Consistency of orthology and paralogy constraints in the presence of gene transfers

Mark Jones[1], Manuel Lafond[2], Celine Scornavacca[3]

[1]Delft Institute of Applied Mathematics
Delft University of Technology, The Netherlands
`M.E.L.Jones@tudelft.nl`
[2] Departement d'informatique, Université de Sherbrooke, Canada
`manuel.lafond@USherbrooke.ca`
[3] Institut des Sciences de l'Evolution, Université de Montpellier, CNRS, IRD, EPHE
34095 Montpellier Cedex 5 - France
`Celine.Scornavacca@umontpellier.fr`

Abstract. Orthology and paralogy relations are often inferred by methods based on gene sequence similarity that yield a graph depicting the relationships between gene pairs. Such relation graphs frequently contain errors, as they cannot be explained via a gene tree that contains the depicted orthologs/paralogs while being consistent with the species evolution. Previous research has mostly focused on correcting such errors in some minimal way, for instance by changing a minimum number of relations to attain consistency.

In this work, we ask: could the errors in the orthology predictions be explained by lateral gene transfer? We formalize this question by allowing gene transfers to behave either as a speciation or as a duplication, expanding the space of valid orthology graphs. We then provide a variety of algorithmic results regarding the underlying problems. Namely, we show that deciding if a relation graph R is consistent with a given species network N with known transfer highways is NP-hard, and that it is W[1]-hard under the parameter "minimum number of transfers". During the process, we define a novel algorithmic problem called *Antichain on trees*, which may be useful for other reductions. We then present an FPT algorithm for the decision problem based on the degree of the gene tree associated with R. We also study analogous problems in the case that the transfer highways on a species tree are unknown.

1 Introduction

In phylogenetics, evolutionary relationships between genes and species are often represented via phylogenetic trees. *Species trees* are phylogenetic trees displaying the evolutionary relationships among a set of species, while *gene trees* are phylogenetic trees displaying the evolutionary relationships among genes. Vertical descent with modification (speciation) constitutes only part of the events shaping a gene history; other such events include, for example, duplications, losses and transfers of genes.

When gene trees are used to estimate the evolutionary relationships of the species containing those genes, only *homologous* genes – genes sharing a common ancestor – should be compared. Homology can be refined into the concepts of *orthology* and *paralogy*: two genes from two different species are said to be *orthologous* if they are derived from a single gene present in the last common ancestor of the two species via a speciation event, and *paralogous* if they were derived via a duplication event [11].

Orthology inference is the starting point of several comparative genomics studies, and is also a key instrument for functional annotation of new genomes [13]. Several methods have been designed to distinguish orthologs from paralogs. These can be roughly divided in two groups [1]. The first group of methods, based on phylogenetic inference, reconstruct a *gene tree* and deduce orthology relationships from this tree by comparing it with the species tree via *reconciliation algorithms* (see [4] for a review). Another class of methods estimates orthology using sequence similarity (see e.g. [35,8, among others] and [29] for a survey), hypothesising that orthologs are more similar than paralogs. Both methods can yield a *relation graph*, in which vertices are genes, edges represent putative orthologous gene pairs and non-edges represent putative paralogs. Phylogeny-based methods require a prior knowledge of the species tree, and are very dependent on the accuracy of the gene trees. Unfortunately, the species phylogeny is not always known and gene trees can be highly inaccurate as a result of several kinds of reconstruction artefact, e.g. long-branch attraction (LBA) [2]. Similarity-based methods do not suffer from these drawbacks but still have an important weakness: the inferred relation graph R may fail to be *consistent*, meaning that there is no gene tree, labeled by speciation and duplication events, that can both explain the relations depicted by R and "agree" with a known species tree S. Moreover, approaches based on sequences tend to miss orthologs whose evolutionary path involves a duplication followed by high divergence, which occurs for instance in neofunctionalisation [33].

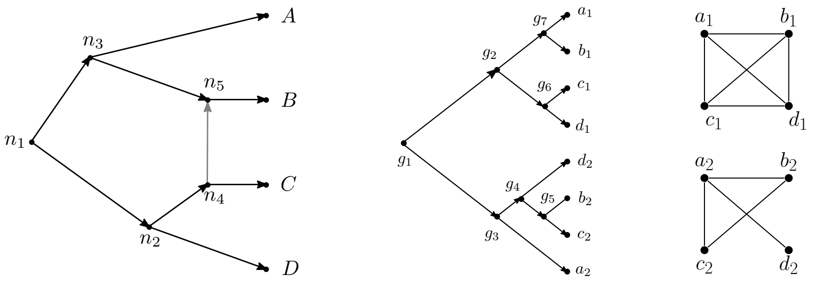
In recent years, the decision problems of consistency of orthology/paralogy relations have been extensively studied [25,21,31,24,26,30,6]. Two possible explanations for the inconsistency of a relation graph R are that either the set of relations contains errors, or the evolutionary model used to assess consistency is not appropriate for the gene family at hand. Most of the previous work in this field has been devoted to detection and correction of errors in relation graphs. The second possibility has recently been considered in [22]. The authors ask, given a event-labeled gene tree G that displays a given set of relations, whether there is a species network N that can be reconciled with G. In a similar vein, in this paper we ask: can inconsistent relations be explained by extending the usual speciation/duplication model to lateral gene transfers? Two genes are said to be *xenologous* if at least one of the two genes has been acquired by gene transfer. As discussed in [27], genes related by transfer may appear either as orthologs or paralogs, even though they are not related by speciation or duplication at their lowest common ancestor. The terms *pseudoorthologs* and *pseudoparalogs* were used to designate homologous genes mimicking orthology and paralogy, respectively, after one or more lateral gene transfers. Here, we provide a variety of algorithmic results regarding the question of explaining inconsistent relations using these new types of relations.

The paper is organized as follows. In Section 2, we introduce the notion of orthology/paralogy consistency with a given species network N, and show how it relates to DS-trees, which are gene trees labeled by speciation and duplication only. Then, in Section 3 we study the question of deciding whether a relation graph R is consistent with N, meaning that R can be represented by a gene history, possibly undergoing lateral transfers, that agrees with N. We show that, unfortunately, this is an NP-hard problem. Furthermore, the problem is unlikely to be fixed-parameter tractable with respect to the number of transfers, as this parameterized version of the problem is $W[1]$-hard. On the positive side, we show in Section 4 that these problems can be solved in time $O(2^k k! k |V(R)||V(N)|^4)$, where here k is the maximum degree of the smallest DS-tree exhibiting the relations of R. In Section 5, we turn to the variant where we have a species tree S rather than a network, and ask if transfer arcs can be inserted into S so that R becomes consistent. Some proofs are quite technical and can be found in the Appendix.

2 Preliminaries

We use the notation $[n] = \{1, 2, \ldots, n\}$. Across the paper, let Γ a set of genes, Σ a set of species, and $\sigma : \Gamma \rightarrow \Sigma$ the mapping between genes and species.

All trees in this paper are assumed to be rooted and directed, each edge being oriented away from the root. A *species network* N on Σ is a directed acyclic graph with a single indegree-0 node (the *root*) and $|\Sigma|$



(a) An LGT network N (b) A gene tree G (c) A relation graph R

Fig. 1: An illustration of an LGT network with secondary arc (n_4, n_5), a gene tree and a relation graph. The genes a_i, b_i, c_i and d_i, with $i \in \{1, 2\}$, belong respectively to species A, B, C and D. Internal nodes are labeled only for the purpose of giving an example of a reconciliation between N and G, see main text. R is not $T_0(N)$-consistent but it is N-consistent using 1 transfer.

outdegree-0 nodes (the *leaves*), such that each leaf is labeled by a different element of Σ. Here we will consider only *binary* species networks, in which internal nodes have either indegree 1 and outdegree 2 (*principal* nodes) or indegree 2 and outdegree 1 (*secondary* nodes or *reticulations*). A *Lateral Gene Transfer (LGT) network* N [5] is a species network along with a partition of $E(N) = E_p \cup E_s$ into a set of *principal arcs* E_p and a set of *secondary arcs* E_s. The E_p edges correspond to vertical descent, whereas the E_s edge correspond to pairs of species that may transfer genetic content. The subnetwork $N' = (V(N), E_p)$ obtained after removing the E_s edges must be a tree in which the root has outdegree 2. We denote by $T_0(N)$ the tree obtained from N' after suppressing indegree-1 outdegree-1 nodes. Roughly speaking, an LGT network can also be seen as a network obtained by starting with a species tree $S = T_0(N)$, and then adding secondary arcs with endpoints located on the edges of S. Note that LGT networks are *tree-based networks*, where $T_0(N)$ is a *distinguished base tree* [12]. As defined in [16,38], we say that an LGT network N is *time-consistent* if there exists a function $t : V(N) \to \mathbb{N}$ such that:

1. $t(u) = t(v)$, if $(u, v) \in E_s$, and
2. $t(u) < t(v)$, if $(u, v) \in E_p$.

Note that although time-consistency forbids directed cycles, not all directed acyclic graphs are time-consistent. For instance, one can easily construct an acyclic LGT network that contains two principal arcs (a, b) and (c, d), and secondary arcs (a, d) and (b, c); no time-consistent labeling is possible for a, b, c, d. It is also worth mentioning that LGT networks that admit a time-consistent map were characterized in [16], where a linear-time algorithm is given to find such a map.

Here a *gene tree* G on Γ is a binary tree with $|\Gamma|$ leaves such that each leaf is labeled by a different element of Γ.

For a binary network N, the root node is denoted by $r(N)$, the set of leaves is denoted by $L(N)$ and the set of internal nodes is denoted by $I(N)$. An internal node x of N has either two children, which we will usually denote x_l and x_r, or one child, which we will denote x_l. The parent of a node x of in-degree 1 is denoted $p(x)$. If x has out-degree 2, the subnetwork rooted at x, denoted N_x, is the network consisting of the root x and all the nodes reachable from x (hence if N is a tree, then N_x is a subtree). If N is a rooted tree, $\textsc{lca}(x, y)$ denotes the lowest common ancestor of x and y. Note that all these notations apply to LGT networks and to gene trees (which are special cases of networks). If N is a species network, since $L(N)$ and Σ are in bijection, we will not make the distinction between a leaf of N and a member of Σ. The same applies to gene tree leaves and Γ.

2.1 Reconciliations between gene trees and species networks

A $\mathbb{DTL}$ reconciliation aims at explaining how an evolutionary history for a family of genes (given by a gene tree) may fit within a given species network N, using speciation, duplication, transfer and gene loss events. The internal nodes of gene trees, representing ancestral genes, are mapped to ancestral species. Furthermore, the branches of a gene tree may hide multiple events that have not been observed, mainly due to losses. Hence, a reconciliation α maps a node x of G to the sequence of species for the genes that should appear on its parent branch. Possible mappings are restricted by few conditions aimed at describing only biologically-meaningful evolutionary histories.

A reconciliation model for gene trees and time-consistent LGT networks (called H-trees) was proposed in [17,19], along with algorithms to minimize the duplication, loss and transfer cost. We use [39, Definition 3], which uses the following formalization:

Definition 1 ([39]). *Given an LGT network N and a gene tree G, let α be a function that maps each node u of G onto a directed path of N, denoted $\alpha(u) = (\alpha_1(u), \ldots, \alpha_\ell(u))$. Then α is a* $\mathbb{DTL}$ reconciliation *between*

G and N if and only if exactly one of the following events occurs for each node u of G and each $\alpha_i(u)$. For each $\alpha_i(u)$ we also specify a label $e_\alpha(u,i)$ corresponding to the case that holds between u and $\alpha_i(u)$, given in square brackets below (for simplicity, let $x := \alpha_i(u)$ below):

a) if x is the last node of $\alpha(u)$, one of the cases below is true:
  1. *$u \in L(G)$, $x \in L(N)$ and $\sigma(u) = x$;* [extant leaf]
  2. *$\{\alpha_1(u_l), \alpha_1(u_r)\} = \{x_l, x_r\}$, where $(x, x_l), (x, x_r) \in E_p$;* [$\mathbb{S}$]
  3. *$\alpha_1(u_l) = x$ and $\alpha_1(u_r) = x$;* [$\mathbb{D}$]
  4. *$\{\alpha_1(u_l), \alpha_1(u_r)\} = \{x, y\}$, where $(x, y) \in E_s$;* [$\mathbb{T}$]

b) otherwise, one of the cases below is true:
  5. *$\alpha_{i+1}(u) = y$, where (x, y) is one of the two outgoing arcs of x in E_p;* [$\mathbb{SL}$]
  6. *$\alpha_{i+1}(u) = y$, where (x, y) is in E_s;* [$\mathbb{TL}$]
  7. *$\alpha_{i+1}(u) = y$ and (x, y) is the only outgoing arc of x in E_p;* [$\emptyset$]

When α is a $\mathbb{DTL}$ reconciliation between G and N, we call the pair (G, α) a reconciled gene tree*.*

By a slight abuse of notation, we may write $|\alpha(u)|$ to denote the number of vertices on the path $\alpha(u)$. If α is clear from the context, we may write $e(u,i)$ in place of $e_\alpha(u,i)$. With a slight abuse of terminology, we will write $e(\alpha_i(u))$ to denote $e(u,i)$. We will also write $\alpha_{\text{LAST}}(u)$ to denote $\alpha_\ell(u)$ and $e(u, \text{LAST})$ or $e(\alpha_{\text{LAST}}(u))$ to denote $e(u,\ell)$ where $\ell = |\alpha(u)|$.

A speciation ($\mathbb{S}$) sends its child genes to the child species through principal arcs. A duplication ($\mathbb{D}$) makes two copies of the gene in the current species. A transfer ($\mathbb{T}$) corresponds to transferring the lineage of a child of a gene to another branch of the species tree, while the sibling lineage still evolves within the lineage of the parent. A speciation-loss ($\mathbb{SL}$) is a speciation where one of the descending genes is absent. A transfer-loss ($\mathbb{TL}$) is a transfer of one of the two descendants of a gene combined with the loss of its sibling lineage. A no event ($\emptyset$) indicates that the gene is not transferred and follows the primary species history. Note that, if N is time-consistent, all $\mathbb{T}$ and $\mathbb{TL}$ events can be guaranteed to happen between co-existing species. Moreover, it is not hard to see that for a given root-to-leaf path $g_1, \ldots, g_k$ of G, the concatenation of the $\alpha(g_i)$ paths correspond to a directed path in N (with some nodes that may occur multiple times in a row because of $\mathbb{D}$ nodes). Hence, if N is time-consistent, α ensures that genes evolve without going back in time. Also note that some models only specify the last element of each $\alpha(u)$ (e.g. the μ map in [32,38]).

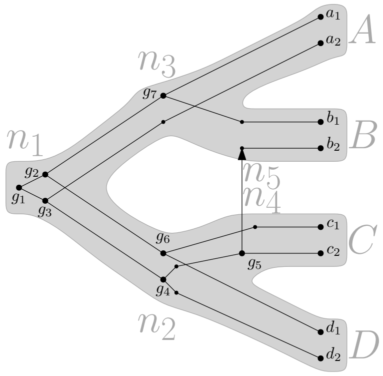


Fig. 2: Illustration of a $\mathbb{DTL}$ reconciliation α between the LGT network N in Figure 1a and the gene tree G in Figure 1b. In cases where $\alpha(x)$ is a path with more than one vertex, only the last vertex of this path is labeled with x. Labels in grey denote the vertices of N.

An example of a $\mathbb{DTL}$ reconciliation between the LGT network in Figure 1a and the gene tree in Figure 1b is as follows: $\alpha(g_1) = (n_1)$, $\alpha(g_2) = (n_1)$, $\alpha(g_3) = (n_1)$, $\alpha(g_4) = (n_2)$, $\alpha(g_5) = (n_2, n_4)$, $\alpha(g_6) = (n_2)$, $\alpha(g_7) = (n_3)$, $\alpha(a_1) = (A)$, $\alpha(b_1) = (n_5, B)$, $\alpha(c_1) = (n_4, C)$, $\alpha(d_1) = (D)$, $\alpha(a_2) = (n_3, A)$, $\alpha(b_2) = (n_5, B)$, $\alpha(c_2) = (C)$, $\alpha(d_2) = (n_2, D)$. See Figure 2. For this $\mathbb{DTL}$ reconciliation, we have $e(\alpha_1(g_1)) = e(\alpha_1(g_4)) = \mathbb{D}$, $e(\alpha_1(g_2)) = e(\alpha_1(g_3)) = e(\alpha_1(g_6)) = e(\alpha_1(g_7)) = \mathbb{S}$, $e(\alpha_2(g_5)) = \mathbb{T}$, $e(\alpha_1(b_1)) = e(\alpha_1(b_2)) = \emptyset$, $e(\alpha_1(c_1)) = \mathbb{TL}$, $e(\alpha_1(a_2)) = e(\alpha_1(d_2)) = e(\alpha_1(g_5)) = \mathbb{SL}$, and $e(\alpha_{\text{LAST}}(u)) =$ extant leaf for all $u \in L(G)$.

Given $x, y \in \mathit{\Gamma}$, let $u = \text{LCA}_G(x, y)$. Then we say that x and y are *orthologs* w.r.t a reconciled gene tree G if $e(\alpha_{\text{LAST}}(u)) = \mathbb{S}$, *paralogs* if $e(\alpha_{\text{LAST}}(u)) = \mathbb{D}$, and *xenologs* if $e(\alpha_{\text{LAST}}(u)) = \mathbb{T}$. Note that one of these cases must hold for all distinct $x, y \in \mathit{\Gamma}$.

