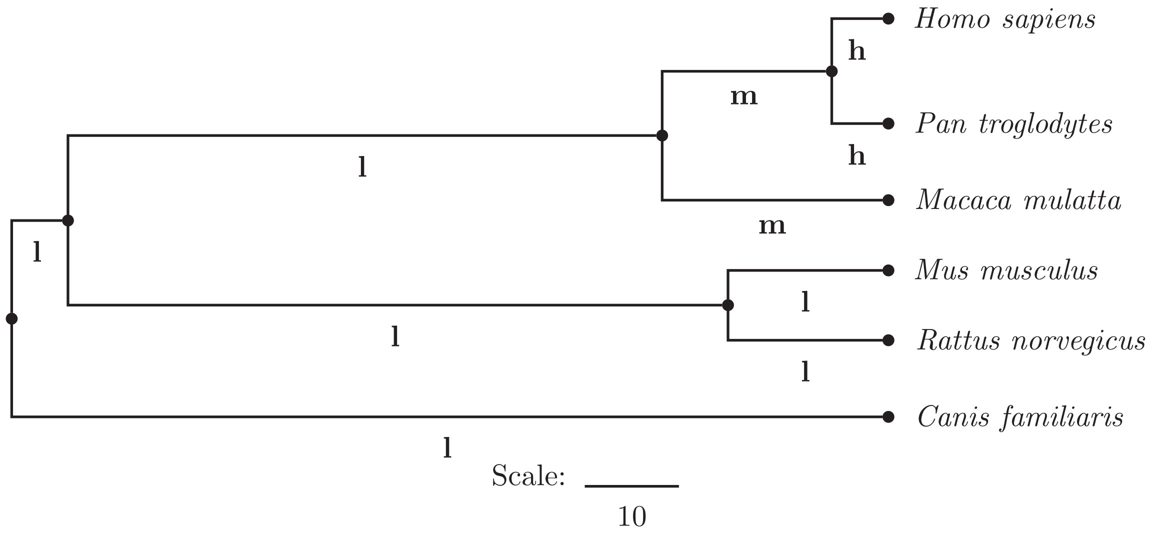

FIG. 5. Species tree for the mammalian group (Figure 1 in Hahn et al., 2007a), where divergence time is in Million Years and the gene duplication/loss rate for each branch is as follows: h, high rate (0.0039 gene/MY); m, medium rate (0.0024 gene/MY); and l, low rate (0.0014 gene/MY).

