Reconciling multiple genes trees via segmental duplications and losses

Riccardo Dondi, Manuel Lafond, Celine Scornavacca

Dondi *et al. Algorithms Mol Biol* (2019) 14:7
https://doi.org/10.1186/s13015-019-0139-6

Algorithms for Molecular Biology

RESEARCH Open Access

Reconciling multiple genes trees via segmental duplications and losses

Riccardo Dondi[1], Manuel Lafond[2*] and Celine Scornavacca[3]

Abstract

Reconciling gene trees with a species tree is a fundamental problem to understand the evolution of gene families. Many existing approaches reconcile each gene tree independently. However, it is well-known that the evolution of gene families is interconnected. In this paper, we extend a previous approach to reconcile a set of gene trees with a species tree based on segmental macro-evolutionary events, where segmental duplication events and losses are associated with cost δ and λ, respectively. We show that the problem is polynomial-time solvable when $\delta \leq \lambda$ (via LCA-mapping), while if $\delta > \lambda$ the problem is NP-hard, even when $\lambda = 0$ and a single gene tree is given, solving a long standing open problem on the complexity of multi-gene reconciliation. On the positive side, we give a fixed-parameter algorithm for the problem, where the parameters are δ/λ and the number d of segmental duplications, of time complexity $O\left(\lceil\frac{\delta}{\lambda}\rceil^d \cdot n \cdot \frac{\delta}{\lambda}\right)$. Finally, we demonstrate the usefulness of this algorithm on two previously studied real datasets: we first show that our method can be used to confirm or raise doubt on hypothetical segmental duplications on a set of 16 eukaryotes, then show how we can detect whole genome duplications in yeast genomes.

Keywords: Phylogenetics, Gene trees, Species trees, Reconciliation, Segmental duplications, Fixed-parameter tractability, NP-hardness, Whole genome duplications

Introduction

It is nowadays well established that the evolution of a gene family can differ from that of the species containing these genes. This can be due to quite a number of different reasons, including gene duplication, gene loss, horizontal gene transfer or incomplete lineage sorting, to only name a few [22]. While this discongruity between the gene phylogenies (the *gene trees*) and the species phylogeny (the *species tree*) complicates the process of reconstructing the latter from the former , every cloud has a silver lining: "plunging" gene trees into the species tree and analyzing the differences between these topologies, one can infer the macro-evolutionary events that shaped gene evolution. This is the rationale behind the *species tree-gene tree reconciliation*, a concept introduced in [13] and first formally defined in [24]. Understanding these macro-evolutionary events allows us to better understand the mechanisms of evolution with applications ranging from orthology detection [9, 18, 19, 30] to ancestral genome reconstruction [11], and recently in dating phylogenies [5, 7].

It is well-known that the evolution of gene families is interconnected. However, in current pipelines, each gene tree is reconciled independently with the species tree, even when posterior to the reconciliation phase the genes are considered as related, e.g. [11]. A more pertinent approach would be to reconcile the set of gene trees at once and consider *segmental* macro-evolutionary events, i.e. events that concern a chromosome segment instead of a single gene.

Some work has been done in the past to model segmental gene duplications and three models have been considered: the EC (*Episode Clustering*) problem, the ME (*Minimum Episodes*) problem [1, 14], and the MGD (*Multiple Gene Duplication*) problem [12]. The EC and MGD problems both aim at clustering duplications together by minimizing the number of locations in the species tree where at least one duplication occurred, with the additional requirement that a cluster cannot contain two gene

*Correspondence: manuel.lafond@USherbrooke.ca
[2] Department of Computer Science, Université de Sherbrooke, Sherbrooke, Canada
Full list of author information is available at the end of the article

duplications from a same gene tree in the MGD problem. The ME problem is more biologically-relevant, because it aims at minimizing the actual number of segmental duplications (more details in "Reconciliation with segmental duplications" section). Most of the exact solutions proposed for the ME problem [1, 20, 26] deal with a constrained version, since the possible mappings of a gene tree node are limited to given intervals, see for example [1, Def. 2.4]. In [26], a simple $O^*(2^k)$ time algorithm is presented for the unconstrained version (here O^* hides polynomial factors), where k is the number of speciation nodes that have descending duplications under the LCA-mapping. This shows that the problem is fixed-parameter tractable (FPT) in k. But since the LCA-mapping maximizes the number of speciation nodes, there is no reason to believe that k is a small parameter, and so more practical FPT algorithms are needed. Recently, Delabre et al. [8] studied the problem of reconstructing the evolution of syntenic blocks. Their model allows segmental duplications, but is more constrained since it requires every gene family of every block to have evolved along the same tree.

In this paper, we extend the unconstrained ME model to gene losses and provide a variety of new algorithmic results. We allow weighing segmental duplication events and loss events by separate costs δ and λ, respectively. We show that if $\delta \leq \lambda$, then an optimal reconciliation can be obtained by reconciling each gene tree separately under the usual LCA-mapping, even in the context of segmental duplications. On the other hand, we show that if $\delta > \lambda$ and both costs are given, reconciling a set of gene trees while minimizing segmental gene duplications and gene losses is NP-hard. The hardness also holds in the particular case that we ignore losses, i.e. when $\lambda = 0$. This solves a long standing open question on the complexity of the reconciliation problem under this model (in [1], the authors already said *"it would be interesting to extend the [...] model of Guigó et al. (1996) by relaxing the constraints on the possible locations of gene duplications on the species tree"*. The question is stated as still open in [26]). The hardness holds also when only a single gene tree is given. On the positive side, we describe an algorithm that is practical when δ and λ are not too far apart. More precisely, we show that multi-gene tree reconciliation is fixed-parameter tractable in the ratio δ/λ and the number d of segmental duplications, and can be solved in time $O\left(\lceil \frac{\delta}{\lambda} \rceil^d \cdot n \cdot \frac{\delta}{\lambda}\right)$. The algorithm has been implemented and tested and is freely available[1] at https://github.com/AEVO-lab/MultRec. We first evaluate the potential of multi-gene reconciliation on a set of 16 eukaryotes, and show that our method can find scenarios with less duplications than other approaches. While some previously identified segmental duplications are confirmed by our results, it casts some doubt on others as they do not occur in our optimal scenarios. We then show how the algorithm can be used to detect whole genome duplications in yeast genomes. Further work includes incorporating in the model segmental gene losses and segmental horizontal gene transfers, with a similar flavor than the heuristic method discussed in [10].

[1] To our knowledge, this is the first publicly available reconciliation software for segmental duplications.

Preliminaries

Basic notions

For our purposes, a *rooted phylogenetic tree* $T = (V(T), E(T))$ is an oriented tree, where $V(T)$ is the set of nodes, $E(T)$ is the set of arcs, all oriented away from $r(T)$, the root. Unless stated otherwise, all trees in this paper are rooted phylogenetic trees. A *forest* $F = (V(F), E(F))$ is a directed graph in which every connected component is a tree. Denote by $t(F)$ the set of trees of F that are formed by its connected components. Note that a tree is itself a forest. In what follows, we shall extend the usual terminology on trees to forests.

For an arc (x, y) of F, we call x the *parent* of y, and y a *child* of x. If there exists a path that starts at x and ends at y, then x is an *ancestor* of y and y is a *descendant* of x. We say y is a *proper* descendant of x if $y \neq x$, and then x is a proper ancestor of y. This defines a partial order denoted by $y \leq_F x$, and $y <_F x$ if $x \neq y$ (we may omit the F subscript if clear from the context). If none of $x \leq y$ and $y \leq x$ holds, then x and y are *incomparable*. The set of children of x is denoted $ch(x)$ and its parent x is denoted $par(x)$ [which is defined to be x if x itself is a root of a tree in $t(F)$]. For some integer $k \geq 0$, we define $par^k(x)$ as the k-th parent of x. Formally, $par^0(x) = par(x)$ and $par^k(x) = par(par^{k-1}(x))$ for $k > 0$. The number of children $|ch(x)|$ of x is called the *out-degree* of x. Nodes with no children are *leaves*, all others are *internal nodes*. The set of leaves of a tree F is denoted by $L(F)$. The leaves of F are bijectively labeled by a set $\mathcal{L}(F)$ of labels. A forest is *binary* if $|ch(x)| = 2$ for all internal nodes x. Given a set of nodes X that belong to the same tree $T \in t(F)$, the *lowest common ancestor* of X, denoted $LCA_F(X)$, is the node z that satisfies $x \leq z$ for all $x \in X$ and such that no child of z satisfies this property. We leave $LCA_F(X)$ undefined if no such node exists (when elements of X belong to different trees of $t(F)$). We may write $LCA_F(x, y)$ instead of $LCA_F(\{x, y\})$. The *height* of a forest F, denoted $h(F)$, is the number of nodes of a longest directed path from a root to a leaf in a tree of F (note that the height is sometimes defined as the number of arcs on such a path—here we

use the number of nodes instead). Observe that since a tree is a forest, all the above notions also apply on trees.

Reconciliations

A reconciliation usually involves two rooted phylogenetic trees, a *gene tree* G and a *species tree* S, which we always assume to be both binary. In what follows, we will instead define reconciliation between a gene forest $\mathcal{G}$ and a species tree. Here $\mathcal{G}$ can be thought of as a set of gene trees. Each leaf of $\mathcal{G}$ represents a distinct extant gene, and $\mathcal{G}$ and S are related by a function $s : \mathcal{L}(\mathcal{G}) \rightarrow \mathcal{L}(S)$, which means that each extant gene belongs to an extant species. Note that s does not have to be injective (in particular, several genes from a same gene tree G of $\mathcal{G}$ can belong to the same species) or surjective (some species may not contain any gene of $\mathcal{G}$). Given $\mathcal{G}$ and S, we will implicitly assume the existence of the function s.

In a $\mathbb{DL}$ reconciliation, each node of $\mathcal{G}$ is associated to a node of S and an event—a speciation ($\mathbb{S}$), a duplication ($\mathbb{D}$) or a contemporary event ($\mathbb{C}$)—under some constraints. A contemporary event $\mathbb{C}$ associates a leaf u of $\mathcal{G}$ with a leaf x of S such that $s(u) = x$. A speciation in a node u of $\mathcal{G}$ is constrained to the existence of two separated paths from the mapping of u to the mappings of its two children, while the only constraint given by a duplication event is that evolution of $\mathcal{G}$ cannot go back in time. More formally:

Definition 1 (*Reconciliation*) Given a gene forest $\mathcal{G}$ and a species tree S, a *reconciliation* between $\mathcal{G}$ and S is a function α that maps each node u of $\mathcal{G}$ to a pair $(\alpha_r(u), \alpha_e(u))$ where $\alpha_r(u)$ is a node of $V(S)$ and $\alpha_e(u)$ is an event of type $\mathbb{S}, \mathbb{D}$ or $\mathbb{C}$, such that:

1. if u is a leaf of $\mathcal{G}$, then $\alpha_e(u) = \mathbb{C}$ and $\alpha_r(u) = s(u)$;
2. if u is an internal node of $\mathcal{G}$ with children u_1, u_2, then exactly one of following cases holds:
   - $\alpha_e(u) = \mathbb{S}$, $\alpha_r(u) = LCA_S(\alpha_r(u_1), \alpha_r(u_2))$ and $\alpha_r(u_1), \alpha_r(u_2)$ are incomparable;
   - $\alpha_e(u) = \mathbb{D}$, $\alpha_r(u_1) \leq \alpha_r(u)$ and $\alpha_r(u_2) \leq \alpha_r(u)$

Note that if $\mathcal{G}$ consists of one tree, this definition coincides with the usual one given in the literature (first formally defined in [24]). We say that α is an *LCA-mapping* if, for each internal node $u \in V(\mathcal{G})$ with children u_1, u_2, $\alpha_r(u) = LCA_S(\alpha_r(u_1), \alpha_r(u_2))$. Note that there may be more than one LCA-mapping, since the $\mathbb{S}$ and $\mathbb{D}$ events on internal nodes can vary. The number of duplications of α, denoted by $d(\alpha)$ is the number of nodes u of $\mathcal{G}$ such that $\alpha_e(u) = \mathbb{D}$. For counting the losses, first define for $y \leq x$ the distance $dist(x, y)$ as the number of arcs on the path from x to y. Then, for every internal node u with children $\{u_1, u_2\}$, the number of losses associated with u in a reconciliation α, denoted by $l_\alpha(u)$, is defined as follows:

- if $\alpha_e(u) = \mathbb{S}$, then $l_\alpha(u) = dist(\alpha_r(u), \alpha_r(u_1)) + dist(\alpha_r(u), \alpha_r(u_2)) - 2$;
- if $\alpha_e(u) = \mathbb{D}$, then $l_\alpha(u) = dist(\alpha_r(u), \alpha_r(u_1)) + dist(\alpha_r(u), \alpha_r(u_2))$.

The number of losses of a reconciliation α, denoted by $l(\alpha)$, is the sum of $l_\alpha(\cdot)$ for all internal nodes of $\mathcal{G}$. The usual cost of α, denoted by $cost(\alpha)$, is $d(\alpha) \cdot \delta + l(\alpha) \cdot \lambda$ [21], where δ and λ are respectively the cost of a duplication and a loss event (it is usually assumed that speciations do not incur cost). A *most parsimonious reconciliation*, or MPR, is a reconciliation α of minimum cost. It is not hard to see that finding such an α can be achieved by computing a MPR for each tree in $t(\mathcal{G})$ separately. This MPR is the unique LCA-mapping α in which $\alpha_e(u) = \mathbb{S}$ whenever it is allowed according to Definition 1 [3].

Reconciliation with segmental duplications

Given a reconciliation α for $\mathcal{G}$ in S, and given $s \in V(S)$, write $D(\mathcal{G}, \alpha, s) = \{u \in V(\mathcal{G}) : \alpha_e(u) = \mathbb{D} \text{ and } \alpha_r(u) = s\}$ for the set of duplications of $\mathcal{G}$ mapped to s. We define $\mathcal{G}[\alpha, s]$ to be the subgraph of $\mathcal{G}$ induced by the nodes in $D(\mathcal{G}, \alpha, s)$. Note that $\mathcal{G}[\alpha, s]$ is a forest.