2.2 Orthology/paralogy relation graphs

An undirected graph R is called a *relation graph* if $V(R) = \mathit{\Gamma}$ (see Figure 1c). Since R is undirected, we may denote an edge $\{x, y\}$ of R as xy. Relation graphs are often used to depict orthology and paralogy relationships [21]: for any pair x, y of distinct vertices in R, xy is an edge in R if x and y are orthologs, otherwise x and y are paralogs. Several orthology-detection methods such as OrthoMCL [35], ProteinOrtho [34] and OrthoFinder [8] use sequence similarity as a proxy for orthology. Roughly speaking, similar sequences are presumed more likely to be orthologs. When transfers are present, sequence similarity predictions get trickier: xenologs can be "interpreted" as either orthologs, in case the two copies retained the same function (and thus their sequences are likely to be similar), or paralogs, if they did not (and thus their sequences are likely to be different). In the following, we adapt the framework described in [21] to the presence of xenologs. Note that in [23,15,14,32], the authors approach this problem from a different angle, supposing the xenology relationships are given in the relation graph.

We say that a reconciled gene tree (G, α) *displays* a relation graph R, if there is a way of reinterpreting transfers as either speciation or duplication events, such that for any pair x, y of vertices in R, xy is an edge in R if and only if x and y are orthologs according to (G, α). More precisely, we introduce two new types of events $\mathbb{T}^{\mathbb{S}}, \mathbb{T}^{\mathbb{D}}$, which correspond to transfers that behave as a speciation and a duplication, respectively. We then have the following definition:

Definition 2. *Let N be an LGT network, $R = (\Gamma, E)$ a relation graph, and (G, α) a reconciled gene tree with respect to N. We say that (G, α)* displays *R if there exists a labeling e^* of α satisfying:*

- $e^*(u, i) \in \{\mathbb{T}^{\mathbb{S}}, \mathbb{T}^{\mathbb{D}}\}$ *if* $e(u, i) = \mathbb{T}$;
- $e^*(u, i) = e(u, i)$ *if* $e(u, i) \neq \mathbb{T}$;
- *for any distinct* $x, y \in \Gamma$, *if* $xy \in E$ *then* $e^*(\textsc{lca}_G(x, y), \textsc{last}) \in \{\mathbb{S}, \mathbb{T}^{\mathbb{S}}\}$, *and otherwise* $e^*(\textsc{lca}_G(x, y), \textsc{last}) \in \{\mathbb{D}, \mathbb{T}^{\mathbb{D}}\}$.

Note that, if (G, α) and R are known, there is only one relabeling e^* that ensures that (G, α) displays R. Indeed, if $e(u, i) \neq \mathbb{T}$ then $e^*(u, i) = e(u, i)$ and thus fixed by (G, α); otherwise, $\alpha_i(u)$ is the last element of $\alpha(u)$ and $\alpha_i(u) \notin L(N)$, and thus the value of $e^*(u, i)$ (either $\mathbb{T}^{\mathbb{S}}$ or $\mathbb{T}^{\mathbb{D}}$) depends on whether $xy \in E$, for any $x, y \in \Gamma$ such that $\alpha_i(u) = \textsc{lca}_G(x, y)$. The question of interest in this paper is, if only R is known, whether there exists a gene tree that displays R and that can be reconciled with a given network N.

Definition 3. *Let N be a species network and $R = (\Gamma, E)$ a relation graph. We say that R is* consistent *with N (or N-consistent) if there exists a reconciled gene tree (G, α) with respect to N that displays R. In addition we say that R is N*-consistent using k transfers *if (G, α) contains at most k transfers, that is, $e(u, i) = \mathbb{T}$ or $\mathbb{TL}$ for at most k choices of (u, i).*

For an example, see Figure 1: R is consistent using one transfer with N because (G, α) displays R (setting $e^*(g_5, \textsc{last}) = \mathbb{T}^{\mathbb{S}}$) and can be reconciled using one transfer (see the reconciliation given above). It is straightforward to see that R is not consistent using no transfers, thus R is not consistent according to the definition of consistency without xenology [25,21,31,24,26]. It is worth mentioning the question studied in [22] can be interpreted as asking whether R is consistent with *some* network N. It turns out that the answer is always yes, albeit a slightly different model is used.

The main question of interest is to decide whether a set of orthology/paralogy relations can be explained by a gene tree that be reconciled with a given species network.

NETWORK CONSISTENCY (NC):
Input: A relation graph R and a time-consistent species network N.
Question: Is R N-consistent?

We can also consider the minimization version. It is the same as NC, but we are also given a parameter k and ask whether R is N-consistent using k transfers.

TRANSFER MINIMIZATION NETWORK CONSISTENCY (TMNC):
Input: A relation graph R, a time-consistent species network N, and an integer k.
Question: Is R N-consistent using at most k transfers?

2.3 Relation graphs and least-resolved DS-trees

It will be useful to view the problem in terms of a gene tree instead of dealing with relations directly. Before proceeding with our algorithmic results, we establish the equivalence between relation graphs and *least-resolved DS trees*. This relationship was initially established in [3]. In essence, a DS-tree is simply a gene tree D in which each internal node is labeled $\mathbb{S}$ or $\mathbb{D}$. This labeling does not have to be valid with respect to any species tree or network.

More formally, a *DS-tree for Γ* is a pair (D, l), where D is a rooted tree with $L(D) = \Gamma$, and $l : I(D) \to \{\mathbb{D}, \mathbb{S}\}$ is a function labeling each internal node of G as a *duplication* or *speciation*. Note that D is not necessarily binary. The graph $R(D, l) = (\Gamma, E)$ is the relation graph such that for any pair $\{x, y\}$ of genes in Γ, if $l(\textsc{lca}_D(x, y)) = \mathbb{S}$ then $xy \in E$, and if $l(\textsc{lca}_D(x, y)) = \mathbb{D}$ then $xy \notin E$. We say that (D, l) *displays* a relation graph R if $R(D, l) = R$.

An l-contraction in a DS-tree (D, l) consists of contracting an arc (u, v) of D with $u, v \in I(D)$ and $l(u) = l(v)$, and assigning the same label to the node resulting from the contraction. We say that (D, l) is *least-resolved* if no l-contraction is possible. Note that if (D, l) is least-resolved, then it has *alternating* duplication and speciation nodes. That is, each child of a speciation node is a duplication or a leaf, and each child of a duplication node is a speciation or a leaf.

A DS-tree (D, l) is a *refinement* of another DS-tree (D', l') if (D', l') can be obtained from (D, l) by a sequence of l-contractions. If D is binary, then (D, l) is a *binary refinement* of (D', l'). Observe that l-contractions do not change $l(\textsc{lca}_{D'}(x, y))$ for any pair of genes (x, y). Thus if (D, l) is a refinement of (D', l'), then $R(D, l) = R(D', l')$.

It is known that all DS-trees that display R, if any exist, are refinements of the same least-resolved DS-tree.

Lemma 1 ([21,31]). *Assume that some DS-tree displays a relation graph R. Then the least-resolved DS-tree (D, l) that displays R is unique. Moreover, (D, l) can be found in linear time.*

We now want to relate DS-trees with $\mathbb{DTL}$ reconciliations by reinterpreting some internal nodes as transfers.

Definition 4. *Let N be an LGT network and (D, l) a DS-tree with D binary. We say (D, l) is N-*reconcilable *if there exists a $\mathbb{DTL}$ reconciliation α between D and N such that for every internal node $u \in I(D)$, the following holds:*

- *if $l(u) = \mathbb{S}$, then $e(\alpha_{\textsc{last}}(u)) \in \{\mathbb{S}, \mathbb{T}\}$;*
- *if $l(u) = \mathbb{D}$, then $e(\alpha_{\textsc{last}}(u)) \in \{\mathbb{D}, \mathbb{T}\}$.*

Moreover, (D, l) is N-reconcilable using k transfers if α uses k transfers.

If D is non-binary, we say that (D, l) is N-reconcilable (using k transfers) if there exists a binary refinement (D', l') of (D, l) such that D' is N-reconcilable (using k transfers).

Since relation graphs correspond to a unique least-resolved DS-tree, asking about the consistency of a relation graph R is equivalent to asking a similar question about a least-resolved DS-tree (D, l) that displays R, if it exists (see Appendix for a proof).

Lemma 2. *Let N be an LGT network and $R = (\Gamma, E)$ a relation graph. Then R is N-consistent (using k transfers) if and only if there exists a*

DS-tree (D, l) for Γ such that $R(D, l) = R$ and such that (D, l) is N-reconcilable (using k transfers).

Note that in particular, Lemma 2 implies that for R to be N-consistent for an LGT network N, there must exist a DS-tree (D', l') such that $R(D', l') = R$. Moreover, we may assume that (D', l') is a binary refinement of the unique least-resolved DS-tree (D, l) that displays R. By Lemma 1, we can check in linear time whether (D, l) exists, and if so construct it. Therefore, we will often describe an instance of our problem by giving the least-resolved DS-tree (D, l) satisfying $R(D, l) = R$.

We close this subsection by mentioning that the notion of consistency of a gene tree (or DS-tree) has been studied the other way around. That is, in [36,18], we are instead given a species tree and a gene family, and must find a feasible gene scenario under certain constraints.

2.4 Basics of parameterized complexity

We finish this section with some basics of parameterized complexity. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed alphabet and Σ^* are the strings over this alphabet. A pair $(x, k) \in \Sigma^* \times \mathbb{N}$ is a Yes-instance of a parameterized problem L if $(x, k) \in L$. We call the second element k the *parameter* of the instance. A parameterized problem is *fixed-parameter tractable (FPT)* if there exists an algorithm that decides whether a given instance (x, k) is a Yes-instance in time $f(k) \cdot |x|^{O(1)}$, where f is a computable function depending only on k; such an algorithm is called an *FPT algorithm*. The class $W[1]$ is a class of parameterized problems which are strongly believed to not be FPT. A parameterized problem L is $W[1]$-*hard* if there exists $L' \in W[1]$ such that an FPT algorithm for L would imply an FPT algorithm for L'. For more information we refer the reader to [7].

3 Hardness of minimizing transfers on LGT networks

In this section, we consider the NC and TMNC problems. We will show that NC is NP-hard. Moreover, we will show that the minimization version TMNC is not only NP-hard, but also $W[1]$-hard parameterized by k, the number of transfers. We give a reduction from the following problem, which is known to be NP-hard and $W[1]$-hard with respect to k [9]:

k-Multicolored Clique:
Input: A graph $H = (V, E)$, a partition of V into color classes $V_1, \ldots, V_k$.
Parameter: k.

Question: Is there a clique C in H containing exactly one vertex from each color class V_i?

The full version of the reduction can be found in the Appendix, but we can sketch the essential ideas here. We describe the NP-hardness proof – the $W[1]$-hardness is similar but ensures that the reduction is parameterized by k. We first reduce **k-Multicolored Clique** to a novel intermediate problem, ANTICHAIN ON TREES (ACT), then reduce ACT to NC. ACT is formally defined below, but the intuition is as follows: we are given a tree T, a set X of elements to place on the nodes of T, and a weight function $w : X \times V(T) \to \mathbb{N}_0 \cup \{\infty\}$ indicating the cost of placing $x \in X$ on $v \in V(T)$. We interpret $w(x, v) < \infty$ as "x can go on v" and $w(x, v) = \infty$ as "x cannot go on v". Our goal is to place each $x \in X$ on an allowable node such that the elements of X are pairwise incomparable (i.e. none is an ancestor of the other).

ANTICHAIN ON TREES (ACT):
Input: An rooted tree T, a set X, a cost function $w : X \times V(T) \to \mathbb{N}_0 \cup \{\infty\}$.
Question: Does there exist an assignment $f : X \to V(T)$ such that $f(x)$ and $f(y)$ are incomparable in T (that is, neither is an ancestor of the other) for each $x \neq y \in X$, and $w(x, f(x)) < \infty$ for each $x \in X$?

We call an assignment f an *incomparable assignment* if it satisfies the conditions of an ACT instance. In the minimization version of ACT, which we call MINIMUM WEIGHT ANTICHAIN ON TREES (MWACT), we are given a parameter k and ask if there is an incomparable assignment of weight at most k.

MINIMUM WEIGHT ANTICHAIN ON TREES (MWACT):
Input: A rooted tree T, a set X, a cost function $w : X \times V(T) \to \mathbb{N}_0 \cup \{\infty\}$, and an integer k.
Question: Does there exist an assignment $f : X \to V(T)$ such that $f(x)$ and $f(y)$ are incomparable in T (that is, neither is an ancestor of the other) for each $x \neq y \in X$, and such that $\sum_{x \in X} w(x, f(x)) \leq k$?

To see the relationship between ACT and NC, consider an ACT instance (T, X, w). In the NC setting, N is obtained from T after incorporating some specific secondary arcs, and the given relations R have, as their unique least-resolved DS-tree (D, l), a speciation root with $|X|$ children, each child being a duplication corresponding to an element of X. Then being able to place $x \in X$ on $v \in V(T)$ represents "$\alpha_{\text{LAST}}(x) = v$ is possible", i.e. the x node of D is *mappable* onto v. That is, the node

v has a directed path to every species present at a leaf below x, and the weight $w(x, v)$ is the number of transfers required to do so. To enforce the $\alpha_{\text{LAST}}(x)$ to be pairwise incomparable, we ensure that transfers can only be undertaken by descendants of the X nodes of D. Thus the speciation root of D cannot be explained by any transfer whatsoever, ensuring that its children must be incomparable. We now proceed with the formalization of these ideas, and direct the reader to the Appendix for the details of the constructions.

We first show that ACT is NP-hard and MWACT is $W[1]$-hard even under certain restrictions; these will allow us to reduce ACT to NC and MWACT to TMNC. The main idea is that the incomparability requirement can be used to create gadgets as subtrees of an ACT or MWACT instance – if some parent node is assigned to a variable in X, then none of its children can be assigned to any variable in X. In addition, the weight function allows to limit the number of places that can be assigned to a given variable. Using these ideas, we can create an instance of ACT, such that an incomparable assignment of finite weight exists if and only if a given instance of k-Multicolored Clique is a Yes-instance.

Lemma 3. *Let $H = (V = V_1 \cup V_2 \cup \cdots \cup V_k, E)$ be an instance of k-*Multicolored Clique*. Then in polynomial time, we can construct an instance (T, X, w) of* ACT *such that (T, X, w) has an incomparable assignment of weight $< \infty$ if and only if H has a k-multicolored clique. Furthermore, if an incomparable assignment of weight $w < \infty$ exists, then there exists an incomparable assignment with weight $\leq k' = k^2 + 2k$, and (T, X, w) satisfies the following properties:*

- *$w(x, v) \in \{0, 1, \infty\}$ for all $x \in X, v \in V(T)$;*
- *$w(x, v) = 0$ for exactly one v for each $x \in X$;*
- *if $w(x, v) = 0$ then $w(y, v) = \infty$ for all $y \neq x$;*
- *for any $x \in X$, $u, v \in V(T)$ such that $w(x, u), w(x, v) < \infty$, u and v are incomparable.*

As (T, X, w) is a Yes-instance of ACT if and only if the corresponding instance of k-multicolored clique is a Yes-instance, we have that ACT is NP-hard. Moreover, let (T, X, w, k') be the instance of MWACT with $k' = k^2 + k$ and T, X, w as in Lemma 3. Then Lemma 3 also implies that (T, X, w, k') is a Yes-instance of MWACT if and only if the corresponding instance of k-multicolored clique is a Yes-instance. As k' is expressible as a function of k, any FPT algorithm for Lemma 3 implies a FPT algorithm for k-multicolored clique. Therefore, as

k-MULTICOLORED CLIQUE is $W[1]$-hard, so is MWACT. Moreover as (T, X, w) satisfies the properties of Lemma 3, we have the following:

Lemma 4. ACT *is NP-hard and* MWACT *is* $W[1]$*-hard, even under the following conditions:*

- $w(x, v) \in \{0, 1, \infty\}$ *for all* $x \in X, v \in V(T)$*;*
- $w(x, v) = 0$ *for exactly one* v *for each* $x \in X$*;*
- *if* $w(x, v) = 0$ *then* $w(y, v) = \infty$ *for all* $y \neq x$*;*
- *for any* $x \in X$*,* $u, v \in V(T)$ *such that* $w(x, u), w(x, v) < \infty$*,* u *and* v *are incomparable.*

We next reduce ACT to NC. The main idea behind this reduction is that every element of X can be represented by a child of the same speciation node in a least-resolved DS-tree. The tree T can be represented by the distinguished base tree in the species network, and secondary arcs can be added in such a way that, for any $\mathbb{DTL}$ reconciliation, the node corresponding to $x \in X$ can only be mapped to nodes v for which $w(x, v) < \infty$.

