manuscript No.
(will be inserted by the editor)

Gene tree reconciliation including transfers with replacement is hard and FPT

Damir Hasić · Eric Tannier

Abstract Phylogenetic trees illustrate the evolutionary history of genes and species. In most cases, although genes evolve along with the species they belong to, a species tree and gene tree are not identical, because of evolutionary events at the gene level like duplication or transfer. These differences are handled by phylogenetic reconciliation, which formally is a mapping between gene tree nodes and species tree nodes and branches. We investigate models of reconciliation with a gene transfer that replaces existing gene, which is a biological important event but never included in reconciliation models. Also the problem is close to a dated version of the classical subtree prune and regraft (SPR) distance problem, where a pruned subtree has to be regrafted only on a branch closer to the root. We prove that the reconciliation problem including transfer and replacement is NP-hard, and that if speciations and transfers with replacement are the only allowed evolutionary events, then it is fixed-parameter tractable (FPT) with respect to the reconciliation's weight. We prove that the results extend to the dated SPR problem.

Keywords phylogenetic reconciliation · dated subtree prune and regraft SPR · gene transfer · transfer with replacement (replacing transfer) · NP hard/complete · fixed parameter tractable FPT

Damir Hasić
Department of Mathematics, Faculty of Science, University of Sarajevo, 71000 Sarajevo, Bosnia and Herzegovina
E-mail: damir.hasic@gmail.com, d.hasic@pmf.unsa.ba

Eric Tannier
Inria Grenoble Rhône-Alpes, F-38334 Montbonnot, France
Univ Lyon, Université Lyon 1, CNRS, Laboratoire de Biométrie et Biologie Évolutive UMR5558, F-69622 Villeurbanne, France

1 Introduction

Duplications and *transfers* are events in evolution of genes, and one of the major reasons for discordance between species and gene trees. These differences are explained by *phylogenetic reconciliation* in Doyon et al. (2011).

The main evolutionary event, investigated in this paper, is *gene transfer* (*horizontal gene transfer*, Figure 1). It involves two (possibly ancient) species existing at the same moment. The species that provides a transfered gene is called a *donor species*, and the species that receives the gene is called a *recipient species*.

As phylogenetic analysis never includes the totality of living species, in particular ancient species which can be extinct or not sampled (see *transfer from the dead* in Szöllősi et al. (2013)), the donor species is not assumed to belong to the species phylogeny, but is related to it through one of its ancestors. By extension, this ancestor is considered as the donor, which yields *diagonal transfers*, or *transfers to the future* (Figure 1 $(b-c)$).

The recipient species either receives a new gene copy, or replaces an existing one (Figure 1 (d)). The latter event is called *replacement transfer* or *transfer with replacement*. In Choi et al. (2012) replacement transfer is called *replacing horizontal gene transfer* (HGT). They found that the replacing HGT and the additive HGT affect differently gene functions in *Streptococcus*. In Rice and Palmer (2006) HGT in plastid genomes is studied and the evidence of transfer with replacement is found. HGT with replacement also occurred in the evolution of eukaryotes (Keeling (2008)).

In this article we explore the algorithmic aspects of transfers with replacements when time constraints are imposed on transfers (*i.e.* they should not be directed to the past).

1.1 A review of previous results

There are two usual ways of detecting transfers by comparing species trees and gene trees. One is reconciliation, the other is the computation of SPR scenarios. Our work lies at the intersection of the two. Indeed, reconciliations can consider time constraints but never include replacing transfers. On the other hand SPR scenarios are a good model for replacing transfers but never consider time constraints.

Time constraints result in —fully or partially— dated species trees. For dated species tree, finding a reconciliation of minimum cost, in a model with gene transfer, is usually polynomial (Merkle et al. 2010; Doyon et al. 2010; Bansal et al. 2012). With undated species tree, and partial time constraints it is usually NP-hard (Hallett and Lagergren 2001; Tofigh et al. 2011; Bansal et al. 2012), and can be fixed parameter tractable (Hallett and Lagergren 2001; Tofigh et al. 2011), or inapproximable (Dasgupta et al. 2006). If a constraint on time consistency of reconciliation scenarios is relaxed, the problem becomes polynomial (Tofigh et al. 2011; Bansal et al. 2012).

There are some results that go beyond finding one optimal solution. In Bansal et al. (2013) an algorithm that uniformly samples the space of optimal solutions is given, and it runs in polynomial time (per sample). In Chan et al. (2015) the space of all optimal solutions is also explored, and formula for the number of optimal solutions is given. Continuous and discrete frameworks for finding a minimum duplication-transfer-loss (DTL) reconciliation are equivalent (Ranwez et al. 2016). Only recently, DTL model is expanded by Chan et al. (2017), where incomplete lineage sorting is included, and FPT algorithm that returns a minimal reconciliation is given. Probabilistic models allow sampling of solutions in larger spaces, according to likelihood distributions Szöllősi et al. (2013).

Until now only reconciliations with transfers without replacements are investigated. However, note that duplications with replacement, *i.e. conversions*, were recently introduced by Hasić and Tannier (2017). For a more detailed review on reconciliations see Szöllősi et al. (2015), Nakhleh (2012), and Doyon et al. (2011).

Transfers that replace existing genes are in close relation to the classical tree rearrangement operation *subtree prune and regraft* (SPR). For the definition of SPR refer to Song (2006) (for the rooted trees) and Allen and Steel (2001) (for the unrooted trees). This operation has never been integrated in reconciliation models but is used to detect transfers when it is the only allowed evolutionary event at the gene scale. In consequence these studies are limited to datasets where genes appear in at most one copy per species.

Computing SPR distance between two rooted binary phylogenetic trees on the same label set is NP-hard. The first proof (Hein et al. 1996) has a mistake (see Allen and Steel (2001)). Nevertheless, the results from Hein et al. (1996) can be used for *tree bisection and reconnection*. The correct proof is in Bordewich and Semple (2005). They used a modified approach from Hein et al. (1996). The problem is also NP-hard for unrooted binary trees (Hickey et al. 2008). If we take the SPR distance as a parameter, then both rooted (Bordewich and Semple 2005) and unrooted (Whidden and Matsen 2015) versions are FPT.

There is an approximation algorithm of ratio 3 (Hein et al. 1996), as well as an ILP algorithm for calculating an exact rooted SPR distance (Wu 2009). An exact rooted SPR distance is also determined by reducing it to CNF (Bonet and John 2009) and using existing SAT solvers.

A probabilistic model of gene transfer with replacement using (undated) SPR, where time consistent and *transfers to the past* are not distinguished, is given by Suchard (2005). Model for host-parasite cophylogeny by Huelsenbeck et al. (2000) could also be used to detect gene transfers between loci.

Rooted SPR distance is equivalent to the number of trees in the *maximal agreement forest* (MAF) (Bordewich and Semple 2005), while in the unrooted version SPR distance is greater than or equal to MAF (Allen and Steel 2001).

A divide and conquer approach with MAF is used for computing an exact SPR distance in Linz and Semple (2011). A 2.5-approximation algorithm for the MAF problem on two rooted binary phylogenetic trees is presented in

Shi et al. (2016). In Chen et al. (2015) an FPT algorithm for rooted SPR, with complexity $O(2.344^k \cdot n)$ is presented, which is an improvement compared to $O(2.42^k \cdot n)$ (Whidden et al. 2010). Rooted SPR is investigated also for non binary trees (Whidden et al. 2016), and MAF for multiple trees (Chen et al. 2016). For a more complete review see Shi et al. (2013) and Whidden et al. (2016).

Dated SPR, that is, SPR distance on a dated species tree, where only contemporaneous or transfers to the future are allowed, is mentioned in Song (2006), where it is investigated how many dated trees are one SPR operation away from a given dated tree. The complexity of the dated SPR distance computation is left open, and has an answer as a consequence of our results here.

1.2 The contribution of this paper

In this paper, we analyze the algorithmic complexity of finding a minimum reconciliation with replacing transfers, in the presence of a dated species tree. If speciations and replacement transfers are the only evolutionary events in the reconciliation, then finding an optimal dated SPR scenario is an equivalent problem.

We define a model of reconciliation with gene transfer followed by a gene replacement, *i.e.* transfered gene replaces a gene that is already present in the recipient species (Figure 1 (*d*)). We will call this event *transfer with replacement*, and it is represented by a transfered gene and a loss (the gene that is replaced). We prove that finding a minimum reconciliation that includes transfer with replacement, as well as transfer, duplication and loss, is NP-hard. If speciation and transfer with replacement are the only allowed events, then it is fixed parameter tractable with respect to the output size, and it is easily reducible to dated SPR problem. Therefore dated SPR is also NP-hard and FPT. Note that the hardness of dated SPR is not easily deduced from the hardness of general SPR because all the known proofs make an extensive usage of the possibility of time inconsistent SPRs.

We prove NP-hardness by a reduction from Max 2-Sat. The gadgets for the variables and clauses are constructed, and used to assemble a reconciliation that we call a *proper reconciliation*. Hence gadgets are sub-reconciliations of a proper reconciliation. Next, we state an obvious claim that relates an optimal Max 2-Sat solution with an optimal proper reconciliation. Then we prove that any optimal reconciliation can be transformed (in polynomial time) into a proper reconciliation of the same weight (therefore optimal).

In order to prove parametrized tractability, we introduce a *normalized reconciliation*. Intuitively, this reconciliation can be obtained from any reconciliation by raising nodes of G as much as possible, that is, mapping them to species tree nodes and edges closer to the root, without affecting transfers, hence keeping the weight of a reconciliation. Then we give a branch and bound algorithm that returns an optimal reconciliation that is also normal-

ized. Thanks to the normalization, we can have at most three cases in the branching algorithm, and every branching produces at least one transfer, so the depth of any branching procedure is at most k, *i.e.* $3^k \cdot n$ is an approximate complexity of the algorithm.

2 Definitions

A *phylogenetic tree* here is a rooted tree T, such that the root vertex $root(T)$ has degree 1, and the incident edge is called *the root edge*, or degree of $root(T)$ is 2 and there is no root edge. With $L(T)$ is denoted the set of all leaves of the tree T. Trees are considered binary, meaning that the nodes have at most two children. We say that they are fully binary when all internal nodes have exactly two children. If x is a node/edge in a rooted tree T, then $p_T(x) = p(x)$ denotes its parent, x_l, x_r denotes its (left and right) children.

If x is an ancestor of y in a rooted tree T, *i.e.* if x is in the path from y to $root(T)$, then we write $y \leq_T x$ or $y \leq x$, defining a partial order on the nodes. Also, let $e_1 = (p(s_1), s_1), e_2 = (p(s_2), s_2) \in E(T)$ and $p(s_1) \leq s_2$, then we can write $s_1 < e_1 < p(s_1) \leq s_2 < e_2 < p(s_2)$. With this, we define a partial order on the set $V(T) \cup E(T)$. If x is a node/edge in T, then $T(x)$ is the maximal rooted subtree with root node/edge x, and if $A \subseteq L(T)$, then $T(A)$ is the subtree of T with a root vertex of degree 2, and $L(T(A)) = A$.

The next definition extends the partial order on the set $V(T) \cup E(T)$ to the total order by introducing the *date function*. Intuitively, to every node and edge from T a date (*i.e.* a point in the past) is assigned. This derives from the fact that phylogenetic trees and reconciliations represent evolutionary events that happened at some point in the past.

Definition 1 (Date function. Dated tree) Let T be a rooted tree and $\tau : V(T) \cup E(T) \to [0, +\infty)$ such that $\tau(L(T)) = \{0\}$, $x_1, x_1 \in V(T) \cup E(T)$, $x_1 < x_2 \implies \tau(x_1) < \tau(x_2)$. Function $\tau = \tau_T$ is a *date function* on the tree T, and T is a *dated tree*.

Note that the edges of T are assigned a date. Although it might seem more natural to assign an interval to an edge, here it is more convenient to assign a point (*i.e.* a date).

By a species tree S we mean a dated, fully binary tree with function $\tau_S = \tau$, and $\tau(s_1) \neq \tau(s_2), \forall s_1, s_2 \in V(S) \backslash L(S), s_1 \neq s_2$. *Subdividing* an edge means that a vertex is added to the edge. Formally, edge $e = (x, y)$ is subdivided if a node z is added to the graph along with edges $(x, z), (y, z)$, and the edge e is removed.

Definition 2 (Subdivision of a species tree) Let S' be a tree obtained from S by subdividing some edges, and $\forall e = (p(s), s) \in E(S)$, and $\forall s_1 \in V(S)$ for which $\tau(s) < \tau(s_1) < \tau(p(s))$, $\exists s' \in V(S') \backslash V(S)$, $\tau(s_1) = \tau(s')$, and $s < s' < p(s)$. Tree S' with these properties, and with the minimum number of nodes is called the *subdivision* of the species tree S.

Note that the node s' from Definition 2 is obtained by subdividing some edge, and $deg(s') = 2$. Subdivision of a species tree is unique (Figure 2). Also, $L(S') = L(S)$ and $root(S') = root(S)$. If $e \in E(S')$, then v_e denotes the maximum element from the set $\{x \in V(S) \mid x < e\}$. We assume that $\tau(V(S') \cup E(S')) = \{0, 1, \ldots, 2n\}$ (see Figure 2), where n is the number of the extant species in S, $\tau(L(S)) = \{0\}$, $\tau(root(S)) = 2n$. Therefore if $x \in V(S') \cup E(S')$, then $\tau(p_{S'}(x)) = \tau(x) + 1$.

We now define gene tree species tree reconciliations. Note G a gene tree, which is a fully binary tree which comes with a mapping $\phi : L(G) \to L(S)$ that indicates the species in which genes are found in the data.

Definition 3 (Extension) A tree T' is said to be an *extension* of a tree T if T can be obtained from T' by pruning some subtrees and suppressing nodes of degree 2.

The next definition is fundamental for the notion of phylogenetic reconciliation. Function ρ indicates positions of genes inside the species tree.

Definition 4 (Semi-reconciliation) Let G' be an extension of a gene tree G, and S is a species tree. Let $\phi : L(G) \to L(S)$, and $\rho : V(G') \to V(S) \cup E(S')$ such that $\rho/L(G) = \phi$ and $\rho(root(G')) = \rho(root(G)) = root(S)$. If $x, y \in V(G')$, $x < y$, and $\rho(x)$ and $\rho(y)$ are comparable in S, then $\rho(x) \leq \rho(y)$. The 6-tuple $\mathfrak{R} = (G, G', S, \phi, \rho, \tau)$ is called *semi-reconciliation*.

Note that the nodes from G' are not mapped into $V(S') \backslash V(S)$. If $\rho(x) = e' \in E(S')$ and e' is a part of $e \in E(S)$, then we will write $\rho(x) \in e$.

The next definition introduces the notion of subtree of G' that is not in G.

Definition 5 (Lost subtree) Let $\mathfrak{R}$ be a reconciliation. A maximal subtree T of G' such that $V(T) \cap V(G) = \emptyset$ is called a *lost subtree*.