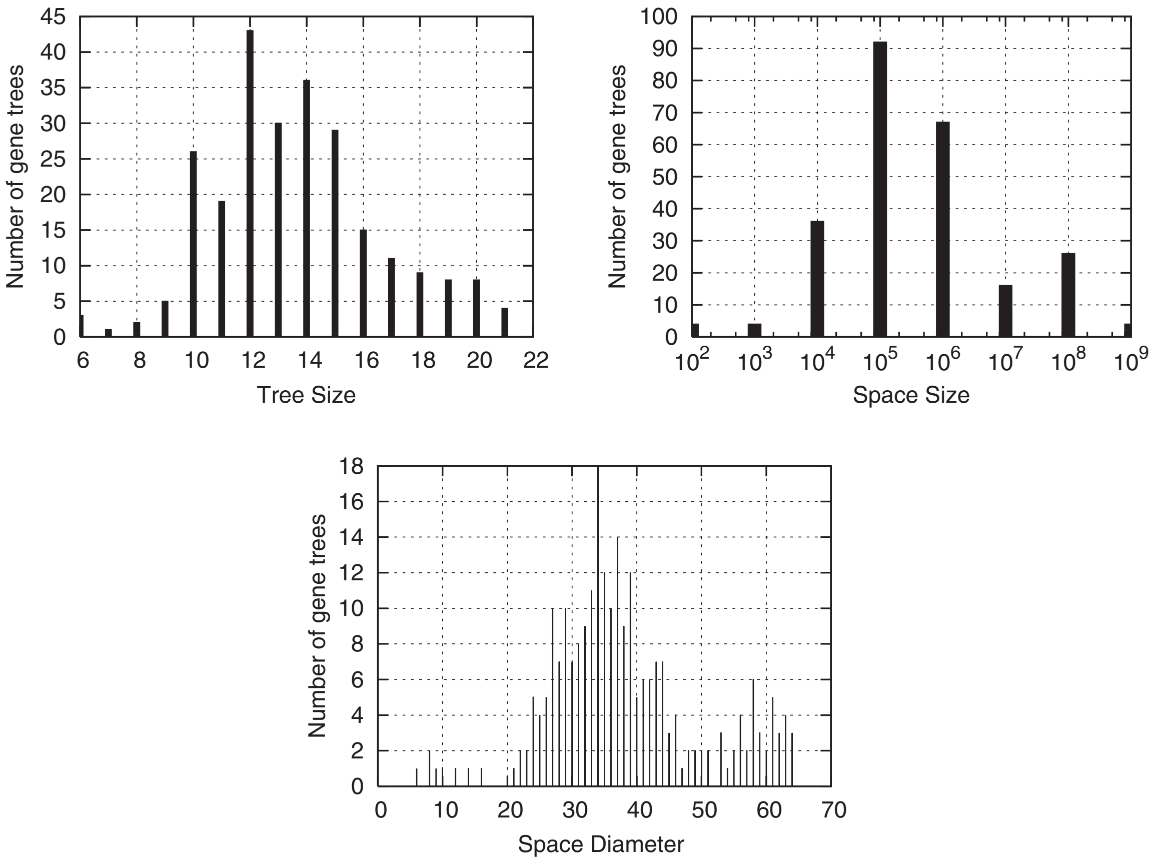

FIG. 6. Distribution of the 249 gene trees according to their number of leaves (**above left**); the reconciliation space size (**above right**), where a gene tree G is counted in the bar 10^i iff $10^{i-1} \leq |\psi(G, S)| < 10^i$; and the diameter of $\mathcal{G}(G, S)$ (**below**), with an average diameter of 34.142.

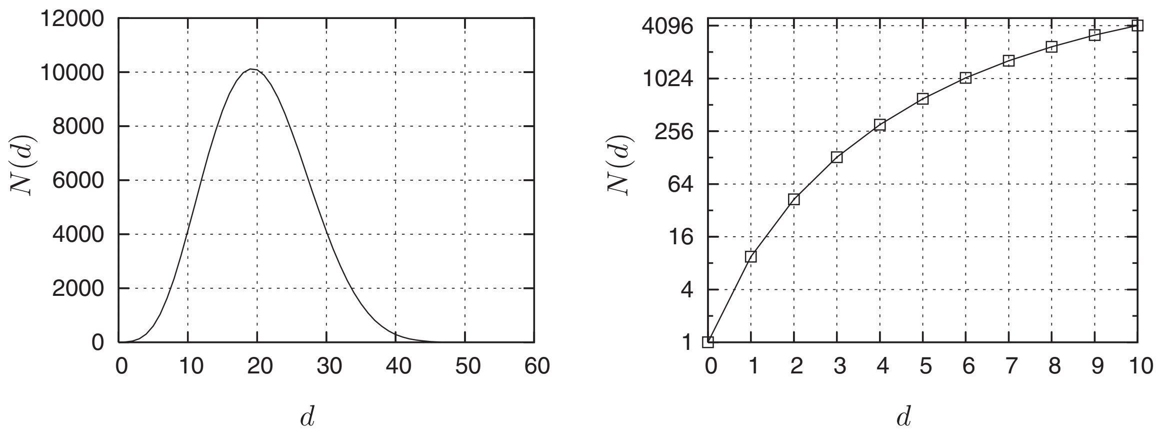

FIG. 7. Over all 249 gene trees, average distribution of the number $N(d)$ of reconciliations α such that D_{NMC} $(\alpha_{min}, \alpha) = d$, where $0 \leq d \leq D_{NMC}(\alpha_{min}, \alpha_{max})$ (i.e., the diameter).

this may generate evolutionary scenarios relatively far from the parsimonious ones. However, the results of the two species trees are similar according to both the "average shape" of the reconciliation spaces and the performance of the LCA reconciliation to retrieve the real evolutionary scenarios. Section 5.1 presents the results of the Drosophila gene families, and Section 5.2 briefly summarizes the ones of the mammals.