Lemma 5. *Let* (T, X, w) *be an instance of* ACT*, such that* $w(x, v) \in \{0, 1, \infty\}$ *for all* $x \in X, v \in V(T)$*,* $w(x, v) = 0$ *for exactly one* v *for each* $x \in X$*, if* $w(x, v) = 0$ *then* $w(y, v) = \infty$ *for all* $y \neq x$*, and for any* $x \in X$*,* $u, v \in V(T)$ *such that* $w(x, u), w(x, v) < \infty$*,* u *and* v *are incomparable.*

Then in polynomial time, we can construct a least-resolved DS-tree (D, l) *and time-consistent LGT network* N *such that for any integer* k*,* (T, X, w) *has an incomparable assignment of cost at most* k *if and only if there exists a binary refinement* (D', l') *of* (D, l) *such that* (D', l') *is* N*-reconcilable using at most* $2k$ *transfers.*

By setting $R = R(D, l)$, Lemma 5 implies that R is N-consistent if and only if (T, W, x) has an incomparable assignment of cost $< \infty$, i.e. (T, W, x) is a YES-instance of ACT. As ACT is NP-hard (under the restrictions in Lemma 5), so is NC. Moreover, for any integer k, Lemma 5 implies that R is N-consistent using at most $k' = 2k$ transfers if and only if (T, W, x, k) is a YES-instance of MWACT. As MWACT is $W[1]$-hard (under the restrictions in Lemma 5), so is TMNC.

Theorem 1. NC *is NP-hard and* TMNC *is* $W[1]$*-hard.*

4 Dynamic programming for bounded degree DS-trees

In this section, we show that given a relation graph R and its least-resolved DS-tree (D, l), if every node of D has degree at most k, then one can decide if (D, l) is N-reconcilable in time $O(2^k k!k|V(D)||V(N)|^4)$. Moreover, if (D, l) is N-reconcilable, our algorithm finds the minimum number of transfers required by any possible reconciliation. In particular, if D is binary, then TMNC can be solved in polynomial time. Note that in [22], it is shown that a DS-tree can always be reconciled with some network in a similar reconciliation model, and the authors characterized precisely when a DS-tree can be reconciled with a given network (although transfers are not studied and, hence, not minimized as we do here). Let us also mention that in a series of papers [20,38,22], it is shown how, given a DS-tree with known transfer events but no species phylogeny, one can find a species tree/network that it can be reconciled with.

The idea of the algorithm is similar to those of [39,28,37]. We use dynamic programming over $V(D)$, from the leaves to the root, and when we encounter a non-binary node, we try every way of refining it. This is a relatively standard procedure, although ensuring a valid reconciliation while minimizing transfers requires care.

For each $g \in V(D)$ and each $s \in V(N)$, we denote by $f(g, s)$ the minimum number of transfers needed by a reconciliation (D_g, α) with respect to N if we require $\alpha_{\text{LAST}}(g) = s$ (recall that D_g is the subtree of D rooted at g). If g is a binary node, we try mapping g_l and g_r to every pair of species s_1 and s_2 that allow $e(g, \text{LAST}) \in \{l(g), \mathbb{T}\}$, and $f(g, s)$ is the minimum over all possibilities. For fixed s, s_1 (resp. s_2), the number

Algorithm 1: minTransferCost(D, N)

```
Data: A DS-tree $D$, an LGT network $N$
Result: $\infty$ if $D$ is not $N$-reconcilable, or otherwise the minimum number of
        transfers
Initialize $f(g, s) = \infty$ for all $g \in V(D), s \in V(S)$
for $g \in V(D)$ in post-order traversal do
    for $s \in V(N)$ in post-order traversal do
        if $g$ is a leaf then
            $f(g, s) = 0$ if $\sigma(g) = s$, otherwise $f(g, s) = \infty$
        else
            $best = \infty$
            for $(D', l') \in \mathcal{B}(g)$ do
                $b = reconcileLBR((D', l'), N, s, f)$
                if $b < best$ then $best = b$;
            $f(g, s) = best$
return $\min_{s \in V(N)} f(r(D), s)$
```

```
Algorithm 2: reconcileLBR($D', N, s, f$)
Data: A binary DS-tree $(D', l')$ which is an LBR of some subtree of $D$, an
      LGT network $N$, the desired species $s$ for $r(D')$, a cost function $f$ on
      the leaves of $D'$
Result: The minimum cost to reconcile $D'$ with $N$ such that $\alpha_{\text{LAST}}(r(D')) = s$
Set $f' = f$ (we maintain temporary costs $f'$ for $D'$)
for $g \in I(D')$ in post-order traversal do
    for $s' \in V(N)$ in post-order traversal do
        if $l'(g) = \mathbb{S}$ then
            if $s'$ has two children and $(s', s'_l), (s', s'_r) \in E_p$ then
                $cost12 = \min_{(s_1,s_2) \in P(s'_l) \times P(s'_r)}(f'(g_l, s_1) + t(s'_l, s_1) +$
                  $f'(g_r, s_2) + t(s'_r, s_2))$
                $cost21 = \min_{(s_1,s_2) \in P(s'_l) \times P(s'_r)}(f'(g_r, s_1) + t(s'_l, s_1) +$
                  $f'(g_l, s_2) + t(s'_r, s_2))$
                $f'(g, s') = \min(cost12, cost21)$
            else if $s'$ is the tail of a secondary arc $(s', s'')$ $(s'' \in \{s'_l, s'_r\})$
              then
                $cost12 = 1 + \min_{(s_1,s_2) \in P(s') \times P(s'')}(f'(g_l, s_1) + t(s', s_1) +$
                  $f'(g_r, s_2) + t(s'', s_2))$
                $cost21 = 1 + \min_{(s_1,s_2) \in P(s') \times P(s'')}(f'(g_r, s_1) + t(s', s_1) +$
                  $f'(g_l, s_2) + t(s'', s_2))$
                $f'(g, s') = min(cost12, cost21)$
        else if $l'(g) = \mathbb{D}$ then
            $f'(g, s') =$
              $\min_{(s_1,s_2) \in P(s') \times P(s')}(f'(g_l, s_1) + t(s', s_1) + f'(g_r, s_2) + t(s', s_2))$
return $f'(r(D'), s)$
```

of transfers required on the branch (g, g_l) (resp. (g, g_r)) is the minimum number of secondary arcs on a path from s to s_1 (resp. s_2). This path would constitute the sequence $\alpha(g_l)$ (resp. $\alpha(g_r)$). Then $f(g, s)$ can be computed from these values, plus those of $f(g_l, s_1)$ and $f(g_r, s_2)$. If g is a non-binary node with children $g_1, \ldots, g_k$, we simply try to refine g in every possible way, then do as in the binary case. In such a binary refinement B of g, we may treat the $g_1, \ldots, g_k$ nodes of B as leaves and use the previously computed $f(g_i, s')$ values for each (g_i, s') pair. Let us turn to the algorithmic details.

Let $g \in I(D)$ with children $g_1, \ldots, g_k$. A binary DS-tree (D', l') with root g and leafset $g_1, \ldots, g_k$ such that $l'(g') = l(g)$ for every $g' \in I(D')$ will be called a *local binary refinement* of g (we write LBR for short). We denote by $\mathcal{B}(g)$ the set of possible LBRs of g. For $s \in V(N)$, denote by $P(s)$ the set of vertices of N that can be reached by some directed path starting from s, and let $t(s, s')$ denote the minimum number of secondary arcs necessary to go from s to s' (note that $t(s, s')$ is easy to compute using weighted shortest path algorithms). We let $t(s, s') = \infty$ if there is no path from s to s'.

The algorithm $minTransferCost$ traverses D in a post-order traversal and, for each node g and each LBR D' in $\mathcal{B}(g)$, calls $reconcileLBR$ to reconcile D'. Note that in the case that g is binary, only one LBR is tested, namely the tree with two leaves g_l and g_r.

The proof of correctness can be done by induction over the height of D_g and can be found in the Appendix. For the complexity, we first compute the all-pairs shortest paths in N in time $O(|V(N)|^3)$ (this is only done once and will not contribute to the final complexity). It is known that the number of binary trees on k leaves is $(2k-3)!! = O(2^k k!)$ [10] which bounds the size of each set of LBRs. The main algorithm computes $\mathcal{B}(g)$ up to $|V(D)||V(N)|$ times. Each member of each $\mathcal{B}(g)$ results in a call to $reconcileLBR$, which is done with a tree D' on at most k leaves. Then in this subroutine for each (g, s) pair with $g \in V(D')$ and $s \in V(N)$, $O(|V(N)|^2)$ pairs of the form (s_1, s_2) are tested – this takes time $O(k|V(N)|^3)$. The total time is thus $O(2^k k! k|V(D)||V(N)|^4)$. The space taken by the algorithm is $O(|V(D)||V(N)|+|V(N)|^2)$. To see this, observe that $O(|V(N)|^2)$ space is needed to store the aforementioned all-pairs shortest path values and $O(|V(D)||V(N)|)$ space is needed for the $f(g, s)$ values. Each enumerated $(D', l') \in \mathcal{B}(g)$ takes space $O(k) = O(|V(D)|)$, which does not add to the space complexity if only the current such (D', l') is kept in memory at all time. Also, one can check that $reconcileLBR$ can be done without additional space (the $P(s)$ sets can be computed on the fly each time when needed).

Theorem 2. *Algorithm* $minTransferCost$ *is correct and runs in time* $O(2^k k! k|V(D)||V(N)|^4)$ *and space* $O(|V(N)||V(D)| + |V(N)|^2)$*.*

Note that while we focused on minimizing the contribution of the k parameter in the above algorithm, it is plausible that techniques developed for similar dynamic programming algorithms in [28,37] could help reduce the $|V(D)||V(N)|^4$ portion of the complexity. In essence, a factor of $|V(N)|^2$ is saved in [28,37] by defining $f(g, s)$ as the best cost of a reconciliation in which $\alpha_{\text{LAST}}(g)$ is mapped to any node reachable from s (instead of requiring s itself), which avoids having to minimize over all reachable pairs (s_1, s_2) for every node of D as in our algorithm.

5 With unknown transfer highways

The set of secondary arcs on a species network cannot always be known with confidence. In fact, reconciliation is sometimes used to infer such arcs on a given species tree [40].

In this section, we remove the assumption that transfer arcs are known. We are given a species *tree* S with $|L(S)| > 1$, and the secondary arcs E_s are to be determined in a time-consistent manner. The question is whether, for a relation graph R, there is a species network N with base tree $T_0(N) = S$ such that R is N-consistent.

Definition 5. *Let S be a species tree. We say that a relation graph R is S-base-consistent (using k transfers) if there exists a time-consistent LGT network N such that $T_0(N) = S$ and R is N-consistent (using k transfers).*

We will show that a relation graph R is always *S-base-consistent*, provided there is a DS-tree (D, l) that displays R. In fact, we prove that *any* binary DS-tree can be made to "agree" with any species tree, no matter how inconsistent they appear to be (provided that each DS-tree leaf can be mapped to a corresponding species tree leaf).

Beforehand, we can easily establish the equivalence between relation graphs and DS-trees as we did for N-consistency. We say that a DS-tree (D, l) is *S-base-reconcilable* (using k transfers) if there exists a time-consistent species network N such that $T_0(N) = S$ and (D, l) is N-reconcilable (using k transfers).

Lemma 6. *Let R be a relation graph and S be a species tree. Then R is S-base-consistent (using k transfers) if and only if there exists a least-resolved DS-tree (D, l) that displays R and a binary refinement (D', l') of (D, l) such that (D', l') is S-base-reconcilable (using k transfers).*

To show that any DS-tree (D, l) is S-base-reconcilable, we add to S a set of secondary arcs E_s of size $O(h(D)|V(S)|^2)$, where $h(D)$ is the height of D (see below). We then obtain a reconciliation α in which $e(\alpha_{\text{LAST}}(u)) = \mathbb{T}$ for every internal node u of D, which might be necessary in some cases. For a node $v \in V(D)$, we denote by $d(v)$ the *depth* of v, which is the number of edges on the path between v and $r(D)$. The *height* of D, denoted $h(D)$, is the maximum depth of a node of D. Let $m = |L(S)|$, and let $(s_1, \ldots, s_m)$ be an arbitrary ordering of $L(S)$. Recall that for $i \in [m]$,

s_i is a leaf of S, and that $p(s_i)s_i$ refers to the edge from the parent of s_i to s_i. We construct the network $N(D)$ from S using the following algorithm:

```
Algorithm 3: constructNetwork($D, S$)
for $d = 0$ to $h(D) + 1$ do
    for $i = 1$ to $m$ do
        for $j = 1$ to $m$, $j \neq i$ do
            Subdivide the arc $p(s_i)s_i$, creating a donor node $don^d_{i \to j}$ ;
            Subdivide the arc $p(s_j)s_j$, creating a receiver node $rec^d_{j \leftarrow i}$ ;
            Add the secondary arc $(don^d_{i \to j}, rec^d_{j \leftarrow i})$ to $E_s$ ;
```

Thus we add every transfer from the s_1 branch to the s_i branch with $i \neq 1$, then every transfer from the s_2 branch to the other s_i branches, and so on, and repeat this process $h(D)+2$ times. Note that $p(s_i)$ changes with each subdivision. It is not hard to see that $N(D)$ is time-consistent, since each time we insert a new arc (x, y), its two endpoints x and y are below every other previously inserted node.

Lemma 7. *Let (D, l) be any binary DS-tree and let $N := N(D)$ be the species network obtained from S after applying Algorithm 3. Then (D, l) is N-reconcilable.*

The detailed proof can be found in the Appendix. The idea is that each $v \in I(D)$ at depth $d(v)$ has the secondary edge $(don^{d(v)}_{i \to j}, rec^{d(v)}_{j \leftarrow i})$ at its disposal. It can be shown that for any $v \in I(D)$ and any distinct $s_i, s_j \in L(N)$, D_v can be reconciled with N such that $\alpha(v) = (don^{d(v)}_{i \to j})$. The idea is illustrated in Figure 3. The highest node of D is mapped to a highest donor node of N, and the descendants transfer back and forth, each time being mapped to a deeper donor node of N.

Theorem 3. *A relation graph R is S-base-consistent if and only if there exists a DS-tree (D, l) that displays R.*

Therefore, deciding if a relation graph R is S-base-consistent can be done in polynomial time.

Thus, unlike N-consistency, deciding S-base-consistency of R can be done quickly by verifying if R admits a DS-tree. However, the explanation of R resulting from the above algorithm will produce scenarios with many transfers, all of which are located between a leaf and its parent. Thus it makes sense to ask if there is a scenario with at most k transfers. This

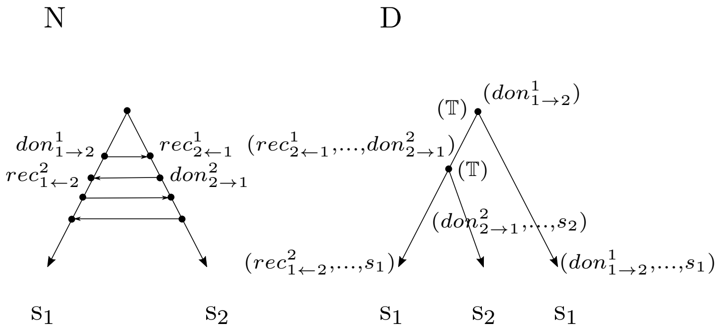


Fig. 3: On the left, a species tree with two leaves, with the horizontal arcs inserted by the algorithm On the right, a DS-tree in which every internal node is labeled $\mathbb{S}$ initially (not shown) which each become a transfer node which we can then label $\mathbb{T}^{\mathbb{S}}$ (the leaves of D depict the species of the gene, we omit giving each gene a name).

problem is closely related to reconciling a gene tree with a species tree while minimizing the number of transfers. In [40], this problem is shown to be NP-hard.

In fact, we present a reduction for minimizing transfers that is very similar in spirit to the one given in [40]. There are, however, many differences between their problem and ours that prevent us from using the previous reduction as a black box for our purposes. First, our definition of reconciliation is different, and in particular, in [40], transfer-loss events are not allowed. Also, in the DS-tree formulation derived from Lemma 2, we are given which nodes of D must be speciations, and which must be duplications. Finally, the authors require that the output network contains no directed cycle, whereas we require time-consistency, which is more restrictive. We invite the interested reader to consult the last section of the Appendix for details.

Theorem 4. *The problem of deciding if a relation graph R is S-base-consistent using k transfers is NP-hard, even if the least-resolved DS-tree (D, l) for R is binary.*

6 Discussion

In this work, we have shown that consistency of relations in the presence of transfers is computationally hard to deal with, making its application difficult in practice. One possible avenue would be to attempt to apply our FPT algorithm to real datasets. A similar algorithm was reported in [28] to be able to handle nodes with up to 8 children, so a next step would be to check the size of non-binary nodes of DS-trees. It would also be interesting to study the problem of error correction of relations in the presence of transfers - although this is almost certainly NP-hard, approximation or FPT algorithms may be applicable.