Here we want to tackle the problem of reconciling several gene trees at the same time and counting segmental duplications only once. Given a set of duplications nodes $\mathcal{D} \in V(\mathcal{G})$ occurring in a given node s of the species tree, it is easy to see that the minimum number of segmental duplications associated with s is the minimal number of parts in a partition of $\mathcal{D}$ in which each part does not contain comparable nodes. See Fig. 1(4) for an example. This number coincides [1] with $h_\alpha(s) := h(\mathcal{G}[\alpha, s])$, i.e. the height of the forest of the duplications in s. Now, denote $\hat{d}(\alpha) = \sum_{s \in V(S)} h_\alpha(s)$. For instance in Fig. 1, under the mapping μ in (2), we have $\hat{d}(\mu) = 6$, because $h_\mu(s) = 1$ for $s \in \{A, B, C, E\}$ and $h_\mu(F) = 2$. But under the mapping α in (3), $\hat{d}(\alpha) = 4$, since $h_\alpha(A) = 1$ and $h_\alpha(F) = 3$. Note that this does not consider losses though—the α mapping has more losses than μ.

The cost of α is $cost^{SD}(\mathcal{G}, S, \alpha) = \delta \cdot \hat{d}(\alpha) + \lambda \cdot l(\alpha)$. If $\mathcal{G}$ and S are unambiguous, we may write $cost^{SD}(\alpha)$. We have the following problem:

Problem 1 Most parsimonious reconciliation of a set of trees with segmental duplications (MPRST-SD)

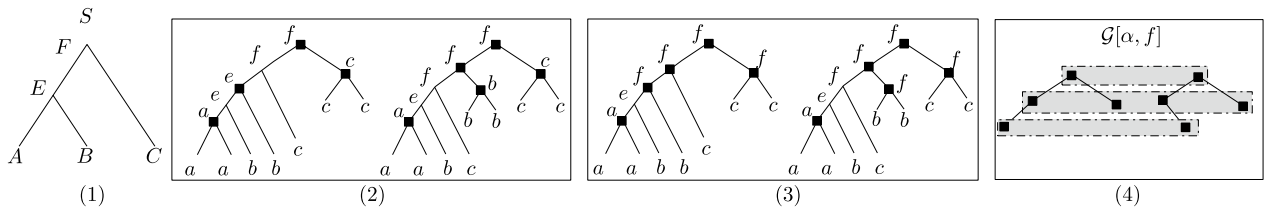

Fig. 1 (1) A species tree S. (2) A gene forest $\mathcal{G}$ with two gene trees reconciled under the MPR that we denote μ. The nodes are labeled by the same name of the species they are mapped to but in lowercase. Black squares indicate duplication nodes. Losses are not shown. (3) The same forest $\mathcal{G}$ but with another reconciliation α for the internal nodes. (4) The forest $\mathcal{G}[\alpha, f]$, along with a partition into (possible) segmental duplications

Instance: *a species tree S, a gene forest $\mathcal{G}$, costs δ for duplications and λ for losses.*

Output: *a reconciliation α of $\mathcal{G}$ in S such that $cost^{SD}(\mathcal{G}, S, \alpha)$ is minimum.*

Note that, when $\lambda = 0$, $cost^{SD}$ coincides with the unconstrained ME score defined in [26] (where it is called *ME under the FHS model*).

Properties of multi-gene reconciliations

We finish this section with some additional terminology and general properties of multi-gene reconciliations that will be useful throughout the paper. The next basic result states that in a reconciliation α, we should set the events of internal nodes to $\mathbb{S}$ whenever it is allowed.

Lemma 1 *Let α be a reconciliation for $\mathcal{G}$ in S, and let $u \in V(\mathcal{G})$ such that $\alpha_e(u) = \mathbb{D}$. Let α' be identical to α, with the exception that $\alpha'_e(u) = \mathbb{S}$, and suppose that α' satisfies the requirements of Definition 1. Then $cost^{SD}(\alpha') \leq cost^{SD}(\alpha)$.*

Proof Observe that changing $\alpha_e(u)$ from $\mathbb{D}$ to $\mathbb{S}$ cannot increase $\hat{d}(\alpha)$. Moreover, as $dist(\alpha'_r(u), \alpha'_r(u_1))$ and $dist(\alpha'_r(u), \alpha'_r(u_2))$ are the same as in α for the two children u_1 and u_2 of u, by definition of duplications and losses this decreases the number of losses by 2. Thus $cost^{SD}(\alpha') \leq cost^{SD}(\alpha)$, and this inequality is strict when $\lambda > 0$. □

Since we are looking for a most parsimonious reconciliation, by Lemma 1 we may assume that for an internal node $u \in V(\mathcal{G})$, $\alpha_e(u) = \mathbb{S}$ whenever allowed, and $\alpha_e(u) = \mathbb{D}$ otherwise. Therefore, $\alpha_e(u)$ is implicitly defined by the α_r mapping. To alleviate notation, we will treat α as simply as a mapping from $V(\mathcal{G})$ to $V(S)$ and thus write $\alpha(u)$ instead of $\alpha_r(u)$. We will assume that the events $\alpha_e(u)$ can be deduced from this mapping α by Lemma 1.

Therefore, treating α as a mapping, we will say that α is *valid* if for every $v \in V(\mathcal{G})$, $\alpha(v) \geq \alpha(v')$ for all descendants v' of v. We denote by $\alpha[v \to s]$ the mapping obtained from α by remapping $v \in V(\mathcal{G})$ to $s \in V(S)$, i.e. $\alpha[v \to s](w) = \alpha(w)$ for every $w \in V(\mathcal{G}) \setminus \{v\}$, and $\alpha[v \to s](v) = s$. Since we are assuming that $\mathbb{S}$ and $\mathbb{D}$ events can be deduced from α, the LCA-mapping is now unique: we denote by $\mu : V(\mathcal{G}) \to V(S)$ the LCA-mapping, defined as $\mu(v) = s(v)$ if $v \in L(\mathcal{G})$, and otherwise $\mu(v) = LCA_S(\mu(v_1), \mu(v_2))$, where v_1 and v_2 are the children of v. Note that for any valid reconciliation α, we have $\alpha(v) \geq \mu(v)$ for all $v \in V(\mathcal{G})$. We also have the following, which will be useful to establish our results.

Lemma 2 *Let α be a mapping from $\mathcal{G}$ to S. If $\alpha(v) > \mu(v)$, then v is a $\mathbb{D}$ node under α.*

Proof Let v_1 and v_2 be the two children of v. If $\alpha(v) \neq LCA_S(\alpha(v_1), \alpha(v_2))$, then v must be a duplication, by the definition of $\mathbb{S}$ events. The same holds if $\alpha(v_1)$ and $\alpha(v_2)$ are not incomparable. Thus assume $\alpha(v) = LCA_S(\alpha(v_1), \alpha(v_2)) > \mu(v)$ and that $\alpha(v_1)$ and $\alpha(v_2)$ are incomparable. This implies that one of $\alpha(v_1)$ or $\alpha(v_2)$ is incomparable with $\mu(v)$, say $\alpha(v_1)$ w.l.o.g. But $\mu(v_1) \leq \alpha(v_1)$, implying that $\mu(v_1)$ is also incomparable with $\mu(v)$, a contradiction to the definition of $\mu = LCA_S(\mu(v_1), \mu(v_2))$. □

Lemma 3 *Let α be a mapping from $\mathcal{G}$ to S, and let $v \in V(\mathcal{G})$. Suppose that there is some proper descendant v' of v such that $\alpha(v') \geq \mu(v)$. Then v is a duplication under α.*

Proof If $\alpha(v) = \mu(v)$, we get $\mu(v) \leq \alpha(v') \leq \alpha(v) = \mu(v)$, and so $\alpha(v') = \mu(v)$. We must then have $\alpha(v'') = \mu(v)$ for every node v'' on the path between v' and v. In particular, v has a child v_1 with $\alpha(v) = \alpha(v_1)$ and thus v is a duplication. If instead $\alpha(v) > \mu(v)$, then v is a duplication by Lemma 2. □

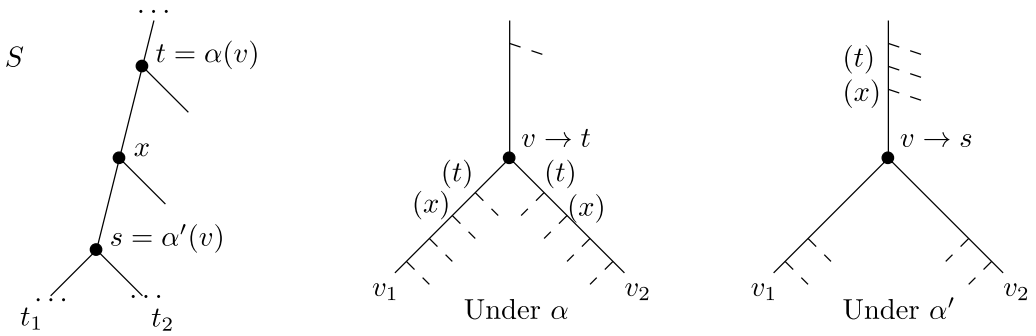

Fig. 2 The Shift-down lemma in action. Here $t = par^2(s)$, and from α to α', we remap v from t to s and save 2 losses—4 losses are saved below v and 2 are added above

The *Shift-down lemma* will prove very useful to argue that we should shift mappings of duplications down when possible, as it saves losses (see Fig. 2). For future reference, do note however that this may increase the height of some duplication forest $\mathcal{G}[\alpha, s]$.

Lemma 4 (Shift-down lemma) *Let α be a mapping from $\mathcal{G}$ to S, let $v \in V(\mathcal{G})$, let $s \in V(S)$ and $k > 0$ be such that $par^k(s) = \alpha(v)$. Suppose that $\alpha[v \to s]$ is a valid mapping. Then $l(\alpha[v \to s]) \leq l(\alpha) - k$.*

Proof Let v_1 and v_2 be the children of v, and denote $t := \alpha(v), t_1 := \alpha(v_1)$ and $t_2 := \alpha(v_2)$. Moreover denote $\alpha' := \alpha[v \to s]$. Let P be the set of nodes that appear on the path between s and t, excluding s but including t (note that s is a proper descendant of t but an ancestor of both t_1 and t_2, by the validity of α'). For instance in Fig. 2, $P = \{x, t\}$. Observe that $|P| = k$. Under α, there is a loss for each node of P on both the (v, v_1) and (v, v_2) branches. (noting that v is a duplication by Lemma 2). These $2k$ losses are not present under α'. On the other hand, there are at most k losses that are present under α' but not under α, which consist of one loss for each node of P on the $(par(v), v)$ branch (in the case that v is not the root of its tree—otherwise, no such loss occurs). This proves that $l(\alpha') \leq l(\alpha) - k$. □

A "proof-by-picture" of the above Lemma appears in Fig. 2. When shifting down the mapping of v, some losses that appear left and right of v get "combined" on the branch above it.

The computational complexity of the MPRST-SD problem

We separate the study of the complexity of the MPRST-SD problem into two subcases: when $\lambda \geq \delta$ and when $\lambda < \delta$.

The case of $\lambda \geq \delta$

The following theorem states that, when $\lambda \geq \delta$, the MPR (ie the LCA-mapping) is a solution to the MPRST-SD problem.

Theorem 1 *Let $\mathcal{G}$ and S be an instance of* MPRST-SD, *and suppose that $\lambda \geq \delta$, Then the LCA-mapping μ is a reconciliation of minimum cost for $\mathcal{G}$ and S. Moreover if $\lambda > \delta$, μ is the unique reconciliation of minimum cost.*

Proof Let α be a mapping of $\mathcal{G}$ into S of minimum cost. Let $v \in V(\mathcal{G})$ be a minimal node of $\mathcal{G}$ with the property that $\alpha(v) \neq \mu(v)$ (i.e. all proper descendants v' of v satisfy $\alpha(v) = \mu(v)$). Note that v must exists since, for every leaf $l \in \mathcal{L}(\mathcal{G})$, we have $\alpha(l) = \mu(l)$. Because $\alpha(v) \geq \mu(v)$, it follows that $\alpha(v) > \mu(v)$. Denote $s = \mu(v)$ and $t = \alpha(v)$. Then there is some $k \geq 1$ such that $t = par^k(s)$. Consider the mapping $\alpha' = \alpha[v \to s]$. This possibly increases the sum of duplications by 1, so that $\hat{d}(\alpha') \leq \hat{d}(\alpha) + 1$. But by the Shift-down lemma, $l(\alpha') \leq l(\alpha) - 1$. Thus we have at most one duplication but save at least one loss.

If $\lambda > \delta$, this contradicts the optimality of α, implying that v cannot exist and thus that $\alpha = \mu$. This proves the uniqueness of μ in this case.

If $\delta = \lambda$, then $\delta\hat{d}(\alpha') + \lambda l(\alpha') \leq \delta\hat{d}(\alpha) + \lambda l(\alpha)$. By applying the above transformation successively on the minimal nodes v that are not mapped to $\mu(v)$, we eventually reach the LCA-mapping μ with an equal or better cost than α. □

The case of $\delta > \lambda$

We show that, in contrast with the $\lambda \geq \delta$ case, the MPRST-SD problem is NP-hard when $\delta > \lambda$ and the costs are given as part of the input. More specifically, we show that the problem is NP-hard when one only wants to minimize the sum of duplication heights, i.e. $\lambda = 0$. Note that

if $\lambda > 0$ but is small enough, the effect will be the same and the hardness result also holds—for instance, putting $\delta = 1$ and say $\lambda < \frac{1}{2|V(\mathcal{G})||V(S)|}$ ensures that even if a maximum number of losses appears on every branch of $\mathcal{G}$, it does not even amount to the cost of one duplication.

