Reconciling Multiple Genes Trees via Segmental Duplications and Losses

Riccardo Dondi
Dipartimento di Filosofia, Lettere, Comunicazione, Università degli Studi di Bergamo, Bergamo, Italy,
riccardo.dondi@unibg.it

Manuel Lafond
Department of Computer Science, Université de Sherbrooke, Québec, Canada,
manuel.lafond@USherbrooke.ca

Celine Scornavacca
ISEM, CNRS, Université de Montpellier, IRD, EPHE, Montpellier, France,
celine.scornavacca@umontpellier.fr

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 the reconciliation problem. 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 $O(\lceil \frac{\delta}{\lambda} \rceil^d \cdot n \cdot \frac{\delta}{\lambda})$. 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 refute hypothetical segmental duplications on a set of 16 eukaryotes, then show how we can detect whole genome duplications in yeast genomes.

2012 ACM Subject Classification Applied computing → Computational biology, Theory of computation → Fixed parameter tractability, Theory of computation → Problems, reductions and completeness

Keywords and phrases Gene trees/species tree reconciliation, phylogenetics, computational complexity, fixed-parameter algorithms

Digital Object Identifier 10.4230/LIPIcs.WABI.2018.5

Related Version A full version of the paper is available at `https://arxiv.org/abs/1806.03988`.

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

1 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,

© Riccardo Dondi, Manuel Lafond, and Celine Scornavacca;
licensed under Creative Commons License CC-BY

to only name a few [17]. 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 gather hints to unveil the macro-evolutionary events that shaped gene evolution. This is the rationale behind the *species tree-gene tree reconciliation*, a concept introduced in [9] and first formally defined in [18]. Gathering intelligence on these macro-evolutionary events permits us to better understand the mechanisms of evolution with applications ranging from orthology detection [14, 21] to ancestral genome reconstruction [7], and recently in dating phylogenies [5, 4].

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. [7]. A more pertinent approach would be to reconcile the set of gene trees at once and consider *segmental* macro-evolutionary events, i.e. 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 [10, 1], and the MGD (*Multiple Gene Duplication*) problem [8]. 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 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 Section 2.3). Most of the exact solutions proposed for the ME problem [1, 15, 20] 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 [20], 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.

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 [20]). 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 $O(\lceil \frac{\delta}{\lambda} \rceil^d \cdot n \cdot \frac{\delta}{\lambda})$. The algorithm has been implemented and tested and is available[1] at `https://github.com/manuellafond/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 [6].

2 Preliminaries

2.1 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.

2.2 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.

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

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 [18]). 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 $d(\alpha) \cdot \delta + l(\alpha) \cdot \lambda$ [16], 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].

2.3 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 the 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 Figure 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 Figure 1, under the mapping μ in

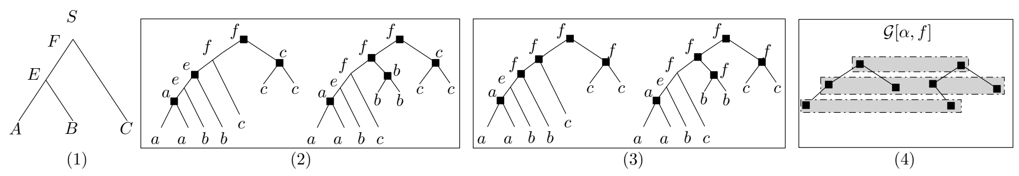

Figure 1 (1) A species tree S. (2) A gene forest $\mathcal{G}$ with two gene trees reconciled under the MPR that we denote μ. The nodes are labeled by the lowercase name of the species they are mapped to. Black squares indicate duplication nodes. Losses are not shown. (3) The same forest $\mathcal{G}$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

(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 :

Most Parsimonious Reconciliation of a Set of Trees with Segmental Duplications (MPRST-SD)
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 [20] (where it is called the FHS model).

2.4 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 2.** *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 2 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 2.

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.

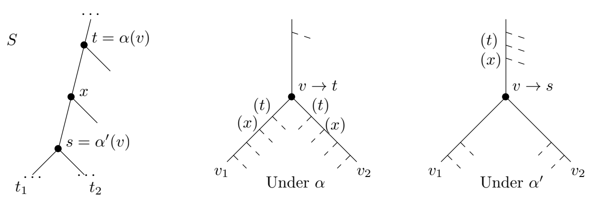

Figure 2 The Shift-down lemma in action. Here $t = par^2(s)$, and remapping v from t to s saves 2 losses – 4 losses are saved below v and 2 are added above.

▶ **Lemma 3.** *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 4.** *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 3. ◀

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 Figure 2). For future reference, do note however that this may increase the height of some duplication forest $\mathcal{G}[\alpha, s]$.