References

1. A. M. Altenhoff and C. Dessimoz. Inferring orthology and paralogy. *Evolutionary Genomics: Statistical and Computational Methods, Volume 1*, pages 259–279, 2012.
2. J. Bergsten. A review of long-branch attraction. *Cladistics*, 21(2):163–193, 2005.
3. S. Böcker and A. W. Dress. Recovering symbolically dated, rooted trees from symbolic ultrametrics. *Advances in mathematics*, 138(1):105–125, 1998.
4. B. Boussau and C. Scornavacca. Reconciling gene trees with species trees. In C. Scornavacca, F. Delsuc, and N. Galtier, editors, *Phylogenetics in the Genomic Era*, pages 3.2:1–3.2:23. No commercial publisher — Authors open access book, 2020.
5. G. Cardona, J. C. Pons, and F. Rosselló. A reconstruction problem for a class of phylogenetic networks with lateral gene transfers. *Alg. Mol. Biol.*, 10(1):1–15, 2015.
6. R. Dondi, M. Lafond, and N. El-Mabrouk. Approximating the correction of weighted and unweighted orthology and paralogy relations. *AMB*, 12(1):4, 2017.
7. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Springer Publishing Company, Incorporated, 2013.
8. D. M. Emms and S. Kelly. Orthofinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. *Genome Biology*, 16(1):1, 2015.
9. M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theo. Comp. Sci.*, 410(1):53 – 61, 2009.
10. J. Felsenstein and J. Felenstein. *Inferring phylogenies*, volume 2. Sinauer associates Sunderland, MA, 2004.
11. W. M. Fitch. Distinguishing homologous from analogous proteins. *Systematic Biology*, 19(2):99–113, 1970.
12. A. R. Francis and M. Steel. Which phylogenetic networks are merely trees with additional arcs? *Systematic biology*, 64(5):768–777, 2015.
13. T. Gabaldón and E. V. Koonin. Functional and evolutionary implications of gene orthology. *Nature Reviews Genetics*, 14(5):360–366, 2013.
14. M. Geiß, J. Anders, P. F. Stadler, N. Wieseke, and M. Hellmuth. Reconstructing gene trees from fitch's xenology relation. *Journal of mathematical biology*, 77(5):1459–1491, 2018.
15. M. Geiß, M. Hellmuth, Y. Long, and P. F. Stadler. A short note on undirected fitch graphs. *arXiv preprint arXiv:1712.01544*, 2017.
16. P. Górecki. Reconciliation problems for duplication, loss and horizontal gene transfer. In *RECOMB*, pages 316–325. ACM, 2004.
17. P. Górecki. H-trees: a model of evolutionary scenario with horizontal gene transfer. *Fundamenta Informaticae*, 103:105–128, 2010.
18. P. Górecki, A. Markin, and O. Eulenstein. Feasibility algorithms for the duplication-loss cost. In *International Computing and Combinatorics Conference*, pages 206–218. Springer, 2019.

19. P. Górecki and J. Tiuryn. Inferring evolutionary scenarios in the duplication, loss and horizontal gene transfer model. In *Logic and Program Semantics*, pages 83–105. Springer, 2012.
20. M. Hellmuth. Biologically feasible gene trees, reconciliation maps and informative triples. *Algorithms for Molecular Biology*, 12(1):1–16, 2017.
21. M. Hellmuth, M. Hernandez-Rosales, K. T. Huber, V. Moulton, P. F. Stadler, and N. Wieseke. Orthology relations, symbolic ultrametrics, and cographs. *Journal of Mathematical Biology*, 66(1-2):399–420, 2013.
22. M. Hellmuth, K. T. Huber, and V. Moulton. Reconciling event-labeled gene trees with mul-trees and species networks. *Journal of mathematical biology*, 79(5):1885–1925, 2019.
23. M. Hellmuth, P. F. Stadler, and N. Wieseke. The mathematics of xenology: Di-cographs, symbolic ultrametrics, 2-structures and tree-representable systems of binary relations. *arXiv preprint arXiv:1603.02467*, 2016.
24. M. Hellmuth, N. Wieseke, M. Lechner, H.-P. Lenhof, M. Middendorf, and P. F. Stadler. Phylogenomics with paralogs. *PNAS*, 112(7):2058–2063, 2015.
25. M. Hernandez-Rosales, M. Hellmuth, N. Wieseke, K. T. Huber, V. Moulton, and P. F. Stadler. From event-labeled gene trees to species trees. *BMC Bioinformatics*, 13(Suppl 19):S6, 2012.
26. M. Jones, C. Paul, and C. Scornavacca. On the consistency of orthology relationships. Recomb Comparative Genomics 2016; to appear, 2016.
27. E. V. Koonin. Orthologs, paralogs, and evolutionary genomics 1. *Annu. Rev. Genet.*, 39:309–338, 2005.
28. M. Kordi and M. S. Bansal. Exact algorithms for duplication-transfer-loss reconciliation with non-binary gene trees. In *Proceedings of the 7th ACM Int. Conf. on Bioinformatics, Computational Biology, and Health Informatics*, pages 297–306. ACM, 2016.
29. D. M. Kristensen, Y. I. Wolf, A. R. Mushegian, and E. V. Koonin. Computational methods for gene orthology inference. *Briefings in bioinformatics*, 12(5):379–391, 2011.
30. M. Lafond, R. Dondi, and N. El-Mabrouk. The link between orthology relations and gene trees: a correction perspective. *AMB*, 11(1):4, 2016.
31. M. Lafond and N. El-Mabrouk. Orthology and paralogy constraints: satisfiability and consistency. *BMC genomics*, 15(Suppl 6):S12, 2014.
32. M. Lafond and M. Hellmuth. Reconstruction of time-consistent species trees. *Algorithms for Molecular Biology*, 15(1):1–27, 2020.
33. M. Lafond, M. Meghdari Miardan, and D. Sankoff. Accurate prediction of orthologs in the presence of divergence after duplication. *Bioinformatics*, 34(13):i366–i375, 2018.
34. M. Lechner, S. Findeiß, L. Steiner, M. Marz, P. F. Stadler, and S. J. Prohaska. Proteinortho: detection of (co-) orthologs in large-scale analysis. *BMC bioinformatics*, 12(1):1–9, 2011.
35. L. Li, C. J. Stoeckert, and D. S. Roos. Orthomcl: identification of ortholog groups for eukaryotic genomes. *Genome research*, 13(9):2178–2189, 2003.
36. A. Markin, V. S. K. T. Vadali, and O. Eulenstein. Solving the gene duplication feasibility problem in linear time. In *International Computing and Combinatorics Conference*, pages 378–390. Springer, 2018.
37. A. Mykowiecka, P. Szczęsny, and P. Górecki. Inferring gene-species assignments in the presence of horizontal gene transfer. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(5):1571–1578, 2017.

38. N. Nøjgaard, M. Geiß, D. Merkle, P. F. Stadler, N. Wieseke, and M. Hellmuth. Time-consistent reconciliation maps and forbidden time travel. *Algorithms for Molecular Biology*, 13(1):1–17, 2018.
39. C. Scornavacca, J. C. P. Mayol, and G. Cardona. Fast algorithm for the reconciliation of gene trees and lgt networks. *Journal of Theoretical Biology*, 418:129–137, 2017.
40. A. Tofigh, M. Hallett, and J. Lagergren. Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM TCBB*, 8(2):517–535, 2011.

Appendix

Here we include the details of the proofs that were left out of the main text.

Lemma 2. *Let N be an LGT network and $R = (\Gamma, E)$ a relation graph. Then R is N-consistent (using k transfers) if and only if there exists a DS-tree (D, l) for Γ such that $R(D, l) = R$ and such that (D, l) is N-reconcilable (using k transfers).*

Proof. ($\Rightarrow$) Let (G, α) be a gene tree reconciled with N such that (G, α) displays R using k transfers, and let e^* be a labeling such that $e^*(u, i) \in \{\mathbb{T}^{\mathbb{S}}, \mathbb{T}^{\mathbb{D}}\}$ if $e(u, i) = \mathbb{T}$, $e^*(u, i) = e(u, i)$ if $e(u, i) \neq \mathbb{T}$, and if $xy \in E$ then $e^*(\textsc{lca}_G(x, y), \textsc{last}) \in \{\mathbb{S}, \mathbb{T}^{\mathbb{S}}\}$, and otherwise $e^*(\textsc{lca}_G(x, y), \textsc{last}) \in \{\mathbb{D}, \mathbb{T}^{\mathbb{D}}\}$.

Now define a binary DS-tree (D, l) as follows. Let $D = G$, and let $l(u) = \mathbb{S}$ if $e^*(\alpha_{\textsc{last}}(u)) \in \{\mathbb{S}, \mathbb{T}^{\mathbb{S}}\}$, and $l(u) = \mathbb{D}$ otherwise (in which case $e^*(\alpha_{\textsc{last}}(u)) \in \{\mathbb{D}, \mathbb{T}^{\mathbb{D}}\}$). Observe that by definition of e^*, if $l(\textsc{lca}_D(x, y)) = \mathbb{S}$ then $xy \in E$, and if $l(\textsc{lca}_D(x, y)) = \mathbb{D}$ then $xy \notin E$. Thus we have that $R = R(D, l)$. Also, note that (D, l) is N-reconcilable using k transfers, since α satisfies the conditions of Definition 4.

($\Leftarrow$): let (D, l) be a DS-tree such that $R(D, l) = R$. Note that D is not necessarily binary. Let (D', l') be a binary refinement of (D, l) such that (D', l') is N-reconcilable (such a refinement is assumed to exist by the lemma statement and by the definition of N-reconcilable for non-binary gene trees). Since (D', l') is N-reconcilable, there exists α' such that (D', α') is a reconciled gene tree with respect to N such that for every $u \in I(D')$, $l'(u) = \mathbb{S}$ implies $e(\alpha_{\textsc{last}}(u)) \in \{\mathbb{S}, \mathbb{T}\}$ and $l'(u) = \mathbb{D}$ implies $e(\alpha_{\textsc{last}}(u)) \in \{\mathbb{D}, \mathbb{T}\}$. Define e^* as follows: if $e(u, i) \neq \mathbb{T}$, then $e^*(u, i) = e(u, i)$; otherwise if $e(u, i) = \mathbb{T}$, if $l'(u) = \mathbb{S}$ then $e^*(u, i) = \mathbb{T}^{\mathbb{S}}$ and if $l'(u) = \mathbb{D}$ then $e^*(u, i) = \mathbb{T}^{\mathbb{D}}$. Note that no additional transfer is created in this manner, and hence e^* still uses k transfers. Also, for any pair of distinct genes $x, y \in \Gamma$ with $u = \textsc{lca}_{D'}(x, y)$, $l'(u) = \mathbb{S}$ implies

$e^*(\alpha_{\text{LAST}}(u)) \in \{\mathbb{S}, \mathbb{T}^{\mathbb{S}}\}$ and $l'(u) = \mathbb{D}$ implies $e^*(\alpha_{\text{LAST}}(u)) \in \{\mathbb{D}, \mathbb{T}^{\mathbb{D}}\}$. It follows that (D', α') display R.

Lemma 3. *Let $H = (V = V_1 \cup V_2 \cup \cdots \cup V_k, E)$ be an instance of k-*MULTICOLORED CLIQUE*. Then in polynomial time, we can construct an instance (T, X, w) of* ACT *such that (T, X, w) has an incomparable assignment of weight $< \infty$ if and only if H has a k-multicolored clique. Furthermore, if an incomparable assignment of weight $w < \infty$ exists, then there exists an incomparable assignment with weight $\leq k' = k^2 + 2k$, and (T, X, w) satisfies the following properties:*

- $w(x, v) \in \{0, 1, \infty\}$ *for all* $x \in X, v \in V(T)$*;*
- $w(x, v) = 0$ *for exactly one* v *for each* $x \in X$*;*
- *If* $w(x, v) = 0$ *then* $w(y, v) = \infty$ *for all* $y \neq x$*;*
- *for any* $x \in X$*,* $u, v \in V(T)$ *such that* $w(x, u), w(x, v) < \infty$*,* u *and* v *are incomparable.*

Proof. **Construction of ACT instance:**

Let $H = V = (V_1 \cup V_2 \cup \cdots \cup V_k, E)$ be an instance of k-MULTICOLORED CLIQUE. We now construct a tree T together with a set X and cost function $w : X \times V(T) \to \mathbb{N}_0 \cup \{\infty\}$. For each element $x \in X$, there will be a single "in"-element *x_in* of $V(T)$, for which $w(x, x\_in) = 0$. There will also be some number of "out"-elements v for which $w(x, v) = 1$.

We begin by describing T. T is made up of a series of subtrees, each of which will act as a gadget in our reduction from k-MULTICOLORED CLIQUE. Every subtree consists of a root with several leaves as children.

The subtrees of T are as follows:

- A tree **Start**, with root *s_in* and children *class_i_in* for each $i \in [k]$;
- For each $i \in [k]$, $v \in V_i$, a tree **Choose**_*v*, with root *v_in*, and children *class_i_out_v*, together with *u_to_i_out_v* for each $u \in V \setminus V_i$ such that $uv \in E$;
- For each $i \in [k]$, $v \in V_i$, a tree **Cover**_*v*, with root *v_out*, and children *count_v_in* , together with *v_to_j_in* for each $j \neq i \in [k]$.
- For each $i \in [k]$, a singleton tree consisting of the node *count_i_out*.

See Figure 4. Finally we add a root node whose children are the roots of all the subtrees given above. This concludes our construction of T.

The set X contains all vertices from V. In addition it contains a 'start' element s, an element *class_i* for each $i \in [k]$, an element *count_v* for each $v \in V$, and an element *v_to_j* for each $v \in V_i$ and $j \neq i \in [k]$.

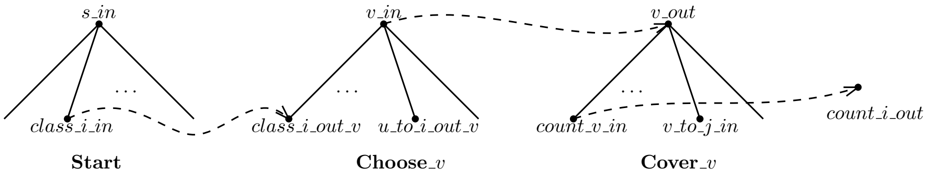


Fig. 4: Figures used in the reduction from k-Multicolored Clique to ACT. Dashed lines represent some of the relations between nodes: If an assignment f does not assign $f(\mathit{class\_i}) = \mathit{class\_i\_in}$, then it must assign $f(\mathit{class\_i}) = \mathit{class\_i\_out\_v}$ (for some $v \in V_i$). Similarly if f does not assign $f(v) = \mathit{v\_in}$, then it must assign $f(v) = \mathit{v\_out}$. If f does not assign $f(\mathit{count\_v}) = \mathit{count\_v\_in}$, then it must assign $f(\mathit{count\_v}) = \mathit{count\_i\_out}$. Note also that if f does not assign $f(\mathit{v\_to\_j}) = \mathit{v\_to\_j\_in}$, then it must assign $f(\mathit{v\_to\_j}) = \mathit{v\_to\_j\_out\_u}$ for some $u \in V_j$ adjacent to v, though that relation is not depicted here.

The cost function $w : X \times V(T) \to \mathbb{N}_0 \cup \{\infty\}$ is defined as follows: For each $i \in [k], v \in V_i$ and $j \neq i \in [k]$, set $w(s, \mathit{s\_in}) = w(\mathit{class\_i}, \mathit{class\_i\_in}) = w(v, \mathit{v\_in}) = w(\mathit{count\_v}, \mathit{count\_v\_in}) = w(\mathit{v\_to\_j}, \mathit{v\_to\_j\_in}) = 0$. For each $i \in [k]$ and $v \in V_i$, set $w(\mathit{class\_i}, \mathit{class\_i\_out\_v}) = 1$, set $w(v, \mathit{v\_out}) = 1$, and set $w(\mathit{count\_v}, \mathit{count\_i\_out}) = 1$. (Note that there are therefore multiple elements $x \in X$ for which $w(x, \mathit{count\_i\_out}) = 1$.) Finally, for each $i \in [k]$ and $v \in V_i$, and each edge $uv \in E$ with $u \in V_j, j \neq i \in [k]$, set $w(\mathit{v\_to\_j}, \mathit{v\_to\_j\_out\_u}) = 1$. For all other $x \in X$ and $v \in V(T)$, set $w(x, v) = \infty$.

This concludes our construction of our ACT instance (X, T, w). The construction can be done in polynomial time.We observe that by construction, $w(x, v) \in \{0, 1, \infty\}$ for all $x \in X, v \in V(T)$, $w(x, v) = 0$ for exactly one v for each $x \in X$, and if $w(x, v) = 0$ then $w(y, v) = \infty$ for all $y \neq x$. To see that u and v are incomparable for $x \in X$, $u, v \in V(T)$ such that $w(x, u), w(x, v) < \infty$, observe that each subtree in the construction contains at most one node z with $w(x, z) < \infty$ for each $x \in X$.

It remains to show that (T, X, w) has an incomparable assignment of weight $< \infty$ if and only if H has a k-multicolored clique and that if an incomparable assignment of weight $w < \infty$ exists, then there exists an incomparable assignment with weight $\leq k'$. To do this, we will first show that the existence of a k-multicolored clique implies the existence of an incomparable assignment with weight $\leq k'$, and then show that the existence of an incomparable assignment of weight $w < \infty$ implies the existence of a k-multicolored clique.

k-multicolored clique implies assignment of weight $\leq k'$:

First suppose that a k-multicolored clique C exists, and let v_i denote the single vertex in $C \cap V_i$, for each $i \in [k]$. Let $f : X \to V(T)$ be defined as follows: Set $f(s) = s\_in$. For each $i \in [k]$, set $f(class\_i) = class\_i\_out\_v_i$. For each $i \in [k]$, set $f(v_i) = v_i\_out$, and for all other $v \in V$ set $f(v) = v\_in$. For each $i \in [k]$, set $f(count\_v_i) = count\_i\_out$, and for all other $v \in V$ set $f(v) = count\_v\_in$. For each $i \in [k]$, $j \neq i \in [k]$, set $f(v_i\_to\_j) = v_i\_to\_j\_out\_v_j$ (note that $v_i\_to\_j\_out\_v_j$ exists because $v_j \in V_j$ and v_i, v_j are adjacent). For all other $v \in V_i$, set $f(v\_to\_j) = v\_to\_j\_in$.