We briefly outline the main ideas of the reduction. We start from the Vertex Cover problem, where we are given a graph G and must find a subset of vertices $V' \subseteq V(G)$ of minimum size such that each edge has at least one endpoint in V'. The species tree S and the forest $\mathcal{G}$ are constructed so that, for each vertex $v_i \in V(G)$, there is a gene tree A_i in $\mathcal{G}$ with a long path of duplications, all of which could either be mapped to a species called y_i or another species z_i. We make it so that mapping to y_i introduces one more duplication than mapping to z_i, hence we have to "pay" for each y_i. On the other hand, we construct a tree C_h in $\mathcal{G}$ for each edge in $e_h = v_i v_j \in E(G)$ such that if, in A_i or A_j, we chose to map to either y_i or y_j, then C_h can "reuse" these y_i or y_j duplications with no extra cost. However if we did not choose either y_i or y_j, C_h will introduce a large number of new duplications. We are therefore forced to pick a minimum number of y_i's to "cover" all the C_h trees.

The construction

We will first need particular trees as described by the following Lemma. These trees guarantee that a prescribed set of leaves L are at distance exactly k from the root, and any two of the leaves in L have their LCA at distance at least k / 2. Recall that for a tree T and $u, v \in V(T)$, $dist_T(u, v)$ denotes the number of edges on the path between u and v in T (we write $dist(u, v)$ for short). A *caterpillar* is a binary rooted tree in which each internal node has one child that is a leaf, with the exception of one internal node which has two such children.

Lemma 5 *Let $k \geq 8$ be an integer, and let L be a given set of at most k labels. Then there exists a rooted tree T with leaf set L' with $L \subseteq L'$ such that for any $l \in L$, $dist(l, r(T)) = k$ and for any two distinct $l_1, l_2 \in L$, $dist(l_1, LCA_T(l_1, l_2)) \geq k/2$. Moreover, T can be constructed in polynomial time with respect to k.*

Proof Let p be the smallest integer such that $k \leq 2^p$. First consider a fully balanced binary tree T on 2^p leaves, so that each leaf is at distance p from the root. Then replace each leaf by the root of a caterpillar of height $k - p + 1$ (hence in the caterpillars the longest root-to-leaf path has $k - p$ edges). The resulting tree is T. Choose k of these caterpillars, and in each of them, assign a distinct label of L to a deepest leaf (any of the two). Thus $dist(l, r(T)) = k$ for each $l \in L$, and clearly T can be built in polynomial time. We also have $dist(l_1, LCA_S(l_1, l_2)) \geq k - p$ for each distinct $l_1, l_2 \in L$. As $2^p \leq 2k$, we have $p \leq \log(2k)$. This implies $k - p \geq k - \log(2k) \geq k/2$ for $k \geq 8$. □

In the following, we will assume that $\lambda = 0$ and $\delta = 1$. We reduce the Vertex Cover problem to that of finding a mapping of minimum cost for given $\mathcal{G}$ and S. Recall that in the decision version of Vertex Cover, we are given a graph $G = (V, E)$ and an integer $\beta < n$ and are asked if there exists a subset $V' \subseteq V$ with $|V'| \leq \beta$ such that every edge of E has at least one endpoint in V'. For such a given instance, denote $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ (so that $n = |V|$ and $m = |E|$). The ordering of the v_i's and e_j's can be arbitrary, but must remain fixed for the remainder of the construction.

Let $K := (n + m)^{10}$, and observe in particular that $\beta < n \ll K$. We construct a species tree S and a gene forest $\mathcal{G}$ from G. The construction is relatively technical, but we will provide the main intuitions after having fully described it. For convenience, we will describe $\mathcal{G}$ as a set of gene trees instead of a single graph. Figure 3 illustrates the constructed species tree and gene trees. The construction of S is as follows: start with S being a caterpillar on $3n + 2$ leaves. Let

$$(x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n, x_{n+1})$$

be the path of this caterpillar consisting of the internal nodes, ordered by decreasing depth (i.e. x_1 is the deepest internal node, and x_{n+1} is the root). For each $i \in [n]$, call p_i, q_i and r_i, respectively, the leaf child of x_i, y_i and z_i. Note that x_1 has two leaf children: choose one to name p_1 arbitrarily among the two. Then for each edge uv of S, graft a large (but polynomial) number, say K^{100}, of leaves on the uv edge (grafting t leaves on an edge uv consists of subdividing uv t times, thereby creating t new internal nodes of degree 2, then adding a leaf child to each of these nodes, see Fig. 3). We will refer to these grafted leaves as the *special uv leaves*, and the parents of these leaves as the *special uv nodes*. These special nodes are the fundamental tool that lets us control the range of duplication mappings.

Finally, for each $i \in \{1, \ldots, n\}$, replace the leaf p_i by a tree P_i that contains K distinguished leaves $c_{i,1}, \ldots, c_{i,K}$ such that $dist(c_{i,j}, r(P_i)) = K$ for all $j \in [K]$, and such that $dist(c_{i,j}, lca(c_{i,j}, c_{i,k})) \geq K/2$ for all distinct $j, k \in [K]$. By Lemma 5, each P_i can be constructed in polynomial time. Note that the edges inside a P_i subtree do not have special leaves grafted onto them. This concludes the construction of S.

We proceed with the construction of the set of gene trees $\mathcal{G}$. Most of the trees of $\mathcal{G}$ consist of a subset of the nodes of S, to which we graft additional leaves to introduce duplications—some terminology is needed before proceeding. For $w \in V(S)$, *deleting* w consists in

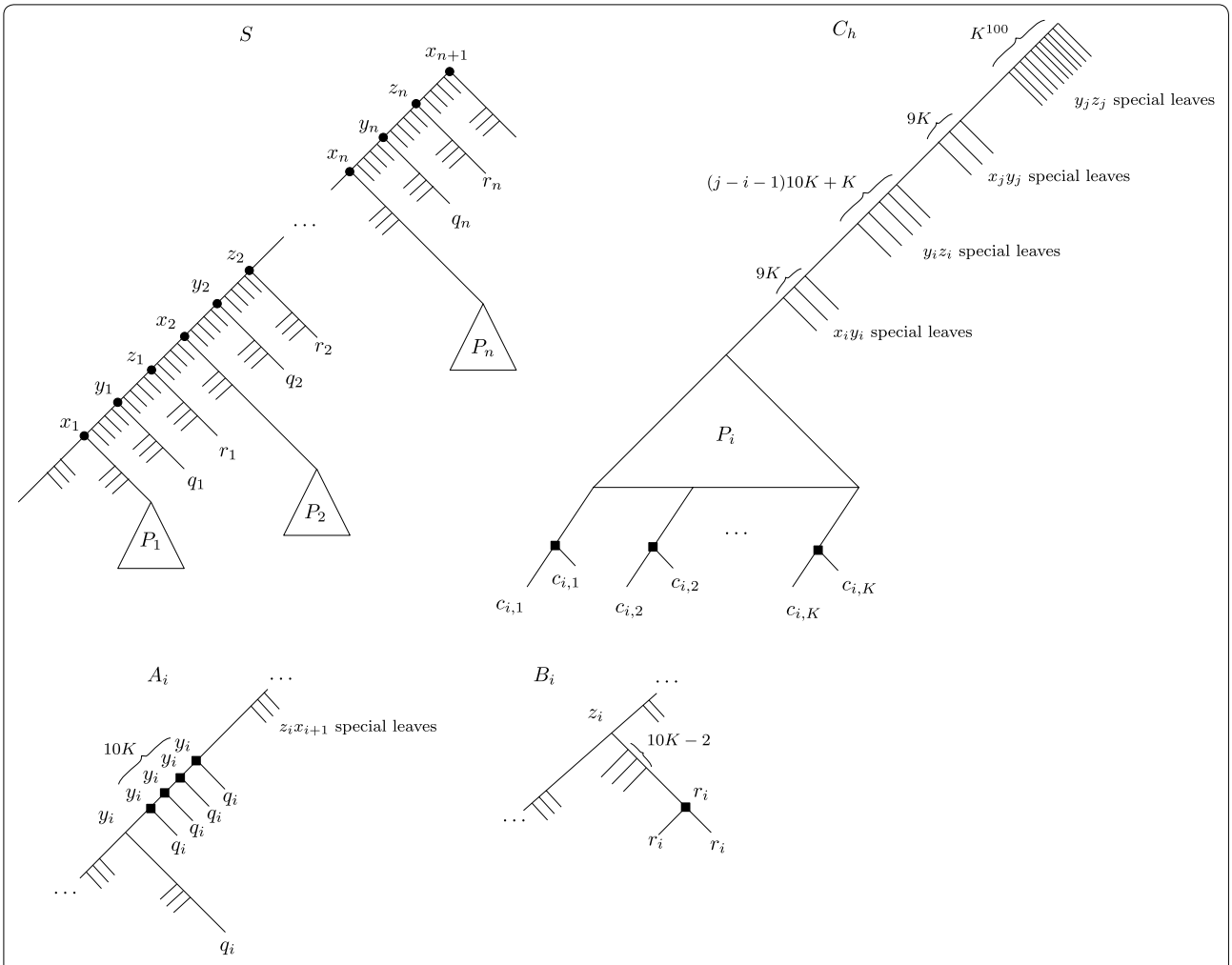

Fig. 3 The species tree S, and the A_i, B_i and C_h trees. The internal nodes are labeled by their LCA-mapping μ, and black squares on the gene trees represent duplication nodes under μ

removing w and all its descendants from S, then contracting the possible resulting degree two vertex (which was the parent of w if $p(w) \neq r(S)$). If this leaves a root with only one child, we contract the root with its child. For $X \subseteq \mathcal{L}(S)$, *keeping* X consists of successively deleting every node that has no descendant in X until none remains (the tree obtained by keeping X is sometimes called the *restriction* of S to X).

The forest $\mathcal{G}$ is the union of three sets of trees A, B, C, so that $\mathcal{G} = A \cup B \cup C$. Roughly speaking A is a set of trees corresponding to the choice of vertices in a vertex cover, B is a set of trees to ensure that we "pay" a cost of one for each vertex in the cover, and C is a set of trees corresponding to edges. For simplicity, we shall describe the trees of $\mathcal{G}$ as having leaves labeled by elements of $\mathcal{L}(S)$—a leaf labeled $s \in \mathcal{L}(S)$ in a gene tree $T \in \mathcal{G}$ is understood to be a unique gene that belongs to species s.

- *The A trees* Let $A = \{A_1, \ldots, A_n\}$, one tree for each vertex of G. For each $i \in [n]$, obtain A_i by first taking a copy of S, then deleting all the special $y_i z_i$ leaves. Then on the resulting $z_i y_i$ branch, graft $10K$ leaves labeled q_i. Then delete the child of z_i that is also an ancestor of r_i (removing z_i in the process). Figure 3 bottom-left might be helpful.

  As a result, under the LCA-mapping μ, A_i has a path of $10K$ duplications mapped to y_i. One can choose whether to keep this mapping in A_i, or to remap these duplications to z_i.
- *The B trees* Let $B = \{B_1, \ldots, B_n\}$. For $i \in [n]$, B_i is obtained from S by deleting all except $10K - 2$ of the special $r_i z_i$ leaves, and grafting a leaf labeled r_i on the edge between r_i and its parent, thereby creating a single duplication mapped to r_i under μ.

- *The C trees* Let $C = \{C_1, C_2, \ldots, C_m\}$, where for $h \in [m]$, C_h corresponds to edge e_h. Let v_i, v_j be the two endpoints of edge $e_h = \{v_i, v_j\}$, where $i < j$. To describe C_h, we list the set of leaves that we keep from S. Keep all the leaves of the P_i subtree of S, and keep a subset of the special leaves defined as follows:

  - Keep $9K$ of the special $x_i y_i$ leaves;
  - Keep $(j - i - 1)10K + K$ of the special $y_i z_i$ leaves;
  - Keep $9K$ of the special $x_j y_j$ leaves;
  - Keep all the special $y_j z_j$ leaves.

  No other leaves are kept. Next, in the tree obtained by keeping the aforementioned list of leaves, for each $k \in [K]$ we graft, on the edge between $c_{i,k}$ and its parent, another leaf labeled $c_{i,k}$. Thus C_h has K duplications, all located at the bottom of the P_i subtree. This concludes our construction.

Let us pause for a moment and provide a bit of intuition for this construction. We will show that G has a vertex cover of size β if and only if there exists a mapping α of $\mathcal{G}$ of cost at most $10Kn + \beta$. As we will show later on, two A_i trees cannot have a duplication mapped to the same species of S, so these trees alone account for $10Kn$ duplications. The r_i duplications in the B_i trees account for n more duplications, so that if we kept the LCA-mapping, we would have $10Kn + n > 10Kn + \beta$ duplications. But some of these r_i duplications could be remapped to z_i, at the cost of creating a path of $10K$ duplications to z_i. This is fine if A_i also has a path of $10K$ duplications to z_i, as this does not incur additional height. In this case, the A_i, B_i pairs introduces $10K$ duplications. If instead this path in A_i is mapped to y_i, we will show that remapping r_i is forbidden, summing up to $10K + 1$ duplications for such a particular A_i, B_i pair. The disadvantage of remapping every to z_i will become apparent when we consider the C_h trees. The idea is that mapping the duplications of A_i to y_i represents including vertex v_i in the vertex cover, and mapping them to z_i represents not including v_i. Because each time we map the A_i duplications to y_i, we have the additional r_i duplication in B_i, we cannot do that more than β times.

Now consider a C_h tree, $e_h = \{v_i, v_j\}$. Under the LCA-mapping, the $c_{i,k}$ duplications at the bottom enforce an additional K duplications. This can be avoided by, say, mapping all these duplications to the same species. For instance, we could remap all these duplications to some y_k species of S. But in this case, because of Lemma 3, every node v of C_h above a $c_{i,k}$ duplication for which $\mu(v) \leq y_k$ will become a duplication. This will create a duplication subtree D in C_h with a large height, and our goal will be to "reuse" the duplications we chose in the $A_{k'}$ and $B_{k'}$ trees. As it turns out, this "reuse" of duplications will be feasible only if some y_i or y_j has duplication height $10K$. If this does not occur, any attempt at mapping the $c_{i,k}$ nodes to a common species will induce a chain reaction of too many duplications created above.