Definition 6 (Sub-branch) If G' is an extension of G, $(x_1, x_2) \in E(G)$, $(x'_1, x'_2) \in E(G')$ and $x_2 \leq x'_2 \leq x'_1 \leq x_1$, then we say that (x'_1, x'_2) is *a sub-branch* of (x_1, x_2). Similarly, we can define sub-branch for S and S'.

Semi-reconciliation is a reconciliation without established evolutionary events. The next definitions introduce these events.

Definition 7 (Speciation) Let $\mathfrak{R}$ be a semi-reconciliation, $x \in V(G')$, x'_l, x'_r are the children of x in G'. Let $\rho(x) \in V(S) \backslash L(S)$ and $\rho(x)_l \leq \rho(x'_l) < \rho(x)$, $\rho(x)_r \leq \rho(x'_r) < \rho(x)$. Then x is called a *speciation*. The set of all speciations is denoted by $\Sigma(\mathfrak{R})$ or Σ.

Definition 8 (Duplication) Let $\mathfrak{R}$ be a semi-reconciliation, $x \in V(G')$, x'_l, x'_r are the children of x in G'. Let $\rho(x) = e \in E(S')$, $v_e \leq \rho(x'_l)$, and $v_e \leq \rho(x'_r)$. Then x is called a *duplication*. The set of all duplications is denoted by $\Delta(\mathfrak{R})$ or Δ.

From now on, we will assume $\tau(x) = \tau(\rho(x))$, for all $x \in V(G')$.

Definition 9 (Transfer) Let $\mathfrak{R}$ be a semi-reconciliation, $x \in V(G')$, x'_l, x'_r are the children of x in G', $\rho(x) = e \in E(S')$ and for one of the $\rho(x'_l), \rho(x'_r)$ (say $\rho(x'_l)$) holds $v_e \leq \rho(x'_l)$ and for the other one (i.e. $\rho(x'_r)$) $\rho(x'_r) = e' \in E(S')$, $\tau(e') \leq \tau(e)$, $deg(x'_r) = 2$, and $v_{e'} \leq \rho(x''_r)$, where x''_r is the only child of x'_r in G'. Then x is called a *transfer parent*, x'_r is a *transfer child*, and the edge $e = (x, x'_r) \in E(G')$ is a *transfer*. If $\tau(x'_r) = \tau(x)$, the transfer is *horizontal transfer*, and if $\tau(x'_r) < \tau(x)$, the transfer is *diagonal transfer* or *transfer to the future*. The set of all transfers is denoted by $\Theta(\mathfrak{R})$ or Θ.

Definition 10 (Loss) Let $\mathfrak{R}$ be a semi-reconciliation, and $x \in L(G')\backslash L(G)$. Then x is called a *loss*. The set of all losses is denoted by $\Lambda(\mathfrak{R})$ or Λ.

The next two events that we are going to define are created by pairing some of the previously defined events with a loss.

Definition 11 (Replacement transfer) Let $(G, G', S, \phi, \rho, \tau)$ be a semi-reconciliation, $\delta_T : \Theta \to \Lambda$ is an injective partial function such that $\rho(x_2) = \rho(\delta_T(e))$ for all $e \in \delta_T^{-1}(\Lambda)$, where $e = (x_1, x_2) \in \Theta$. If $e \in \delta_T^{-1}(\Lambda)$, then e is called a *replacement transfer* or *transfer with replacement*, and $l = \delta_T(e)$ is its associate loss. The set of all replacement transfers is denoted by Θ' and the set of all associate losses by Λ'_T.

In the previous definition, mapping δ_T pairs transfer e (or we can say a transfer child x_2) with the loss l (see Figure 1). In this way, we get that gene x_2 is replacing gene l, hence the name *transfer with replacement*. Requirement $\rho(x_2) = \rho(l)$ is necessary if x_2 replaces l.

Conversion is to duplication what replacement transfer is to transfer.

Definition 12 (Conversion) Let $(G, G', S, \phi, \rho, \tau)$ be a semi-reconciliation, $\delta_D : \Delta \to \Lambda$ is an injective partial function such that $\rho(x) = \rho(\delta_D(x))$ for all $x \in \delta_D^{-1}(\Lambda)$. If $x \in \delta_D^{-1}(\Lambda)$, then x is called a *conversion*, and $\delta_D(x)$ is its associate loss. The set of all conversions is denoted by Δ' and the set of associate losses by Λ'_D.

The elements of $\Lambda' = \Lambda'_T \cup \Lambda'_D$ are called *free losses*. The set of all (evolutionary) events is $\{S, D, T, L, C, T_R\}$.

Definition 13 (Reconciliation) Let $(G, G', S, \phi, \rho, \tau)$ be a semi-reconciliation, and $A \subseteq \{D, T, L, C, T_R\}$. To every node from $V(G')\backslash L(G)$ some event from $A \cup \{S\}$ is attached. Then $\mathfrak{R} = (G, G', S, \phi, \rho, \tau, \delta_T, \delta_D, A)$ is called *A reconciliation*.

If transfers with replacement or conversions are not included in a reconciliation, then $\delta_T^{-1}(\Lambda) = \emptyset$, or $\delta_D^{-1}(\Lambda) = \emptyset$. Note that if $x \in V(G')$ and $deg(x) = 2$, then $(p_{G'}(x), x) \in \Theta(\mathfrak{R})$.

Speciations, duplications, transfers, losses, conversions, and transfers with replacement are called *evolutionary events*. A reconciliation can allow only some of these events. For example, if a reconciliation $\mathfrak{R}$ allows speciations, duplications and losses, we will call it *DL reconciliation*. If $\mathfrak{R}$ also allows transfers,

we call it *DTL reconciliation*. Speciations are assumed to be allowed in every reconciliation, so they are not emphasized in the type of a reconciliation. If transfers are not allowed in a reconciliation, then the date function is not necessary, and can be disregarded. Note that if $A \subseteq B$, then any A reconciliation is also a B reconciliation. If conversions or transfers with replacement are included in a reconciliation, then we assume that free losses are allowed. Therefore T_R reconciliation allows speciations, replacement transfers, and free losses, while $T_R L$ reconciliation additionally allows non-free losses.

Not every semi-reconciliation can produce a reconciliation. For example, if a node from G' is mapped under its LCA (*Last Common Ancestor* - see Goodman et al. (1979), Chauve and El-Mabrouk (2009)) position, then the transfers must be allowed as an event in order to obtain a reconciliation.

Definition 14 (Weighted reconciliation) Let $\mathfrak{R}$ be an A reconciliation, and $A = \{a_1, \ldots, a_k\}$. If $c_i \geq 0$ are associated with the events a_i $(i = 1, \ldots, k)$, then $\omega(\mathfrak{R}) = \sum c_i \cdot |a_i|$ is called *the weight or cost of* $\mathfrak{R}$, where $|a_i|$ denotes the number of nodes in G' that are associated with the event a_i, for $i = 1, \ldots, k$.

We see that speciations do not affect the weight of a reconciliation, thus take that their weight is 0. In this paper free losses (losses assigned to a conversion or replacement transfer) have weight 0. Other events, included in a reconciliation, have weight 1.

Definition 15 (Minimum A Reconciliation problem) Let G and S be gene and species trees. The problem of finding an A reconciliation of minimum weight is called Minimum A Reconciliation.

The next definition introduces the weight of a subtree of G'. This is necessary because we estimate the weight of a reconciliation by decomposing G' into subtrees and evaluating the weight of every subtree.

Definition 16 (The weight of a subtree) Let $\mathfrak{R}$ be a reconciliation and T is a subtree of G'. By $\omega_{\mathfrak{R}}(T)$ or $\omega(T)$ is denoted the sum of weights of all events assigned to the nodes and edges of T.

3 Finding an optimal DTLCT$_\mathbf{R}$ reconciliation is NP-hard

In this section, and the rest of the paper, we assume that all events are of weight 1, except speciations and free losses, which are of weight 0. We prove that finding a minimum reconciliation that includes transfers with replacement is NP-hard. We first prove the NP-hardness of the problem of finding a minimum reconciliation that includes all events (duplication, transfer, loss, conversion, transfer with replacement).

We will use a reduction from Max 2-Sat.

Max 2-Sat:
Input: $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$; $C_j = x'_{j_1} \vee x'_{j_2}$, $j = 1, \ldots, m$; $K \leq m$.

Output: Is there a truth assignment for logical variables $x_1, \ldots, x_n$ such that there are at least K true clauses.

This problem is NP-hard (Garey et al. 1976; Garey and Johnson 1979), solvable in polynomial time if $K = m$ (Even et al. 1976; Garey and Johnson 1979). It remains NP-hard even if every variable appears in at most three clauses (Raman et al. 1998). We assume that every variable appears in exactly three clauses, and both positive and negative literals are present. We also assume the optimization version of this problem that asks for the minimum number of false clauses.

Formula F is called a logical expression/formula. Variables $x_1, \ldots, x_n$ are (logical) variables. If x is a variable, then x is called a *positive literal*, and $\neg x$ is a *negative literal*. To a variable x_i literals x_i^1, x_i^2, x_i^3 are assigned. We can assume that x_i^1 and x_i^2 have the same logical value, which is different from the logical value of x_i^3. Variables can be *true* or *false*. Literal is *true* if it is positive and the variable is true, or if it is negative and the variable is false. Similarly, literal is *false* if it is positive and the variable is false, or if it is negative and the variable is true.

3.1 Variable and clause gadgets

In order to construct a polynomial reduction from Max 2-Sat to Optimal $DTLCT_R$ Reconciliation, suppose we have a logical formula F of Max 2-Sat, with n variables and m clauses, such that each variable appears exactly three times as a literal, and both positive and negative literals are present. We will construct a species tree, a gene tree, and a function ϕ mapping the gene tree leaves to the species tree leaves, an instance of the reconciliation problem.

First, we introduce the *border line* that corresponds to some date, depicted by horizontal dashed line in Figures 3, 4, 5a, 5b, 6, 7. Some nodes of the constructed gene tree will be assigned to literals of x_i ($i = 1, \ldots, n$), and in an optimal reconciliation, their mapping above or under this border line will decide if the literals are true or false. In consequence, the positive and negative version of a same variable must be mapped on the opposite sides of the border line in reconciliations.

For each variable and each clause we define a piece of a gene tree and a piece of a species tree with appropriate function ϕ. The gadget for a variable x_i is illustrated in Figure 3. The species subtree S_{x_i} consists in 28 leaves named $A_1^i, \ldots, A_{28}^i$, organized in two subtrees. Seven cherry trees are under the border line on each part, and then linked by two combs, one fully above and one fully under the border line. The gene subtree G_{x_i} is also organized in two subtrees, each consisting in 7 cherry trees linked by a comb. One of the subtrees is identified as the "positive literal subtree" and the other as the "negative literal subtree". The function ϕ mapping the leaves of the gene tree and the leaves of the species tree is such that $\phi(r_{i,k}) = A_{2k-1}^i$ ($k \in \{1, \ldots, 7\}$), $\phi(r_{i,k}) = A_{2k}^i$ ($k \in \{8, \ldots, 14\}$), $\phi(l_{i,k}) = A_{29-2k}^i$ ($k \in \{1, \ldots, 7\}$), $\phi(l_{i,k}) =$

A^i_{30-2k} ($k \in \{8, \ldots, 14\}$), where $(r_{i,k}, l_{i,k})$ is the kth cherry of the gene tree G_{x_i} (i.e. $r_{i,k}, l_{i,k}$ are the children of b^k_i).

Then both trees G and S are anchored by an outgroup comb of size $P(n)$, a polynomial with sufficient size, with respective leaf sets $a^1_i, \ldots, a^{P(n)}_i$ and $A_{i,1}, \ldots, A_{i,P(n)}$, and $\phi(a^k_i) = A_{i,1}$ as illustrated by Figure 4.

Figures 5a and 5b illustrate the gadget for a clause C_j. The species subtree S_{C_j} is a fully balanced binary tree with 8 leaves, noted $B^j_1, \ldots, B^j_8$. The internal nodes of the subtree leading to $B^j_1, \ldots, B^j_4$ are all above the border line, while the internal nodes of the subtree leading to $B^j_5, \ldots, B^j_8$ are all under the border line. To each literal from the clause corresponds a fully balanced gene tree with four leaves, respectively mapping by function ϕ to $((B^j_1, B^j_7), (B^j_2, B^j_6))$ and $((B^j_3, B^j_5), (B^j_4, B^j_8))$ (which is an arbitrary way of mapping each cherry to the two different species subtrees). The internal nodes of the two gene subtrees are respectively labeled r_{j1}, r^0_{j1}, r^1_{j1}, and r_{j2}, r^0_{j2}, r^1_{j2}. The forest of these two gene subtrees is noted F_j.

Clauses and variables gadgets are linked to form the full trees G and S. First, each gene subtree in a clause C_j, representing a literal of a variable x_i, is linked by its root r_{j_1} or r_{j_2} to G_{x_i} (i.e. to x^1_i, x^2_i, or x^3_i), in the middle of the comb of the appropriate subtree (positive literal subtree if the literal is positive, or conversely). Second, the species subtrees and the gene subtrees are linked by a comb containing all variables and clauses in the order $x_1, \ldots, x_n, C_1, \ldots, C_m$ as described by Figure 6.

3.2 Proper reconciliation

Now that we have constructed an instance for the reconciliation problem from a logical formula, we need to be able to translate a reconciliation into an assignment of the variables. This is possible for a type of reconciliation named *proper*. Proper reconciliations are illustrated in Figure 6.