Observe that $\sum_{x \in X} w(x, f(x)) = k+k+k+k(k-1) = k^2+2k = k'$. It remains to show that $f(x)$ and $f(y)$ are incomparable for each $x \neq y \in X$. As each of the subtrees described above are incomparable, it is enough to show that for each subtree, there are no comparable y, z with y, z assigned to different elements of X.

In **Start**, the root s_in is assigned but none of the children $class\_i\_in$ are assigned, so we have no comparable assigned nodes.

In **Choose**$\_v$, if $v = v_i$ for some $i \in [k]$, then the root v_i_in is not assigned, and as all other nodes are children of v_i_in, there are no comparable assigned nodes. For all other v in class V_i, the root v_i_in is assigned. However, the child $class\_i\_out\_v$ is not assigned (as $class\_i$ is assigned to $class\_i\_out\_v_i$), and the other children $u\_to\_i\_out\_v$ are not assigned ($u\_to\_i\_out\_v$ is only assigned if $v = v_i, u = v_j$ for some $i \neq j \in [k]$).

In **Cover**$\_v$, if $v = v_i$ for some $i \in [k]$, then the root v_i_out is assigned, but none of its children $v_i\_to\_j\_in$ or $count\_v_i\_in$ are assigned, as $v_i\_to\_j$ is assigned to $v_i\_to\_j\_out\_v_j$ and $count\_v_i$ is assigned to $count\_i\_out$. For other $v \in V$, the root v_out is not assigned, and as all other nodes are children of v_out, there are no comparable assigned nodes.

The nodes $count\_i\_out$ are the only nodes in T that may be assigned to more than one element of X. However, by definition of f we have that for each $i \in [k]$, $count\_v_i$ is the only element assigned to $count\_i\_out$.

As $\sum_{x \in X} w(x, f(x)) \leq k'$ and $f(x), f(y)$ are incomparable for all $x \neq y \in X$, we have that (X, T, w, k') is a Yes-instance, as required.

Assignment of finite weight implies k-multicolored clique:

Suppose that $f : X \to V(T)$ is an incomparable assignment such that $\sum_{x \in X} w(x, f(x)) < \infty$.

Note that $f(s) = s\_in$, as there is no other node z for which $w(s, z) < \infty$. It follows that $f(class\_i) \neq class\_i\_in$ for each $i \in [k]$. Therefore $f(class\_i) = class\_i\_out\_v$ for some $v \in V_i$. Denote this v by v_i. As $class\_i\_out\_v_i$ is a child of v_i_in in **Choose**$\_v_i$, we must have that $f(v_i) \neq v_i\_in$, and so instead $f(v_i) = v_i\_out$. As v_i_out is the root of **Cover**$\_v_i$, it follows that for each $j \neq i \in [k]$, we cannot have $f(v_i\_to\_j) = v_i\_to\_j\_in$. Therefore $f(v_i\_to\_j) = v_i\_to\_j\_out\_u$ for some $u \in V_j$ adjacent to v_i. Denote this u by u_{ij}.

It remains to show that $u_{ij} = v_j$ for each $i \neq j \in [k]$, as this implies that $v_1, \ldots, v_k$ form a clique. As $f(v_i\_to\_j) = v_i\_to\_j\_out\_u_{ij}$ is a child of u_{ij}_in in **Choose**$\_u_{ij}$, we must have that $f(u_{ij}) \neq u_{ij}\_in$, and so instead $f(u_{ij}) = u_{ij}\_out$. As $count\_u_{ij}\_in$ is a child of u_{ij}_out in **Cover**$\_u_{ij}$, we must have that $f(count\_u_{ij}) \neq count\_u_{ij}\_in$ and so instead $f(count\_u_{ij}) = count\_j\_out$ (recall that $u_{ij} \in V_j$). By a similar argument, since $f(v_j) = v_j\_out$ we also have $f(count\_v_j) = count\_j\_out$. But then f is not an incomparable assignment unless $u_{ij} = v_j$ (since $f(count\_u_{ij})$ and $f(count\_v_j)$ are the same node, and therefore comparable). Therefore we must have that $u_{ij} = v_j$ for all $i \neq j \in [k]$, as required.

Lemma 5. *Let (T, X, w) be an instance of* ACT*, such that $w(x, v) \in \{0, 1, \infty\}$ for all $x \in X, v \in V(T)$, $w(x, v) = 0$ for exactly one v for each $x \in X$, if $w(x, v) = 0$ then $w(y, v) = \infty$ for all $y \neq x$, and for any $x \in X$, $u, v \in V(T)$ such that $w(x, u), w(x, v) < \infty$, u and v are incomparable. Then in polynomial time, we can construct a least-resolved DS-tree (D, l) and time-consistent LGT network N such that for any integer k, (T, X, w) has an incomparable assignment of cost at most k if and only if there exists a binary refinement (D', l') of (D, l) such that (D', l') is N-reconcilable using at most $2k$ transfers.*

Proof. Let (T, X, w) be an instance of ACT satisfying the specified properties. We begin by adjusting T to ensure that it is binary. If an internal node u has a single child, we add an additional child of u as a leaf of the tree. If u has more than two children, we refine u into a binary tree with the same leaf set (treating u as the root of this binary tree). For any new node v introduced in this way, we set $w(x, v) = \infty$ for all $x \in X$. Observe that for the resulting tree T', two nodes $u, v \in V(T)$ are incomparable in T if and only if they are incomparable in T'. Thus, changing T in this way gives us an equivalent instance.

So we may now assume that T is binary. We next describe how to construct a least-resolved DS-tree (D, l) .

Let Γ be a set of genes as follows. For each $x \in X$, Γ contains two new genes x_left and x_right. Let Σ contain species $spec\_x\_left$ and $spec\_x\_right$ for each $x \in X$, with $\sigma(x\_left) = spec\_x\_left$, $\sigma(x\_right) = spec\_x\_right$.

Let the DS-tree (D, l) contain a speciation node r as the root, and let $\{gene\_x : X\}$ be the set of children of r. For each $x \in X$, let $gene\_x$ be a duplication node with children x_left and x_right. Note that (D, l) is a least-resolved DS-tree.

We next describe how to construct the LGT network N, beginning with the distinguished base tree $T_0(N)$. Initially, let $T_0(N) = T$, the input tree of our ACT instance (in its binary version). To avoid confusion with the MWACT instance later, we rename each node $v \in V(T)$ to $spec\_v$. In addition, for each $x \in X$ let u_x be the unique node in T for which $w(x, u_x) = 0$, with $spec\_u_x$ the corresponding node in N.

Now for each $v \in V(T)$, we will add $spec\_v\_left$ and $spec\_v\_right$ as descendants (not necessarily children) of $spec\_v$, as follows. If $spec\_v$ is a leaf in $T_0(N)$, then add $spec\_v\_left$ and $spec\_v\_right$ as children of $spec\_v$. Otherwise, add $spec\_v\_left$ and $spec\_v\_right$ as descendants of different children of $spec\_v$. (This can be be done by subdividing any arc incident to leaf descended from a given child of $spec\_v$, and adding $spec\_v\_left$ or $spec\_v\_right$ as a child of the newly added node). Observe that after $spec\_v\_left$ and $spec\_v\_right$ have been added, $spec\_v$ is the least common ancestor of $spec\_v\_left$ and $spec\_v\_right$. Furthermore this process does not change the least common ancestor of any pair of leaves. Therefore, after doing this process for each $v \in V(T)$, we will have that for every $v \in V(T)$, $spec\_v$ is the least common ancestor of $spec\_v\_left$ and $spec\_v\_right$. When $v = u_x$ for some $x \in X$, we also denote $spec\_v\_left$ and $spec\_v\_right$ by $spec\_x\_left$ and $spec\_x\_right$ respectively.

This completes the construction of the distinguished base tree; now we describe how to add secondary arcs. For each $x \in X$ and each $v \in V(T)$ with $w(x, v) = 1$, we do the following. Add a new tail node between $spec\_v\_left$ and its parent, add a new head node between $spec\_x\_left$ and its parent, and add an arc from the tail to the head as a secondary arc. Similarly, add a new tail node between $spec\_v\_right$ and its parent, and add a new head node between $spec\_x\_right$ and its parent, and add an arc from the tail to the head as a secondary arc. Observe that after this, $spec\_v$ has paths to $spec\_x\_left$ and $spec\_x\_right$ in N, and these paths each use one secondary arc. See Figure 5. Furthermore (by virtue of the fact that $w(y, u_x) \neq 1$ for any $x, y \in X$, and therefore a tail node is never added above $spec\_u_x\_left$ or $spec\_u_x\_right$), every path in N has at most one secondary arc.

This completes the construction of the species network N, and our problem instance. Observe that N is time-consistent, since each time we insert a new secondary arc, its two endpoints are below every other previously inserted node. We now show that (T, X, w) has an incomparable assignment of cost at most k if and only if (D, l) is N-consistent using at most $2k$ transfers.

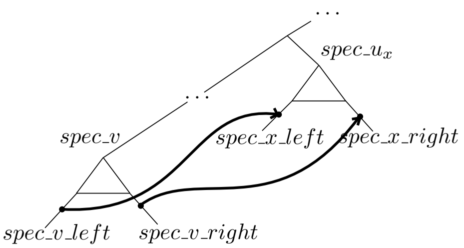


Fig. 5: Part of the species network N constructed in the reduction from ACT to NC. For each $v \in T$, $spec\_v$ is the least common ancestor in N of $spec\_v\_left$ and $spec\_v\_right$. If $w(x, v) = 1$ and $w(x, u_x) = 0$, then secondary arcs (the thick lines) are added from an ancestor of $spec\_v\_left$ to an ancestor of $spec\_x\_left = spec\_u_x\_left$, and from an ancestor of $spec\_v\_right$ to an ancestor of $spec\_x\_right = spec\_u_x\_right$. Thus, there are paths from $spec\_u_x$ to each of $spec\_x\_left$ and $spec\_x\_right$ using 0 transfers in total, and paths from $spec\_v$ to each of $spec\_x\_left$ and $spec\_x\_right$ using 2 transfers in total.

First suppose that (D, l) is N-consistent using at most $2k$ transfers. We will first show the following claim. In this claim and its proof, we use the terms 'ancestor' and 'descendant' to exclusively refer to ancestors or descendants with respect to the distinguished base tree $T_0(N)$:

Claim 1 *For $x \in X$, suppose $u \in V(N)$ is such that there exist paths from u to $spec\_x\_left$ and from u to $spec\_x\_right$, using at most k_x secondary arcs in total. If $k_x = 0$ then u is an ancestor of $spec\_u_x$, and otherwise u is an ancestor of some $spec\_v$ such that $w(x, v) \leq 1$. Moreover, if u is not an ancestor of $spec\_u_x$ then $k_x = 2$.*

Proof. First, recall that $spec\_u_x$ is the least common ancestor of $spec\_x\_left$ and $spec\_x\_right$ in $T_0(N)$. Since $k_x = 0$ implies that u is an ancestor of both $spec\_x\_left$ and $spec\_x\_right$, we have that if $k_x = 0$ then u is an ancestor of $spec\_u_x$.

Since there is a path from u to $spec\_x\_left$, u must be an ancestor of $spec\_v\_left$ for some v such that $w(v, x) \leq 1$ (such nodes are the only ones that have a path to $spec\_x\_left$, either using exclusively principal arcs or a using a single secondary arc). Similarly, u must be an ancestor of $spec\_v'\_right$ for some v' such that $w(v', x) \leq 1$. If $v = v'$ then u is an ancestor of both $spec\_v\_left$ and $spec\_v\_right$ and is therefore an ancestor of $spec\_v$, as required. So assume that $v \neq v'$. If u is an ancestor of $spec\_v$ or $spec\_v'$ then we are done, and otherwise u must be a descendant of both $spec\_v$ and $spec\_v'$ (since it is an ancestor of descendants of both of these). But this implies that v and v' are comparable, a contradiction as $w(x, v), w(x, v') < \infty$.

Finally, we observe that $k_x < 1$ only if u is an ancestor of at least one of $spec\_x\_left$ and $spec\_x\_right$. Therefore if $k_x < 2$ and u is not an ancestor of $spec\_u_x$, it is a descendant of $spec\_u_x$. But this again implies a contradiction as u is an ancestor of some $spec\_v$ with $w(x, v) = 1$, which would then be a descendant of $spec\_u_x$.

Now consider the binary refinement (D', l') of (D, l) that is N-consistent using at most $2k$ transfers. Thus there exists α such that (D', α) is a reconciled gene tree with respect to N. Note that by construction of (D, l), there is a rooted subtree in D' whose leaves are the set of duplication nodes $\{gene\_x : x \in X\}$ and whose internal nodes are all speciation nodes according to l'. For each $x \in X$, there are paths in D' from $gene\_x$ to x_left and to x_right, and so there are paths in N from $\alpha_{\text{LAST}}(gene\_x)$ to $\sigma(x\_left) = spec\_x\_left$ and to $\sigma(x\_right) = spec\_x\_right$. It follows from Claim 1 that $\alpha_{\text{LAST}}(gene\_x)$ is an ancestor of v for some $v \in V(T)$ such that $w(x, v) \leq 1$. By construction of N, there are no paths to such a v using a secondary arc, and therefore as all ancestors of x in D' are speciation nodes, $\{\alpha_{\text{LAST}}(gene\_x) : x \in X\}$ must form the leaves of a subtree in T. It follows that $\alpha_{\text{LAST}}(gene\_x)$ and $\alpha_{\text{LAST}}(gene\_y)$ are incomparable for any $x \neq y \in X$.

Now we can define $f : X \to V(T)$ as follows. For each $x \in X$, let $f(x) = u_x$ if $\alpha_{\text{LAST}}(gene\_x)$ is an ancestor of $spec\_u_x$, and otherwise let $f(x)$ be a $v \in V(T)$ such that $w(x, v) \leq 1$ and $\alpha_{\text{LAST}}(gene\_x)$ is an ancestor of $spec\_u_x$. As their ancestors $\alpha_{\text{LAST}}(gene\_x)$ and $\alpha_{\text{LAST}}(gene\_y)$ are incomparable, it follows that $f(x)$ and $f(y)$ are also incomparable, for any $x \neq y \in X$. Furthermore, by Claim 1 we have that either $\alpha_{\text{LAST}}(gene\_x)$ is an ancestor of $spec\_u_x$, or the paths from $\alpha_{\text{LAST}}(gene\_x)$ to $\sigma(x\_left)$ and to $\sigma(x\_right)$ use 2 secondary arcs. Therefore the number of transfer arcs used by α is 2 for every $x \in X$ with $w(x, f(x)) = 1$. Thus $2k \geq 2\sum_{x \in X} w(x, f(x))$, and so f is an incomparable assignment with $\sum_{x \in X} w(x, f(x)) \leq k$, as required.

Now suppose that (T, X, w) has an incomparable assignment $f : X \to V(T)$ such that $\sum_{x \in X} w(x, f(x)) \leq k$. We will show that (D, l) has a binary refinement (D', l') that is N-reconcilable using at most $2k$ transfers. In particular, we will show that there is a reconciliation α such that $\alpha_{\text{LAST}}(gene\_x) = spec\_f(x)$ for all $x \in X$.

Observe first that as f is an incomparable assignment, there exists a subtree T' of T whose leaves are $\{f(x) : x \in X\}$. By refining the root r of D into a subtree isomorphic to T', we get a refinement (D', l') such that D' with the leaves $\{x\_left, x\_right : x \in X\}$ removed has a reconciliation with N using 0 transfers. Furthermore this reconciliation α is such that $\alpha_{\text{LAST}}(gene\_x) = spec\_f(x)$ for all $x \in X$. It remains to show how to extend α to the leaves $\{x\_left, x\_right : x \in X\}$ of D'.

For each $x \in X$, let P_{x_left} be a path in N from $spec\_f(x)$ to $spec\_x\_left$ using a minimum number of secondary arcs. By construction, this path uses 0 secondary arcs if $w(x, f(x)) = 0$, and at most 1 secondary arc if $w(x, f(x)) = 1$. Similarly, let P_{x_right} be a path in N from $spec\_f(x)$ to $spec\_x\_right$ using a minimum number of secondary arcs. Then for each $x \in X$, we let $\alpha(x\_left) = P_{x\_left}$ and $\alpha(gene\_x\_right) = P_{x\_right}$. It can be seen that (D', α) is a valid reconciliation with respect to N that agrees with (D', l'). Furthermore, α uses 2 transfers for each $x \in X$ such that $w(x, f(x)) = 1$, and no others. Therefore D' is reconcilable using at most $\sum_{x \in X} 2w(x, f(x)) \leq 2k$ transfers, as required.

Theorem 2. *Algorithm minTransferCost is correct and runs in time* $O(2^k k! k |V(D)||V(N)|^4)$.