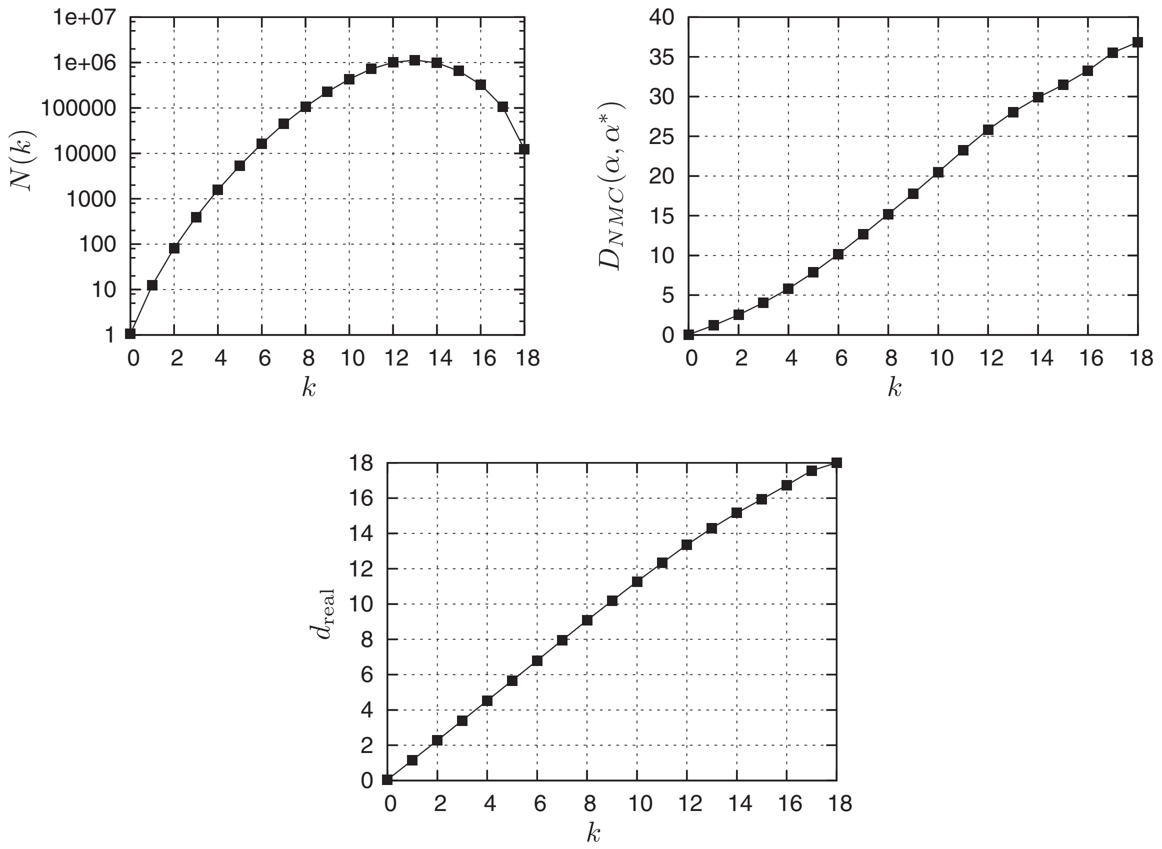

FIG. 8. (**Above left**) Over all 249 gene trees, average distribution of the number $N(k)$ of reconciliations α such that $d_{\text{cost}}(\alpha) = dup(\alpha) - dup(\alpha^*) = k$, for $k \in \mathbb{N}$. α^* is a global minimum that minimizes the NMC distance D_{NMC} (α, α^*). (**Above right, resp. below**) The same distribution with the average value of D_{NMC} (α, α^*) (resp. d_{real}) for all reconciliations $\alpha \in \Psi(G, S)$ such that $d_{\text{cost}}(\alpha) = k$, for a given $k \in \mathbb{N}$. The real distance d_{real} is the number of nodes u of G such that $\alpha(u) \neq \alpha_{\text{real}}(u)$.

5.1. Drosophila group

Along the species tree S of the 12 Drosophila (Fig. 4), we generated 1000 synthetic gene trees and obtained 249 unique ones after removing multiple copies. Figure 6 describes the distribution of the size of each gene tree G, the cardinality and the diameter of the reconciliation space $\mathcal{G}(G, S)$. We also computed the average diameter over all the 249 reconciliation spaces, which is $\mu_{diam} = 34.142$.