Definition 17 (Proper reconciliation) We call a reconciliation $\mathfrak{R} = (G, G', S, \phi, \rho, \tau, \delta_T, \delta_D, \{T_R\})$, where G and S are constructed from a logical formula, a *proper* reconciliation if

- all transfers are horizontal;
- in variable gadgets, the gene tree vertices in the anchor comb are mapped by ρ to the species tree vertices in the anchor comb, that is, $\rho(c^0_i) = s^0_{x_i}$, $\rho(d^k_i) = D^i_k$ (for all $k \in \{1, \ldots, P(n)\}$), $\rho(d_i) = s^1_{x_i}$;
- in variable gadgets, the two gene tree comb internal vertices (c^k_i in the figure) are mapped to the two species tree combs (vertices $C'_{i,k}$ in the Figure), in one of the two possible combinations (the two gene tree combs may map to either species tree combs).
- in clause gadgets, the mapping ϕ corresponds to one of the cases drawn on Figures 5a and 5b, that is:
  - $\rho(r_{j_1}) = B^j_{1,2}$, $\rho(r_{j_2}) = B^j_{5,6,7,8}$, $\rho(r^0_{j_1}) \in (B^j_{1,2}, B^j_1)$, $\rho(r^1_{j_1}) \in (B^j_{1,2}, B^j_2)$, $\rho(r^0_{j_2}) \in (B^j_{5,6}, B^j_5)$, $\rho(r^1_{j_2}) \in (B^j_{7,8}, B^j_8)$;

- $\rho(r_{j_1}) = B^j_{5,6,7,8}$, $\rho(r_{j_2}) = B^j_{3,4}$, $\rho(r^0_{j_1}) \in (B^j_{5,6}, B^j_6)$, $\rho(r^1_{j_1}) \in (B^j_{7,8}, B^j_7)$, $\rho(r^0_{j_2}) \in (B^j_{3,4}, B^j_3)$, $\rho(r^1_{j_2}) \in (B^j_{3,4}, B^j_4)$;
- $\rho(r_{j_1}) = B^j_{1,2}$, $\rho(r_{j_2}) = B^j_{3,4}$, $\rho(r^0_{j_1}) \in (B^j_{1,2}, B^j_1)$, $\rho(r^1_{j_1}) \in (B^j_{1,2}, B^j_2)$, $\rho(r^0_{j_2}) \in (B^j_{3,4}, B^j_3)$, $\rho(r^1_{j_2}) \in (B^j_{3,4}, B^j_4)$;
- $\rho(r_{j_1}) = B^j_{5,6,7,8}$, $\rho(r_{j_2}) \in (B^j_{3,4}, B^j_3)$, $\rho(r^0_{j_1}) \in (B^j_{5,6}, B^j_6)$, $\rho(r^1_{j_1}) \in (B^j_{7,8}, B^j_7)$, $\rho(r^0_{j_2}) \in (B^j_{3,4}, B^j_3)$, $\rho(r^1_{j_2}) \in (B^j_{3,4}, B^j_4)$;

Note that a proper reconciliation is a T_R reconciliation, *i.e.* the only events are speciations, replacement transfers, and free losses. Hence the weight of a proper reconciliation is the number of transfers.

Let F be a logical formula and G, S are gene and species tree assigned to F, as previously described. There is an obvious relation between value assignment to logical variables and a proper reconciliation between G and S. The next lemma and its proof describes and quantifies this relation.

Lemma 1 *Let $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a logical formula, and G, S are gene and species trees assigned to F. Let $\mathfrak{R}$ be a proper reconciliation between G and S. There is an assignment of the logical variables which satisfies exactly $17n + 5m + \omega(\mathfrak{R})$ clauses.*

Proof The assignment is constructed from the proper reconciliation according to the positions of the corresponding vertices above or under the border line. The definition of proper reconciliation ensures that two opposite literals are always on the opposite side of the border line. Every variable gadget has 17 transfers (counting the ones incident with x^1_i, x^2_i, x^3_i), and the total number of transfers generated by these gadgets is $17n$. Clause gadget has 5 or 4 transfers (not counting incoming transfers, because they are already counted at the variable gadgets), depending if the clause's literals are both false or not. We have f clause gadgets with 5 transfers corresponding to unsatisfied clauses. Hence the number of transfers generated in the clause gadgets is $4(m - f) + 5f = 4m + f$. This yields $\omega(\mathfrak{R}) = 17n + 4m + f$, so there are $17n + 5m + \omega(\mathfrak{R})$ satisfied clauses. □

We see that if we minimize the cost of a proper reconciliation, we also minimize the number of false clauses in the logical formula. As an immediate consequence of Lemma 1, we have the next lemma.

Lemma 2 *To a proper optimal reconciliation corresponds an optimal logical formula.*

In order to prove NP-hardness, we need to show that there is an optimal, proper reconciliation, which can be easily (in polynomial time) obtained from an arbitrary optimal ($DTLCT_R$) reconciliation.

3.3 Optimal proper reconciliation

In this section we describe how to construct a proper optimal reconciliation, given an optimal reconciliation.

Let $\mathfrak{R}$ be an arbitrary reconciliation. If $\mathfrak{R}$ is a proper reconciliation, then $\omega(G_{x_i}) = 17$ (here we also count three transfers incident with x_i^1, x_i^2, x_i^3), and $\omega(F_j) \in \{4, 5\}$ for all variable and clause gadgets.

Lemma 3 *Let $\mathfrak{R}$ be a reconciliation between G and S, where G and S are gene and species trees constructed from a logical formula, and $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$. We have:*

(a) $\omega(F_j) \geq 4$;
(b) if both r_{j_1} and r_{j_2} are under the border line, then $\omega(F_j) \geq 5$;
(c) $\omega(G_{x_i}) \geq 17$;
(d) if x_i^1 or x_i^2 is on the same side of the border line as x_i^3, then $\omega(G_{x_i}) \geq 19$.

Proof To prove (a) and (b), we identify two cases, according to the position of r_{j_1} and r_{j_2}.

Case 1. Node r_{j_1} or r_{j_2} is above the border line. In $\mathcal{G}_{C_j}$ each of $r_{j_1}^0, r_{j_2}^0, r_{j_1}^1, r_{j_2}^1$ is incident with exactly one transfer. In order to obtain $\omega(F_j) < 4$, we need to achieve that some of the nodes $r_{j_1}^0, r_{j_2}^0, r_{j_1}^1, r_{j_2}^1$ is neither duplication nor incident with a transfer. The only way to achieve this is to place some of them in $s_{C_j}^0 = root(S_{C_j})$. Let us take $\rho(r_{j_1}^0) = s_{C_j}^0$. Then $\rho(r_{j_1}) > s_{C_j}^0$, or $\rho(r_{j_1})$ and $s_{C_j}^0$ are incomparable. If $\rho(r_{j_1}^1) < s_{C_j}^0$, then $(r_{j_1}, r_{j_1}^1)$ is a transfer, and the weight of F_j is not decreased. If $\rho(r_{j_1}^1) = s_{C_j}^0$, then r_{j_1} is a duplication, or one of the edges $(r_{j_1}, r_{j_1}^0)$ and $(r_{j_1}, r_{j_1}^1)$ contains a transfer. In this way we eliminate two transfers (that were incident with $r_{j_1}^0$ and $r_{j_1}^1$), and obtain one transfer or duplication. But we generate at least one non-free loss in S_{C_j}. Similar considerations apply to the other nodes of F_j. Hence we cannot obtain $\omega(F_j) < 4$.

Case 2. Both nodes r_{j_1} and r_{j_2} are under the border line. None of the nodes $r_{j_1}^0, r_{j_1}^1, r_{j_2}^0, r_{j_2}^1$ can be placed at $s_{C_j}^0$, therefore every one of them is incident with at least one transfer. If we wish to eliminate transfers starting at r_{j_1} or r_{j_2}, then we need to place them both in $lca(B_5, B_6, B_7, B_8)$, i.e. in the minimal node in S_{C_j} that is ancestor of B_5, B_6, B_7, and B_8 (in Figure 5b (d) the node that is placed in $lca(B_5, B_6, B_7, B_8)$ is r_{j_1}, i.e. $\rho(r_{j_1}) = lca(B_5, B_6, B_7, B_8)$). In this case we increase the number of non-free losses. Whichever placement we choose, we have $\omega(F_j) \geq 5$.

(c) The proof is similar in spirit to the proof of (a). Observe Figures 3 and 4. First, note that moving nodes (i.e. speciations) that belong to the part of variable gadget called anchor (i.e. nodes $d_i^1, \ldots, d_i^{P(n)}$), in order to position some nodes from G_{x_i} would produce extra transfers. Also, moving all $P(n)$ nodes would create a reconciliation more expensive than any proper reconciliation. Hence we can assume that anchoring nodes $d_i^1, \ldots, d_i^{P(n)}$ are not moved. Because if this, when we move nodes of G_{x_i} out of S_{x_i}, we cannot raise them above $D_1^i = p(s_{x_i}^0)$, hence transfers are created.

There are 14 transfers incident with b_i^s $(s = 1, \ldots, 14)$. We can achieve that no transfer or duplication is incident with b_i^s only if $\rho(b_i^s) = s_{x_i}^0$. Then a parent of b_i^s (i.e. c_i^{s-1}), as well as c_i^0, becomes a duplication, or is incident

with a transfer, and two or more (depending on which b_i^s is moved) non-free losses are created. Therefore taking $\rho(b_i^s) = s^0_{x_i}$, for some values of s, does not decrease $\omega(G_{x_i})$.

Observe transfers incident with x_i^1, x_i^2, and eliminate them by moving nodes x_i^1, x_i^2 to an appropriate node of S, and assume that (x_i^2, x_i^1) is not a transfer. By eliminating these transfers, we obtain at least two new transfers (at edges $(c_i^4, x_i^2), (x_i^1, c_i^3)$, or at some other edges). Similar considerations apply for x_i^3. Therefore, in this case too we cannot decrease the number of transfers.

Can we have less than 17 transfers if take $\rho(b_i^s) = s^0_{x_i}$, for some values of s, and eliminate transfers incident with x_i^1, x_i^2 ? Let us take $\rho(b_i^7) = s^0_{x_i}$. Then we need to move nodes c_i^6 and c_i^0, which generates at least two new transfers or duplications, and new non-free losses. Also, moving nodes x_i^1, x_i^2 will generate at least one transfer, different from the previous two newly generated. Hence we have at least three new events, and we cannot obtain less than 17 events (transfers, duplications, non-free losses).

(d) Let us take that x_i^1 and x_i^3 are under the border line. Then at least three of the nodes $c_i^1, \ldots, c_i^{12}$ are not on the gadgets positions. Some of these nodes are c_i^1, c_i^2, c_i^3, because they are descendants of x_i^1 in G. The paths (c_i^1, b_i^2, A_i^3), (c_i^2, b_i^3, A_i^5), (c_i^3, b_i^4, A_i^7) generate extra three transfers. An extra transfer is created on the edge (x_i^2, x_i^1), or at some other edge, if we move x_i^2 together with x_i^1. Even if we assume that, by moving x_i^1 (and x_i^2), we eliminate two transfers that were incident with them, we gain 4 more. Hence $\omega(G_{x_i}) \geq 19$.

□

The proof of next theorem describes a polynomial algorithm that transforms an optimal reconciliation $\mathfrak{R}$ into a reconciliation $\mathfrak{R}'$ that is both optimal and proper.

Theorem 1 *Let G and S be a gene and species tree. There is an optimal reconciliation that is proper.*

Proof Let $\mathfrak{R}$ be an optimal reconciliation. We use $\mathfrak{R}$ to construct $\mathfrak{R}'$ that is both optimal and proper.

Move the vertices of G_{x_i} and position them in S_{x_i} to obtain $\rho(c_i^0) = s^0_{x_i}$, $\rho(d_i^k) = D_k^i$ (for all $k \in \{1, \ldots, P(n)\}$), $\rho(d_i) = s^1_{x_i}$ $(i = 1, \ldots, n)$. If x_i^1 and x_i^2 were not on the same side of the border line as x_i^3 (in $\mathfrak{R}$), then they remain on the same side in $\mathfrak{R}'$ as in $\mathfrak{R}$. If x_i^1 or x_i^2 was on the same side as x_i^3 (in $\mathfrak{R}$), then place x_i^1, x_i^2 above, and x_i^3 under the border line (in $\mathfrak{R}'$).

Next, move the vertices of F_j (in $\mathfrak{R}$) and position them in S_{C_j} to obtain the conditions of Definition 17. Nodes r_{j_1} and r_{j_2} are positioned on the same side of the border line as x'_{j_1} and x'_{j_2}, respectively. A reconciliation, obtained in this way, denote by $\mathfrak{R}'$. It is obvious that $\mathfrak{R}'$ is a proper reconciliation (by the construction). Let us prove that it is an optimal reconciliation.

We have $\omega_{\mathfrak{R}}(G_{x_i}) \geq 17 = \omega_{\mathfrak{R}'}(G_{x_i})$, $\omega_{\mathfrak{R}}(F_j) \geq 4$, and $\omega_{\mathfrak{R}'}(F_j) \in \{4, 5\}$ (Lemma 3).

Let $i \in \{1, \ldots, n\}$, x_i^1, x_i^2, x_i^3 are connected with $r_{a_1} \in V(F_a), r_{b_1} \in V(F_b)$, $r_{c_1} \in V(F_c)$ via transfers, and $\Omega_{\mathfrak{R}}(i) = \omega_{\mathfrak{R}}(G_{x_i}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_b) + \omega_{\mathfrak{R}}(F_c)$.

Case 1. Assume that $\omega_{\mathfrak{R}}(F_a) \geq \omega_{\mathfrak{R}'}(F_a)$, $\omega_{\mathfrak{R}}(F_b) \geq \omega_{\mathfrak{R}'}(F_b)$, $\omega_{\mathfrak{R}}(F_c) \geq \omega_{\mathfrak{R}'}(F_c)$. Then $\omega_{\mathfrak{R}}(G_{x_i}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_b) + \omega_{\mathfrak{R}}(F_c) \geq \omega_{\mathfrak{R}'}(G_{x_i}) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + \omega_{\mathfrak{R}'}(F_c)$, *i.e.* $\Omega_{\mathfrak{R}}(i) \geq \Omega_{\mathfrak{R}'}(i)$.

Case 2. Assume that $\omega_{\mathfrak{R}}(F_a) = 4$, $\omega_{\mathfrak{R}'}(F_a) = 5$, $\omega_{\mathfrak{R}}(F_b) \geq \omega_{\mathfrak{R}'}(F_b)$, $\omega_{\mathfrak{R}}(F_c) \geq \omega_{\mathfrak{R}'}(F_c)$. Since $\omega_{\mathfrak{R}'}(F_a) = 5$, we have that x_i^1 is under the border line (in $\mathfrak{R}'$). Because of the transformation rules, at the beginning of the proof, we have that x_i^1, x_i^2 are under the border line (in $\mathfrak{R}$ and $\mathfrak{R}'$), while x_i^3 is above the line (in $\mathfrak{R}$ and $\mathfrak{R}'$).

Let y_1 be a literal of variable x_s (*i.e.* $y_1 \in \{x_s^1, x_s^2, x_s^3\}$) connected with $r_{a_2} \in V(F_a)$ via transfer. Since $\omega_{\mathfrak{R}}(F_a) = 4$, $\omega_{\mathfrak{R}'}(F_a) = 5$, we have that y_1 is above the border line in $\mathfrak{R}$, and under the line in $\mathfrak{R}'$, hence $y_1 = x_s^3$, $\omega_{\mathfrak{R}'}(F_{a'}) = \omega_{\mathfrak{R}'}(F_{b'}) = 4$, $\omega_{\mathfrak{R}}(G_{x_s}) \geq 19$, where $F_{a'}, F_{b'}$ are connected with x_s^1, x_s^2 via transfers. We have $\omega_{\mathfrak{R}}(F_{a'}) \geq 4 = \omega_{\mathfrak{R}'}(F_{a'})$ and $\omega_{\mathfrak{R}}(F_{b'}) \geq 4 = \omega_{\mathfrak{R}'}(F_{b'})$.

From the previous arguments, $\omega_{\mathfrak{R}}(G_{x_s}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_a) \geq 19 + 4 + 4 = 17 + 5 + 5 = \omega_{\mathfrak{R}'}(G_{x_s}) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_a)$.