Proof. We prove the following statement by induction: for each $g \in V(D)$ and $s \in V(N)$, the algorithm finds the minimum number of required transfers for a reconciliation between the subtree D_g and N such that g is mapped to s. If g is a leaf of D, the statement is easy to see, so suppose $g \in I(D)$. Let $(\hat{D}_g, \alpha)$ be an optimal solution for D_g, s and N, i.e. $\hat{D}_g$ is a binary refinement of D_g, α is a reconciliation between $\hat{D}_g$ and N such that $\alpha_{\text{LAST}}(g) = s$, and the pair $(\hat{D}_g, \alpha)$ minimizes the number t of required transfers. If g is binary, then g_l and g_r are children of g in both D_g and $\hat{D}_g$. Let $s_1 = \alpha_{\text{LAST}}(g_l)$ and $s_2 = \alpha_{\text{LAST}}(g_r)$. It is clear that α restricted to $\hat{D}_{g_l}$[1] yields a reconciliation of $\hat{D}_{g_l}$ using $f(g_l, s_1)$ transfers, since if there was a better refinement of D_{g_l} admitting a better reconciliation with g_l mapped to s_1, then we could include this subsolution in $(\hat{D}, \alpha)$ and obtain a lower transfer cost. The same argument holds for g_r and $f(g_r, s_2)$. We thus need to show that the algorithm will, at some point, consider the scenario of mapping g_l with s_1 and g_r with s_2. If $l(g) = \mathbb{S}$, two cases may occur,

---

[1] By the restriction α' of α to $\hat{D}_{g_l}$, we mean $\alpha'(v) = \alpha(v)$ for all strict descendants v of g_l, and $\alpha'(g_l) = (\alpha_{\text{LAST}}(g_l))$

according to Definition 1: (1) $e(\alpha_{\text{LAST}}(g)) = \mathbb{S}$, in which case $\alpha_1(g_l) = s_l$ and $\alpha_1(g_r) = s_r$ (or vice-versa, w.l.o.g.). This implies $s_1 \in P(s_l)$ and $s_2 \in P(s_r)$, and this scenario is tested on line 8 of *reconcileLBR*; (2) $e(\alpha_{\text{LAST}}(g)) = \mathbb{T}$, in which case (s, s') is a transfer-arc, say $s' = s_r$ without loss of generality. Then $\alpha_1(g_l) \in \{s, s_l\}$ and $\alpha_1(g_r) = s_r$ (or vice-versa, w.l.o.g.), which imply $s_1 \in P(s)$ and $s_2 \in P(s_r)$. This is tested by line 12 of *reconcileLBR*. If $l(g) = \mathbb{D}$, we have $\alpha_1(g_l) = \alpha_1(g_r) = s$ and thus it is only required that $\alpha_{\text{LAST}}(g_l) \in P(s)$ and $\alpha_{\text{LAST}}(g_r) \in P(s)$, which is tested on line 14. Therefore, the desired scenario of mapping g_l to s_1 and g_r to s_2 is considered.

One can also observe that no invalid mappings of g_l and g_r are considered by the algorithm (if $l(g) = \mathbb{S}$, we test only the s_1 and s_2 that allow $e(\alpha_{\text{LAST}}(g)) \in \{\mathbb{S}, \mathbb{T}\}$, and similarly for $l(g) = \mathbb{D}$). The fact that the computed value $f'(g, s)$ (and hence $f(g, s)$) is minimum follows from the induction hypothesis on g_l and g_r.

Suppose instead that g has children $g_1, \ldots, g_k$, $k \geq 3$. For a fixed $(D', l') \in \mathcal{B}(g)$, by the induction hypothesis we have that $f(g_i, s')$ is correct for every $i \in [k]$ and $s' \in V(N)$. Using the argumentation for the binary case, it follows that after calling *reconcileLBR*, we have correctly computed the minimum number of transfers for the tree obtained from D_g after replacing g by its local binary refinement D'. The connected subtree B_g of $\hat{D}$ induced by $g, g_1, \ldots, g_k$ is in $\mathcal{B}(g)$, and hence *minTransferCost* will find $f(g, s)$ correctly when trying $D' = B_g$. This concludes the proof, since the time and space complexity of the algorithm was argued in the main text. □

Lemma 6. *Let R be a relation graph and S be a species tree. Then R is S-base-consistent (using k transfers) if and only if there exists a least-resolved DS-tree (D, l) that displays R and a binary refinement (D', l') of (D, l) such that (D', l') is S-base-reconcilable (using k transfers).*

Proof. ($\Rightarrow$) Assume that R is S-base-consistent using k transfers. Then there exists an LGT network N such that $T_0(N) = S$ and R is N-consistent using k transfers. Then by Lemma 2, there is a DS-tree (D, l) and a binary refinement (D', l') such that (D', l') is N-reconcilable using k transfers. Thus by definition, (D', l') is S-base-reconcilable using k transfers.

($\Leftarrow$) Assume that there is a DS-tree (D, l) that displays R and a binary refinement (D', l') of (D, l) such that (D', l') is S-base-reconcilable using k transfers. Then there is an LGT network N such that $T_0(N) = S$ and (D', l') is N-reconcilable using k transfers. Again, by Lemma 2, R

is N-consistent using k transfers. So R is also S-base-consistent using k transfers. $\square$

Lemma 7. *Let (D, l) be a binary DS-tree and let $N := N(D)$ be the species network obtained from S after applying Algorithm 3. Then (D, l) is N-reconcilable.*

Proof. We show that for any $v \in I(D)$, the subtree (D_v, l) is N-reconcilable (where here, we slightly abuse notation by using l to label D_v). Moreover, we show that if v is not a leaf and $s_i, s_j \in L(S)$ are distinct, then there is a reconciliation (D_v, α) with respect to N such that $\alpha(v) = (don_{i \to j}^{d(v)})$ and $e(\alpha_{\text{LAST}}(v)) = \mathbb{T}$ (here, and for the rest of the proof, $d(v)$ refers to the depth of v in D, and *not* its depth in D_v). We use induction on the height $h(D_v)$. First note that if $h(D_v) = 0$, then the statement is trivially true.

As an additional base case, suppose that $h(D_v) = 1$ and fix some $don_{i \to j}^{d(v)}$, with $i \neq j$. Then both children v_l and v_r of v are leaves. Let $s_p = \sigma(v_l)$ and $s_q = \sigma(v_r)$ for some $p, q \in [m]$. Note that $p = q$ is possible.

We find two paths P_1 and P_2 that correspond to $\alpha(v_l)$ and $\alpha(v_r)$. We first claim that in N, there exists a directed path $P_1 = (don_{i \to j}^{d(v)} = x_1, x_2, \ldots, x_{k_1} = s_p)$ such that $x_2 = rec_{j \leftarrow i}^{d(v)}$ (i.e. P_1 starts with the $(don_{i \to j}^{d(v)}, rec_{j \leftarrow i}^{d(v)})$ arc). Observe that there exists a directed path P_1' from $rec_{j \leftarrow i}^{d(v)}$ to s_p. Indeed, if $s_j = s_p$, then $rec_{j \leftarrow i}^{d(v)} = rec_{p \leftarrow i}^{d(v)}$ is an ancestor of s_p and P_1' obviously exists. Otherwise, P_1' starts from $rec_{j \leftarrow i}^{d(v)}$, goes to its descendant $don_{j \to p}^{d(v)+1}$, takes the $(don_{j \to p}^{d(v)+1}, rec_{p \leftarrow j}^{d(v)+1})$ arc and then goes to s_p (observe that $don_{j \to p}^{d(v)+1}$ does exist, since the first loop of the algorithm creating N takes c from 1 to $h(D) + 1$, and $d(v) \leq h(D)$). Since P_1' exists and $(don_{i \to j}^{d(v)}, rec_{j \leftarrow i}^{d(v)})$ is an arc of N, the P_1 path exists.

By the same arguments, there is a path $P_2 = (don_{i \to j}^{d(v)} = y_1, y_2, \ldots, y_{k_2} = s_q)$.

Now, the existence of P_1 and P_2 imply that we can make v a transfer node. More precisely, we let $\alpha(v) = (don_{i \to j}^{d(v)})$, $\alpha(v_l) = (x_2, x_3, \ldots, x_{k_1} = s_p)$ and $\alpha(v_r) = (don_{i \to j}^{d(v)}, y_2, y_3, \ldots, y_{k_2} = s_q)$. Set $e(\alpha_{\text{LAST}}(v)) = \mathbb{T}$ and $e(v_l, k) \in \{\mathbb{SL}, \mathbb{TL}, \emptyset\}$ for $k \in [|\alpha(v_l)| - 1]$ depending on what type of arc $x_k x_{k+1}$ is, then do the same for each $e(v_r, k)$ and $k \in [|\alpha(v_r)| - 1]$. We have $\alpha_{\text{LAST}}(v) = don_{i \to j}^{d(v)}$, $\alpha_1(v_l) = rec_{i \leftarrow j}^{d(v)}$ and $\alpha_1(v_r) = don_{i \to j}^{d(v)}$, and since $(don_{i \to j}^{d(v)}, rec_{j \leftarrow i}^{d(v)}) \in E_s(N)$, condition a.4 of Definition 1 is satisfied, and so α is a reconciliation in which $e(\alpha_{\text{LAST}}(v)) = \mathbb{T}$. This proves the base case.

Let $v \in V(D)$ such that $h(D_v) > 1$, and assume now by induction that the claim holds for any internal node v' such that $D_{v'}$ has height smaller than $h(D_v)$. Let v_l, v_r be the children of v. At least one of v_l, v_r must be an internal node, say v_r without loss of generality. Suppose first that v_l is a leaf. As before, in N there is a path $P_1 = (x_1, x_2, \ldots, x_{k_1})$ starting with the $(x_1, x_2) = (don^{d(v)}_{i \to j}, rec^{d(v)}_{j \leftarrow i})$ arc and that goes to $x_{k_1} = \sigma(v_l)$. As for v_r, by induction D_{v_r} is N-reconcilable by some reconciliation (D_{v_r}, α') such that $\alpha'(v_r) = (don^{d(v)+1}_{i \to j})$. Now, in N there is a path $P_2 = (don^{d(v)}_{i \to j} = y_1, y_2, \ldots, y_{k_2} = don^{d(v)+1}_{i \to j})$ from $don^{d(v)}_{i \to j}$ to $don^{d(v)+1}_{i \to j}$ in which each arc is in $E_p(N)$. We can obtain the desired reconciliation α from α' in the following manner. First let $\alpha(v) = (don^{d(v)}_{i \to j})$ and $\alpha(v_l) = (x_2, x_3, \ldots, x_{k_1})$. For every strict descendant v'_r of v_r, let $\alpha(v'_r) = \alpha'(v_r)$, and finally let $\alpha(v_r) = (don^{d(v)+1}_{i \to j} = y_1, y_2, y_3, \ldots, y_{k_2} = don^{d(v)+1}_{i \to j})$. As in the base case, we can set $e(\alpha_{\text{LAST}}(v)) = \mathbb{T}$ and satisfy condition a.4 of Definition 1. We set $e(v_r, k) \in \{\mathbb{SL}, \mathbb{TL}, \emptyset\}$ accordingly for every $k \in [|\alpha(v_r)| - 1]$ (depending on what type of arc $x_k x_{k+1}$ is) and set $e(\alpha_{\text{LAST}}(v_r)) = e(\alpha'_{\text{LAST}}(v_r))$. Finally we set $e(\alpha_k(v'_r)) = e(\alpha'_k(v'_r))$ for every strict descendant v'_r of v_r and every $k \in [|\alpha(v'_r)|]$. We have that $\alpha(v), \alpha(v_l)$ and $\alpha(v_r)$ satisfy Definition 1, $e(\alpha_{\text{LAST}}(v_r)) = e(\alpha'_{\text{LAST}}(v_r))$ and every other gene-species mapping and event is unchanged from α'. It follows that α is a reconciliation. Since $e(\alpha_{\text{LAST}}(v)) = \mathbb{T}$, the claim is proved for this case.

If instead both $v_l, v_r \in I(D)$, then by induction, D_{v_l} is N-reconcilable with reconciliation α^l such that $\alpha^l(v_l) = (don^{d(v)+1}_{j \to i})$ (notice the use of $j \to i$ and not $i \to j$). Moreover, D_{v_r} is N-reconcilable with reconciliation α^r such that $\alpha^r(v_r) = (don^{d(v)+1}_{i \to j})$. In N, there is a path $P_1 = (x_1, x_2, \ldots, x_{k_1})$ starting with the $(x_1, x_2) = (don^{d(v)}_{i \to j}, rec^{d(v)}_{j \leftarrow i})$ arc that goes to $x_{k_1} = don^{d(v)+1}_{j \to i}$. There is also a path $P_2 = (y_1, y_2, \ldots, y_{k_2})$ from $y_1 = don^{d(v)}_{i \to j}$ to $y_{k_2} = don^{d(v)+1}_{i \to j}$ that uses only arcs from $E_p(N)$. Thus as before, we can make v a transfer node. That is we set $\alpha(v) = (don^{d(v)}_{i \to j})$ and $e(\alpha_{\text{LAST}}(v)) = \mathbb{T}$, $\alpha(v_l) = (x_2, \ldots, x_{k_1} = don^{d(v)+1}_{j \to i})$ and $\alpha(v_r) = (don^{d(v)}_{i \to j} = y_1, y_2, \ldots, y_{k_2} = don^{d(v)+1}_{i \to j})$. We set $e(v_l, k), e(v_r, k') \in \{\mathbb{SL}, \mathbb{TL}, \emptyset\}$ accordingly for every $k \in [|\alpha(v_l)| - 1], k' \in [|\alpha(v_r)| - 1]$, set $e(\alpha_{\text{LAST}}(v_l)) = e(\alpha^l_{\text{LAST}}(v_l)), e(\alpha_{\text{LAST}}(v_r)) = e(\alpha^r_{\text{LAST}}(v_r))$, and keep every other gene-species mapping and event from α^l and α^r unchanged. In this manner $\alpha(v)$ satisfies Definition 1, and α is a reconciliation. Again since $e(\alpha_{\text{LAST}}(v)) = \mathbb{T}$, the claim is proved. $\square$

Theorem 3. *A relation graph R is S-base-consistent if and only if there exists a DS-tree (D, l) such that $R(D, l) = R$.*

Proof. If there is no DS-tree (D, l) such that $R(D, l) = R$, then by Lemma 2 there exists no species network N with which R is consistent, and thus R cannot be S-base-consistent. Conversely, let $(D'l')$ be a DS-tree such that $R(D', l') = R$, and let (D, l) be a binary refinement of (D', l') (recalling that $R(D, l) = R(D', l') = R$). Then by Lemma 7, (D, l) is $N(D)$-reconcilable, where the network $N(D)$ is the one constructed from S by the algorithm described above. By Lemma 2, R is $N(D)$-consistent and thus R is also S-base-consistent. $\square$

Proof of Theorem 4: NP-hardness of minimizing transfers with unknown transfer highways

The formal problem that we show NP-hard here in the following.

TRANSFER MINIMIZATION SPECIES TREE CONSISTENCY (TMSTC):
Input: A relation graph R, a species tree S, an integer k.
Question: Is R S-base-consistent using at most k transfers?

We reduce the feedback arc set problem to TMSTC.

Feedback Arc Set (FAS):
Input: A directed graph $H = (V, A)$ and an integer k.
Question: Does there exist a *feedback arc set* of size at most k, i.e. a set of arcs $A' \subseteq A$ of size at most k such that $H' = (V, A \setminus A')$ contains no directed cycle?

Given a FAS instance $H = (V, A)$, we construct a DS-tree (D, l) and a species tree S such that H admits a feedback arc set of size at most k if and only if $R(D, l)$ is S-base-consistent using at most $K = 2|A| + k$ transfers.

A *caterpillar* is a rooted binary tree in which every internal node has exactly one child that is a leaf, except for one node that has two leaf children. We denote a caterpillar on leafset $x_1, x_2, \ldots, x_n$ by $(x_1|x_2|\ldots|x_n)$, where the x_i nodes are ordered by depth in non-decreasing order (thus x_1 is the leaf child of the root). A *subtree caterpillar* is a rooted binary tree obtained by replacing some leaves of a caterpillar by rooted subtrees. If each x_i is replaced by a subtree X_i, we denote this by $(X_1|X_2|\ldots|X_n)$. If some X_i is a leaf x_i (i.e. X_i a tree with one vertex x_i), we may write $(X_1|\ldots|X_{i-1}|x_i|\ldots|X_n)$.

Given the FAS instance $H = (V, A)$, first order V and A arbitrarily, and denote $V = (v_1, v_2, \ldots, v_n)$ and $A = (a_1, a_2, \ldots, a_m)$. The species tree S has a corresponding subtree for each vertex of V and each arc of A. For each vertex $v_i \in V$, let S_{v_i} be a caterpillar $(v_{i,1}|v_{i,2}|\ldots|v_{i,2K})$ with $2K$ leaves. For each $j \in [2K]$, denote $z_{i,j} = p(v_{i,j})$ (noting that $z_{i,2K-1} = z_{i,2K}$). Then, for each arc $a \in A$, let S_a be the binary tree on two leaves p_a, q_a. Then S is the subtree-caterpillar $(S_{a_1}|S_{a_2}|\ldots|S_{a_m}|S_{v_1}|S_{v_2}|\ldots|S_{v_n})$. See Figure 6.