The complete proof is somewhat technical. The interested reader can find it in Appendix.

Theorem 2 *The* MPRST-SD *problem is NP-hard for* $\lambda = 0$ *and for given* $\delta > \lambda$.

The above hardness supposes that δ and λ can be arbitrarily far apart. This leaves open the question of whether MPRST-SD is NP-hard when δ and λ are fixed constants—in particular when $\delta = 1 + \epsilon$ and $\lambda = 1$, where $\epsilon < 1$ is some very small constant. We end this section by showing that the above hardness result persists event if only one gene tree is given. The idea is to reduce from the MPRST-SD shown hard just above. Given a species tree S and a gene forest $\mathcal{G}$, we make $\mathcal{G}$ a single tree by incorporating a large number of speciations (under μ) above the root of each tree of $\mathcal{G}$ (modifying S accordingly), then successively joining the roots of two trees of $\mathcal{G}$ under a common parent until $\mathcal{G}$ has only one tree.

Theorem 3 *The* MPRST-SD *problem is NP-hard for* $\lambda = 0$ *and for given* $\delta > \lambda$, *even if only one gene tree is given as input.*

Proof We reduce from the MPRST-SD problem in which multiple trees are given. We assume that $\delta = 1$ and $\lambda = 0$ and only consider duplications—we use the same argument as before to justify that the problem is NP-hard for very small λ. Let S be the given species tree and $\mathcal{G}$ be the given gene forest. As we are working with the decision version of MPRST-SD, assume we are given an integer t and asked whether $cost^{SD}(\mathcal{G}, S, \alpha) \leq t$ for some α. Denote $n = |\mathcal{L}(\mathcal{G})|$ and let $G_1, \ldots, G_k$ be the $k > 1$ trees of $\mathcal{G}$. We construct a corresponding instance of a species tree S' and a single gene tree T as follows (the construction is illustrated in Fig. 4). Let S' be a species tree obtained by adding $2(t + k)$ nodes "above" the root of S. More precisely, first let C be a caterpillar with $2(t + k)$ internal nodes. Let l be a deepest leaf of C. Obtain S' by replacing l by the root of S. Then, obtain the gene tree T by taking k copies $C_1, \ldots, C_k$ of C, and for each leaf l' of each C_i other than l, put $s(l')$ as the corresponding leaf in S'. Then for each $i \in [k]$, replace the l leaf of C_i by the tree G_i (we keep the leaf mapping s of G_i), resulting in a tree we call T_i. Finally, let T' be a caterpillar with k leaves $h_1, \ldots, h_k$, and replace each h_i by the T_i tree. The resulting tree is T. We show that $cost(\mathcal{G}, S, \alpha) \leq t$ for some α if and only if $cost(T, S', \alpha') \leq t + k - 1$ for some α'.

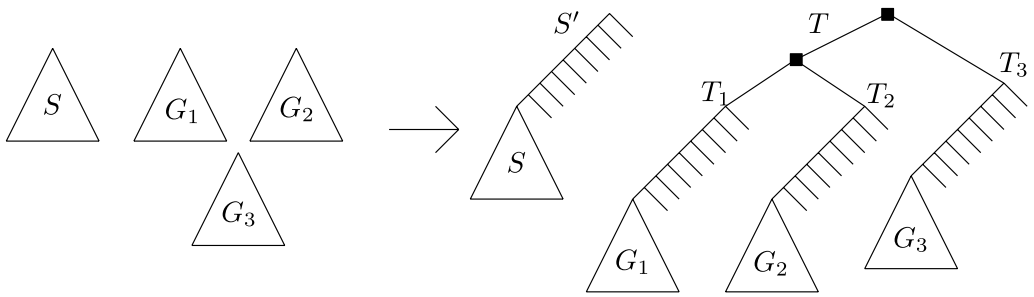

Fig. 4 The construction of S' and T from S and the set of gene trees $G_1, \ldots, G_k$ (here $k = 3$). The black squares indicate the path of $k-1$ duplications that must be mapped to $r(S')$

Notice the following: in any mapping α of T, the $k-1$ internal nodes of the T' caterpillar must be duplications mapped to $r(S')$, so that $h_\alpha(r(S')) \geq k-1$. Also note that under the LCA mapping μ for T and S', the only duplications other than those $k-1$ mentioned above occur in the G_i subtrees. The ($\Rightarrow$) is then easy to see: given α such that $cost(\mathcal{G}, S, \alpha) \leq t$, we set $\alpha'(v) = \alpha(v)$ for every node v of T that is also in $\mathcal{G}$ (namely the nodes of $G_1, \ldots, G_k$), and set $\alpha'(v) = \mu(v)$ for every other node. This achieves a cost of $t + k - 1$.

As for the ($\Leftarrow$) direction, suppose that $cost^{SD}(T', S', \alpha') \leq t + k - 1$ for some mapping α'. Observe that under the LCA-mapping in T, each root of each G_i subtree has a path of $2(t+k)$ speciations in its ancestors. If any node in a G_i subtree of T is mapped to $r(S')$, then all these speciations become duplications (by Lemma 3), which would contradict $cost^{SD}(T', S', \alpha') \leq t + k - 1$. We may thus assume that no node belonging to a G_i subtree is mapped to $r(S')$. Since $h_{\alpha'}(r(S')) \geq k-1$, this implies that the restriction of α' to the G_i subtrees has cost at most t.

More formally, consider the mapping α'' from $\mathcal{G}$ to S' in which we put $\alpha''(v) = \alpha'(v)$ for all $v \in V(\mathcal{G})$. Then $cost^{SD}(\mathcal{G}, S', \alpha'') \leq cost^{SD}(T, S', \alpha') - (k-1) \leq t$, because α'' does not contain the top $k-1$ duplications of α', and cannot introduce longer duplication paths than in α'.

We are not done, however, since α'' is a mapping from $\mathcal{G}$ to S', and not from $\mathcal{G}$ to S. Consider the set $Q \subseteq V(\mathcal{G})$ of nodes v of $\mathcal{G}$ such that $\alpha''(v) \in \overline{V(S)} := V(S') \setminus V(S)$. We will remap every such node to $r(S)$ and show that this cannot increase the cost. Observe that if $v \in Q$, then every ancestor of v in $\mathcal{G}$ is also in Q. Also, every node in Q is a duplication (by invoking Lemma 2).

Consider the mapping α^* from $\mathcal{G}$ to S' in which we put $\alpha^*(v) = \alpha''(v)$ for all $v \notin Q$, and $\alpha^*(v) = r(S)$ for all $v \in Q$. It is not difficult to see that α^* is valid.

Now, $h_{\alpha^*}(s) = 0$ for all $s \in \overline{V(S)}$ and $h_{\alpha^*}(s) = h_{\alpha''}(s)$ for all $s \in V(S) \setminus \{r(S)\}$. Moreover, the height of the $r(S)$ duplications under α^* cannot be more than the height of the forest induced by Q and the duplications mapped to $r(S)$ under α''. In other words,

$$
\begin{aligned}
h_{\alpha^*}(r(S)) &\leq \max_{G_i} \left(h_{\alpha''}(r(S)) + \sum_{s' \in \overline{V(S)}} h(G_i[\alpha'', s']) \right) \\
&= h_{\alpha''}(r(S)) + \max_{G_i} \left(\sum_{s' \in \overline{V(S)}} h(G_i[\alpha'', s']) \right) \\
&\leq h_{\alpha''}(r(S)) + \sum_{s' \in \overline{V(S)}} \max_{G_i} \left(h(G_i[\alpha'', s']) \right) \\
&= h_{\alpha''}(r(S)) + \sum_{s' \in \overline{V(S)}} h\left(\mathcal{G}[\alpha'', s']\right) \\
&= h_{\alpha''}(r(S)) + \sum_{s' \in \overline{V(S)}} h_{\alpha''}(s')
\end{aligned}
$$

Therefore, the sum of duplication heights cannot have increased. Finally, because α^* is a mapping from $\mathcal{G}$ to S, we deduce that $cost^{SD}(\mathcal{G}, S, \alpha^*) \leq cost^{SD}(\mathcal{G}, S', \alpha'') \leq t$, as desired. □

An FPT algorithm

In this section, we show that for costs $\delta > \lambda$ and a parameter $d > 0$, if there is an optimal reconciliation α of cost $cost^{SD}(\mathcal{G}, S)$ satisfying $\hat{d}(\alpha) \leq d$, then α can be found in time $O(\lceil \frac{\delta}{\lambda} \rceil^d \cdot n \cdot \frac{\delta}{\lambda})$.

In what follows, we allow mappings to be partially defined, and we use the $\perp$ symbol to indicate undetermined mappings. The idea is to start from a mapping in which every internal node is undetermined, and gradually determine those in a bottom-up fashion. We need an additional set of definitions. We will assume that $\delta > \lambda > 0$ (although the algorithm described in this section can solve the $\lambda = 0$ case by setting λ to a very small value).

We say that the mapping $\alpha : V(\mathcal{G}) \to V(S) \cup \{\perp\}$ is a *partial mapping* if $\alpha(l) = s(l)$ for every leaf $l \in \mathcal{L}(\mathcal{G})$, and it holds that whenever $\alpha(v) \neq \perp$, we have $\alpha(v') \neq \perp$ for every descendant v' of v. That is, if a node is determined, then all its descendants also are. This also implies that every ancestor of a $\perp$-node is also a

$\perp$-node. A node $v \in V(\mathcal{G})$ is a *minimal* $\perp$*-node* (under α) if $\alpha(v) = \perp$ and $\alpha(v') \neq \perp$ for each child v' of v. If $\alpha(v) \neq \perp$ for every $v \in V(\mathcal{G})$, then α is called *complete*. Note that if α is partial and $\alpha(v) \neq \perp$, one can already determine whether v is an $\mathbb{S}$ or a $\mathbb{D}$ node, and hence we may say that v is a speciation or a duplication under α. Also note that the definitions of $\hat{d}(\alpha), l(\alpha)$ and $h_\alpha(s)$ extend naturally to a partial mapping α by considering the forest induced by the nodes not mapped to $\perp$.

If α is a partial mapping, we call α' a *completion* of α if α' is complete, and $\alpha(v) = \alpha'(v)$ whenever $\alpha(v) \neq \perp$. Note that such a completion always exists, as in particular one can map every $\perp$-node to the root of S (such a mapping must be valid, since all ancestors of a $\perp$-node are also $\perp$-nodes, which ensures that $r(S) = \alpha'(v) \geq \alpha'(v')$ for every descendant v' of a newly mapped $\perp$-node v). We say that α' is an *optimal completion of* α if $cost^{SD}(\alpha')$ is minimum among every possible completion of α. For a minimal $\perp$-node v with children v_1 and v_2, we denote $\mu_\alpha(v) = LCA_S(\alpha(v_1), \alpha(v_2))$, i.e. the lowest species of S to which v can possibly be mapped to in any completion of α. Observe that $\mu_\alpha(v) \geq \mu(v)$. Moreover, if v is a minimal $\perp$-node, then in any completion α' of α, $\alpha'[v \to \mu_\alpha(v)]$ is a valid mapping. A minimal $\perp$-node v is called a *lowest minimal* $\perp$*-node* if, for every minimal $\perp$-node w distinct from v, either $\mu_\alpha(v) \leq \mu_\alpha(w)$ or $\mu_\alpha(v)$ and $\mu_\alpha(w)$ are incomparable.

The following Lemma forms the basis of our FPT algorithm, as it allows us to bound the possible mappings of a minimal $\perp$-node.

Lemma 6 *Let α be a partial mapping and let v be a minimal $\perp$-node. Then for any optimal completion α^* of α, $\alpha^*(v) \leq par^{\lceil \delta/\lambda \rceil}(\mu_\alpha(v))$.*

Proof Let α^* be an optimal completion of α and let $\alpha' := \alpha^*[v \to \mu_\alpha(v)]$. Note that $\hat{d}(\alpha') \leq \hat{d}(\alpha^*) + 1$. Now suppose that $\alpha^*(v) > par^{\lceil \delta/\lambda \rceil}(\mu_\alpha(v))$. Then by the Shift-down lemma, $l(\alpha^*) - l(\alpha') > \lceil \delta/\lambda \rceil \geq \delta/\lambda$. Thus $cost^{SD}(\alpha^*) - cost^{SD}(\alpha') > -\delta + \lambda(\delta/\lambda) = 0$. This contradicts the optimality of α^*. □

A node $v \in V(\mathcal{G})$ is a *required duplication* (under α) if, in any completion α' of α, v is a duplication under α'. We first show that required duplications are easy to find.

Lemma 7 *Let v be a minimal $\perp$-node under α, and let v_1 and v_2 be its two children. Then v is a required duplication under α if and only if $\alpha(v_1) \geq \mu(v)$ or $\alpha(v_2) \geq \mu(v)$.*

Proof Suppose that $\alpha(v_1) \geq \mu(v)$, and let α' be a completion of α. If $\alpha'(v) = \alpha'(v_1)$, then v is a duplication by definition. Otherwise, $\alpha'(v) > \alpha'(v_1) = \alpha(v_1) \geq \mu(v)$, and v is a duplication by Lemma 2. The case when $\alpha(v_2) \geq \mu(v)$ is identical.