▶ **Lemma 5** (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 Figure 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 3). 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 $l(\alpha') \leq l(\alpha) - k$. ◀

An illustration of the Shift-down lemma can be found in Figure 2.

3 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 $\lambda < \delta$.

3.1 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 6.** *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 $\alpha' = \alpha[v \rightarrow s]$. 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. ◀

3.2 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. The hardness proof is quite technical, and we refer the interested reader to the Appendix of the arXiv version of the paper (`https://arxiv.org/abs/1806.03988`) for the details.

We briefly outline the main ideas of the reduction. The reduction is 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. We also have a gene tree C_h in $\mathcal{G}$ for each edge $e_h \in E(G)$, with say v_i and v_j being the endpoints of e_h. In C_h, there is a large set of duplications $\mathcal{D}$ under the LCA-mapping μ. We make it so that if we either mapped the duplications in A_i or A_j to y_i or y_j, respectively, then we may map all the $\mathcal{D}$ nodes to y_i or y_j without adding more duplications. However if we did not choose y_i nor y_j, then it will not be possible to remap the $\mathcal{D}$ nodes without incurring a large duplication cost. Therefore, the goal becomes to choose a minimum number of y_i's from the A_i trees so that for each edge $e_h = \{v_i, v_j\}$, one of y_i or y_j is chosen for the tree C_h. This establishes the correspondence with the vertex cover instance.

▶ **Theorem 7.** *The* MPRST-SD *problem is NP-hard for $\lambda = 0$ and for given $\delta > \lambda$.*

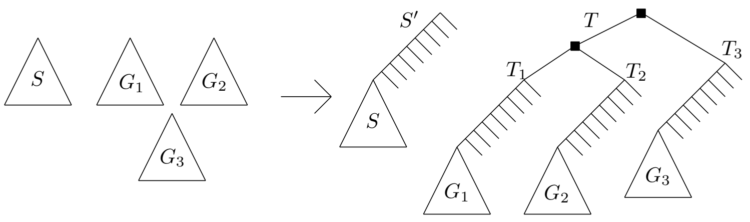

Figure 3 The construction of S' and T from S and the set of gene trees $G_1, \ldots, G_k$ (here $k = 3$). The black squares indicate the path of $k - 1$ duplications that must be mapped to $r(S')$.

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 even if only one gene tree is given. The idea is to reduce from the MPRST-SD show 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 8.** *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 Figure 3). 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'.

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 4), 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 3).

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,

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

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. ◀

4 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 $\alpha(v) \neq \bot$. Note that such a completion always exists, as in particular one can map every $\bot$-node to the root of S (such a mapping must be valid, since all ancestors of a $\bot$-node are also $\bot$-nodes, which ensures that $r(S) = \alpha'(v) \geq \alpha'(v')$ for every descendant v' of a newly mapped $\bot$-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 $\bot$-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 $\bot$-node, then in any completion α' of α, $\alpha'[v \to \mu_\alpha(v)]$ is a valid mapping. A minimal $\bot$-node v is called a *lowest minimal $\bot$-node* if, for every minimal $\bot$-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 $\bot$-node.

▶ **Lemma 9.** *Let α be a partial mapping and let v be a minimal $\bot$-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 10.** *Let v be a minimal $\bot$-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 3. 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. ◀

Lemma 11 and Lemma 12 allow us to find minimal $\bot$-nodes of $\mathcal{G}$ that are the easiest to deal with, as their mapping in an optimal completion can be determined with certainty.

▶ **Lemma 11.** *Let v be a minimal $\bot$-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 10 we have $\alpha(v_1) < \mu(v)$ and $\alpha(v_2) < \mu(v)$ and, as argued in the proof of Lemma 10, $\alpha(v_1)$ 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 12.** *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_v). 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 11 or Lemma 12. Formally, v is easy if either v is a speciation mapped to $\mu_\alpha(v)$ under any optimal completion of α (Lemma 11), or $\hat{d}(\alpha) = \hat{d}(\alpha[v \to \mu_\alpha(v)])$ (Lemma 12). 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 13.** *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 12. 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

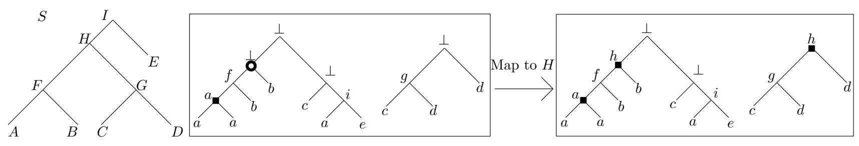

Figure 4 An illustration of one pass through the algorithm. The species tree S is left and $\mathcal{G}$ has two trees (middle) and has partial mapping α (labels are the lowercase of the species). Here α is in a clean state, and the algorithm will pick a lowest minimal $\perp$-node (white circle) and try to map it to, say, H. The forest on the right is the state of α after applying this and cleaning up.