The DS-tree (D, l) has one subtree for each arc of A. For each $a = (v_i, v_j) \in A$, let $D_a = D_{i,j}$ be a caterpillar with $4K + 2$ leaves such that $D_{i,j} = (v^1_{i,1}|v^2_{i,1}|v^1_{i,2}|v^2_{i,2}|\ldots|v^1_{i,2K}|v^2_{i,2K}|w^i_{j,1}|w^i_{j,2})$ (we will interchangeably use the D_a and $D_{i,j}$ notations whenever convenient). Here the indices of the leaf labels indicates the species containing them, i.e. for each $h \in [2K], \sigma(v^1_{i,h}) = \sigma(v^2_{i,h}) = v_{i,h}$, and $\sigma(w^i_{j,1}) = v_{j,1}, \sigma(w^i_{j,2}) = v_{j,2}$. Thus all the leaves of $L(D_{i,j})$ are from the S_{v_i} subtree, with the exception of $w^i_{j,1}$ and $w^i_{j,2}$ at the bottom. For each $h \in [2K]$, the parent of $v^1_{i,h}$ is labeled by $\mathbb{D}$ whereas the parent of $v^2_{i,h}$ is labeled by $\mathbb{S}$. The parent of $w^i_{j,1}$ and $w^i_{j,2}$ is labeled by $\mathbb{D}$. We define another tree $D'_a = D'_{i,j} = (p^1_a|p^2_a|q^1_a|q^2_a|D_{i,j})$. The parents of p^1_a and q^1_a are labeled $\mathbb{D}$, whereas the parents of p^2_a and q^2_a are labeled $\mathbb{S}$ (here $\sigma(p^1_a) = \sigma(p^2_a) = p_a$ and $\sigma(q^1_a) = \sigma(q^2_a) = q_a$).

Finally, we let

$$D = (D'_{a_1}|p^3_{a_2}|D'_{a_2}|p^3_{a_3}|D'_{a_3}|\ldots|p^3_{a_{m-2}}|D'_{a_{m-2}}|p^3_{a_{m-1}}|D'_{a_{m-1}}|D'_{a_m})$$

where each $p^3_{a_i}$ is a new leaf with $\sigma(p^3_{a_i}) = p_{a_i}$. The purpose of the $p^3_{a_i}$ is to enforce a binary DS-tree. The root is a speciation, and the main path of D alternates labelings, i.e. for each $1 < i < [m]$, the parent of $p^3_{a_i}$

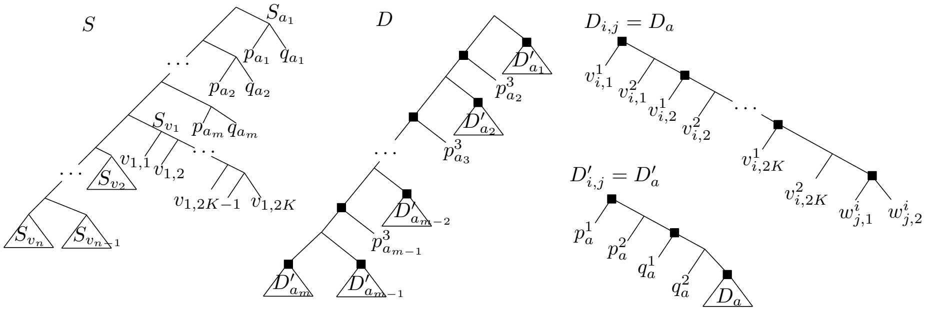


Fig. 6: The S and D trees constructed for our reduction. Duplication nodes appear as squares, and the absence of a square indicates speciation.

is labeled $\mathbb{D}$ and the parent of $r(D'_{a_i})$ is labeled $\mathbb{S}$. The parent of $r(D'_{a_m})$ is labeled $\mathbb{S}$.

It is not hard to see that this construction can be carried out in polynomial time. Note that D is binary and is also a least-resolved DS-tree. Thus by Lemma 6, $R(D,l)$ is S-base-consistent using K transfers if and only if (D,l) is S-base-reconcilable using K transfers.

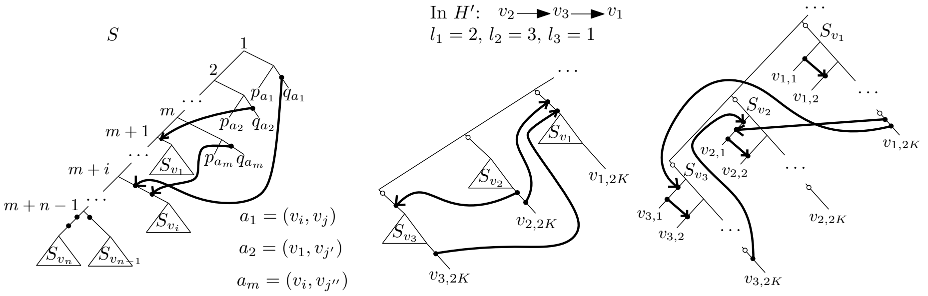
Lemma 8. *If H admits a feedback arc set $A' \subset A$ of size k, then (D,l) is S-base-reconcilable using at most $K = 2m + k$ transfers.*

Proof. The intuition behind the proof is as follows. Each $D_{i,j}$ subtree and its $\mathbb{D}, \mathbb{S}$ labeling could be part of a valid reconciliation with respect to S, if it were not for the $w^i_{j,1}$ and $w^i_{j,2}$ leaves at the bottom, which prevent their ancestors to be speciations. These need to be handled by either making the two edges incident to $w^i_{j,1}$ and $w^i_{j,2}$ a transfer to $v_{j,1}$ and $v_{j,2}$ respectively, or better, by making the edge above their common parent a transfer to some common ancestor of $v_{j,1}$ and $v_{j,2}$. The latter option is preferred as it requires one less transfer, but it cannot be taken for every $D_{i,j}$ subtree because we will likely create time-inconsistencies. As it turns out, given a feedback arc set A' of size k, we have a way of taking these 'double-transfers' only k times. As mentioned before, this is similar to the proof in [40]. The difficulty here however, is to ensure that time-consistency is preserved and that the $\mathbb{D}, \mathbb{S}$ labeling can be preserved.

We first show how to add secondary arcs to S in a time-consistent manner in order to obtain N, by making the time function t explicit. We will add more arcs than necessary, but this simplifies the exposition. Let $s_1, \ldots, s_{n+m-1}$ be the vertices on the $r(S) - r(S_{v_n})$ path in S (excluding $r(S_{v_n})$), ordered by depth in increasing order. Assign time slot $t(s_\ell) = \ell$ for each $\ell \in [n + m - 1]$. We then describe the transformation from S to N in three steps.

Step 1: transfer arcs from q_{a_ℓ} to S_{v_i}. We process each arc $a_\ell \in A$ for $\ell = 1, 2, \ldots, m$ in increasing order as such: first let $(v_i, v_j) = a_\ell$ (i.e. v_i, v_j are the vertices of the a_ℓ arc in H). Assign time slot $\ell + 1$ to the parent of nodes p_{a_ℓ} and q_{a_ℓ}. Then, subdivide $(q_{a_\ell}, p(q_{a_\ell}))$, creating a new node that we call $send\_q_{a_\ell}\_to\_i$. Next, subdivide $(p(r(S_{v_i})), r(S_{v_i}))$, creating a new node that we call $recv\_i\_from\_q_{a_\ell}$. Then add the secondary arc $(send\_q_{a_\ell}\_to\_i, recv\_i\_from\_q_{a_\ell})$. See Figure 7(1) for an illustration. Assign the time slot $m + n + \ell$ to the two newly created nodes.

Note that this process is repeated for each arc a_ℓ in order. Therefore, $p(r(S_{v_i}))$ may change during the process as new secondary arcs are inserted. In the end, there is exactly one outbound transfer node inserted above each q_{a_ℓ}, and $|N^+(v_i)|$ inbound transfer nodes inserted above each $r(S_{v_i})$, where $N^+(v_i)$ is the set of out-neighbors of v_i in H. One can check that no time inconsistency is created so far, since every time a node is inserted, it is added below every other internal node having a defined time slot so far, and it is assigned a higher time slot (since $m + n + \ell$ is always the highest time slot so far, for each $\ell \in [m]$). Also note for later reference that, assuming $n \leq m$, $t(p(r(S_{v_i}))) \leq m+n+\ell$ for some $\ell \leq m$, and therefore $t(p(r(S_{v_i}))) \leq 3m$ after these operations.



(1) (2) (3)

Fig. 7: An illustration of the modifications from S to N. (1) We first add the transfers between the S_{a_ℓ} subtrees to the S_{v_i} subtrees. For the purpose of the example, we have only illustrated the arcs $a_1 = (v_i, v_j), a_2 = (v_1, v_{j'}), a_m = (v_i, v_{j''})$ (the j, j', j'') indices are irrelevant for this step). Here the node added above q_{a_1} would be named *send_*q_{a_1}*_to_i* and its endpoint is *recv_i_from_*q_{a_1}. (2) We then add "forward-transfers", which are secondary arcs from the bottom of $S_{v_{l_i}}$ to the top of $S_{v_{l_j}}$, where $j > i$. Here we illustrate this step on a small example of H', with the topological sorting (v_2, v_3, v_1). The white nodes indicate that other transfer nodes could be on the subpath due to the previous step. (3) We finally allow transferring "backwards" from $v_{i,2K}$ to $v_{j,1}$, $j < i$, then from $v_{j,1}$ to $v_{j,2}$.

For what follows, let $H' = (V, A \setminus A')$. Since H' is a directed acyclic graph, it admits a topological sort, i.e. an ordering $(v_{l_1}, v_{l_2}, \ldots, v_{l_n})$ of V such that if $i < j$, then (v_{l_j}, v_{l_i}) is not an arc of H' (in other words, there are no backwards arcs). We now add two new sets of arcs that are entirely based on the ordering $(v_{l_1}, \ldots, v_{l_n})$.

Step 2: transfer arcs from $v_{l_i,2K}$ to its successor subtrees. What we want to achieve in this step is that for each v_{l_i}, we can transfer from the parent of $v_{l_i,2K}$ to any subtree $S_{v_{l_h}}$ such that $h > i$. An example is provided in Figure 7(2). Process each vertex $v_{l_i} \in V$ for $i = 1, 2, \ldots, n$ in increasing order as follows. First we create the transfer nodes above $r(S_{v_{l_i}})$ that are destined to receive from the predecessors of v_{l_i}. For each $j = 1, 2, \ldots, i-1$ in order, add a node $recv\_l_i\_from\_l_j$ on the edge between $r(S_{v_{l_i}})$ and its parent, and assign the time slot

$$t(recv\_l_i\_from\_l_j) = (4 + i)Km + j$$

Then, we create the nodes above $v_{l_i,2K}$ that are destined to send to the successor subtrees of v_{l_i}. For each $j = i + 1, i + 2, \ldots, n$ in increasing order, add a node $send\_l_i\_to\_l_j$ on the $(p(v_{l_i,2K}), v_{l_i,2K})$ arc. For each such j, assign time slot

$$t(send\_l_i\_to\_l_j) = (4 + j)Km + i$$

Then, for each $i, j \in [n]$ with $i < j$, add a transfer arc from $send\_l_i\_to\_l_j$ to $recv\_l_j\_from\_l_i$. Note that this transfer arc satisfies our time consistency requirement since $t(send\_l_i\_to\_l_j) = (4+j)Km+i = t(recv\_l_j\_from\_l_i)$. Also note that for each arc (v_{l_i}, v_{l_j}) in $A \setminus A'$, there is a corresponding secondary arc from $send\_l_i\_to\_l_j$ to $recv\_l_j\_from\_l_i$.

We argue that S is still time-consistent. We know already that secondary arcs so far have the same timing, so we must show that (1) no node has a child with a greater time slot, and (2) there is a way to assign a time slot to the nodes $z_{i,1}, \ldots, z_{i,2K-1}$ within the S_{v_i} trees. For (1), all the receiving and sending nodes inserted at the last step have a time slot greater than $3m$ and are inserted below the nodes that had a time slot assigned at the previous step (which were assigned a time slot at most $3m$). Moreover, the $recv\_l_i\_from\_l_j$ nodes are inserted on the $p(r(S_{v_{l_i}}))r(S_{v_{l_i}})$ arc in increasing order of time, as well as the $send\_l_i\_to\_l_j$ nodes on the $(p(v_{l_i,2K}), v_{l_i,2K})$ arc. Hence no inconsistency is created within the S_{v_i} trees. For (2), note that for each $i \in [m]$, the nodes $z_{i,1}, \ldots, z_{i,2K-1}$ of S_{v_i} lying on the path between $recv\_l_i\_from\_l_{i-1}$ (above $r(S_{v_i})$) and $send\_l_i\_to\_l_{i+1}$ (at the bottom of S_{v_i}) all have an available time slot between $(4 + i)Km + i - 1$ and $(4 + i + 1)Km + i$, since there are $2K - 1$ such nodes and there are $Km + 1$ available time slots. Therefore, we can assign a time to each $z_{i,h}$ so that time consistency holds. Note that all internal nodes of S have been assigned a time slot so far.

Step 3: escape route from $v_{l_i,2K}$ to $v_{l_j,1}$, then to $v_{l_j,2}$. Again, process each vertex v_{l_i} for $i = 1, 2, \ldots, n$ in increasing order. We make, for $j < i$, a "last-resort escape route" from $v_{l_i,2K}$ to $v_{l_j,1}$, followed by a transfer arc going from $v_{l_j,1}$ to $v_{l_j,2}$. Taking these arcs in a reconciliation corresponds to taking "backwards arcs", i.e. that belong to A'. For that purpose, we add, on the arc between $v_{l_i,2K}$ and its parent, $i-1$ transfer nodes to send backwards. Then on the arc between $v_{l_i,1}$ and its parent, we add $n-i$ transfer nodes to receive from the front. This step is illustrated on Figure 7(3).

More precisely, for each $j = 1, 2, \ldots, i-1$, add a node $backsend\_l_i\_to\_l_j$ on the edge between $v_{l_i,2K}$ and its parent. Assign a high time slot to this node, say $t(backsend\_l_i\_to\_l_j) = (Km)^{10} + i + j$. Then for each $j = i+1, i+2, \ldots, n$, add a node $backrecv\_l_i\_from\_l_j$ on the edge between $v_{l_i,1}$ and its parent. Assign $t(backrecv\_l_i\_from\_l_j) = (Km)^{10} + i + j$. Note that time consistency is still preserved by these node insertions. Then for each $i, j \in [n]$ with $i > j$, add a secondary arc from $backsend\_l_i\_to\_l_j$ to $backrecv\_l_j\_from\_l_i$. Again, these arcs are time-consistent since $t(backsend\_l_i\_to\_l_j) = (Km)^{10} + i + j = t(backrecv\_l_j\_from\_l_i)$.

To finish the network, for each $i \in [n]$, add a secondary arc $(send12\_i, recv12\_i)$ between the $(p(v_{i,1}), v_{i,1})$ arc and the $(p(v_{i,2}), v_{i,2})$ arc. In order to preserve time-consistency, assign a large enough time slot, say m^{100} to both newly created nodes. This finally concludes the construction. Let us call the resulting network N.

For the remainder, let $u, v \in V(N)$ and suppose that there is a path from u to v in N that does not use a secondary arc. We denote this path by $[u \mathinner{..} v]$. We will also denote by $]u \mathinner{..} v]$ the path $[u \mathinner{..} v]$, but excluding u from this path.

Reconciling (D, l) with N. We are finally ready to show that (D, l) is N-reconcilable using at most K transfers. We begin by showing how to reconcile $D_{i,j}$ for $a = (v_i, v_j) \in A$. For reasons that will become apparent later, the edge above $r(D_{i,j})$ will always be contain a transfer. To be more precise, set $\alpha_1(p(v^1_{i,1})) = recv\_i\_from\_q_a$ with $e(p(v^1_{i,1}), 1) = \emptyset$ (setting it up to receive a transfer). Then set $\alpha_{\text{LAST}}(p(v^1_{i,1})) = z_{i,1}$ with $e(p(v^1_{i,1}), \text{LAST}) = \mathbb{D}$. Since there is a directed path from $recv\_i\_from\_q_a$ to $z_{i,1}$ that uses no secondary arc of N, $\alpha(p(v^1_{i,1}))$ can be completed with the appropriate $\mathbb{SL}$ events. Set $\alpha(p(v^2_{i,1})) = (z_{i,1})$ and for each $2 \leq h \leq 2K-1$, set $\alpha(p(v^1_{i,h})) = \alpha(p(v^2_{i,h})) = (z_{i,h})$ (we will handle the case $h = 2K$ later). Then set $e(\alpha_{\text{LAST}}(p(v^1_{i,h}))) = \mathbb{D}$ and $e(\alpha_1(p(v^2_{i,h}))) = \mathbb{S}$. Note that the assigned events are the same as in the DS labeling l of D, and that so far α satisfies Definition 1. It is straightforward to set $\alpha(v^1_{i,h})$ and $\alpha(v^2_{i,h})$ appropriately.