Conversely, suppose that $\alpha(v_1) < \mu(v)$ and $\alpha(v_2) < \mu(v)$. Then $\alpha(v_1)$ and $\alpha(v_2)$ must be incomparable descendants of $\mu(v)$ (because otherwise if e.g. $\alpha(v_1) \leq \alpha(v_2)$, then we would have $\mu(v) = LCA_S(\mu(v_1), \mu(v_2)) \leq LCA_S(\alpha(v_1), \alpha(v_2)) = \alpha(v_2)$, whereas we are assuming that $\alpha(v_2) < \mu(v)$). Take any completion α' of α such that $\alpha'(v) = \mu(v)$. To see that v is a speciation under α', it remains to argue that $\alpha'(v) = \mu(v) = LCA_S(\alpha(v_1), \alpha(v_2))$. Since $\mu(v)$ is an ancestor of both $\alpha(v_1)$ and $\alpha(v_2)$, we have $LCA_S(\alpha(v_1), \alpha(v_2)) \leq \mu(v)$. We also have $\mu(v) = LCA_S(\mu(v_1), \mu(v_2)) \leq LCA_S(\alpha(v_1), \alpha(v_2))$, and equality follows. □

Lemmas 8 and 9 allow us to find minimal $\perp$-nodes of $\mathcal{G}$ that are the easiest to deal with, as their mapping in an optimal completion can be determined with certainty.

Lemma 8 *Let v be a minimal $\perp$-node under α. If v is not a required duplication under α, then $\alpha^*(v) = \mu_\alpha(v)$ for any optimal completion α^* of α.*

Proof Let v_1, v_2 be the children of v, and let α^* be an optimal completion of α. Since v is not a required duplication, by Lemma 7 we have $\alpha(v_1) < \mu(v)$ and $\alpha(v_2) < \mu(v)$ and, as argued in the proof of Lemma 7, $\alpha(v_1)$ and $\alpha(v_2)$ are incomparable. We thus have that $\mu_\alpha(v) = \mu(v)$. Then $\alpha^*[v \to \mu(v)]$ is a valid mapping, and v is a speciation under this mapping. Hence $\hat{d}(\alpha^*[v \to \mu(v)]) \leq \hat{d}(\alpha^*)$. Then by the Shift-down lemma, this new mapping has fewer losses, and thus attains a lower cost than α^*. □

Lemma 9 *Let v be a minimal $\perp$-node under α, and let $\alpha_v := \alpha[v \to \mu_\alpha(v)]$. If $\hat{d}(\alpha) = \hat{d}(\alpha_v)$, then $\alpha^*(v) = \mu_\alpha(v)$ for any optimal completion α^* of α.*

Proof Let α^* be an optimal completion of α. Denote $s := \mu_\alpha(v)$, and assume that $\alpha^*(v) > s$ (as otherwise, we are done). Let $\alpha' = \alpha^*[v \to s]$. We have that $l(\alpha') < l(\alpha^*)$ by the Shift-down lemma. To prove the Lemma, we then show that $\hat{d}(\alpha') \leq \hat{d}(\alpha^*)$. Suppose otherwise that $\hat{d}(\alpha') > \hat{d}(\alpha^*)$. As only v changed mapping to s to go from α^* to α', this implies that $h_{\alpha'}(s) > h_{\alpha^*}(s)$ because of v. Since under α^*, no ancestor of v is mapped to s, it must be that under α', v is the root of a subtree T of height $h_{\alpha'}(s)$ of duplications in s. Since T contains only descendants of v, it must also be that $h_{\alpha_v}(s) = h_{\alpha'}(s)$ (here α_v is the mapping defined in the Lemma statement). As we are assuming that $h_{\alpha'}(s) > h_{\alpha^*}(s)$, we get $h_{\alpha_v}(s) > h_{\alpha^*}(s)$. This is a contradiction, since $h_{\alpha^*}(s) \geq h_\alpha(s) = h_{\alpha_v}(s)$ (the left inequality because α^* is a completion of α, and the

right equality by the choice of α_ν). Then $l(\alpha') < l(\alpha^*)$ and $\hat{d}(\alpha') \leq \hat{d}(\alpha^*)$ contradicts the fact that α^* is optimal. □

We say that a minimal $\perp$-node $v \in V(\mathcal{G})$ is *easy* (under α) if v falls into one of the cases described by Lemma 8 or Lemma 9. Formally, v is easy if either v is a speciation mapped to $\mu_\alpha(v)$ under any optimal completion of α (Lemma 8), or $\hat{d}(\alpha) = \hat{d}(\alpha[v \to \mu_\alpha(v)])$ (Lemma 9). Our strategy will be to "clean-up" the easy nodes, meaning that we map them to $\mu_\alpha(v)$ as prescribed above, and then handle the remaining non-easy nodes by branching over the possibilities. We say that a partial mapping α is *clean* if every minimal $\perp$-node v satisfies the two following conditions:

[**C1**] v is not easy;
[**C2**] for all duplication nodes w (under α with $\alpha(w) \neq \perp$), either $\alpha(w) \leq \mu_\alpha(v)$ or $\alpha(w)$ is incomparable with $\mu_\alpha(v)$.

Roughly speaking, C2 says that all further duplications that may "appear" in a completion of α will be mapped to nodes "above" the current duplications in α. The purpose of C2 is to allow us to create duplication nodes with mappings from the bottom of S to the top. Our goal will be to build our α mapping in a bottom-up fashion in $\mathcal{G}$ whilst maintaining this condition. The next lemma states that if α is clean and some *lowest* minimal $\perp$-node v gets mapped to species s, then v brings with it every minimal $\perp$-node that can be mapped to s.

Lemma 10 *Suppose that α is a clean partial mapping, and let α^* be an optimal completion of α. Let v be a lowest minimal $\perp$-node under α, and let $s := \alpha^*(v)$. Then for every minimal $\perp$-node w such that $\mu_\alpha(w) \leq s$, we have $\alpha^*(w) = s$.*

Proof Denote $\alpha' := \alpha[v \to s]$. Suppose first that $s = \mu_\alpha(v)$. Note that since α is clean, v is not easy, which implies that $h_{\alpha'}(s) = h_\alpha(s) + 1$. Since v is a lowest minimal $\perp$-node, if w is a minimal $\perp$-node such that $\mu_\alpha(w) \leq s$, we must have $\mu_\alpha(w) = s$, as otherwise v would not have the 'lowest' property. Moreover, because v and w are both minimal $\perp$-nodes under the partial mapping α, one cannot be the ancestor of the other and so v and w are incomparable. This implies that mapping w to s under α' cannot further increase $h_{\alpha'}(s)$ (because we already increased it by 1 when mapping v to s). Thus $\hat{d}(\alpha') = \hat{d}(\alpha'[w \to s])$, and w is easy under α' and must be mapped to s by Lemma 9. This proves the $\alpha^*(v) = \mu_\alpha(s)$ case.

Now assume that $s > \mu_\alpha(v)$, and let w be a minimal $\perp$-node with $\mu_\alpha(w) \leq s$. Let us denote $s' := \alpha^*(w)$. If $s' = s$, then we are done. Suppose that $s' < s$, noting that $h_{\alpha^*}(s') > 0$ (because w must be a duplication node, due to α being clean). If $s' = \mu_\alpha(v)$, then w is also a lowest minimal $\perp$-node. In this case, using the arguments from the previous paragraph and swapping the roles of v and w, one can see that v is easy in $\alpha[w \to s']$ and must be mapped to $s' < s$, a contradiction. Thus assume $s' > \mu_\alpha(v)$. Under α^*, for each child v' of v, we have $\alpha^*(v') \leq \mu_\alpha(v) < s'$, and for each ancestor v'' of v, we have $\alpha^*(v'') \geq \alpha^*(v) = s > s'$. Therefore, by remapping v to s', v is the only duplication mapped to s' among its ancestors and descendants. In other words, because $h_{\alpha^*}(s') > 0$, we have $\hat{d}(\alpha^*[v \to s']) \leq \hat{d}(\alpha^*)$. Moreover by the Shift-down lemma, $l(\alpha^*[v \to s']) < l(\alpha^*)$, which contradicts the optimality of α^*.

The remaining case is $s' > s$. Note that $h_{\alpha^*}(s) > 0$ (because v must be a duplication node, due to α being clean). Since it holds that v is a minimal $\perp$-node, that α is clean and that $s > \mu_\alpha(v)$, it must be the case that α has no duplication mapped to s (by the second property of cleanness). In particular, w has no descendant that is a duplication mapped to s under α (and hence under α^*). Moreover, as $s' = \alpha^*(w) > s$, w has no ancestor that is a duplication mapped to s. Thus $\hat{d}(\alpha^*[w \to s]) \leq \hat{d}(\alpha^*)$, and the Shift-down lemma contradicts the optimality of α^*. This concludes the proof. □

We are finally ready to describe our algorithm. We start from a partial mapping α with $\alpha(v) = \perp$ for every internal node v of $\mathcal{G}$. We gradually "fill-up" the $\perp$-nodes of α in a bottom-up fashion, maintaining a clean mapping at each step and ensuring that each decision leads to an optimal completion α^*. To do this, we pick a lowest minimal $\perp$-node v, and "guess" $\alpha^*(v)$ among the $\lceil \delta/\lambda \rceil$ possibilities. This increases some $h_\alpha(s)$ by 1. For each such guess s, we use Lemma 10 to map the appropriate minimal $\perp$-nodes to s, then take care of the easy nodes to obtain another clean mapping. We repeat until we have either found a complete mapping or we have a duplication height higher than d. An illustration of a pass through the algorithm is shown in Fig. 5.

Notice that the algorithm assumes that it receives a clean partial mapping α. In particular, the initial mapping α that we pass to the first call should satisfy the two properties of cleanness. To achieve this, we start with a partial mapping α in which every internal node is a $\perp$-node. Then, while there is a minimal $\perp$-node v that is not a required duplication, we set $\alpha(v) = \mu_\alpha(v)$, which makes v a speciation. It is straightforward to see that the resulting α is clean: C1 is satisfied because we cannot make any

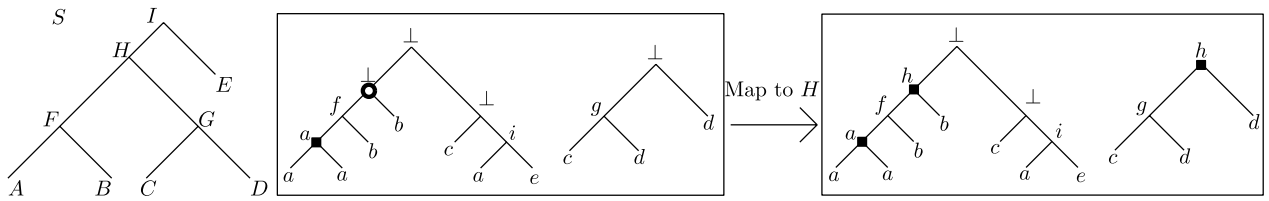

Fig. 5 An illustration of one pass through the algorithm. The species tree S is left and $\mathcal{G}$ has two trees (middle) and partial mapping α (labels are the lowercase of the species). Here α is in a clean state, and the algorithm will pick a lowest minimal $\perp$-node (white circle) and try to map it to, say, H. The forest on the right is the state of α after applying this and cleaning up

more minimal $\perp$-nodes become speciations, and we cannot create any duplication node without increasing $cost^{SD}$ because α has no duplication. C2 is met because there are no duplications at all.

an optimal completion α^* of α having $\hat{d}(\alpha^*) \leq d$, or ∞ if no such completion exists—assuming that the algorithm receives a clean mapping α as input. Thus in order to use induction, we must also show that at each recursive call

Algorithm 1 FPT algorithm for parameter d.

1: **procedure** SUPERRECONCILE($\mathcal{G}, S, \alpha, d, \delta, \lambda$)
$\mathcal{G}$ is the set of input trees, S is the species tree, α is a *clean* partial mapping, d is the maximum value of $\hat{d}(\alpha)$ and δ and λ are the costs of a segmental duplication and loss, respectively.
2: **if** $\hat{d}(\alpha) > d$ **then**
3: Return ∞
4: **else if** α is a complete mapping **then**
5: Return $cost^{SD}(\alpha)$
6: **else**
7: Let v be a lowest minimal $\perp$-node
8: $bestCost \leftarrow \infty$
9: **for** s such that $\mu_\alpha(v) \leq s \leq par^{\lceil \delta/\lambda \rceil}(\mu_\alpha(v))$ **do**
10: Let $\alpha' = \alpha[v \rightarrow s]$
11: **for** minimal $\perp$-node $w \neq v$ under α such that $\mu_\alpha(w) \leq s$ **do**
12: Set $\alpha' = \alpha'[w \rightarrow s]$
13: **while** there is a minimal $\perp$-node w that is easy under α' **do**
14: Set $\alpha' = \alpha'[w \rightarrow \mu_{\alpha'}(w)]$
15: $cost \leftarrow superReconcile(\mathcal{G}, S, \alpha', d)$
16: **if** $cost < bestCost$ **then** $bestCost \leftarrow cost$
17: Return $bestCost$

The complexity follows from the fact that the algorithm creates a search tree of degree $\lceil \delta/\lambda \rceil$ of depth at most d. The main technicality is to show that the algorithm maintains a clean mapping before each recursive call.[2]

Theorem 4 *Algorithm* 1 *is correct and finds a minimum cost mapping* α^* *satisfying* $\hat{d}(\alpha^*) \leq d$, *if any, in time* $O(\lceil \frac{\delta}{\lambda} \rceil^d \cdot n \cdot \frac{\delta}{\lambda})$.

Proof We show by induction over the depth of the search tree that, in any recursive call made to Algorithm 1 with partial mapping α, the algorithm returns the cost of done on line 15, α' is a clean mapping. We additionally claim that the search tree created by the algorithm has depth at most d. To show this, we will also prove that every α' sent to a recursive call satisfies $\hat{d}(\alpha') = \hat{d}(\alpha) + 1$.

The base cases of lines 3–5 are trivial. For the induction step, let v be the lowest minimal $\perp$-node chosen on line 7. By Lemma 6, if α^* is an optimal completion of α and $s = \alpha^*(v)$, then $\mu_\alpha(v) \leq s \leq par^{\lceil \delta/\lambda \rceil}(\mu_\alpha(v))$. We try all the $\lceil \delta/\lambda \rceil$ possibilities in the for-loop on line 9. The for-loop on line 11 is justified by Lemma 10, and the for-loop on line 13 is justified by Lemmas 8 and 9. Assuming that α' is clean on line 15, by induction the recursive call will return the cost of an optimal completion α^* of α' having $\hat{d}(\alpha^*) \leq d$, if any such completion exists. It remains to argue that for every α' sent to a recursive call on line 15, α' is clean and $\hat{d}(\alpha') = \hat{d}(\alpha) + 1$.