Finally, $\big(\omega_{\mathfrak{R}}(G_{x_i}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_b) + \omega_{\mathfrak{R}}(F_c)\big) + \big(\omega_{\mathfrak{R}}(G_{x_s}) + \omega_{\mathfrak{R}}(F_{a'}) + \omega_{\mathfrak{R}}(F_{b'}) + \omega_{\mathfrak{R}}(F_a)\big) \geq \big(\omega_{\mathfrak{R}'}(G_{x_i}) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + \omega_{\mathfrak{R}'}(F_c)\big) + \big(\omega_{\mathfrak{R}'}(G_{x_s}) + \omega_{\mathfrak{R}'}(F_{a'}) + \omega_{\mathfrak{R}'}(F_{b'}) + \omega_{\mathfrak{R}'}(F_a)\big)$, i.e. $\Omega_{\mathfrak{R}}(i) + \Omega_{\mathfrak{R}}(s) \geq \Omega_{\mathfrak{R}'}(i) + \Omega_{\mathfrak{R}'}(s)$.

Case 3. Assume that $\omega_{\mathfrak{R}}(F_b) = 4$, $\omega_{\mathfrak{R}'}(F_b) = 5$, $\omega_{\mathfrak{R}}(F_a) \geq \omega_{\mathfrak{R}'}(F_a)$, $\omega_{\mathfrak{R}}(F_c) \geq \omega_{\mathfrak{R}'}(F_c)$. This case is analogous to Case 2.

Case 4. Assume that $\omega_{\mathfrak{R}}(F_c) = 4$, $\omega_{\mathfrak{R}'}(F_c) = 5$, $\omega_{\mathfrak{R}}(F_a) \geq \omega_{\mathfrak{R}'}(F_a)$, $\omega_{\mathfrak{R}}(F_b) \geq \omega_{\mathfrak{R}'}(F_b)$. Then x_i^3 is under, and x_i^1, x_i^2 are above the border line in $\mathfrak{R}'$. We have two subcases.

Case 4.1. Assume that x_i^1 or x_i^2 was on the same side of the line as x_i^3 (in $\mathfrak{R}$). Then $\omega_{\mathfrak{R}}(G_{x_i}) \geq 19$. Hence $\omega_{\mathfrak{R}}(G_{x_i}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_b) + \omega_{\mathfrak{R}}(F_c) \geq 19 + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + 4 > 17 + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + 5 = \omega_{\mathfrak{R}'}(G_{x_i}) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + \omega_{\mathfrak{R}'}(F_c)$, i.e. $\Omega_{\mathfrak{R}}(i) > \Omega_{\mathfrak{R}'}(i)$.

Case 4.2. Assume that x_i^1 and x_i^2 were not on the same side of the line as x_i^3 (in $\mathfrak{R}$). Then x_i^3 is under the line (in $\mathfrak{R}$ and $\mathfrak{R}'$). Let $y_3 \in \{x_l^1, x_l^2, x_l^3\}$ and it is connected with $r_{c_2} \in V(F_c)$ via transfer. From $\omega_{\mathfrak{R}}(F_c) = 4$, $\omega_{\mathfrak{R}'}(F_c) = 5$, we have that y_3 in $\mathfrak{R}$ was above the line, and in $\mathfrak{R}'$ is under the line, hence $y_3 = x_l^3$, $\omega_{\mathfrak{R}}(G_{x_l}) \geq 19$, $\omega_{\mathfrak{R}'}(F_{a''}) = \omega_{\mathfrak{R}'}(F_{b''}) = 4$, where $F_{a''}$ and $F_{b''}$ are connected with x_l^1 and x_l^2 via transfers.

It follows that $\omega_{\mathfrak{R}}(G_{x_l}) + \omega_{\mathfrak{R}}(F_c) + \omega_{\mathfrak{R}}(F_c) \geq 19 + 4 + 4 = 17 + 5 + 5 = \omega_{\mathfrak{R}'}(G_{x_l}) + \omega_{\mathfrak{R}'}(F_c) + \omega_{\mathfrak{R}'}(F_c)$.

Next, $\big(\omega_{\mathfrak{R}}(G_{x_i}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_b) + \omega_{\mathfrak{R}}(F_c)\big) + \big(\omega_{\mathfrak{R}}(G_{x_l}) + \omega_{\mathfrak{R}}(F_{a''}) + \omega_{\mathfrak{R}}(F_{b''}) + \omega_{\mathfrak{R}}(F_c)\big) \geq \big(\omega_{\mathfrak{R}'}(G_{x_i}) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + \omega_{\mathfrak{R}'}(F_c)\big) + \big(\omega_{\mathfrak{R}'}(G_{x_l}) + \omega_{\mathfrak{R}'}(F_{a''}) + \omega_{\mathfrak{R}'}(F_{b''}) + \omega_{\mathfrak{R}'}(F_c)\big)$, i.e. $\Omega_{\mathfrak{R}}(i) + \Omega_{\mathfrak{R}}(l) \geq \Omega_{\mathfrak{R}'}(i) + \Omega_{\mathfrak{R}'}(l)$.

Case 5. Assume that $\omega_{\mathfrak{R}}(F_a) = \omega_{\mathfrak{R}}(F_b) = 4$, $\omega_{\mathfrak{R}'}(F_a) = \omega_{\mathfrak{R}'}(F_b) = 5$, and $\omega_{\mathfrak{R}}(F_c) \geq \omega_{\mathfrak{R}'}(F_c)$. By a similar argument as in the previous cases, we have that x_i^1, x_i^2 are under the line (in $\mathfrak{R}$ and $\mathfrak{R}'$), while x_i^3 is above the line (in $\mathfrak{R}$ and $\mathfrak{R}'$). Let $y_1 \in \{x_r^1, x_r^2, x_r^3\}$ be connected with $r_{a_2} \in V(F_a)$, and $y_2 \in \{x_t^1, x_t^2, x_t^3\}$ be connected with $r_{b_2} \in V(F_b)$. As in the previous cases, we have $y_1 = x_r^3$, $y_2 = x_t^3$, and they were above the line in $\mathfrak{R}$, and under the line

in $\mathfrak{R}'$. Hence $\omega_{\mathfrak{R}}(G_{x_r}) \geq 19$ and $\omega_{\mathfrak{R}}(G_{x_t}) \geq 19$. Let $x_r^1, x_r^2, x_t^1, x_t^2$ be connected with $F_{a_r}, F_{b_r}, F_{a_t}, F_{b_t}$. Then $\omega_{\mathfrak{R}'}(F_{a_r}) = \omega_{\mathfrak{R}'}(F_{b_r}) = \omega_{\mathfrak{R}'}(F_{a_t}) = \omega_{\mathfrak{R}'}(F_{b_t}) = 4$.

Therefore $\omega_{\mathfrak{R}}(G_{x_r}) + \omega_{\mathfrak{R}}(G_{x_t}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_b) + \omega_{\mathfrak{R}}(F_b) \geq 19 + 19 + 4 + 4 + 4 + 4 = 17 + 17 + 5 + 5 + 5 + 5 = \omega_{\mathfrak{R}'}(G_{x_r}) + \omega_{\mathfrak{R}'}(G_{x_t}) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + \omega_{\mathfrak{R}'}(F_b)$.

Hence $\big(\omega_{\mathfrak{R}}(G_{x_i}) + \omega_{\mathfrak{R}}(F_a) + \omega_{\mathfrak{R}}(F_b) + \omega_{\mathfrak{R}}(F_c)\big) + \big(\omega_{\mathfrak{R}}(G_{x_r}) + \omega_{\mathfrak{R}}(F_{a_r}) + \omega_{\mathfrak{R}}(F_{b_r}) + \omega_{\mathfrak{R}}(F_a)\big) + \big(\omega_{\mathfrak{R}}(G_{x_t}) + \omega_{\mathfrak{R}}(F_{a_t}) + \omega_{\mathfrak{R}}(F_{b_t}) + \omega_{\mathfrak{R}}(F_b)\big) \geq \big(\omega_{\mathfrak{R}'}(G_{x_i}) + \omega_{\mathfrak{R}'}(F_a) + \omega_{\mathfrak{R}'}(F_b) + \omega_{\mathfrak{R}'}(F_c)\big) + \big(\omega_{\mathfrak{R}'}(G_{x_r}) + \omega_{\mathfrak{R}'}(F_{a_r}) + \omega_{\mathfrak{R}'}(F_{b_r}) + \omega_{\mathfrak{R}'}(F_a)\big) + \big(\omega_{\mathfrak{R}'}(G_{x_t}) + \omega_{\mathfrak{R}'}(F_{a_t}) + \omega_{\mathfrak{R}'}(F_{b_t}) + \omega_{\mathfrak{R}'}(F_b)\big)$, i.e. $\Omega_{\mathfrak{R}}(i) + \Omega_{\mathfrak{R}}(r) + \Omega_{\mathfrak{R}}(t) \geq \Omega_{\mathfrak{R}'}(i) + \Omega_{\mathfrak{R}'}(r) + \Omega_{\mathfrak{R}'}(t)$.

Note that the case when $\omega_{\mathfrak{R}}(F_a) = \omega_{\mathfrak{R}}(F_b) = \omega_{\mathfrak{R}}(F_c) = 4 < \omega_{\mathfrak{R}'}(F_a) = \omega_{\mathfrak{R}'}(F_b) = \omega_{\mathfrak{R}'}(F_c) = 5$ is not possible.

Every $i \in \{1, \ldots, n\}$ belongs to exactly one case. Variables s (from Cases 2 and 3), l (Case 4.2), t and r (Case 5) are equal to some $i \in \{1, \ldots, n\}$, but are different among themselves, i.e. there is no value that repeats itself among variables s, l, r, t. Let A_1 be the set of all values of i from the Case 1 that are different from all s, l, r, t. In a similar manner we introduce sets $A_{2,3}$, $A_{4.1}$, $A_{4.2}$, A_5

We will use the previous cases to prove $\omega(\mathfrak{R}) \geq \omega(\mathfrak{R}')$. We have $2 \cdot \omega(\mathfrak{R}) \geq \sum_i \omega_{\mathfrak{R}}(G_{x_i}) + \sum_i \Omega_{\mathfrak{R}}(i) = \sum_i \omega_{\mathfrak{R}}(G_{x_i}) + \sum_{A_1} \Omega_{\mathfrak{R}}(i) + \sum_{A_{2,3}} \big(\Omega_{\mathfrak{R}}(i) + \Omega_{\mathfrak{R}}(s)\big) + \sum_{A_{4.1}} \Omega_{\mathfrak{R}}(i) + \sum_{A_{4.2}} \big(\Omega_{\mathfrak{R}}(i) + \Omega_{\mathfrak{R}}(l)\big) + \sum_{A_5} \big(\Omega_{\mathfrak{R}}(i) + \Omega_{\mathfrak{R}}(r) + \Omega_{\mathfrak{R}}(t)\big) \geq \sum_i \omega_{\mathfrak{R}'}(G_{x_i}) + \sum_{A_1} \Omega_{\mathfrak{R}'}(i) + \sum_{A_{2,3}} \big(\Omega_{\mathfrak{R}'}(i) + \Omega_{\mathfrak{R}'}(s)\big) + \sum_{A_{4.1}} \Omega_{\mathfrak{R}'}(i) + \sum_{A_{4.2}} \big(\Omega_{\mathfrak{R}'}(i) + \Omega_{\mathfrak{R}'}(l)\big) + \sum_{A_5} \big(\Omega_{\mathfrak{R}'}(i) + \Omega_{\mathfrak{R}'}(r) + \Omega_{\mathfrak{R}'}(t)\big) = 2 \cdot \omega(\mathfrak{R}')$.

Finally, $\omega(\mathfrak{R}) \geq \omega(\mathfrak{R}')$. Therefore $\mathfrak{R}'$ is an optimal reconciliation. □

Theorem 2 MINIMUM $DTLCT_R$ RECONCILIATION *problem is NP-hard.*

Proof We will use a reduction from optimization version of MAX 2-SAT. Let $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, $C_j = x'_{j_1} \vee x'_{j_2}$, $j = 1, \ldots, m$ be an instance of MAX 2-SAT. Trees S and G can be obtained in the polynomial time. After obtaining an optimal reconciliation between S and G as an output of MINIMUM $DTLCT_R$ RECONCILIATION, we can (in polynomial time) obtain a proper optimal reconciliation (the proof of Theorem 1), and from it an optimal logical formula, i.e. a logical formula F with minimum number of false clauses (Lemma 1). □

Since a proper reconciliation is a T_R reconciliation, *i.e.* it has only transfers with replacement and all losses are free, then the next theorem can be proved in the same manner as Theorem 2.

Theorem 3 *Let $A \subseteq \{D, T, L, C, T_R\}$ and $T_R \in A$. Then* MINIMUM A RECONCILIATION *problem is NP-hard.*

4 Minimum T_R Reconciliation problem is fixed parameter tractable

We will give a branch and bound algorithm that solves MINIMUM T_R RECONCILIATION problem, with complexity $O(f(k)p(n))$, where p is a polynomial,

k is a parameter representing an upper bound for the reconciliation's weight, and f is a (computable) function.

4.1 Normalized reconciliation

In order to reduce the search space of an FPT algorithm that searches for an optimal reconciliation, we introduce the notion of *normalized reconciliation*. The principle will be that our algorithm can output every normalized reconciliation with a non null probability, and on the other size every reconciliation can be transformed into a normalized one, without changing its cost, by operations that we call *node raising* and *transfer adjustment*. Figures 8 and 9 depict node raising and transfer adjustment, which are defined as follows.

Definition 18 (The reduction of an extension of a gene tree) If G is a gene tree, and G' is an extension of G. With $r(G')$ we denote *the reduction* of G', obtained by pruning all lost subtrees.

We can obtain G from $r(G')$ by suppressing nodes of degree 2.

The next definition introduces *transfer adjustment* (Figure 9). The purpose of this operation is that all transfer parents are in $V(G)$.

Definition 19 (Transfer adjustment) Let $\mathfrak{R} = (G, G', S, \phi, \rho, \tau, \delta_T, \delta_D, \{T_R\})$ be a T_R reconciliation, $x' \in V(G')\backslash V(G)$, $(x', x_t) \in \Theta(\mathfrak{R})$, (x', x_t) is a sub-branch of a branch in G, x_G is the minimal ancestor of x' in $V(G)$. Let $\mathfrak{R}' = (G, G'', S, \phi, \rho', \tau, \delta'_T, \delta'_D, \{T_R\})$ be a TR reconciliation such that:

(*a*) if x_G is a transfer parent and $(x_G, x'_t) \in \Theta(\mathfrak{R})$, $x' < x_1 < \ldots < x_k < x_G$, then remove (x', x_t), suppress x'_t and x', $\rho'(x_G) = \rho(x'_t)$, $(x_G, x_t) \in \Theta(\mathfrak{R}')$. Next, insert new nodes x'', x''_t such that $\rho'(x'') = \rho(x_G)$, $x_k <_{G''} x'' <_{G''} p_{G'}(x_G)$, $\rho'(x''_t) = \rho'(x_G)$, $x_G <_{G''} x''_t <_{G''} x''$. For all $x \in V(G')\backslash\{x_G, x', x'_t\}$ we have $\rho'(x) = \rho(x)$.