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 13 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 Figure 4.

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 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.

See below for the proof of correctness. 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].

[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 $cost^{SD}(\mathcal{G}, S)$.

Algorithm 1 FPT algorithm for parameter d.

```
1:  procedure SuperReconcile($\mathcal{G}, S, \alpha, d$)
    $\mathcal{G}$ is the set of input trees, $S$ is the species tree, $\alpha$ is a clean partial mapping, $d$ is the maximum value of $\hat{d}(\alpha)$.
2:      if $\hat{d}(\alpha) > d$ then
3:          Return $\infty$
4:      else if $\alpha$ 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 $\mu_\alpha(v) \le s \le par^{\lceil \delta/\lambda \rceil}(\mu_\alpha(v))$ do
10:             Let $\alpha' = \alpha[v \to s]$
11:             for minimal $\perp$-node $w \ne v$ under $\alpha$ such that $\mu_\alpha(w) \le s$ do
12:                 Set $\alpha' = \alpha'[w \to s]$
13:             while there is a minimal $\perp$-node $w$ that is easy under $\alpha'$ do
14:                 Set $\alpha' = \alpha'[w \to \mu_{\alpha'}(w)]$
15:             $cost \leftarrow superReconcile(\mathcal{G}, S, \alpha', d)$
16:             if $cost < bestCost$ then $bestCost \leftarrow cost$
17:         Return $bestCost$
```

▶ **Theorem 14.** *Algorithm 1 is correct and finds a minimum cost mapping α^* satisfying $\hat{d}(\alpha^*) \le d$, 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 an optimal completion α^* of α having $\hat{d}(\alpha^*) \le d$, 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 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 9, if α^* is an optimal completion of α and $s = \alpha^*(v)$, then $\mu_\alpha(v) \le s \le par^{\lceil \delta/\lambda \rceil}(\mu_\alpha(v))$. 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 13, and the for-loop on line 13 is justified by Lemma 11 and Lemma 12. 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 $\hat{d}(\alpha^*) \le d$, 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 $\hat{d}(\alpha') = \hat{d}(\alpha) + 1$.

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 $\mu_{\alpha'}(w) \ge \mu_\alpha(w_0)$. By our assumption, we then have $\alpha'(z) > \mu_{\alpha'}(w) \ge \mu_\alpha(w_0)$. 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 13, 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.

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 $O(\lceil \delta/\lambda \rceil^d n \cdot \frac{\delta}{\lambda})$. ◀

5 Experiments

We used our software to reanalyze a data set of 53 gene trees for 16 eukaryotes presented in [10] and already reanalyzed in [19, 1]. 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 (ExactMGD) 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 (see [19, Fig. 1]) while the other locations of segmental duplications inferred in [10] 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 [12] and implemented in ecceTERA [11] 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 [13], with 216 gene families supporting the event. Other nodes with a

[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).

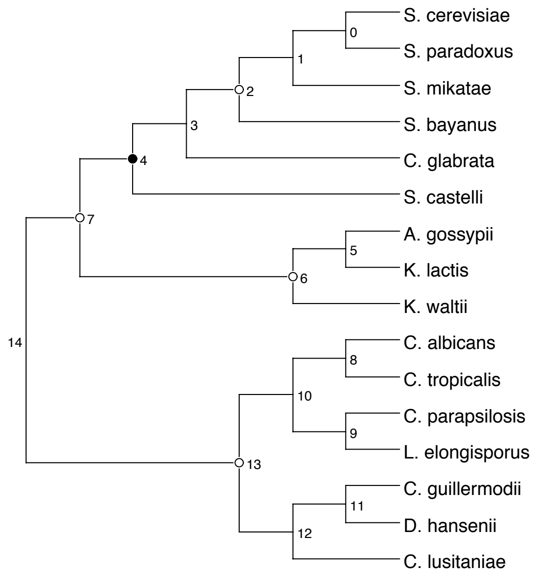

Figure 5 The species tree phylogeny for the yeast data set described in [2]. Numbers at the internal nodes are meaningless and are only used to refer to the nodes in the main text.

signature of segmental duplication are nodes 7, 6, 13 and 2 (refer to Fig. 5) 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.

6 Conclusion

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.

References

1 Mukul S Bansal and Oliver Eulenstein. The multiple gene duplication problem revisited. *Bioinformatics*, 24(13):i132–i138, 2008.