[2] There is a subtlety to consider here. What we have shown is that if there exists a mapping α of minimum cost $cost^{SD}(\mathcal{G}, S)$ with $\hat{d}(\alpha) \leq d$, then the algorithm finds it. It might be that a reconciliation α satisfying $\hat{d}(\alpha) \leq d$ exists, but that the algorithm returns no solution. This can happen in the case that α is not of cost $cost^{SD}(\mathcal{G}, S)$.

Let us first show that such a α' is clean, for each choice of s on line 9. There clearly cannot be an easy node under α' after line 13, so we must show C2, i.e. that for any minimal $\perp$-node w under α', there is no duplication z under α' satisfying $\alpha'(z) > \mu_{\alpha'}(w)$. Suppose instead that $\alpha'(z) > \mu_{\alpha'}(w)$ for some duplication node z. Let w_0 be a descendant of w that is a minimal $\perp$-node in α (note that $w_0 = w$ is possible). We must have $\mu_{\alpha'}(w) \geq \mu_\alpha(w_0)$. By our assumption, we then have $\alpha'(z) > \mu_{\alpha'}(w) \geq \mu_\alpha(w_0)$. Then z cannot be a duplication under α, as otherwise α itself could not be clean (by C2 applied on z and w_0). Thus z is a newly introduced duplication in α', and so z was a $\perp$-node under α. Note that Algorithm 1 maps $\perp$-nodes of $\mathcal{G}$ one after another, in some order $(z_1, z_2, \ldots, z_k)$. Suppose without loss of generality that z is the first duplication node in this ordering that gets mapped to $\alpha'(z)$. There are two cases: either $\alpha'(z) \neq s$, or $\alpha'(z) = s$.

Suppose first that $\alpha'(z) \neq s$. Lines 10 and 11 can only map $\perp$-nodes to s, and line 13 either maps speciation nodes, or easy nodes that become duplications. Thus when $\alpha'(z) \neq s$, we may assume that z falls into the latter case, i.e. z is easy before being mapped, so that mapping z to $\alpha'(z)$ does not increase $h_{\alpha'}(\alpha'(z))$. Because z is the first $\perp$-node that gets mapped to $\alpha'(z)$, this is only possible if there was already a duplication z_0 mapped to $\alpha'(z)$ in α. This implies that $\alpha(z_0) = \alpha'(z) > \mu_\alpha(w_0)$, and that α was not clean (by C2 applied on z_0 and w_0). This is a contradiction.

We may thus assume that $\alpha'(z) = s$. This implies $\mu_{\alpha'}(w) < \alpha'(z) = s$. If w was a minimal $\perp$-node in α, it would have been mapped to s on line 11, and so in this case w cannot also be a minimal $\perp$-node in α', as we supposed. If instead w was not a minimal $\perp$-node in α, then w has a descendant w_0 that was a minimal $\perp$-node under α. We have $\mu_\alpha(w_0) \leq \mu_{\alpha'}(w) < s$, which implies that w_0 gets mapped to s on line 11. This makes $\mu_{\alpha'}(w) < s$ impossible, and we have reached a contradiction. We deduce that z cannot exist, and that α' is clean.

It remains to show that $\hat{d}(\alpha') = \hat{d}(\alpha) + 1$. Again, let s be the chosen species on line 9. Suppose first that $s = \mu_\alpha(v)$. Then $h_{\alpha[v \to s]}(s) = h_\alpha(s) + 1$, as otherwise v would be easy under α, contradicting its cleanness. In this situation, as argued in the proof of Lemma 10, each node w that gets mapped to s on line 11 or on line 13 is easy, and thus cannot further increase the height of the duplications in s. If $s > \mu_\alpha(v)$, then $h_{\alpha[v \to s]}(s) = 1 = h_\alpha(s) + 1$, since by cleanness no duplication under α maps to s. Here, each node w that gets mapped on line 11 has no descendant nor ancestor mapped to s, and thus the height does not increase. Noting that remapping easy nodes on line 13 cannot alter the duplication heights, we get in both cases that $\hat{d}(\alpha[v \to s]) = \hat{d}(\alpha) + 1$. This proves the correctness of the algorithm.

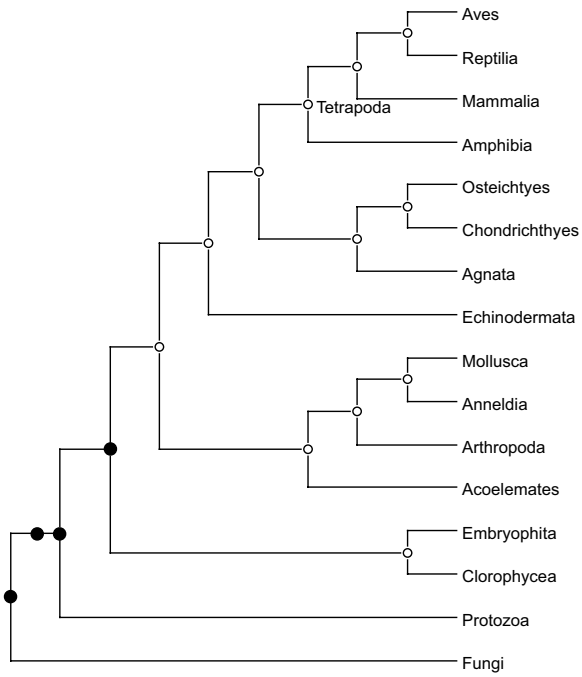

Fig. 6 The species tree phylogeny for the 16 eukaryotes studied by Guigó et al. Black nodes indicate the location of segmental duplications detected by our algorithm (the 2 black circles suggest two consecutive segmental duplications)

As for the complexity, the algorithm creates a search tree of degree $\lceil \delta/\lambda \rceil$ and of depth at most d. Each pass can easily seen to be feasible in time $O(\delta/\lambda \cdot n)$ (with appropriate pre-parsing to compute $\mu_\alpha(v)$ in constant time, and to decide if a node is easy or not in constant time as well), and so the total complexity is $O(\lceil \delta/\lambda \rceil^d n \cdot \frac{\delta}{\lambda})$. □

Experiments

We used our software to reanalyze a data set of 53 gene trees for 16 eukaryotes presented in [14] and already reanalyzed in [1, 25]. In [1], the authors showed that, if segmental duplications are not accounted for, we get a solution having $\hat{d}$ equal to 9, while their software (Exact-MGD) returns a solution with $\hat{d}$ equal to 5. We were able to retrieve the solution with maximum height of 5 fixing $\delta \in [28, 61]$ and $\lambda = 1$, but, as soon as $\delta > 61$, we got a solution with maximum height of 4 where no duplications are placed in the branch leading to the Tetrapoda clade. The result is shown in Fig. 6 (also see [25, Fig. 1]). In [14], the Tetrapoda duplication was only supported by 6 trees (11.3%), whereas in our solution all these

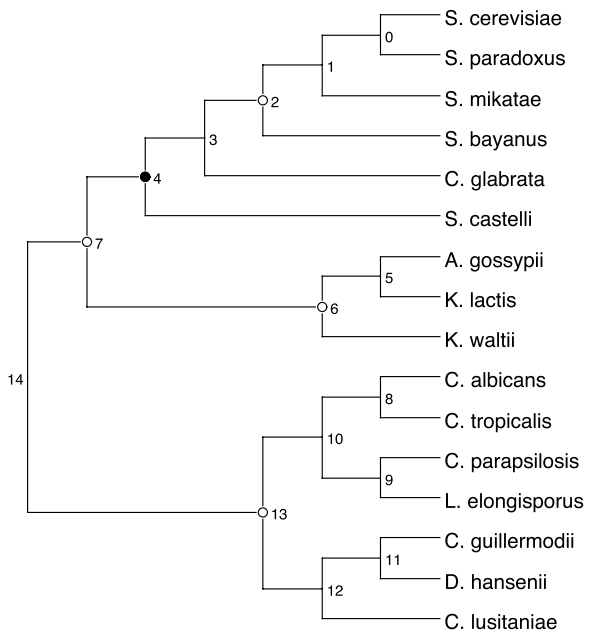

Fig. 7 The species tree phylogeny for the yeast data set described in [2]. Numbers at the internal nodes are meaningless and are only used to refer to the nodes in the main text

duplications were remapped to the common ancestor of Chlorophyceae and Mammalia. The other locations of segmental duplications inferred in [14] are confirmed.[3] This may sow some doubt on the actual existence of a segmental duplication in the LCA of the Tetrapoda clade.

We also reanalyzed the data set of yeast species described in [2]. First, we selected from the data set the 2379 gene trees containing all 16 species and refined unsupported branches using the method described in [16] and implemented in ecceTERA [15] with a bootstrap threshold of 0.9 and $\delta = \lambda = 1$. Using our method with $\delta = 1.5, \lambda = 1$ we were able to detect the ancient genome duplication in Saccharomyces cerevisiae already established using synteny information [17], with 216 gene families supporting the event. Other nodes with a signature of segmental duplication are nodes 7, 6, 13 and 2 (refer to Fig. 7) with respectively 190, 157, 148 and 136 gene families supporting the event. It would be interesting to see if the synteny information supports these hypotheses.

Note on the costs: As for any other parsimony method, the costs associated to the events in Algorithm 1 (i.e. δ and λ) have to be fixed by the user and cannot be estimated. Several possibilities exist to estimate these costs. A method for cost estimation for $\mathbb{DL}$ reconciliation, based on reducing large fluctuations in ancient genome sizes, has been proposed in [6]. Another possibility is to use costs estimated by a maximum likelihood method, as done in [23], where the costs estimated in [27, 28] were used. Uncertainty in the input costs can also be tackled by considering several cost vectors and pareto-optimal reconciliations, or even via sampling; examples of these approach can be found respectively in [29] in the context of $\mathbb{DTL}$ reconciliations (i.e. $\mathbb{DL}$ reconciliations where horizontal gene transfers are also allowed) and in [4], where Boltzmann sampling is used in the context of evolution of gene adjacencies, a problem related to reconciliations.

[3] Note that for this data set we used a high value for $\frac{\delta}{\lambda}$ since, because of the sampling strategy, we expect that all relevant genes have been sampled (recall that in ExactMGD, λ is implicitly set to 0).

Conclusion

We have presented an approach for the reconciliation of a set of gene trees and a species tree, based on segmental macro-evolutionary events, where segmental duplication events and losses are associated with cost δ and λ, respectively. We have shown that the problem is polynomial-time solvable when $\delta \leq \lambda$, since LCA-mapping is already an optimal solution. When $\delta > \lambda$ the problem is NP-hard, even when $\lambda = 0$ and a single gene tree is given. This result solves a long standing open problem on the complexity of the reconciliation of a set of gene trees with a species tree. Moreover, we have given a fixed-parameter algorithm of time complexity $O\left(\lceil\frac{\delta}{\lambda}\rceil^d \cdot n \cdot \frac{\delta}{\lambda}\right)$, where d is the number of segmental duplications, that has been tested on real data, showing its effectiveness.

This work poses a variety of questions that deserve further investigation. The complexity of the problem when δ/λ is a constant remains an open problem. Moreover, our FPT algorithm can handle data sets with a sum of duplication height of about $d = 30$, but in the future, one might consider whether there exist fast approximation algorithms for MPRST-SD in order to attain better scalability. Other future directions include a multivariate complexity analysis of the problem, in order to understand whether it is possible to identify other parameters that are small in practice. Finally, we plan to extend the experimental analysis to other data sets, for instance for the detection of whole genome duplications in plants.

Authors' contributions

RD, ML and CS all participated in writing the manuscript and establishing the theoretical results. ML implemented the algorithm, CS ran the experiments. All authors read and approved the final manuscript.

Author details

[1] Dipartimento di Filosofia, Lettere, Comunicazione, Università degli Studi di Bergamo, Bergamo, Italy. [2] Department of Computer Science, Université de Sherbrooke, Sherbrooke, Canada. [3] ISEM, CNRS, IRD, EPHE, Universit de Montpellier, Montpellier, France.

Acknowledgements

The authors would like to thank Mukul Bansal for providing the eukaryotes data set for the experiments section.

Competing interests
The authors declare that they have no competing interests.

Availability of data and materials
The implementation is available at https://github.com/AEVO-lab/MultRec. The data used in the experiments is available on demand.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Funding
The work of ML was supported by a postdoctoral fellowship from the Natural Sciences and Engineering Research Council (NSERC), Canada.

Appendix: Proof of Theorem 2

See Figs. 8 and 9.

We now prove the hardness of the MPRST-SD problem. For convenience, we repeat the illustration of the construction.