(*b*) if x_G is a speciation, and $x' < x_1 < \ldots < x_k < x_G$, then $\rho'(x_G) = p_{S'}(\rho(x_G))$, $(x_G, x_t) \in \Theta(\mathfrak{R}')$, $x_{k+1} \in V(G'')$, $x_1 <_{G''} \ldots <_{G''} x_k <_{G''} x_{x+1}$, x_{x+1} is a child of x_G (in G''), and $\rho'(x_{k+1}) = \rho(x_G)$. Next, (x', x_t) is removed, x' is suppressed, and $\rho'(x) = \rho(x)$, $\forall x \in V(G')\backslash\{x', x_G\}$.

We say that the transfer $(x', x_t) \in \Theta(\mathfrak{R})$ is *adjusted*.

The next definition introduces *node raising* (Figure 8). The purpose of this operation is transforming a reconciliation in a more suitable form.

Definition 20 (Node raising) Let $\mathfrak{R} = (G, G', S, \phi, \rho, \tau, \delta_T, \delta_D, \{T_R\})$ be an optimal T_R reconciliation, $x' \in V(G')\backslash(\Sigma(\mathfrak{R}) \cup L(G'))$, $\rho(x') = y_1 \in E(S')$, $y_1 < y_2 \in E(S')$. If there exists a T_R reconciliation $\mathfrak{R}' = (G, G'', S, \phi, \rho', \tau, \delta'_T, \delta'_D, \{T_R\})$ such that $r(G')$ can be obtained from $r(G'')$ by supressing some nodes of degree 2, $\rho'(x') = y_2$, and $\rho'(x) = \rho(x)$ $(\forall x \in V(r(G'))\backslash\{x'\})$, then we say that $\mathfrak{R}'$ is obtained from $\mathfrak{R}$ by *raising node* x'.

Note that we do not raise speciations, and do not place raised nodes at speciations from S.

Note that node raising and transfer adjustment does not change the number of transfers.

The next definition introduces the notion of *normalized reconciliation*, which represents the output of the main algorithm. The purpose of introducing this type of reconciliation is to reduce the search space of the branch and bound algorithm that we are going to give.

Definition 21 (Normalized reconciliation) Let $\mathfrak{R} = (G, G', S, \phi, \rho, \tau, \delta_T, \delta_D, \{T_R\})$ be an optimal T_R reconciliation, and for every transfer $(x', x_t) \in E(G')$ we have $x' \in V(G)$. Let $y \in V(G')$ be the maximal element such that $x' \leq y$, $\rho(x') \leq \rho(y)$, and $\tau(y) \leq \tau(x') + 1$. Then ($\tau(y) = \tau(x')$ and $deg(y) = 2$) or ($\tau(y) = \tau(x') + 1$ and $y \in \Sigma(\mathfrak{R}) \cap V(G)$) or ($\tau(y) = \tau(x') + 1$ and $y = root(G)$). If (x', x_t) is a diagonal transfer, l is a loss assigned to x_t, and T_l is a lost subtree with a leaf l, then $|E(T_l)| = 1$. Reconciliation $\mathfrak{R}$ is called a *normalized reconciliation.*

The proof of the next theorem describes how to construct a normalized reconciliation from an arbitrary optimal reconciliation.

Theorem 4 *Let $\mathfrak{R}$ be an optimal reconciliation, then there exists a normalized reconciliation $\mathfrak{R}'$ such that $\omega(\mathfrak{R}') = \omega(\mathfrak{R})$.*

Proof First, adjust all transfers (Definition 19 and Figure 9). Transfer adjustment does not change the number of transfers. Therefore the weight of reconciliation is not changed.

We will describe how to raise transfer nodes in order to obtain a normalized reconciliation. From the transfer adjustments we have that if (x', x_t) is a transfer, then $x' \in V(G)$. We can have three cases.

Case 1. There is $y \in V(G')$ such that $x' < y$, there is no transfer in the $y - x'$ path in G' (i.e. $\rho(x') \leq \rho(y)$ and there is no node of degree 2 in the $y - x'$ path), and $y \in \Sigma(\mathfrak{R}) \cap V(G)$.

Case 2. There is $y \in V(G')$ such that $x' < y$, there is neither transfer nor speciation from $V(G)$ in the $y - x'$ path in G', and $deg(y) = 2$. Then y is a transfer child. Let l be a loss assigned to y and T_l a lost subtree with a leaf l.

Case 3. There is no $y \in V(G')$ that satisfies Case 1 or 2. In this case, there is no transfer and no speciation from $V(G)$ in the path (inside G') from $root(G)$ to x'.

First, raise all $x' \in V(G)$ that satisfy Case 1 to obtain $\tau(x') = \tau(y) - 1$, where y is the minimal node from Case 1.

Then, raise all transfer children y to obtain $\tau(y) = \tau(p_{G'}(y))$ or $|E(T_l)| = 1$, where T_l is a lost subtree with a leaf (i.e. loss) assigned to y.

Next, raise x' (from Case 2) to obtain $\tau(x') = \tau(y)$.

If x' satisfies Case 3, then raise it to obtain $\rho(x') = root_E(G)$.

In this way we obtain a reconciliation $\mathfrak{R}'$. The previous procedure does not move speciations from $V(G')$. Therefore the number of transfers is not

increased. Since $\mathfrak{R}$ is an optimal reconciliation, we cannot decrease the number of transfers. Hence $\omega(\mathfrak{R}') = \omega(\mathfrak{R})$.

Let us prove that $\mathfrak{R}'$ is a normalized reconciliation. Let (x', x_t) be a transfer. From the transfer adjustments, we have $x' \in V(G)$. Let $y \in V(G')$ be the maximal element such that $x' \leq y$, $\rho(x') \leq \rho(y)$, and $\tau(y) \leq \tau(x')+1$. Observe two cases.

Case (a). Assume that $\tau(y) = \tau(x')$. We need to prove that y is a transfer child, *i.e.* $deg(y) = 2$. Assume the opposite, y is not a transfer child. Since $\tau(y) = \tau(x')$, y is not a speciation. Therefore y is a transfer parent. Transfer parents are raised as described in Cases 1, 2, and 3. Hence there is y', such that $\tau(y') = \tau(y) = \tau(x)$, or $\tau(y') = \tau(y) + 1 = \tau(x) + 1$, and y' is a transfer child, speciation from $V(G)$, or $root(G)$. Since y is a transfer parent, we have $y \neq y'$, *i.e.* $y < y'$, which contradicts the maximality of y. We get that y is a transfer child.

Case (b). Assume that $\tau(y) = \tau(x')+1$. We need to prove that $y \in V(G) \cap \Sigma$, or $y = root(G)$. Since x' is a transfer parent, we have $\rho(x') \in E(S')$. From this and $\tau(y) = \tau(x') + 1$, we have $\rho(y) \in V(S)$ (see assumptions about dating S' and Figure 2). Therefore $y \in \Sigma(\mathfrak{R}')$ or $y = root(G)$. If $y = root(G)$, this case is finished. If $y \in \Sigma(\mathfrak{R}')$, we need to prove $y \in V(G)$. If $y \notin V(G)$, then this contradicts Cases 1, 2, 3 and the procedure of node raising.

If (x', x_t) is a diagonal transfer, then $\tau(x_t) < \tau(x')$. By the procedure for node raising, we have $|E(T_l)| = 1$, where l is a loss assigned to x_t and T_l is the lost subtree with a leaf l.

All conditions of Definition 21 are satisfied. Thus $\mathfrak{R}'$ is a normalized reconciliation. □

4.2 Random normalized optimal reconciliation

In this subsection we describe an FPT algorithm that returns a normalized reconciliation with the weight not greater than k, if there is one.

The problem definition follows.

Parametric Optimal R Reconciliation

Input: $G, S, k \geq 0$;

Output: Is there an optimal reconciliation $\mathfrak{R} = \mathfrak{R}(G, S)$ such that $\omega(\mathfrak{R}) \leq k$? If yes, return one such reconciliation.

We are given S, G and ϕ, which is in this particular case a bijection between the leaves of G and S (Figure 10 (i)). Let A_i be the extant species (leafs of S), and a_i are the extant genes (leafs of G) $(i = 1, \ldots, n)$. We will maintain during the execution of the algorithm a set of *active edges* which initially contains the terminal edges of G, *i.e.* the edges with a leaf as an extremity. Every active edge *belongs* to an edge in S, which initially is the edge determined by ϕ. Some of the active edges might be *lost*, while initially none is.

Observe one time slice. Let s_0 be the internal node of S in this time slice. Let E_1, E_2 be edges from S incident with s_0 and e_1, e_2 active edges that belong to E_1, E_2 (Figure 10 (b)). We have several cases.

Case 1. At least one of the edges e_1 or e_2 is lost. Then coalesce them at s_0 (meaning the lca of the two edges in G is mapped to s_0 by ρ), and the edge that is not lost propagates to the next time slice, as well as all other edges (Figure 10 (c)), where they remain active.

Case 2. Edges e_1 and e_2 are incident. Then coalesce them at s_0 (Figure 10 (d)). All other active edges propagate to the next time slice, where they remain active. The parent edge of e_1 and e_2 is also an active edge in the next time slice.

Case 3. Edges e_1 and e_2 are neither lost nor incident. Branch and bound tree is branching into three subtrees (subcases (a_1), (a_2), and (b)).

Case 3–(a_1). Put e_1 *on hold* (Figure 11 (a)). This means that e_1 is not propagated into the next time slice, but stays active as long as it does not become a (diagonal) transfer (see Case 3–(b)). Edge e_2 and all other active edges from the current time slice are propagated into the next time slice.

Case 3–(a_2). The same as Case 3–(a_1), but e_2 is *on hold* instead of e_1.

Case 3–(b). Let x be the minimum node in $V(G)$ that is an ancestor of both e_1 and e_2, $x'_1, \ldots, x'_{k_1}$ are the vertices in the path from e_1 to x, and $x''_1, \ldots, x''_{k_2}$ are the vertices in the path from e_2 to x (Figure 11 (b)). Take $\rho(x) = s_0$. Observe x'_1. Let e_3 be an active edge that is a descendant of x'_1, E_3 is the edge from $E(S)$ that contains e_3, and $x_1^1, \ldots, x_1^{m1}$ are the vertices in the path from e_3 to x'_1. Add these vertices and corresponding edges to E_3, as well as transfer (x'_1, x_1^1). Repeat the process for every child of x'_1. It is possible that some of the added transfers is diagonal. We say that x'_1 is *expanded*. In E_3 add a lost edge e'_3, that is propagated to the next time slice, as an active edge. Next, expand the remaining nodes $x'_2, \ldots, x'_{k_1}$, $x''_1, \ldots, x''_{k_2}$.

Case 4. We reached $root_E(S)$. Then expand all the remaining nodes from $V(G)$ and $\rho(root(G)) = root(S)$.

The rest of the procedure is standard branch and bound. When we reach the first solution (reconciliation) with at most k transfers, we denote it by $\mathfrak{R}^*$. If $\mathfrak{R}$ is some other reconciliation, obtained in the branch and bound process, such that $\omega(\mathfrak{R}) < \omega(\mathfrak{R}^*)$, then we take $\mathfrak{R}^* = \mathfrak{R}$. If $\omega(\mathfrak{R}) = \omega(\mathfrak{R}^*)$, then we randomly take $\mathfrak{R}^* = \mathfrak{R}$.

If during the branch and bound procedure, we obtain a (partial) reconciliation with more than k transfers, then we do not branch, and go one step back.

4.3 Pseudocode and properties

In this section we give pseudocodes, prove some properties of the algorithm, and give a proof that Minimum T_R Reconciliation is fixed parameter tractable.

Algorithm 1 Parametric optimal reconciliation

procedure ParametricOptimalReconciliation(G, S, k)
  create S' - a subdivision of S
  $\mathfrak{R}$ denotes partially constructed reconciliation
  $\mathfrak{R}^*$ denotes current optimal reconciliation
  Initialize($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice$)
  BranchAndBound($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice, k$)
  return($\mathfrak{R}$)
end procedure

Algorithm 2 Initializes parameters.

procedure Initialize($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice$)
  $\mathfrak{R}^* \leftarrow NULL$
  $\omega(NULL) \leftarrow +\infty$
  $curr\_time\_slice \leftarrow 1$
  assign extant genes a_i to the corresponding edges A_i $(i = 1, \ldots, n)$
end procedure

Algorithm 3 Branch and bound

procedure BranchAndBound($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice, k$)
  if $root_E(S)$ is in $curr\_time\_slice$ **then**
    expand remaining nodes from $V(G)$
    if $\omega(\mathfrak{R}) < \omega(\mathfrak{R}^*)$ **then**
      $\mathfrak{R}^* \leftarrow \mathfrak{R}$
    end if
    if $\omega(\mathfrak{R}) == \omega(\mathfrak{R}^*)$ **then**
      $\mathfrak{R}^* \leftarrow \mathfrak{R}$ - **random**
    end if
    return
  end if
  $state_1$ - the state of reconciliation $\mathfrak{R}$
  $s_0 \in V(S)$ - speciation in $curr\_time\_slice$
  E_1, E_2 - edges of S incident with s_0
  e_1, e_2 - active edges of G' that are inside E_1, E_2
  if (e_1 or e_2 is a lost edge) **or** (e_1 and e_2 are incident in G) **then**
    coalesce e_1, e_2 into a speciation at s_0
    all other active edges propagate to the next time slice
    $curr\_time\_slice++$
    BranchAndBound($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice, k$)
  else
    ExpandOneEdge($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice, (a_1), k$)
    ExpandOneEdge($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice, (a_2), k$)
    ExpandOneEdge($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice, (b), k$)
  end if
end procedure

Algorithm 4 Executes one edge incident with branching vertex of branch and bound tree

procedure ExpandOneEdge($\mathfrak{R}, curr\_time\_slice, case, k$)
  if $case == (a_1)$ **or** $case == (a_2)$ **then**
    if $case == (a_1)$ **then**
      $e' \leftarrow e_1$
    else
      $e' \leftarrow e_2$
    end if
    put e' *on hold*
    propagate all other active edges to the next time slice
    $curr\_time\_slice++$
    BranchAndBound($\mathfrak{R}, \mathfrak{R}^*, curr\_time\_slice, k$)
    reset $\mathfrak{R}$ to $state_1$
  else
    $x \leftarrow lca_G(e_1, e_2) \in V(G)$
    $x'_1, \ldots, x'_{k_1}$ nodes from $V(G)$ in the path from e_1 to x
    $x''_1, \ldots, x''_{k_2}$ nodes from $V(G)$ in the path from e_2 to x
    assign x to s_0
    expand $x'_1, \ldots, x'_{k_1}$ and $x''_1, \ldots, x''_{k_2}$
    $t \leftarrow \omega(\mathfrak{R})$ - i.e. the number of transfers in the current partial reconciliation;
    if $t > k$ **then**
      return
    end if

```
23:         if ℜ is a (complete) reconciliation then
24:             if ω(ℜ) < ω(ℜ*) then
25:                 ℜ* ← ℜ
26:             else if ω(ℜ) == ω(ℜ*) then
27:                 ℜ* ← ℜ - random
28:             end if
29:             return
30:         end if
31:         curr_time_slice++
32:         BranchAndBound(ℜ, ℜ*, curr_time_slice)
33:         reset ℜ to state_1
34:     end if
35: end procedure
```

Theorem 5 *Let $\mathfrak{R}$ be a normalized reconciliation and $\omega(\mathfrak{R}) \leq k$. Then $\mathfrak{R}$ is a possible output of Algorithm 1.*

Proof Since $\mathfrak{R}$ is a normalized reconciliation, it is also, by definition, optimal. If I is a time slice, then R_I denotes partial reconciliation induced by I, *i.e.* the part of $\mathfrak{R}$ that is inside I, and all other time slices before I. We will prove that the algorithm constructs $\mathfrak{R}_I$ during the execution. We will use mathematical induction on I.