The first experimental concern that we address here is as follows: what is the average shape of the 249 reconciliation spaces according solely to the NMC operators? Recall that α_{min} and α_{max} are located on the border of $\mathcal{G}(G, S)$ and their NMC distance is the diameter of this space (Corollary 1). Figure 7 plots the average number $N(d)$ of reconciliations that are at a given NMC distance d to α_{min}, and we can easily observe (left plot) that $N(d)$ is inversely proportional to $|d - \mu_{diam}/2|$. This suggests that the two regions of $\mathcal{G}(G, S)$ with the lowest concentration of reconciliations are located around α_{min} and α_{max} and the highest concentrated region is equidistant to these two reconciliations. Although this implies that there is relatively (according to the size of the space) few (sub-)optimal reconciliations different from α_{min}, we can see (right plot) that their number is non-negligible. This justifies the use of our exploration algorithm to visit other parsimonious reconciliations and to consider them as an alternative for the real evolutionary scenario.

For each of the 249 unique gene trees, we used Algorithm 2 to explore the whole space $\Psi(G, S)$ focusing on the duplication cost (for the loss and mutation criteria, the results are similar). For the duplication criterion, 237 gene trees have a unique global minimum, and 12 have two. In each of these 12 cases, the NMC distance between the two global minimums is one. Over all the 249 gene trees, the LCA reconciliation α_{min}, that is a