Let G and β be a given instance of vertex cover, and let $\mathcal{G}$ and S be constructed as described before. Call a node $v \in V(\mathcal{G})$ an *original duplication* if v is a duplication under μ. If $\mu(v) = s$, we might call v an original s-duplication for more precision. For $T \in \mathcal{G}$ and $t \in V(S)$, suppose there is a unique node $w \in V(T)$ such that $\mu(w) = s$. We then denote w by $T[t]$. In particular, any special node that is present in a tree $T \in \mathcal{G}$ satisfies the property, so when we mention the special uv nodes of T, we refer to the special nodes that are mapped to the corresponding special uv nodes in S under μ. For example in the C_h tree of Fig. 3, the indicated set of $(j-i-1)10K+K$ nodes are

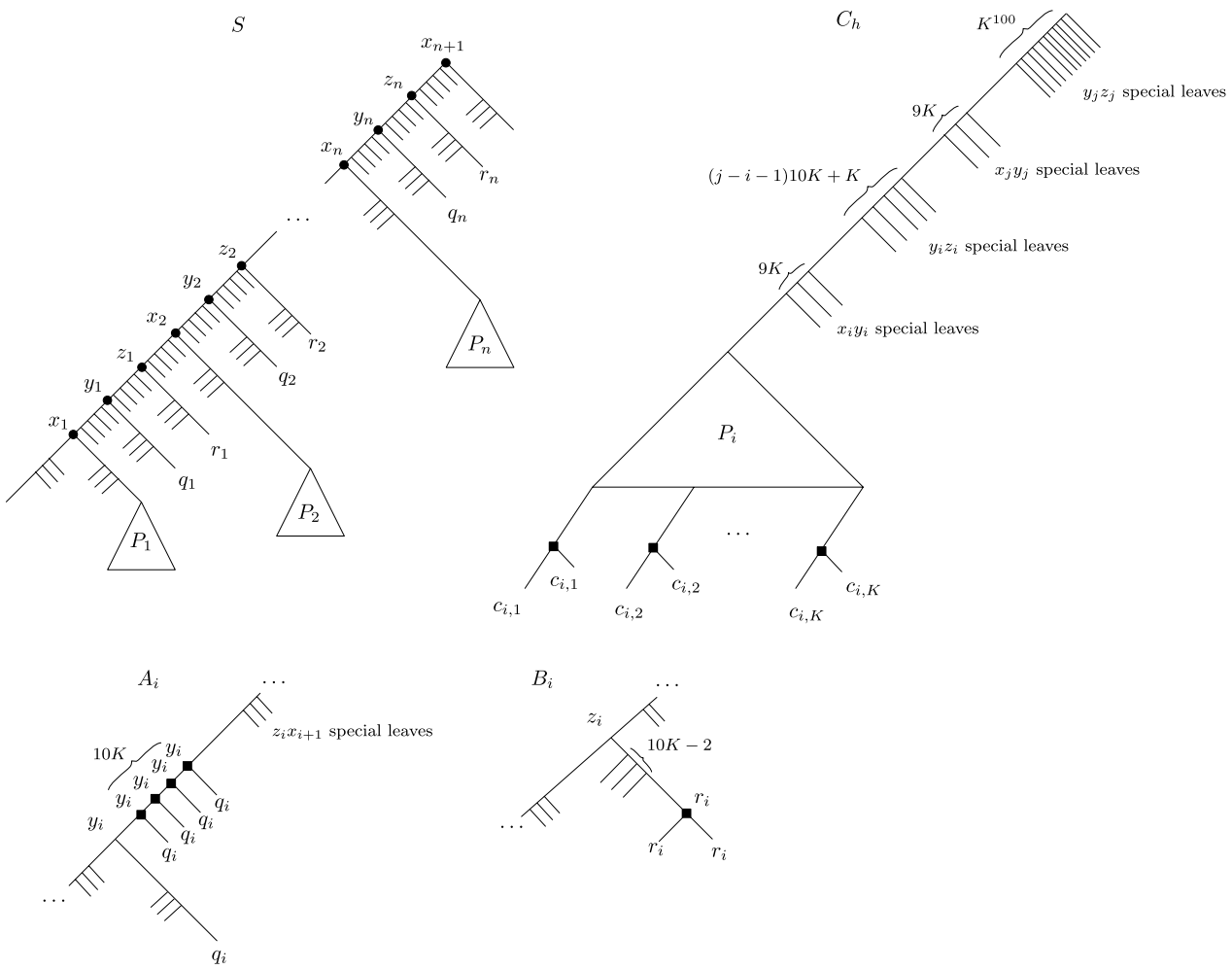

Fig. 8 The species tree S, and the A_i, B_i and C_h trees. The internal nodes are labeled by their LCA-mapping μ, and black squares on the gene trees represent duplication nodes under μ

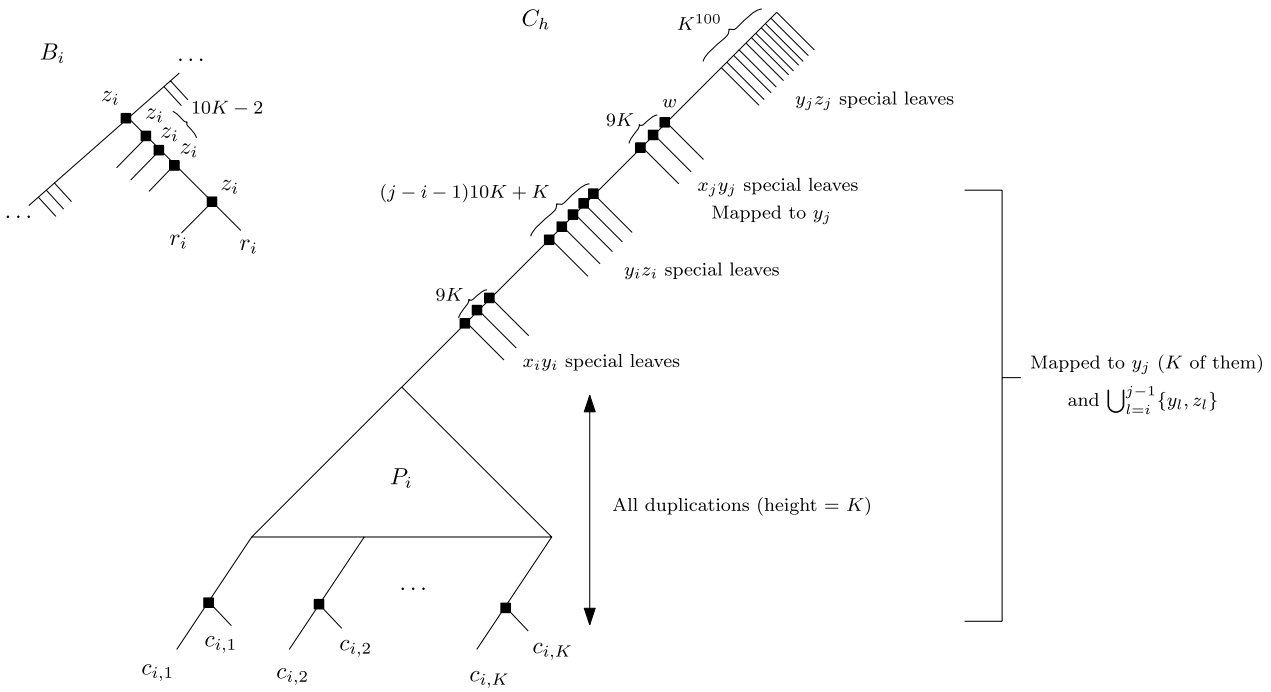

Fig. 9 Mapping for the B_i tree when $v_i \notin V'$, and the C_h tree when $v_i \notin V'$ but $v_j \in V'$

called special $y_i z_i$ nodes as they are mapped to the special $y_i z_i$ nodes of S under μ.

We now show that G has a vertex cover of size β if and only if $\mathcal{G}$ and S admit a mapping α of cost at most $10Kn + \beta$.

($\Rightarrow$) Suppose that $V' = \{v_{a_1}, \ldots, v_{a_\beta}\}$ is a vertex cover of G. We describe a mapping α such that for each $i \in [n]$:

- If $v_i \in V'$, then $h_\alpha(y_i) = 10K, h_\alpha(r_i) = 1$ and $h_\alpha(z_i) = 0$;
- If $v_i \notin V'$, then $h_\alpha(z_i) = 10K, h_\alpha(r_i) = 0$ and $h_\alpha(y_i) = 0$

and $h_\alpha(w) = 0$ for every other node $w \in V(S)$.

Summing over all $i \in [n]$, a straightforward verification show that this mapping α attains a cost of

$$\sum_{s \in V(S)} h_\alpha(s) = \sum_{v_i \in V'} (10K + 1) + \sum_{v_i \notin V'} 10K = 10Kn + \beta$$

It remains to argue that each tree can be reconciled using these duplications heights. In the remainder, we shall view these duplication heights as "free to use", meaning that we are allowed to create a duplication path of nodes mapped to $s \in V(S)$, as long as this path has at most $h_\alpha(s)$ nodes, using h_α defined above.

Let $A_i \in A$ be one of the A trees, $i \in [n]$. If $v_i \in V'$ is in the vertex cover, then we have put $h_\alpha(y_i) = 10K$. In this case, setting $\alpha(w) = \mu(w)$ for every node w in A_i is a valid mapping in which $h_\alpha(s)$ described above is respected for all $s \in V(S)$ (the only duplications in A_i are those $10K$ mapped to y_i). If instead $v_i \notin V'$, then we may set $\alpha(w) = z_i$ for all the $10K$ original y_i-duplications of A_i, and set $\alpha(w) = \mu(w)$ for every other node. This is easily seen to be valid since the ancestors of the original y_i-duplications in A_i are all proper ancestors of z_i.

Let $B_i \in B$ be a B_i tree, $i \in [n]$. If $v_i \in V'$, then $h_\alpha(r_i) = 1$, and so setting $\alpha(w) = \mu(w)$ for all $w \in V(B_i)$ is valid and respects $h_\alpha(s)$ for all $s \in V(S)$ (since the r_i duplication is mapped to r_i and there are no other duplications). If $v_i \notin V'$, then set $\alpha(w) = z_i$ for every node between the original r_i-duplication in B_i and $B_i[z_i]$ and set $\alpha(w) = \mu(w)$ for every other node w. This creates a path of $10K$ duplications mapped to z_i, which is acceptable since $h_\alpha(z_i) = 10K$. This case is illustrated in Fig. 9.

Let $C_h \in W$ be a C tree, $h \in [m]$. Let v_i, v_j be the two endpoints of edge $e_h = \{v_i, v_j\}$, with $i < j$. Since V' is a vertex cover, we know that one of v_i or v_j is in V'. Suppose first that $v_i \in V'$. In this case, we have set $h_\alpha(y_i) = 10K$. Let w be the highest special $x_i y_i$ node of C_h (i.e. the closest to the root). We set $\alpha(w) = y_i$ for each internal node descending from w. All of these nodes become duplications, but the number of nodes of a longest path from w to an internal node descending from w is $10K$ (K for the R_i subtree, plus $9K$ for the special $x_i y_i$ nodes). Thus having $h_\alpha(y_i)$ is sufficient to cover the whole subtree rooted

at w with duplications. All the proper ancestors of w can retain the LCA mapping and be speciations, since $\mu(w') > y_i$ for all these proper ancestors w'.

Now, let us suppose that $v_i \notin V'$, implying $v_j \in V'$. Note that $h_\alpha(y_l) + h_\alpha(z_l) = 10K$ for all $l \in [n]$. This time, let w be highest special x_jy_j node of C_h. Again, we will make w and all its internal node descendants duplications. We map them to the set $\{y_j\} \cup \bigcup_{l=i}^{j-1}\{y_l, z_l\}$. This case is illustrated in Fig. 9.

More specifically, the longest path from w to a descending internal node contains $9K + (j-i-1)10K + K + 9K + K = (j-i+1)10K$, where we have counted the special x_jy_j nodes, the special y_iz_i nodes, the special x_iy_i nodes and the P_i subtree nodes. We have $h_\alpha(y_j) + \sum_{l=i}^{j-1}(h_\alpha(y_l) + h_\alpha(z_l)) = 10K + (j-i)10K = (j-i+1)10K$, just enough to map the whole subtree rooted at w to duplications. It is easy to see that such a mapping can be made valid by first mapping the $9K$ special x_jy_j nodes to y_j, then the other nodes descending from w to the rest of $\{y_j\} \cup \bigcup_{l=i}^{j-1}\{y_l, z_l\}$. This is because all these nodes are ancestors of the special y_iz_i nodes, the special x_iy_i nodes and the P_i nodes (except y_i, but we have $h_\alpha(y_i) = 0$ anyway).

We have constructed a mapping α with the desired duplication heights, concluding this direction of the proof. Let us proceed with the converse direction.

($\Leftarrow$): suppose that there exists a mapping α of the $\mathcal{G}$ trees of cost at most $10Kn + \beta$. We show that there exists a vertex cover of size at most β in G. For some $X \subseteq V(S)$, define $h_\alpha(X) = \sum_{x\in X} h_\alpha(x)$. For each $i \in [n]$, define the sets

$$Y_i = \{s \in V(S) : y_i \leq s < z_i\}$$

and

$$Z_i = \{s \in V(S) : z_i \leq s < x_{i+1}\}$$
$$R_i = \{s \in V(S) : r_i \leq s < z_i\}$$

Our goal is to show that the Y_i's for which $h_\alpha(Y_i) > 9K$ correspond to a vertex cover. The proof is divided into a series of claims.

Claim 1 *For each $i \in [n]$, $h_\alpha(Y_i) + h_\alpha(Z_i) \geq 10K$.*

Proof Consider the A_i tree of A, and let y_i^* be an original y_i-duplication in A_i. If $\alpha(y_i^*) \geq x_{i+1}$, then every node of A_i on the path from y_i^* to $A_i[x_{i+1}]$ is a duplication. This includes all the K^{100} special z_ix_{i+1} nodes, contradicting the cost of α. Thus $y_i \leq \alpha(y_i^*) < x_{i+1}$. That is, $\alpha(y_i) \in Y_i \cup Z_i$. As this is true for all the original y_i-duplications of A_i, and because Y_i and Z_i are disjoint sets, this shows that $h_\alpha(Y_i) + h_\alpha(Z_i) \geq 10K$. □

The above claim shows that we already need a duplication height of $10Kn$ just for the union of the Y_i and Z_i nodes. This implies the following.

Claim 2 *There are at most β duplication in $\mathcal{G}$ that are not mapped to a node in $\bigcup_{i\in[n]}(Y_i \cup Z_i)$. Moreover, for any subset $I \subseteq [n]$, $\sum_{i\in I}(h_\alpha(Y_i) + h_\alpha(Z_i)) \leq 10K|I| + \beta$.*

Proof The first statement follows from Claim 1 and the cost of α. As for the second statement, suppose it does not hold for some $I \subseteq [n]$. Then $\sum_{i\in[n]}(h_\alpha(Y_i) + h_\alpha(Z_i)) = \sum_{i\in I}(h_\alpha(Y_i) + h_\alpha(Z_i)) + \sum_{i\in[n]\setminus I}(h_\alpha(Y_i) + h_\alpha(Z_i)) > 10Kn + \beta$, a contradiction to the cost of α.