Let I_0 be the first time slice, and $s_0 \in V(S)$ is a speciation such that $\tau(s_0) \in I_0$ (Figure 10), $E_1, E_2 \in E(S)$ are incident with s_0, e_1, e_2 are active edges in E_1 and E_2.

Let us prove that $\mathfrak{R}_{I_0}$ can be obtained during the execution of the algorithm. We have several cases.

Case 1. Edges e_1 and e_2 are incident. Let us prove that e_1 and e_2 coalesce at s_0. Assume the opposite, $\rho(x) \neq s_0$, where $x \in V(G)$ is incident with both e_1 and e_2. Then e_1 or e_2 is a transfer. By placing $\rho(x) = s_0$ we decrease the number of transfers in $\mathfrak{R}$, which contradicts the optimality of $\mathfrak{R}$.

Case 2. Edges e_1 and e_2 are not incident. We will investigate subcases. Some subcases are not obtainable by the algorithm. For them, we will prove they cannot occur in $\mathfrak{R}$. Let x be the minimal element from $V(G)$ that is an ancestor of e_1 and e_2.

Case 2.1. Let $\rho(x) = s_0$, $\rho(x'_{i_1}) = E_1$, $\rho(x''_{i_2}) = E_2$ ($i_1 = 1, \ldots, k_1, i_2 = 1, \ldots, k_2$.). This case is obtainable by the algorithm.

Case 2.2. Let e_i contains a diagonal transfer and e_{3-i} is propagated to the next time slice ($i = 1$ or $i = 2$). This case is also obtainable by the algorithm.

Case 2.3. Both e_1 and e_2 are propagated to the next time slice. Then s_0 contains two gene lineages, which is impossible for a TR reconciliation. Therefore this case cannot occur.

Case 2.4 We have $\rho(x) = s_0$, and there is $y_1 \in \{x'_1, \ldots, x'_{k_1}\}$, or $y_2 \in \{x''_1, \ldots, x''_{k_2}\}$ such that $\rho(y_1) \neq E_1$, or $\rho(y_2) \neq E_2$. Since s_0 is the only speciation in S in the current time slice, then all $x'_1, \ldots, x'_{k_1}$ and $x''_1, \ldots, x''_{k_2}$ are transfers. Therefore by moving y_1 (or y_2) into e_1 (or e_2) we remove some of

the transfers. In this way, we obtain a reconciliation cheaper than $\mathfrak{R}$, which contradicts the optimality of $\mathfrak{R}$. Hence this case is impossible.

Case 2.5. Assume that $\tau(x) > \tau(s_0)$, $\tau(y_1) \leq \tau(s_0)$, and $\tau(y_2) \leq \tau(s_0)$ for some $y_1 \in \{x'_1, \ldots, x'_{k_1}\}$, $y_2 \in \{x''_1, \ldots, x''_{k_2}\}$. Then $\mathfrak{R}$ is not a normalized reconciliation. Hence this case is not possible.

Case 2.6. If x is in I_0 and $\rho(x) \neq s_0$, then by taking $\rho(x) = s_0$ we get a reconciliation with fewer transfers, contrary to the optimality of $\mathfrak{R}$.

For the inductive hypothesis part, assume that the statement is true for time slices $I_0, I_1, \ldots, I_{k-1}$. Let us prove that it is true for I_k. Proving the statement for I_k is the same as for I_0, therefore we will not repeat it.

Hence $\mathfrak{R}_I$ is obtainable by the procedure. Since $\mathfrak{R}_I = \mathfrak{R}$ for the final time slice I, $\mathfrak{R}$ is also obtainable by the algorithm. Since it is an optimal reconciliation, $\mathfrak{R}$ is a possible output of the algorithm. □

Theorem 6 *If Algorithm 1 returns a reconciliation $\mathfrak{R}$, then $\omega(\mathfrak{R}) \leq k$ and $\mathfrak{R}$ is a normalized reconciliation.*

Proof It is obvious that $\omega(\mathfrak{R}) \leq k$, because the Algorithm cuts an edge of the branch-and-bound tree if $t > k$.

Let $(x', x_t) \in E(G')$ be a transfer, $y \in V(G')$ is the maximal element such that $x' \leq y$, $\rho(x') \leq \rho(y)$, $\tau(y) \leq \tau(x') + 1$. In the algorithm, transfers are created when nodes are expanded. Since only nodes in $V(G)$ are expanded, every transfer starts in a node from $V(G)$. Hence $x' \in V(G)$.

Transfers are constructed in the Case 3–(b) and Case 4 (see Subsection 4.2). If $x' \in \{x'_1, \ldots, x'_{k_1}, x''_1, \ldots, x''_{k_2}\}$, then $y \in V(G) \cap \Sigma(\mathfrak{R})$. If $y \notin V(G) \cap \Sigma(\mathfrak{R})$, then (x', x_t) is obtained in the expanded part, and $deg(y) = 2$.

If $deg(y) = 2$ and $\tau(y) < \tau(p_{G'}(y))$, then $(p_{G'}(y), y)$ is a diagonal transfer. Diagonal transfers are made by using edges from G that were *on hold*. From Case 3–(a) we have that a loss l, assigned to y, belongs to a lost subtree T_l with one edge and $\tau(root(T_l)) = \tau(y) + 1$.

If transfer (x', x_t) is expanded in $root_E(S)$, then $y = root(G)$.

Now we will prove that $\mathfrak{R}$ is an optimal reconciliation. Assume the opposite, $\mathfrak{R}$ is not an optimal reconciliation. From Theorem 4 there is a normalized optimal reconciliation. Let $\mathfrak{R}'$ be a normalized reconciliation, i.e. $\omega(\mathfrak{R}') < \omega(\mathfrak{R})$.

From Theorem 5, $\mathfrak{R}'$ is a possible output of Algorithm 1. Since the algorithm always replaces current reconciliation, with the less expensive (if it finds one), $\mathfrak{R}$ cannot be an output of Algorithm 1, since it would be replaced by $\mathfrak{R}'$ (or some other reconciliation), a contradiction.

We have thus proved that $\mathfrak{R}$ is a normalized reconciliation, and $\omega(\mathfrak{R}) \leq k$. □

Theorem 7 *Time complexity of Algorithm 1 is $O(3^k \cdot n)$.*

Proof Branching in the algorithm occurs if and only if we add transfers, i.e. with every branching we add at least one transfer. Therefore we can have the branch depth at most k. Since we branch to three cases (a_1, a_2, and b), the size of the branch and bound tree is $O(3^k \cdot n)$. □

Theorem 8 MINIMUM T_R RECONCILIATION *problem is fixed parameter tractable with respect to the parameter that represents an upper bound for the reconciliation's weight.*

Proof Follows directly from Theorems 4, 5, 6, and 7. □

5 Minimum dated SPR scenario is NP-hard and FPT

Finally, we prove that a constrained version of the well known SPR distance problem, the MINIMUM DATED SPR SCENARIO, mentioned in Song (2006), is equivalent to the MINIMUM T_R RECONCILIATION problem.

Definition 22 (Dated SPR operation) Let T be a dated, fully binary, rooted tree, $e_1 = (a_2, a_1)$, $e_2 = (b_2, b_1) \in E(T)$, where $a_2 = p(a_1)$, $b_2 = p(b_1)$, and $\tau(a_1) < \tau(b_2)$. Delete e_1, suppress a_2, subdivide e_2 with node a_2', where $\tau(a_1) \leq \tau(a_2')$, connect a_1 and a_2'. Obtained tree denote by T'. We say that T' is obtained from T by a dated SPR operation.

We will denote this SPR operation by $spr((a_2, a_1), (b_2, b_1)) = a_2'$. Note that if $spr((a_2, a_1), (b_2, b_1)) = a_2'$, $spr((a_2, a_1), (b_2, b_1)) = a_2''$, and $\tau(a_2') \neq \tau(a_2'')$, then these two SPR operations are different.

Definition 23 (Minimum Dated SPR Scenario problem) Let T and T' be rooted, fully binary trees, where T is dated and T' is undated tree. Assigning dates to $V(T')$, and finding a minimum number (over all possible date assignments to $V(T')$) of SPR operations that transform T into T' is called MINIMUM DATED SPR SCENARIO *problem*. The number of SPR operations is called *the length of SPR scenario*.

Now, we introduce parametrized versions of the problems we are interested in.
k-MINIMUM T_R RECONCILIATION:
Input: S, G, k.
Output: Is there an optimal T_R reconciliation $\mathfrak{R}$ such that $\omega(\mathfrak{R}) \leq k$?

k-MINIMUM DATED SPR SCENARIO:
Input: T - dated, T' - undated, full binary, rooted trees
Output: Is there an optimal dated SPR reconciliation with the length not greater than k?

Lemma 4 *The problem* k-MINIMUM DATED SPR SCENARIO *is (polynomially) equivalent to the problem* k-MINIMUM T_R RECONCILIATION.

Proof Note that if $(a_2, a_1) \in E(G)$, then there is a path in G' $(a_2, b_1, \ldots, b_k, a_1)$. The length of this path is at least 1, i.e. $k \geq 0$. Hence every edge from G is a path in G'. Also, (a_2, a_1) can contain a transfer. In this proof we assume that all transfers are adjusted (as described by Definition 19 and Figure 9), i.e. all transfers start in $V(G)$.

We introduce coloring of edges and nodes that were involved in some SPR operation. Let $spr((a_2, a_1), (b_2, b_1)) = a_2'$ be the i-th SPR operation $T_i \to T_{i+1}$. Then we color edge (a_2', a_1) and node a_2' with color C_i. If edge (b_2, b_1) was colored, then edges (b_2, a_2') and (a_2', b_1) are colored with the same color. Let c_1 be the child of a_2 (in T_i) different from a_1, and c_2 is the parent of a_2 (in T_i). Then c_2 is the parent of c_1 (in T_{i+1}). If edge (c_2, a_2) was colored with a color, then edge (c_2, c_1) is colored with the same color.

To the optimal SPR scenario we will assign an optimal T_R reconciliation. Colored edges will represent transfers, colored nodes will be transfer parents, non-colored edges will coincide with the edges of species tree, and non-colored nodes will be speciations.

Let us first demonstrate the reduction from k-Minimum T_R Reconciliation to k-Minimum Dated SPR Scenario. Let S and G be a species and gene tree, $S = T_0 \to T_1 \to \ldots \to T_k = G$ be an optimal SPR scenario transforming S into G. Using this optimal SPR scenario, we will construct an optimal T_R reconciliation $\mathfrak{R} = (G, G', S, \phi, \rho, \tau, \delta_T, \delta_D, \{T_R\})$.

Note that in T_k we have at most k nodes that are colored. Also, colored edges form (colored) subtrees of T_k with colored roots and inner nodes, while the leafs of these trees are not colored.

If $a \in V(T_k)$ is a non-colored node, then it can be observed as a node from S and node from G. Take $\rho(a) = a \in V(S)$, for all non-colored nodes $a \in V(T_k) = V(G)$. Non-colored paths connect non-colored nodes. All non-colored edges from $T_k = G$ place inside S so that they contain no transfer. Note that leafs of T_k are non-colored.

Now, inside S we will place colored nodes and colored edges. Let T_c be an arbitrary colored tree, and c_0 is its root. Then c_0 is on a non-colored path of G, and we will leave it there in S. Next, we will move inner nodes of T_c so we place them inside S. Let $L(T_c) = \{l_1, \ldots, l_s\}$, and $\tau(l_1) \geq \ldots \geq \tau(l_s)$. Assume that $c_1^1, c_2^1, \ldots, c_{i_1}^1$ are inner nodes of T_c in the path from l_1 to c_0 whose placement inside S is not defined. Then place these nodes in the edge of S' just above l_1, i.e. $\rho(c_1^1) = \ldots = \rho(c_{i_1}^1) = p_{S'}(\rho(l_1))$. Repeat the previous process for leafs $l_2, \ldots l_s$. In this way we obtain a reconciliation with transfers, and every edge of S at any moment contains at most one lineage from G', hence if we extend losses we obtain a T_R reconciliation. Since a transfer can start only at a colored node, we have at most k transfers, i.e. $\omega(\mathfrak{R}) \leq k$.

After the next reduction, we will prove that $\mathfrak{R}$ is an optimal reconciliation.

In the second part, we demonstrate a reduction from k-Minimum Dated SPR Scenario to k-Minimum T_R Reconciliation. Let T be a dated and T' is an undated binary rooted tree. We need a minimum dated SPR scenario $T = T_0 \to T_1 \to \ldots \to T_k = T'$.

Take $S = T$ and $G = T'$. Let $\mathfrak{R}$ be an optimal T_R reconciliation, and $\omega(\mathfrak{R}) = k$. We will prove that the length of minimum dated SPR scenario is k, and reconstruct it using $\mathfrak{R}$.

First, let us construct a scenario of the length k. Adjust all transfers in $\mathfrak{R}$, so they start at the nodes from $V(G)$, just like in the first step of the proof of Theorem 4 (Definition 19, Figure 9).

Take $T_k = T'$, $G_k = G$, $G'_k = G'$, and $\mathfrak{R}_k = \mathfrak{R}$. Let (x_2, x_1) be an arbitrary transfer, x'_1 is the child of x_1 in G', l is the loss assigned to x_1, and $l_0 = root(T_l)$, where T_l is a lost subtree such that $l \in L(T_l)$. Let $p_k = (l_0, l_1, \ldots, l_{s-1}, l_s = l)$ be a path in G' (i.e. in T_l), and therefore a lost path. Remove (x_2, x_1) from G'_k, suppress x_2, include the path p_k into G_k (p_k is not a lost path anymore), suppress x_1. Thus we eliminate one transfer, and obtain $G_{k-1}, G'_{k-1}, \mathfrak{R}_{k-1}$, where $\omega(\mathfrak{R}_{k-1}) = \omega(\mathfrak{R}_k) - 1$. Repeating this procedure, we obtain an SPR scenario $T' = T_k \to T_{k-1} \to \ldots \to T_0 = T$, i.e. $T = T_0 \to T_1 \to \ldots \to T_k = T'$.