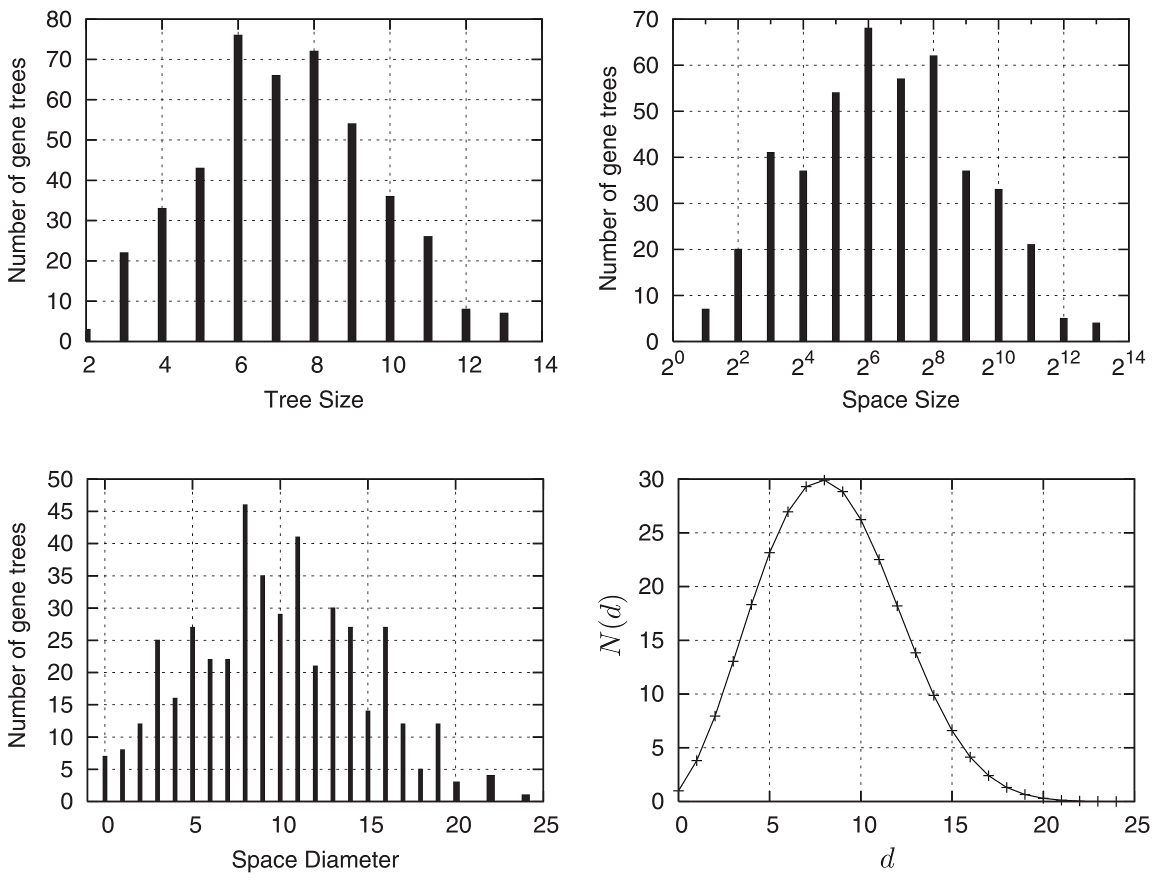

FIG. 9. Distribution of the 446 gene trees: (**Above left**) Their number of leaves. (**Above right**) The reconciliation space size, where a gene tree G is counted in the bar 2^i if $2^{i-1} \leq |\psi\ (G, S)| < 2^i$. (**Below left**) The diameter of $\mathcal{G}(G, S)$. (**Below right**) Over all 446 gene trees, average distribution of the number $N(d)$ of reconciliations α such that D_{NMC} $(\alpha_{min}, \alpha) = d$, where $0 \leq d \leq D_{NMC}(\alpha_{min}, \alpha_{max})$ (i.e., the diameter). Over all 446 gene trees, the average standard deviation of $D_{NMC}(\alpha_{min}, \alpha)$ to the mean $D_{NMC}(\alpha_{min}, \alpha_{max})/2$ is 1.7263.

global minimum, is either identical or, in the worst case, at a distance of a single NMC to the true evolutionary scenario induced by the birth-and-death and noted α_{real}.

For a reconciliation $\alpha \in \Psi(G,S)$, let $d_{\text{cost}}(\alpha) = dup(\alpha) - dup(\alpha^*)$, where α^* is a global minimum (for the duplication cost) that minimizes $D_{NMC}(\alpha, \alpha^*)$. We denote by $N(k)$ the number of reconciliations $\alpha \in \Psi(G,S)$ such that $d_{\text{cost}}(\alpha) = k$, for a given $k \in \mathbb{N}$. Figure 8 (left) shows that, on average over all gene trees, $N(k)$ is proportional to k from $k=0$ to $k=13$ and inversely proportional from $k=13$ to $k=18$. This can be explained by the following facts: the maximum value of d_{cost} is equal to the number of internal nodes u of G that can be mapped on $M(u)$, and the average number of such nodes is 13. All this suggests that, for a given gene tree, $N(k)$ is maximized at this maximum value of $d_{\text{cost}} = k$.

We analyzed the relationship between the NMC and cost distances using the average value of $D_{NMC}(\alpha, \alpha^*)$ over all gene trees G and all reconciliations $\alpha \in \Psi(G,S)$ such that $d_{\text{cost}}(\alpha) = k$, for a given $k \in \mathbb{N}$. We also computed the number of nodes $u \in V(G)$ such that $\alpha(u) \neq \alpha_{\text{real}}(u)$. We observe that the cost distance of a reconciliation α is proportional both to the NMC distance with the closest optimal reconciliation α^* (Fig. 8, center) and to how much α differs from the real reconciliation α_{real} (Fig. 8, right).

5.2. Mammalian group

We generated 10000 gene trees along the species tree of the mammals (Fig. 5), and obtained 9823 non-empty ones and 446 gene trees after removing multiple copies. As with the Drosophila gene families, over all the 446 gene trees, α_{min} is either identical or at a distance of a single NMC to α_{real}. Moreover, the number of global minimums for the duplication cost ranges from 1 to 5, where 374 gene trees have a single