Now, let c be an original duplication in some tree of $T \in \mathcal{G}$. The *c-duplication* path $\mathcal{P}(c)$ is the maximal path of T (treated as an undirected graph) that starts at c and contains only ancestors of c that are duplication nodes under α (in other words, we start at c and include it in $\mathcal{P}(c)$, traverse the ancestors one after another and include every duplication node encountered, and then stop when reaching a speciation or the root—every node in $\mathcal{P}(c)$ is a duplication). We will treat $\mathcal{P}(c)$ as a set of nodes. We say that $\mathcal{P}(c)$ ends at node p if $p \in \mathcal{P}(c)$ and $p \geq p'$ for every $p' \in \mathcal{P}(c)$.

Claim 3 *Let $C_h \in C$ and let v_i, v_j be the two endpoints of edge e_h, with $i < j$. Then there is an original duplication $c \in V(C_h)$ such that $\mathcal{P}(c)$ ends at a node p with $\alpha(p) \geq y_i$.*

Proof Let $c_1, \ldots, c_K$ be the original duplication nodes in the C_h tree, which belong to the P_i subtree of C_h. Assume the claim is false, and that $\alpha(c_k) < y_i$ for every $k \in [K]$. First observe that if $\alpha(c_k) \neq \alpha(c_{k'})$ for every pair $c_k, c_{k'}$ of original duplications in C_h, then $\sum_{k\in[K]} h_\alpha(\alpha(c_k)) \geq K$. Note that $\alpha(c_k) \in Y_l \cup Z_l$ is impossible for $l < i$, by the placement of the $c_{i,k}$ leaves in P_i in the species tree S. As we further assume that $\alpha(c_k) < y_i$ for every k, none of the c_k duplications is mapped to a member of $\bigcup_{l=i}^{n}(Y_l \cup Z_l)$ either. Therefore, none of the c_k duplications is counted in Claim 1, so this implies a cost of at least $10nK + K > 10nK + \beta$, a contradiction. So we may assume that $\alpha(c_k) = \alpha(c_{k'})$ for some distinct original duplications $c_k, c_{k'}$. Notice that $\alpha(c_k) = \alpha(c_{k'})$ must be a common ancestor of $\mu(c_k)$ and $\mu(c_{k'})$. This implies that every node on the path between c_k and $LCA_{C_h}(c_k, c_{k'})$ is a duplication (by Lemma 3), which in turn implies $|\mathcal{P}(c_k)| \geq K/2$, by the construction of P_i. By assumption, no duplication of $\mathcal{P}(c_k)$ is mapped to a member of $\bigcup_{l=1}^{n}(Y_l \cup Z_l)$, and again due to Claim 1, the total cost of α is at least $10nK + K/2 > 10nK + \beta$, a contradiction. □

We can now show that the sets Y_i for which $h_\alpha(Y_i) > 9K$ correspond to a vertex cover.

Claim 4 *Let $e_h \in E$, and let v_i, v_j be the two endpoints of e_h, with $i < j$. Then at least one of $h_\alpha(Y_i) > 9K$ or $h_\alpha(Y_j) > 9K$ holds.*

Proof Assume that $h_\alpha(Y_i) \leq 9K$. By Claim 3, there is an original duplication c in C_h such that $\mathcal{P}(c)$ ends at a node p satisfying $\alpha(p) \geq y_i$. This implies that every special x_iy_i node in C_h is a duplication (by Lemma 3). Thus $\mathcal{P}(c)$ contains all these $9K$ nodes, plus K nodes from the R_i subtree. Since $h_\alpha(Y_i) \leq 9K$, at least K of these nodes are mapped to a node outside Y_i. Call U this set of K nodes that are not mapped in Y_i. By the placement of the P_i subtree, none of the nodes of U is mapped to an element of $Y_l \cup Z_l$ for $l < i$. Also by Claim 2, at most β of the U nodes are mapped to a node outside of $W := Z_i \cup \bigcup_{l=i+1}^{n}(Y_i \cup Z_i)$, so it follows that at least $K - \beta$ nodes of U are mapped to an element of W. Because all the elements of W are ancestors of the special y_iz_i nodes, this implies that all the special y_iz_i nodes in C_h are duplications, of which there are $(j - i - 1)10K + K$.

So far, this makes at least $10K + 10K(j - i - 1) + K = (j - i)10K + K$ duplications in $\mathcal{P}(c)$. Let us now argue that at least one of these is mapped to an ancestor of y_j (not necessarily proper). If not, then all the duplications considered in $\mathcal{P}(c)$ so far are mapped to $\{X\} \cup \bigcup_{l=i}^{j-1}(Y_i \cup Z_i)$, where X is some subset of $V(S)$ satisfying $|X| \leq \beta$. By Claim 2, we know that $h_\alpha(X) + \sum_{l=i}^{j-1}(h_\alpha(Y_i) + h_\alpha(Z_i)) \leq \beta + (j - i)10K + \beta = (j - i)10K + 2\beta < (j - i)10K + K$. In fact, this means that at least $K - 2\beta$ of the duplications considered in $\mathcal{P}(c)$ so far that are unaccounted for, which means that they are mapped to an ancestor of y_j.

Because of this, we now get that the $9K$ special x_jy_j nodes of C_h are duplications (which were not considered in $\mathcal{P}(c)$ so far). That means that at least $K - 2\beta + 9K = 10K - 2\beta$ duplications of $\mathcal{P}(c)$ are mapped to an ancestor of y_j. Notice that $\mathcal{P}(c)$ cannot contain a node w with $\alpha(w) \geq z_j$. Indeed, if this were the case, then all the special y_jz_j nodes of C_h would be duplications, of which there are K^{100}. Thus all the aforementioned $10K - 2\beta$ duplications are mapped to an ancestor of y_j but a proper descendant of z_j, i.e. they are mapped in Y_j. So $h_\alpha(Y_j) \geq 10K - 2\beta > 9K$, proving our claim. □

Let $V' = \{v_i : h_\alpha(v_i) > 9K\}$. Then Claim 4 implies that V' is a vertex cover. It only remains to show that $|V'| \leq \beta$. This will follow from our last claim.

Claim 5 *Suppose that $h_\alpha(Y_i) > 9K$. Then $h_\alpha(Y_i) + h_\alpha(Z_i) + h_\alpha(R_i) \geq 10K + 1$.*

Proof Let r_i^* be the original r_i-duplication in the A_i tree. We show that $\alpha(r_i^*) < z_i$. If $\alpha(r_i^*) \geq z_i$, then all the $10K$ nodes on the path from r_i^* to $A_i[z_i]$, including r_i^* itself, are duplications mapped to a node in Z_i (these nodes cannot be mapped to an ancestor of the Z_i nodes, due to the presence of the special z_ix_{i+1} nodes above $A_i[z_i]$). Thus $h_\alpha(Z_i) \geq 10K$, and so $h_\alpha(Y_i) + h_\alpha(Z_i) > 10K + 9K = 19K$, contradicting Claim 2. It follows that $\alpha(r_i^*) \notin Z_i$. Since we cannot have $\alpha(r_i^*) \in Y_i$, we now know that $\alpha(r_i^*) \notin Y_i \cup Z_i$. Using Claim 1 we have $h_\alpha(Y_i) + h_\alpha(Z_i) + h_\alpha(R_i) \geq 10K + 1$. □

To finish the argument, Claim 5 implies that $|V'| \leq \beta$, since each $v_i \in V'$ implies that $h_\alpha(Y_i) > 9K$ and that $h_\alpha(Y_i) + h_\alpha(Z_i) + h_\alpha(R_i) \geq 10K + 1$. More formally, if we had $|V'| > \beta$, letting $I = \{i : h_\alpha(Y_i) > 9K\}$, with $|I| > \beta$, this would imply $\sum_{i\in[n]}(h_\alpha(Y_i) + h_\alpha(Z_i) + h_\alpha(R_i)) \geq \sum_{i\in I}(h_\alpha(Y_i) + h_\alpha(Z_i) + h_\alpha(R_i)) + \sum_{i\in[n]\setminus I}(h_\alpha(Y_i) + h_\alpha(Z_i)) > 10Kn + \beta$. This concludes the proof.

Publisher's Note

Received: 5 November 2018 Accepted: 23 February 2019
Published online: 20 March 2019

References

1. Bansal MS, Eulenstein O. The multiple gene duplication problem revisited. Bioinformatics. 2008;24(13):132–8.
2. Butler G, Rasmussen MD, Lin MF, Santos Manuel AS, Sakthikumar S, Munro CA, Rheinbay E, Grabherr M, Forche A, Reedy JL, et al. Evolution of pathogenicity and sexual reproduction in eight candida genomes. Nature. 2009;459(7247):657.
3. Cedric C, Nadia E-M. New perspectives on gene family evolution: losses in reconciliation and a link with supertrees. In: Annual international conference on research in computational molecular biology. Springer; 2009. p. 46–58.
4. Chauve C, Ponty Y, Zanetti JPP. Evolution of genes neighborhood within reconciled phylogenies: an ensemble approach. In: Brazilian symposium on bioinformatics. Springer; 2014. p. 49–56.
5. Chauve C, Rafiey A, Davin AA, Scornavacca C, Veber P, Boussau B, Szollosi G, Daubin V, Tannier E. Maxtic: fast ranking of a phylogenetic tree by maximum time consistency with lateral gene transfers. bioRxiv, 2017. https://www.biorxiv.org/content/early/2017/11/07/127548.
6. David LA, Alm EJ. Rapid evolutionary innovation during an archaean genetic expansion. Nature. 2011;469(7328):93.
7. Davín AA, Tannier E, Williams TA, Boussau B, Daubin V, Szöllősi GJ. Gene transfers can date the tree of life. Nat Ecol Evol. 2018;2(5):904.
8. Delabre M, El-Mabrouk N, Huber KT, Lafond M, Moulton V, Noutahi E, Castellanos MS. Reconstructing the history of syntenies through super-reconciliation. In: RECOMB international conference on comparative genomics. Springer; 2018. 179–195.
9. Dondi R, Lafond M, El-Mabrouk N. Approximating the correction of weighted and unweighted orthology and paralogy relations. Algorith Mol Biol. 2017;12(1):1–4. https://doi.org/10.1186/s13015-017-0096-x.
10. Duchemin W. Phylogeny of dependencies and dependencies of phylogenies in genes and genomes. Ph.D. thesis, Universit de Lyon, 2017.
11. Duchemin W, Anselmetti Y, Patterson M, Ponty Y, Bérard S, Chauve C, Scornavacca C, Daubin V, Tannier E. Decostar: reconstructing the ancestral

organization of genes or genomes using reconciled phylogenies. Genome Biol Evol. 2017;9(5):1312–9.
12. Fellows M, Hallett M, Stege U. On the multiple gene duplication problem. In: International symposium on algorithms and computation. Springer; 1998. p. 348–357.
13. Goodman M, Czelusniak J, Moore GW, Romero-Herrera AE, Matsuda G. Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. Syst Biol. 1979;28(2):132–63.
14. Guigo R, Muchnik I, Smith TF. Reconstruction of ancient molecular phylogeny. Mol Phylogenet Evol. 1996;6(2):189–213.
15. Jacox E, Chauve C, Szllsi GJ, Ponty Y, Scornavacca C. Eccetera: comprehensive gene tree-species tree reconciliation using parsimony. Bioinformatics. 2016;32(13):2056–8. https://doi.org/10.1093/bioinformatics/btw105.
16. Jacox E, Weller M, Tannier E, Scornavacca C. Resolution and reconciliation of non-binary gene trees with transfers, duplications and losses. Bioinformatics. 2017;33(7):980–7.
17. Kellis M, Birren BW, Lander ES. Proof and evolutionary analysis of ancient genome duplication in the yeast saccharomyces cerevisiae. Nature. 2004;428(6983):617.
18. Lafond M, Dondi R, El-Mabrouk N. The link between orthology relations and gene trees: a correction perspective. Algorith Mol Biol. 2016;11:4. https://doi.org/10.1186/s13015-016-0067-7.
19. Lafond M, Miardan MM, Sankoff D. Accurate prediction of orthologs in the presence of divergence after duplication. Bioinformatics. 2018;34(13):i366–75.
20. Luo CW, Chen MC, Chen YC, Yang RW, Liu HF, Chao KM. Linear-time algorithms for the multiple gene duplication problems. IEEE/ACM Trans Comput Biol Bioinf. 2011;8(1):260–5.
21. Ma B, Li M, Zhang L. From gene trees to species trees. SIAM J Comput. 2000;30(3):729–52.
22. Maddison WP. Gene trees in species trees. Syst Biol. 1997;46(3):523–36.
23. Nguyen TH, Ranwez V, Pointet S, Chifolleau AMA, Doyon JP, Berry V. Reconciliation and local gene tree rearrangement can be of mutual profit. Algorith Mol Biol. 2013;8(1):12.
24. Page RDM. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. Syst Biol. 1994;43(1):58–77.
25. Page RDM, Cotton JA. Vertebrate phylogenomics: reconciled trees and gene duplications. Pac Symp Biocomput. 2002;7:536–47.
26. Paszek J, Gorecki P. Efficient algorithms for genomic duplication models. IEEE/ACM Trans Comput Biol Bioinf. 2017;15(5):1515–24.
27. Szöllősi GJ, Boussau B, Abby SS, Tannier E, Daubin V. Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. In: Proceedings of the national academy of sciences, 2012. p. 201202997.
28. Szöllősi GJ, Daubin V. Modeling gene family evolution and reconciling phylogenetic discord. In: Evolutionary genomics. Springer; 2012. p. 29–51.
29. To TH, Jacox E, Ranwez V, Scornavacca C. A fast method for calculating reliable event supports in tree reconciliations via pareto optimality. BMC Bioinf. 2015;16(1):384.
30. Ullah I, Sjöstrand J, Andersson P, Sennblad B, Lagergren J. Integrating sequence evolution into probabilistic orthology analysis. Syst Biol. 2015;64(6):969–82.