Since the transfers can be horizontal or diagonal, corresponding SPR operations are dated. We proved that optimal dated SPR scenario transforming T into T' has the length at most k.

Let us prove that previous reductions construct optimal reconciliation (the first reduction) and optimal SPR scenario (the second reduction). Let $T_1 \to \ldots T_k$ be an optimal SPR scenario. Take $S = T_1, G = T_k$ and $\mathfrak{R}$ is a reconciliation obtained in the first reduction. We have $k' = \omega(\mathfrak{R}) \leq k$. Now, let $T_1 = T'_1 \to T'_2 \to \ldots \to T'_{k''} = T_k$ be a SPR scenario obtained from G and S in the second reduction. Then $k'' \leq k' \leq k$. Since there is no SPR scenario, transforming T_1 into T_k, with the length less than k, we have $k'' = k' = k$. □

Theorem 9 Minimum Dated SPR Scenario *is NP-hard*

Proof Since there is a polynomial reduction from Minimum T_R Reconciliation to Minimum Dated SPR Scenario (Lemma 4) and Minimum T_R Reconciliation is NP-hard (Theorem 3), then Minimum Dated SPR Scenario is NP-hard. □

Theorem 10 Minimum Dated SPR Scenario *is FPT with respect to parametrized distance.*

Proof Since there is a polynomial reduction (which is also an FPT reduction) from k-Minimum Dated SPR Scenario to k-Minimum T_R Reconciliation (Lemma 4) and Minimum T_R Reconciliation is FPT (Theorem 8), then Minimum Dated SPR Scenario is FPT. □

6 Conclusion

We propose an integration of two ways of detecting lateral gene transfers, and more generally to construct gene histories and handle the species tree gene tree discrepancies. On one side, SPR scenarios model transfers with replacements and are limited by computational complexity issues, the difficulty to include time constraints and other gene scale events like transfers without replacement, duplications, conversions and losses. On the other side, reconciliation algorithms usually work with dynamic programming, necessitating an independence hypothesis on different gene tree lineages, incompatible with replacing transfers.

We think this is a big issue for biological models, because the results can depend on the type of methodology which is chosen, leading to simplification hypotheses. Moreover, algorithms are often tested with simulations containing the same hypotheses as the inference models. This is why it can be important to explore methodological issues at the edge of both methods, which is what we do here.

Future work include imagining a way to include transfer with replacement in standard reconciliation software. This will require more integration and probably more efficient algorithms so that it does not harm the computing time.

7 Figures

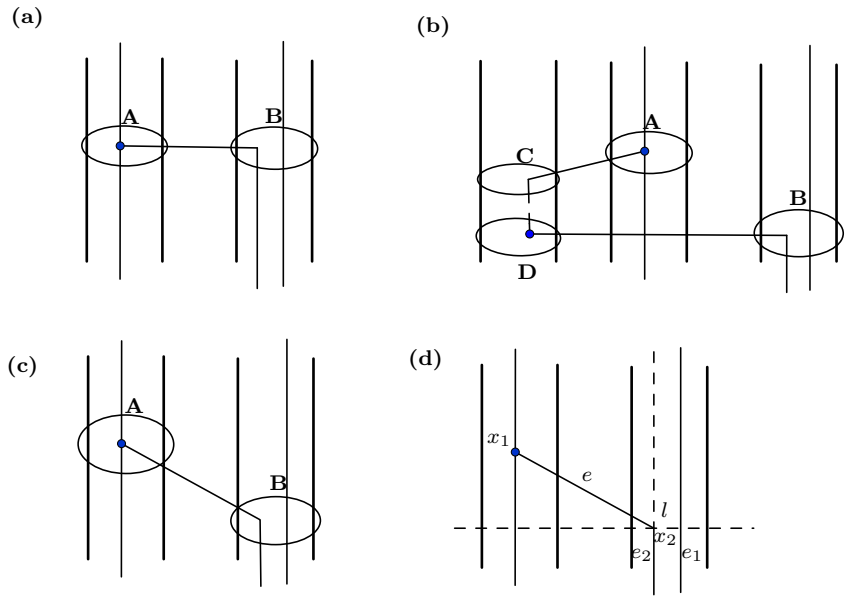

Fig. 1: Gene transfer. **(a)** Horizontal gene transfer between species existing at the same moment. Species A is a donor species, while species B is a recipient species, and it receives a new gene copy. **(b)** Gene exits the observed phylogeny, through a speciation or transfer, then it returns through a horizontal transfer. **(c)** The event from (b) can be represented with a diagonal transfer. **(d)** Transfered gene (x_2) replaces already present gene (l). Replaced gene l is lost. This event is called *replacement transfer* or *transfer with replacement*, and is represented by a transfer and gene loss. Formally, $\delta_T(e) = l$, where $e = (x_1, x_2)$ is a transfer.

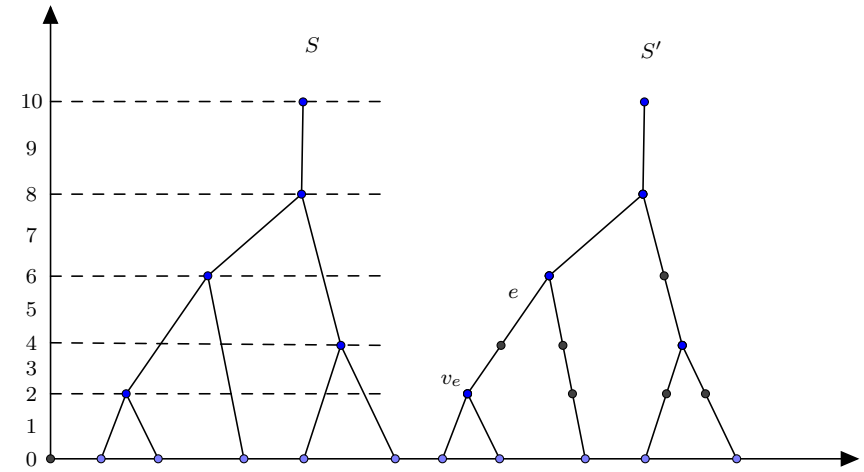

Fig. 2: Tree S' denotes the subdivision of a tree S. To nodes from S' even dates are assigned, while edges from S' are assigned odd dates. The dates are integers from 0 to $2n$, where n is the number of the extant species. If $e \in E(S')$, then v_e is the maximum node from S such that $v_e < e$.

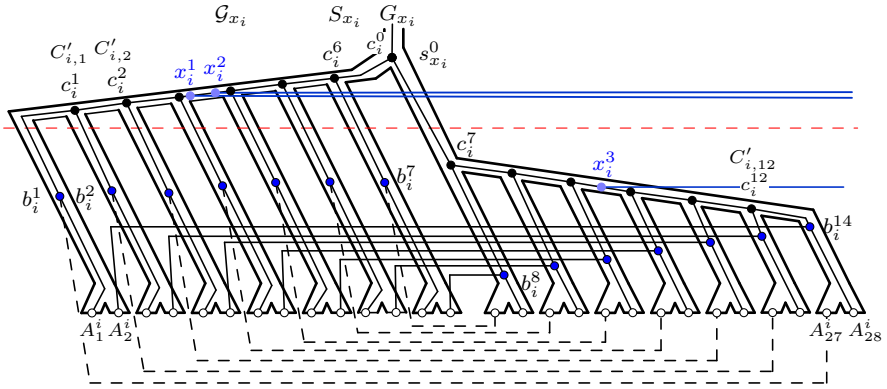

Fig. 3: A variable gadget denoted by $\mathcal{G}_{x_i}$. It is composed of S_{x_i} (a part of the species tree S) and G_{x_i} (a part of gene tree G). Nodes $A_1^i, \ldots, A_{28}^i$ are leaves of S_{x_i}. Nodes $C'_{i,1}, \ldots, C'_{i,12}$ are some of the inner nodes of S_{x_i}, and $s_{x_i}^0$ is the root of S_{x_i}. The rest of the labels denote some of the nodes of G_{x_i}. Variable x_i has two positive (represented by $x_i^1, x_i^2 \in V(G)$) and one negative literal (represented by $x_i^3 \in V(G)$). We can assume that every variable has exact three literals, and there is at least one positive and one negative literal.

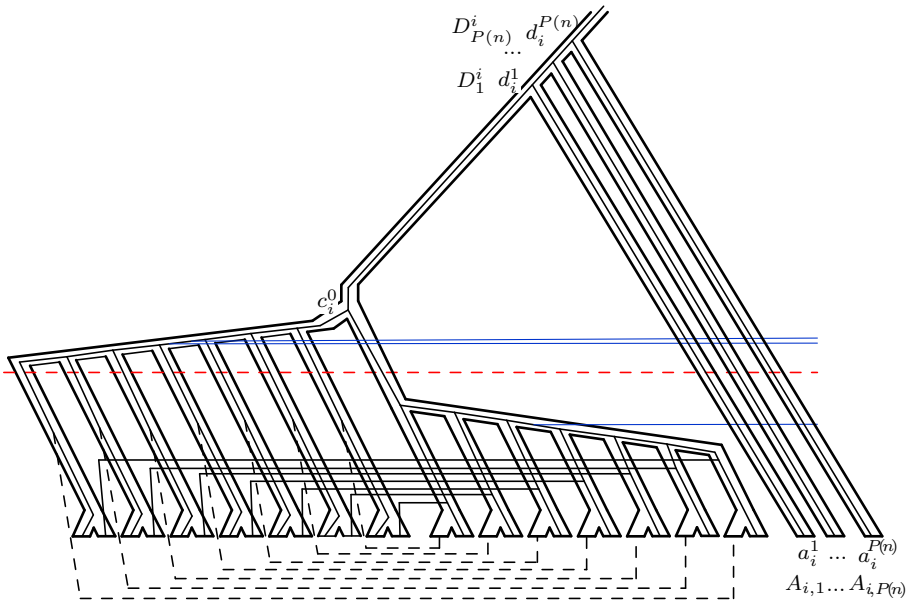

Fig. 4: A variable gadget with anchor. We have $P(n)$ species in the anchor, where P is sufficiently large polynomial. Nodes $d_i^1, \ldots, d_i^{P(n)}, a_i^1, \ldots, a_i^{P(n)}$ belong to gene tree that is part of anchor.

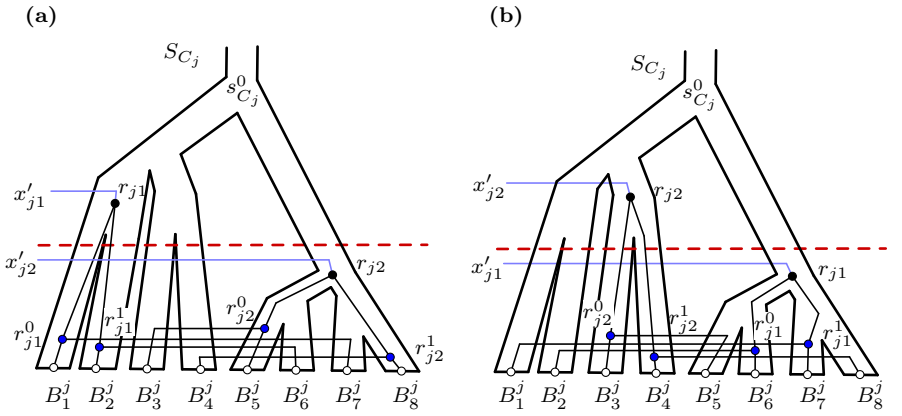

Fig. 5a: Clause gadget that corresponds to a clause $C_j = x'_{j_1} \vee x'_{j_2}$. **(a)** Literal x'_{j_1} is true, and x'_{j_2} is false. **(b)** Literal x'_{j_2} is true, and x'_{j_1} is false.

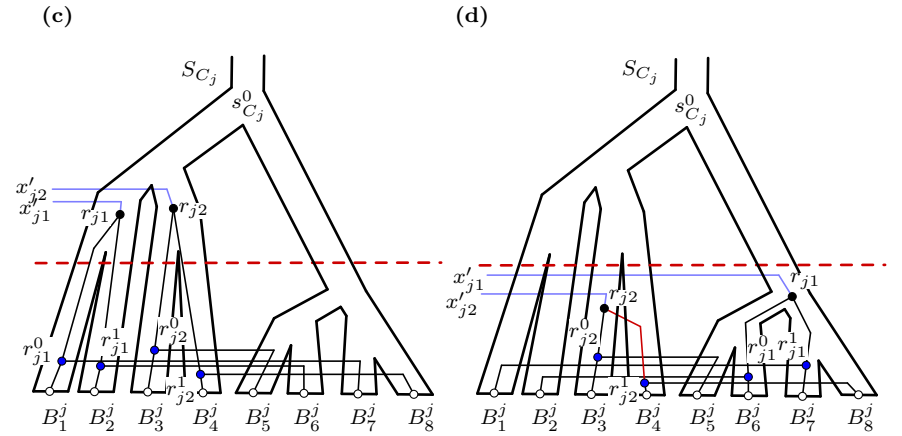

Fig. 5b: **(c)** Both literals are true. **(d)** Both literals are false, hence the clause is false. In this case we have an extra transfer.

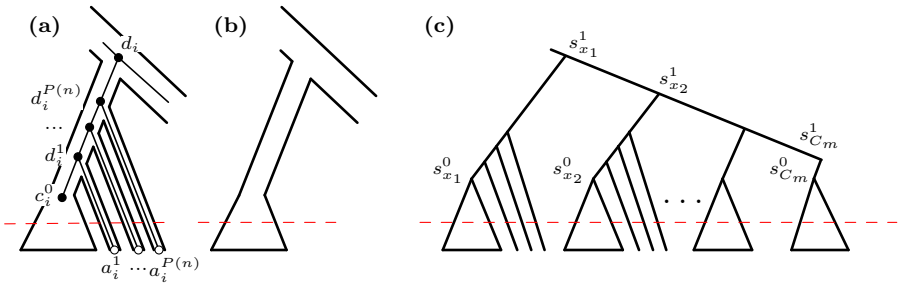

Fig. 6: **(a)** Variable gadget with anchor. For $i = n$, d_i (i.e. d_n) does not exist. **(b)** Clause gadget. **(c)** Proper reconciliation. Nodes s^α_β ($\alpha \in \{0,1\}$, $\beta \in \{x_1, \ldots, C_m\}$) belong to species tree.