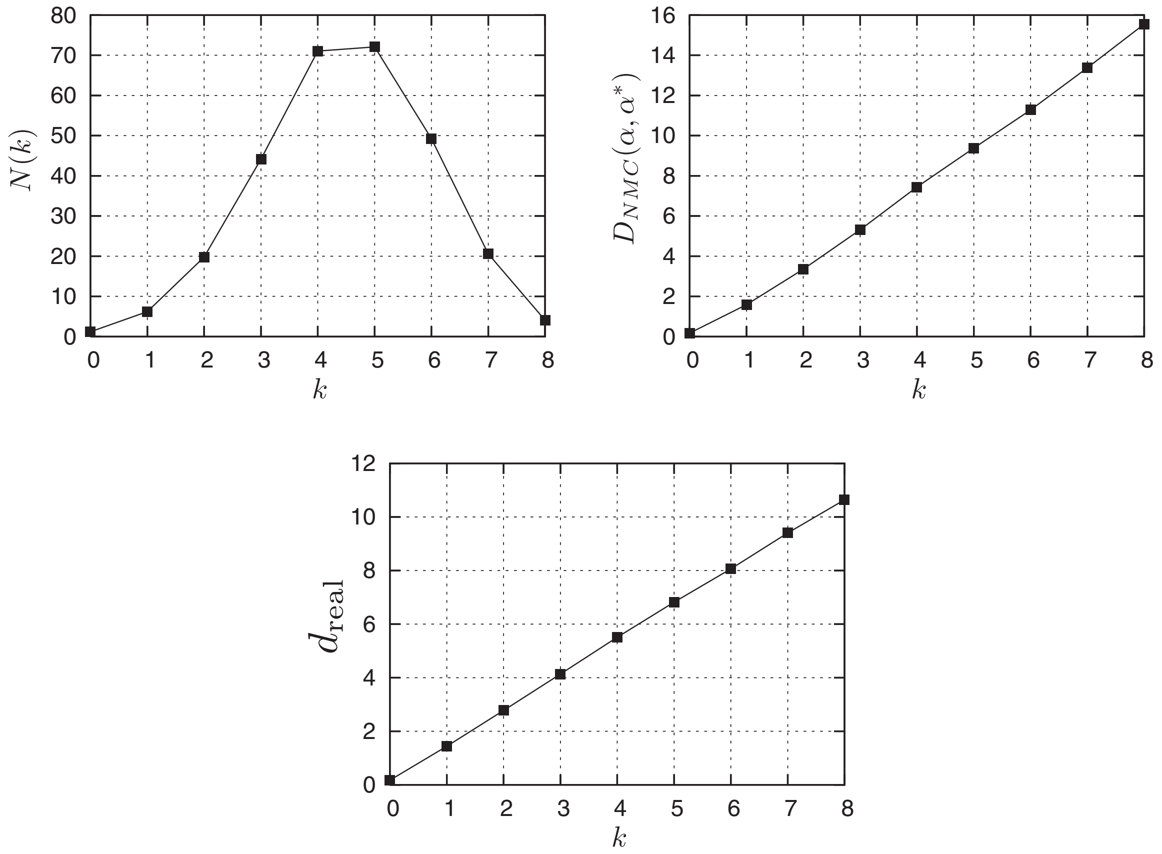

FIG. 10. (**Above left**) Over all 446 gene trees, average distribution of the number $N(k)$ of reconciliations α such that $d_{\text{cost}}(\alpha) = dup(\alpha) - dup(\alpha^*) = k$, for $k \in \mathbb{N}$. α^* is a global minimum that minimizes the NMC distance $D_{NMC}(\alpha, \alpha^*)$. (**Above right, resp. below**) The same distribution with the average value of $D_{NMC}(\alpha, \alpha^*)$ (resp. d_{real}) for all reconciliations $\alpha \in \Psi(G,S)$ such that $d_{\text{cost}}(\alpha) = k$, for a given $k \in \mathbb{N}$. The real distance d_{real} is the number of nodes u of G such that $\alpha(u) \neq \alpha_{\text{real}}(u)$.

minimum and 64 have 2. Figures 9 and 10 present the same experimental results as the ones for the Drosophila.

6. CONCLUSION

We described in this work several algorithms for exploring the space of all reconciliations between a gene tree and a species tree. From an algorithmic point of view, our exhaustive exploration algorithm is optimal as it requires an (amortized) constant time between successive reconciliations. Our experiments on two simulated and real datasets with low duplication/loss rates show that even in this case the number of reconciliations can be very large, but that for all three combinatorial criteria considered there are relatively few optimal or near-optimal reconciliations, always located close (in terms of NMC distance) to the LCA reconciliation. Recall that this parsimonious reconciliation is known to be a minimum for all the three cost models, and the only one for the loss and mutation criteria. This is opposite to the duplication cost, where there can be more than one optimal reconciliation, which can be counted in polynomial time and explored in optimal time. We are also able to perform such controlled exploration of sub-optimal reconciliations for the three cost models, but we know to count the number of such sub-optimal reconciliations only for the duplication cost. It would be interesting to explore further the combinatorial structure of $\mathcal{T}(G, S)$ to see if it is possible to have as much control on gene losses than we currently have on duplication events.