2 Geraldine Butler, Matthew D Rasmussen, Michael F Lin, Manuel AS Santos, Sharadha Sakthikumar, Carol A Munro, Esther Rheinbay, Manfred Grabherr, Anja Forche, Jennifer L Reedy, et al. Evolution of pathogenicity and sexual reproduction in eight candida genomes. *Nature*, 459(7247):657, 2009.

3 Cedric Chauve and Nadia El-Mabrouk. New perspectives on gene family evolution: losses in reconciliation and a link with supertrees. In *Annual International Conference on Research in Computational Molecular Biology*, pages 46–58. Springer, 2009.

4 Cedric Chauve, Akbar Rafiey, Adrian A. Davin, Celine Scornavacca, Philippe Veber, Bastien Boussau, Gergely Szollosi, Vincent Daubin, and Eric Tannier. Maxtic: Fast ranking

of a phylogenetic tree by maximum time consistency with lateral gene transfers. *bioRxiv*, 2017. URL: `https://www.biorxiv.org/content/early/2017/11/07/127548`.

5 Adrián A Davín, Eric Tannier, Tom A Williams, Bastien Boussau, Vincent Daubin, and Gergely J Szöllősi. Gene transfers can date the tree of life. *Nature ecology & evolution*, page 1, 2018.

6 Wandrille Duchemin. *Phylogeny of dependencies and dependencies of phylogenies in genes and genomes*. PhD thesis, Université de Lyon, 2017.

7 Wandrille Duchemin, Yoann Anselmetti, Murray Patterson, Yann Ponty, Sèverine Bérard, Cedric Chauve, Celine Scornavacca, Vincent Daubin, and Eric Tannier. Decostar: Reconstructing the ancestral organization of genes or genomes using reconciled phylogenies. *Genome biology and evolution*, 9(5):1312–1319, 2017.

8 Michael Fellows, Michael Hallett, and Ulrike Stege. On the multiple gene duplication problem. In *International Symposium on Algorithms and Computation*, pages 348–357. Springer, 1998.

9 Morris Goodman, John Czelusniak, G William Moore, Alejo E Romero-Herrera, and Genji Matsuda. Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Biology*, 28(2):132–163, 1979.

10 Roderic Guigo, Ilya Muchnik, and Temple F Smith. Reconstruction of ancient molecular phylogeny. *Molecular phylogenetics and evolution*, 6(2):189–213, 1996.

11 Edwin Jacox, Cedric Chauve, Gergely J. Szöllősi, Yann Ponty, and Celine Scornavacca. ecceterа: comprehensive gene tree-species tree reconciliation using parsimony. *Bioinformatics*, 32(13):2056–2058, 2016.

12 Edwin Jacox, Mathias Weller, Eric Tannier, and Céline Scornavacca. Resolution and reconciliation of non-binary gene trees with transfers, duplications and losses. *Bioinformatics*, 33(7):980–987, 2017.

13 Manolis Kellis, Bruce W Birren, and Eric S Lander. Proof and evolutionary analysis of ancient genome duplication in the yeast saccharomyces cerevisiae. *Nature*, 428(6983):617, 2004.

14 Manuel Lafond, Mona Meghdari Miardan, and David Sankoff. Accurate prediction of orthologs in the presence of divergence after duplication. *Bioinformatics*, In press, 2018.

15 Cheng-Wei Luo, Ming-Chiang Chen, Yi-Ching Chen, Roger WL Yang, Hsiao-Fei Liu, and Kun-Mao Chao. Linear-time algorithms for the multiple gene duplication problems. *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, 8(1):260–265, 2011.

16 Bin Ma, Ming Li, and Louxin Zhang. From gene trees to species trees. *SIAM Journal on Computing*, 30(3):729–752, 2000.

17 Wayne P Maddison. Gene trees in species trees. *Systematic biology*, 46(3):523–536, 1997.

18 Roderic DM Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Systematic Biology*, 43(1):58–77, 1994.

19 Roderic DM Page and JA Cotton. Vertebrate phylogenomics: reconciled trees and gene duplications. In *Pacific Symposium on Biocomputing*, volume 7, pages 536–547, 2002.

20 Jaroslaw Paszek and Pawel Gorecki. Efficient algorithms for genomic duplication models. *IEEE/ACM transactions on computational biology and bioinformatics*, 2017.

21 Ikram Ullah, Joel Sjöstrand, Peter Andersson, Bengt Sennblad, and Jens Lagergren. Integrating sequence evolution into probabilistic orthology analysis. *Systematic Biology*, 64(6):969–982, 2015.