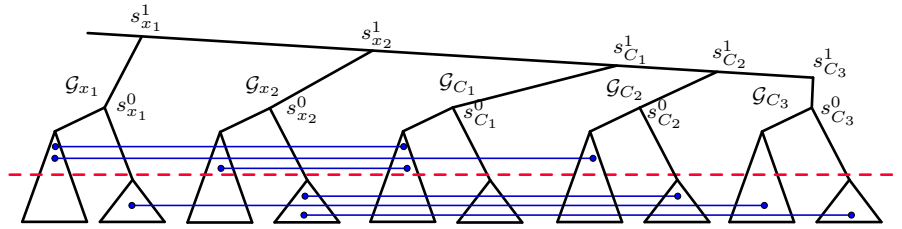

Fig. 7: A proper reconciliation assigned to formula $F_1 = (x_1 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$ with values $x_1 = 1, x_2 = 0$. Some other formulas are also possible, like $F_2 = (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee x_2)$ with values $x_1 = 0, x_2 = 0$. Clauses C_1 and C_2 are true, and clause C_3 is false.

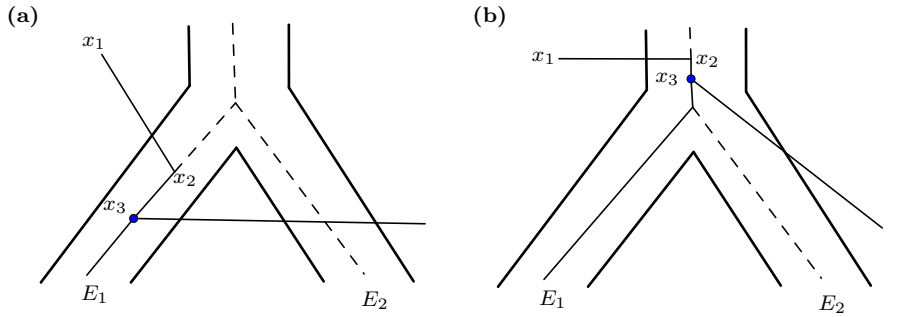

Fig. 8: Node raising. **(a)** Transfer (x_1, x_2), and x_3 is the only child of x_2. **(b)** We first raise x_2, then we raise x_3. Node x_2 cannot be raised higher than x_1. Node x_3 can be arbitrary close to x_2.

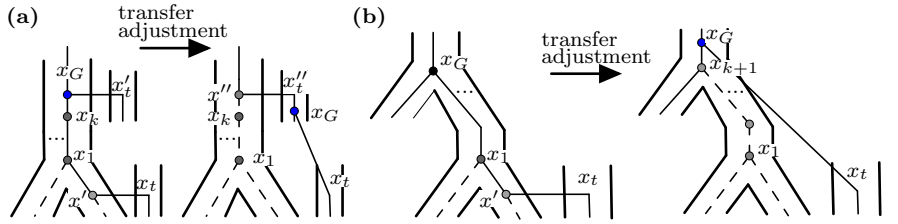

Fig. 9: Transfer adjustment. Observe transfer (x', x_t), where $x' \in V(G')\backslash V(G)$. The path $(x', x_1, \ldots, x_k, x_G)$ is in G', and x_G is the minimum ancestor of x' in G. We adjust transfers in order to obtain that all transfers' parents are in $V(G)$. **(a)** We have $x_G \notin \Sigma$, hence x_G is a transfer parent, and (x_G, x'_t) is a transfer. Node x_G now denote by x'', and move x_G to obtain $\rho(x_G) = \rho(x'_t)$ and x'_t is a parent of x_G. New transfers are (x'', x'_t) and (x_G, x_t). Suppress node x', $(x_1, \ldots, x_k, x'')$ is a lost path, and $x'' \notin V(G)$. **(b)** We have $x_G \in \Sigma$. Raise x_G, suppress x', and the new transfer is (x_G, x_t). The path $(x_1, \ldots, x_k, x_{k+1})$ is lost, and x_{k+1} is a child of x_G.

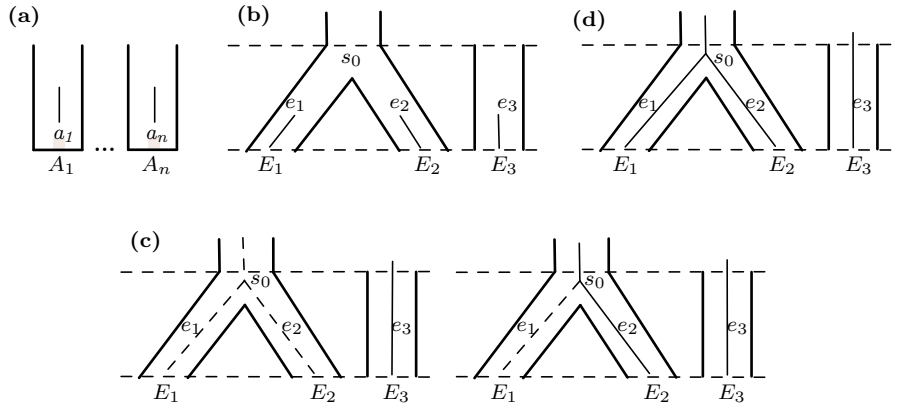

Fig. 10: Initialization and Case 1. **(a)** Initializing the partial reconciliation, at the beginning of B&B. To every extant gene is assigned an active edge. **(b)** We observe current time slice, where s_0 is a corresponding speciation from S. **(c)** If at least one of the edges e_1 and e_2 is lost, then they are coalesced at s_0, and non-lost edge is propagated to the next time slice, as well as all other edges from the current time slice. **(d)** If e_1 and e_2 are incident (i.e. they are siblings), then they coalesce at s_0.

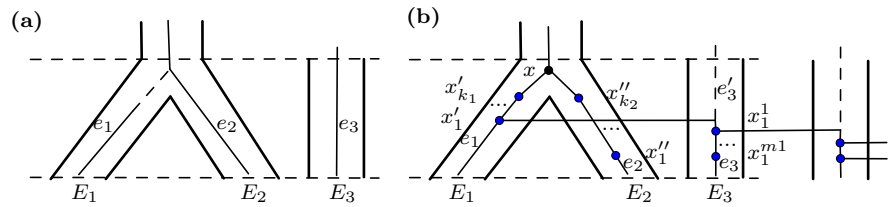

Fig. 11: Cases 2 and 3. **(a)** Edge e_1 is put on hold (staying active), waiting to become a (diagonal) transfer. Edge e_2 is propagated to the next time slice, as well as all other active edges from the current time slice. **(b)** Let x be the minimal ancestor of e_1 and e_2 in $V(G)$. Assign x to s_0, and expand all nodes between x and e_1, and between x and e_2.

Acknowledgements E.T. was supported by the French Agence Nationale de la Recherche (ANR) through grant no. ANR-10-BINF-0101 Ancestrome.

Conflict of Interest: The authors declare that they have no conflict of interest.

References

Allen BL, Steel M (2001) Subtree transfer operations and their induced metrics on evolutionary trees. Ann Comb 5(1):1–15. doi: 10.1007/s00026-001-8006-8

Bansal MS, Alm EJ, Kellis M (2012) Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. Bioinformatics 28(12):283–291. doi: 10.1093/bioinformatics/bts225

Bansal MS, Alm EJ, Kellis M (2013) Reconciliation revisited: Handling multiple optima when reconciling with duplication, transfer, and loss. J Comput Biol 20(10):738–754. doi: 10.1089/cmb.2013.0073

Bonet ML, John KS (2009) Efficiently calculating evolutionary tree measures using SAT. vol 5584 LNCS. Springer Berlin Heidelberg, Berlin, Heidelberg. pp 4–17. doi: 10.1007/978-3-642-02777-2_3

Bordewich M, Semple C (2005) On the computational complexity of the rooted subtree prune and regraft distance. Ann Comb 8(4):409–423. doi: 10.1007/s00026-004-0229-z

Chan Yb, Ranwez V, Scornavacca C (2015) Exploring the space of gene/species reconciliations with transfers. J Math Biol 71(5):1179–1209. doi: 10.1007/s00285-014-0851-2

Chan Yb, Ranwez V, Scornavacca C (2017) Inferring incomplete lineage sorting, duplications, transfers and losses with reconciliations. J Theor Biol 432:1 – 13. doi: 10.1016/j.jtbi.2017.08.008

Chauve C, El-Mabrouk N (2009) New perspectives on gene family evolution: Losses in reconciliation and a link with supertrees. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5541 LNBI:46–58. doi: 10.1007/978-3-642-02008-7_4

Chen J, Shi F, Wang J (2016) Approximating maximum agreement forest on multiple binary trees. Algorithmica 76(4):867–889. doi: 10.1007/s00453-015-0087-6

Chen ZZ, Fan Y, Wang L (2015) Faster exact computation of rSPR distance. J Comb Optim 29(3):605–635. doi: 10.1007/s10878-013-9695-8

Choi SC, Rasmussen MD, Hubisz MJ, Gronau I, Stanhope MJ, Siepel A (2012) Replacing and additive horizontal gene transfer in streptococcus. Mol Biol Evol 29(11):33093320. doi: 10.1093/molbev/mss138

Dasgupta B, Ferrarini S, Gopalakrishnan U, Paryani NR (2006) Inapproximability results for the lateral gene transfer problem. J Comb Optim 11(4):387–405. doi: 10.1007/s10878-006-8212-8

Doyon JP, Scornavacca C, Ranwez V, Berry V (2010) An efficient algorithm for gene / species trees parsimonious reconciliation with losses, duplications, and transfers. Comparative Genomics: International Workshop, RECOMB-

CG 2010, Ottawa, Canada, October 9-11, 2010 Proceedings (October):93–108. doi: 10.1007/978-3-642-16181-0_9

Doyon JP, Ranwez V, Daubin V, Berry V (2011) Models, algorithms and programs for phylogeny reconciliation. Briefings Bioinf 12(5):392–400. doi: 10.1093/bib/bbr045

Even S, Itai A, Shamir A (1976) On the complexity of timetable and multicommodity flow problems. SIAM J Comput 5(4):691–703. doi: 10.1137/0205048

Garey M, Johnson D, Stockmeyer L (1976) Some simplified NP-complete graph problems. Theor Comput Sci 1(3):237 – 267. doi: 10.1016/0304-3975(76)90059-1

Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA

Goodman M, Czelusniak J, Moore GW, Romero-Herrera AE, Matsuda G (1979) Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. Syst Biol 28(2):132–163. doi: 10.1093/sysbio/28.2.132

Hallett MT, Lagergren J (2001) Efficient algorithms for lateral gene transfer problems. In: Proceedings of the Fifth Annual International Conference on Computational Biology. ACM, New York, NY, USA. RECOMB '01. pp 149–156. doi: 10.1145/369133.369188

Hasić D, Tannier E (2017) Gene tree species tree reconciliation with gene conversion. submitted pp 1–38

Hein J, Jiang T, Wang L, Zhang K (1996) On the complexity of comparing evolutionary trees. Discrete Appl Math 71(1-3):153–169. doi: 10.1016/S0166-218X(96)00062-5

Hickey G, Dehne F, Rau-Chaplin A, Blouin C (2008) SPR distance computation for unrooted trees. Evol Bioinform 4:17–27. doi: 10.4137/EBO.S419

Huelsenbeck JP, Rannala B, Larget B (2000) A Bayesian framework for the analysis of cospeciation. Evolution 54(2):352–364. doi: 10.1111/j.0014-3820.2000.tb00039.x

Keeling JD Patrick J; Palmer (2008) Horizontal gene transfer in eukaryotic evolution. Nat Rev Genet 9:605–618. doi: 10.1038/nrg2386

Linz S, Semple C (2011) A cluster reduction for computing the subtree distance between phylogenies. Ann Comb 15(3):465–484. doi: 10.1007/s00026-011-0108-3

Merkle D, Middendorf M, Wieseke N (2010) A parameter-adaptive dynamic programming approach for inferring cophylogenies. BMC Bioinf 11(1):S60. doi: 10.1186/1471-2105-11-S1-S60

Nakhleh L (2012) Computational approaches to species phylogeny inference and gene tree reconciliation. Biophys Chem 34(1):13–23. doi: 10.1016/j.immuni.2010.12.017

Raman V, Ravikumar B, Rao S (1998) A simplified NP-complete MAXSAT problem. Information Processing Letters 65(1):1 – 6. doi: 10.1016/S0020-0190(97)00223-8

Ranwez V, Scornavacca C, Doyon JP, Berry V (2016) Inferring gene duplications, transfers and losses can be done in a discrete framework. J Math Biol

72(7):1811–1844. doi: 10.1007/s00285-015-0930-z

Rice DW, Palmer JD (2006) An exceptional horizontal gene transfer in plastids: gene replacement by a distant bacterial paralog and evidence that haptophyte and cryptophyte plastids are sisters. BMC Biol 4(1):31. doi: 10.1186/1741-7007-4-31

Shi F, Feng Q, Chen J, Wang L, Wang J (2013) Distances between phylogenetic trees: a survey. Tsinghua Sci Technol 18(5):490–499. doi: 10.1109/TST.2013.6616522

Shi F, Feng Q, You J, Wang J (2016) Improved approximation algorithm for maximum agreement forest of two rooted binary phylogenetic trees. J Comb Optim 32(1):111–143. doi: 10.1007/s10878-015-9921-7

Song YS (2006) Properties of subtree-prune-and-regraft operations on totally-ordered phylogenetic trees. Ann Comb 10(1):147–163. doi: 10.1007/s00026-006-0279-5

Suchard MA (2005) Stochastic models for horizontal gene transfer: Taking a random walk through tree space. Genetics 170(1):419–431. doi: 10.1534/genetics.103.025692

Szöllősi GJ, Tannier E, Lartillot N, Daubin V (2013) Lateral gene transfer from the dead. Syst Biol 62(3):386–397. doi: 10.1093/sysbio/syt003

Szöllősi GJ, Tannier E, Daubin V, Boussau B (2015) The inference of gene trees with species trees. Syst Biol 64(1):42–62. doi: 10.1093/sysbio/syu048

Tofigh A, Hallett M, Lagergren J (2011) Simultaneous identification of duplications and lateral gene transfers. IEEE/ACM Trans Comput Biol Bioinformatics 8(2):517–535. doi: 10.1109/TCBB.2010.14

Whidden C, Matsen FA (2015) Calculating the unrooted subtree prune-and-regraft distance. submitted pp 1–37

Whidden C, Beiko RG, Zeh N (2010) Fast FPT Algorithms for Computing Rooted Agreement Forests: Theory and Experiments. Springer Berlin Heidelberg, Berlin, Heidelberg. pp 141–153. doi: 10.1007/978-3-642-13193-6_13

Whidden C, Beiko RG, Zeh N (2016) Fixed-parameter and approximation algorithms for maximum agreement forests of multifurcating trees. Algorithmica 74(3):1019–1054. doi: 10.1007/s00453-015-9983-z

Wu Y (2009) A practical method for exact computation of subtree prune and regraft distance. Bioinformatics 25(2):190–196. doi: 10.1093/bioinformatics/btn606