According to our experimental results on fly and mammalian genomes, parsimonious models (based on combinatorial costs) are fully justified to infer the real evolutionary scenario for all gene families considered here. However, more in-depth studies are required to evaluate the efficiency of parsimony as well as of probabilistic methods in evaluating possible reconciliation scenarios. In this perspective, we are currently studying the impact of lower/higher duplication and loss rates on the shape of the reconciliation space when using both, parsimonious and probabilistic criteria. Other natural generalizations of the algorithms described here are handling either non-binary gene or species trees (Chang and Eulenstein, 2006; Vernot et al., 2008) (or both) and attacking the more difficult problem of multiple gene duplications (Fellows et al., 1998).

ACKNOWLEDGMENTS

C.C. acknowledges the support of NSERC through an Individual Discovery Grant and of Simon Fraser University through a Startup Grant. S.H. acknowledges the support of NSERC through an Individual Discovery Grant. This work was previously published as a proceeding (Doyon et al., 2008).

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Arvestad, L., Berglund, A.-C., Lagergren, J., et al. 2004. Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. *Proc. RECOMB 2004* 326–335.

Bonizzoni, P., Vedova, G.D., and Dondi, R. 2005. Reconciling a gene tree to a species tree under the duplication cost model. *Theor. Comput. Sci.* 347, 36–53.

Chang, W.-C., and Eulenstein, O. 2006. Reconciling gene trees with apparent polytomies. *Proc. COCOON 2006* 235–244.

Chauve, C., Doyon, J.-P., and El-Mabrouk, N. 2008. Gene family evolution by duplication, speciation, and loss. *J. Comput. Biol.* 15, 1043–1062.

Chauve, C., and El-Mabrouk, N. 2009. New perspectives on gene family evolution: losses in reconciliation and a link with supertrees. *Proc. RECOMB 2009* (in press).

Denise, A., and Zimmermann, P. 1997. Uniform random generation of decomposable structures using floating-point arithmetic. *Theor. Comput. Sci.* 218, 233–248.

Doyon, J.-P., Chauve, C., and Hamel, S. 2008. Algorithms for exploring the space of gene tree/species tree reconciliations. *Proc. RECOMB-CG 2008* 1–13.

Fellows, M.R., Hallet, M.T., and Stege, U. 1998. On the multiple gene duplication problem. *Proc. ISAAC 1998* 347–356.

Fitch, W.M. 1970. Distinguishing homologous from analogous proteins. *Syst. Zool.* 19, 99–113.

Fitch, W.M. 2000. Homology—a personal view on some of the problems. *Trends Genet.* 16, 227–231.

Goodman, M., Czelusniak, J., Moore, G.W., et al. 1979. Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Syst. Zool.* 28, 132–163.

Górecki, P., and Tiuryn, J. 2006. Dls-trees: a model of evolutionary scenarios. *Theor. Comput. Sci.* 359, 378–399.

Graur, D., and Li, W.-H. 1999. *Fundamentals of Molecular Evolution*, 2nd ed. Sinauer Associates, Sunderland, MA.

Hahn, M.W., Demuth, J., and Han, S.-G. 2007a. Accelerated rate of gene gain and loss in primates. *Genetics* 177, 1941–1949.

Hahn, M.W., Han, M.V., and Han, S.-G. 2007b. Gene family evolution across 12 *Drosophila* genomes. *PLoS Genet.* 3, e197.

Jensen, R. 2001. Orthologs and paralogs—we need to get it right. *Genome Biol.* 2, 1002.1–1002.3.

Kendall, D.G. 1948. On the generalized "birth-and-death" process. *Ann. Math. Statist.* 19, 1–15.

Ma, B., Li, M., and Zhang, L. 2001. From gene trees to species trees. *SIAM J. Comput.* 30, 729–752.

Ma, J., Ratan, A., Zhang, L., et al. 2007. A heuristic algorithm for reconstructing ancestral gene orders with duplications. *Proc. RECOMB-CG* 122–135.

Page, R.D. 1994. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst. Biol.* 43, 58–77.

Vernot, B., Stolzer, M., Goldman, A., et al. 2008. Reconciliation with non-binary species trees. *J. Comput. Biol.* 15, 981–1006.

Address correspondence to:
Dr. Sylvie Hamel
DIRO
Université de Montréal
CP6128
succ. Centre-Ville
H3C 3J7, Montréal, QC, Canada

E-mail: hamelsyl@iro.umontreal.ca