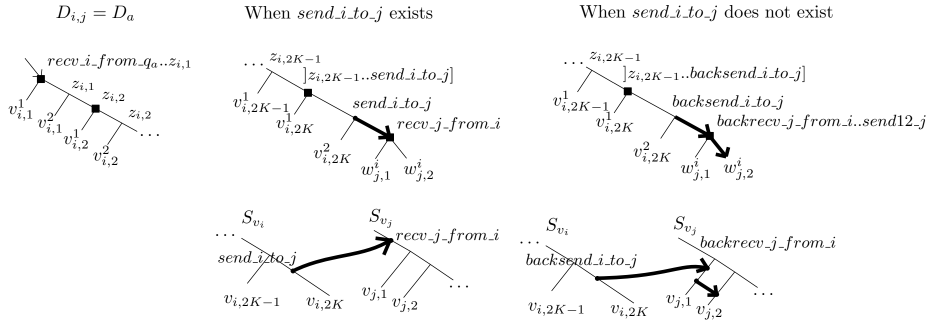


Fig. 8: Top left: how the $D_{i,j}$ subtree is reconciled from its root down to the parent of $v^1_{i,2K-1}$. Top center and top right : the two possible reconciliations of $D_{i,j}$. In the first case, we can handle the w^i_j nodes using a single transfer above S_{v_j}. In the second case, we must transfer on the arc leading to $v_{j,1}$, then use another to get to $v_{j,2}$. Bottom: the transfer highways of N used by both scenarios.

We now handle the nodes $p(v^1_{i,2K})$ and $p(v^2_{i,2K})$ (see Figure 8 for an illustration). First denote by w the parent of both $w^i_{j,1}$ and $w^i_{j,2}$ in $D_{i,j}$. Suppose that $a = (v_i, v_j)$ is not in A'. Recall the ordering $v_{l_1}, \dots, v_{l_n}$ from above. Then there are i' and j' such that $i = l_{i'}$ and $j = l_{j'}$, with $i' < j'$. Therefore N has a secondary arc $(send\_l_{i'}\_to\_l_{j'}, recv\_l_{j'}\_from\_l_{i'}) = (send\_i\_to\_j, recv\_j\_from\_i)$ starting above $v_{i,2K}$ and ending above S_{v_j}. We make the parent edge of w borrow this transfer arc. For that purpose, set $\alpha(p(v^1_{i,2K})) = \;]z_{i,2K-1} \,..\, send\_i\_to\_j]$ and $\alpha(p(v^2_{i,2K})) = (send\_i\_to\_j)$, setting $e(p(v^1_{i,2K}), \textsc{last}) = \mathbb{D}$ and $e(p(v^2_{i,2K}), \textsc{last}) = \mathbb{T}$. For the child leaves, set $\alpha(v^1_{i,2K}) = \alpha(v^2_{i,2K}) = [send\_i\_to\_j \,..\, v_{i,2K}]$. Then we set $\alpha(w) = [recv\_j\_from\_i \,..\, z_{j,1}]$ with $e(w, \textsc{last}) = \mathbb{D}$. It is straightforward to check that $\alpha(w^i_{j,1})$ and $\alpha(w^i_{j,2})$ can be set without requiring any additional transfer, since $z_{j,1}$ is an ancestor of both $v_{j,1}$ and $v_{j,2}$.

Now, suppose instead that $a = (v_i, v_j) \in A'$. Then the transfer arc used in the previous case does not exist, since it is backwards with respect to our ordering. In this case, we must use the last-resort route, namely the secondary arcs $(backsend\_i\_to\_j, backrecv\_j\_from\_i)$ arc, then the $(send12\_j, recv12\_j)$ arc. More precisely, set $\alpha(p(v^1_{i,2K})) = \;]z_{i,2K-1} \,..\, backsend\_i\_to\_j]$ and $\alpha(p(v^2_{i,2K})) = (backsend\_i\_to\_j)$, with $e(p(v^1_{i,2K}), \textsc{last}) = \mathbb{D}$ and $e(p(v^2_{i,2K}), \textsc{last}) = \mathbb{T}$. Then set $\alpha(v^1_{i,2K}) = \alpha(v^2_{i,2K}) = [backsend\_i\_to\_j \,..\, v_{i,2K}]$. Then let $\alpha(w) = [backrecv\_j\_from\_i \,..\, send12\_j]$ with $e(w, \textsc{last}) = \mathbb{T}$. Set $\alpha(w^i_{j,1}) = [send12\_j \,..\, v_{j,1}]$ and $\alpha(w^i_{j,2}) = [recv12\_j \,..\, v_{j,2}]$. One can check that α satisfies Definition 1 and in this case, $D_{i,j}$ requires two transfers.

It remains to reconcile the rest of D. We exhibit α for the nodes of $D'_{i,j}$ that are not in $D_{i,j}$. Denote $a = (v_i, v_j)$. In S, denote $r_a = p(p_a) = p(q_a)$. Set $\alpha(p(p_a^1)) = \alpha(p(p_a^2)) = (r_a)$, and $e(p(p_a^1)) = \mathbb{D}, e(p(p_a^2)) = \mathbb{S}$ (we will adjust $\alpha(p(p_a^1))$ later). Then set $\alpha(p(q_a^1)) = [r_a \,..\, \mathit{send\_q_a\_to\_i}]$ with $e(p(q_a^1), \text{LAST}) = \mathbb{D}$, and $\alpha(p(q_a^2)) = (\mathit{send\_q_a\_to\_i})$. Recall that $p(v_{i,1}^1)$ is a child of $p(q_a^2)$ and that $\alpha_1(p(v_{i,1}^1)) = \mathit{recv\_i\_from\_q_a}$. Thus by setting $e(p(q_a^1)) = \mathbb{T}$ we satisfy Definition 1. It is clear that the α values for the leaves p_a^1, p_a^2, q_a^1 and q_a^2 can be set without requiring any additional transfer. We have now reconciled $D'_{i,j}$ such that $\alpha_{\text{LAST}}(r(D'_{i,j})) = r_a$, adding one transfer in the process.

What remains now are the nodes $g_1, g_2, \ldots, g_\ell$, ordered by increasing depth, that lie on the path between $r(D)$ and $r(D'_{a_m})$ (excluding the latter). We claim that none of these nodes requires any transfer. The node g_ℓ is a speciation and has two children $r(D'_{a_{m-1}})$ and $r(D'_{a_m})$: one mapped by α to species $r_{a_{m-1}}$ and the other to r_{a_m}. Then we can set $\alpha(g_\ell) = (lca_S(r_{a_{m-1}}, r_{a_m}))$ and $e(g_\ell, \text{LAST}) = \mathbb{S}$, and adjust $\alpha(r(D'_{a_{m-1}}))$ and $\alpha(r(D'_{a_m}))$ accordingly. Now, $g_{\ell-1} = p(g_\ell)$ is a duplication whose other child is p_{m-1}^3, and thus it is safe to set $\alpha(g_{\ell-1}) = (lca_S(r_{a_{m-1}}, r_{a_m}))$ as well and set $e(g_{\ell-1}, \text{LAST}) = \mathbb{D}$. Since the D_a subtrees are ordered in the same manner in D as the S_a subtrees in S, it is not hard to see inductively that for $i < \ell - 1$, if $l(g_i) = \mathbb{S}$, then g_i has $r(D'_{a_h})$ as a child for some $h < m-1$, which is mapped to r_{a_h}, and the other child is g_{i+1}, mapped to $x := lca_S(r_{a_{h+1}}, r_{a_{h+2}})$. Hence we can set $\alpha(g_i) = (x, r_{a_h})$ and adjust the α values of the two children of g_i accordingly. If $l(g_i) = \mathbb{D}$, we simply set $\alpha(g_i) = \alpha(g_{i+1})$. We are done with the reconciliation α between D and N.

To sum up, if $a \notin A'$, then D'_a requires 2 transfers, and if $a \in A'$, then D'_a requires 3 transfers, and $|A'| = k$. Thus $K = 2m + k$ transfers are added in total. $\square$

We now undertake the converse direction of the proof. We will make use of the following well-known fact on reconciliations.

Lemma 9. *Let S be a species tree and let N be an LGT network obtained by adding secondary arcs to S. Let (D, α) be a reconciliation with respect to N. Let $u \in I(D)$ such that $e(u, \text{LAST}) = \mathbb{S}$ and let v, w be two leaves descending from u such that, for every node z on the path between u and v or on the path between u and w, $\alpha(z)$ contains no $\mathbb{T}$ or $\mathbb{TL}$ event. Then $\alpha_{\text{LAST}}(u) = lca_S(\sigma(v), \sigma(w))$.*

Proof. First note that by the definition of a reconciliation, $\alpha_{\text{LAST}}(u) = \mathbb{S}$ implies that $\alpha_{\text{LAST}}(u)$ must exist in S, since only those nodes can be the tail of two principal arcs in N (recall that this is required by speciation).

Assume without loss of generality that v descends from u_l and w from u_r. Let $P_v = (u = v_1, \dots, v_a = v)$ be the path from u to v and $P_w = (u = w_1, \dots, w_b = w)$ the path from u to w. By the definition of speciation, $\alpha_1(u_l)$ and $\alpha_1(u_r)$ are the two children of $\alpha_{\text{LAST}}(u)$. Moreover, by appending the paths $\alpha(v_2), \dots, \alpha(v_a)$ and eliminating possible repetitions due to duplications, we obtain a path P'_v of N that uses only principal arcs, starts at $\alpha_1(u_l)$ and ends at v. Similarly, appending the paths $\alpha(w_2), \dots, \alpha(w_b)$, we obtain a path P'_w of N that uses only principal arcs, starts at $\alpha_1(u_r)$ and ends at w. Because $\alpha_1(u_l)$ and $\alpha_1(u_r)$ are the children of $\alpha_{\text{LAST}}(u)$ and P'_v and P'_w use only E_p arcs, P'_v and P'_w are vertex-disjoint. Thus $\alpha_{\text{LAST}}(u)$ is a node of N whose two children can start disjoint paths that lead to v and w, respectively. The only node of N from which this is possible is $lca_S(\sigma(v), \sigma(w))$. □

Lemma 10. *If (D, l) is S-base-reconcilable using at most $K = 2m + k$ transfers, then H admits a feedback arc set $A' \subseteq A$ of size at most k.*

Proof. Suppose that (D, l) is S-base-reconcilable using at most K transfers, let N be the species network such that $T_0(N) = S$ and let (D, α) a reconciliation with respect to N using K transfers showing that (D, l) is N-reconcilable. We divide this proof into a series of claims. Without loss of generality, we assume that the secondary arcs on N are minimal, in the sense that every secondary arc of N is used by α.

Claim 2 *For every arc $a = (v_i, v_j) \in A$, in the $D'_{i,j}$ subtree, there is a node x and an integer h such that $e(x, h) \in \{\mathbb{T}, \mathbb{TL}\}$ and x does not belong to $D_{i,j}$.*

Proof. Suppose for contradiction that the claim is false. Denote $y_p := p(p_a^2)$ and $y_q = p(q_a^2)$. Because there is no transfer, we have $e(y_p, \text{LAST}) = e(y_q, \text{LAST}) = \mathbb{S}$, by the orthology requirements of (D, l). By Lemma 9, $\alpha_{\text{LAST}}(y_p) = lca_S(\sigma(p_a^2), \sigma(q_a^2)) = p(p_a) = p(q_a)$. Now consider $\alpha_{\text{LAST}}(y_q)$. By definition of speciation and by the absence of transfers in $\alpha(p(p_a^1))$ and $\alpha(y_q)$, $\alpha_{\text{LAST}}(y_q)$ must be a strict descendant of $\alpha_{\text{LAST}}(y_p) = p(q_a)$. On the other hand, $\alpha_{\text{LAST}}(y_q)$ is a strict ancestor of q_a since $e(y_q, \text{LAST}) = \mathbb{S}$. Moreover, $\alpha_{\text{LAST}}(y_q)$ is a node of S (this is because $e(y_q, \text{LAST}) = \mathbb{S}$, and thus by definition, $\alpha_{\text{LAST}}(y_q)$ must be a node whose two children are principal arcs). We have reached a contradiction, since S contains no node that is a strict descendant of $p(q_a)$ and a strict ancestor of q_a. □

Claim 3 *Let $(v_i, v_j) \in A$. Then there is an internal node x of $D_{i,j}$ such that $\alpha_{\textsc{last}}(x)$ is a node of S_{v_i}.*

Proof. Suppose that for every internal node x of $D_{i,j}$, $\alpha_{\textsc{last}}(x)$ is not a node of S_{v_i}. Let $h \in [2K]$ such that h is odd. We show that there must be a transfer in some node of the path between $v^2_{i,h}$ and $v^2_{i,h+1}$ in $D_{i,j}$. Let us assume that this is not the case. We can thus assume that $e(p(v^2_{i,h}), \textsc{last}) = \mathbb{S}$ and that $\alpha(v^2_{i,h})$ does not contain a $\mathbb{TL}$ event. It follows that $\alpha_{\textsc{last}}(p(v^2_{i,h}))$ is an ancestor of $v_{i,h}$ which, by assumption, does not belong to S_{v_i}. Since we further assume that there is no transfer in $\alpha(p(v^1_{i,h+1})), \alpha(p(v^2_{i,h+1}))$ or $\alpha(v^2_{i,h+1})$, by Lemma 9, we must have $\alpha_{\textsc{last}}(p(v^2_{i,h}) = lca_S(\sigma(v_{i,h}), \sigma(v_{i,h+1}))$. This node is in S_{v_i}, and we have reached a contradiction. Therefore, some transfer must be present in some node of the $v^2_{i,h} - v^2_{i,h+1}$ path.

This holds for every odd h, so $D_{i,j}$ has a least K transfers. But by the previous claim, $D'_{i,j}$ has at least one transfer that is not in $D_{i,j}$, so in total D has strictly more than K transfers, a contradiction. ☐

Claim 4 *Let $(v_i, v_j) \in A$. Then in N, there is a node s of S_{v_i} such that there exists a directed path P_1 from s to $v_{j,1}$ containing a secondary arc (t_1, t'_1), and a directed path P_2 from s to $v_{j,2}$ containing a secondary arc (t_2, t'_2), and such that $D_{i,j}$ uses these secondary arcs (i.e. for each $h \in \{1, 2\}$, either $(\alpha_i(x), \alpha_{i+1}(x)) = (t_h, t'_h)$ for some $x \in V(D_{i,j})$ and integer i, or $(\alpha_{\textsc{last}}(x), \alpha_1(y)) = (t_h, t'_h)$ for some $x, y \in V(D_{i,j})$). Note that $(t_1, t'_1) = (t_2, t'_2)$ is possible.*

Proof. Let x be a node of $D_{i,j}$ satisfying Claim 3 above. Since $s := \alpha_{\textsc{last}}(x)$ is in the S_{v_i} subtree, and that x has descendants $w^i_{j,1}$ and $w^i_{j,2}$ mapped to $v_{j,1}$ and $v_{j,2}$, there must be a path from s to $v_{j,1}$ and from s to $v_{j,2}$. Since s and $v_{j,1}$ (or $v_{j,2}$) are incomparable in S, these paths must contain a secondary arc. Moreover, there must be such paths P_1 and P_2 and some node of $D_{i,j}$ on the $x - v_{j,1}$ path (resp. the $x - (v_{j,2})$ path) that uses the (t_1, t'_1) arc (resp. the (t_2, t'_2) arc). ☐

As specified in the previous claim, $(t_1, t'_1) = (t_2, t'_2)$ is possible. In essence, this happens when S_{v_i} is able to get to S_{v_j}. In the following, let $\hat{A} \subseteq A$ be the set of arcs such that $(v_i, v_j) \in \hat{A}$ if and only if there is a directed path in N from $r(S_{v_i})$ to $r(S_{v_j})$. The set $A' = A \setminus \hat{A}$ will form our feedback arc set, i.e. the arcs to remove to eliminate all cycles.

Claim 5 *$H' = (V, \hat{A})$ contains no directed cycle.*

Proof. Suppose instead that in H', there is a cycle $C = x_1x_2 \dots x_\ell x_1$. By the definition of $\hat{A}$, in N there is a directed path from $r(S_{x_i})$ to $r(S_{x_{i+1}})$ for every $i \in [\ell - 1]$, and from $r(S_{x_\ell})$ to $r(S_{x_1})$. Thus N contains a cycle, contradicting time-consistency. $\square$

Claim 6 $|\hat{A}| \geq m - k$.

Proof. Recall that by Claim 2, D has a transfer in $D'_{i,j}$ that is not in $D_{i,j}$, and these together take up m transfers. Moreover by Claim 4, each $D_{i,j}$ subtree uses at least one transfer. Since D uses at at most $K = 2m + k$ transfers, there can be at most k of the $D_{i,j}$ subtrees that use more than one transfer, and hence at least $m - k$ that only use one.

By Claim 4, for each $(v_i, v_j) \in A$, there is a directed path P_1 in N from $r(S_{v_i})$ to $v_{j,1}$ and a directed path P_2 from $r(S_{v_i})$ to $v_{j,2}$, such that $D_{i,j}$ uses the transfer arc (t_1, t'_1) from P_1 and (t_2, t'_2) from P_2. If $D_{i,j}$ uses one transfer, we must have $(t_1, t'_1) = (t_2, t'_2)$. This is only possible if $t'_1 = t'_2$ is an ancestor of $\textsc{lca}_S(v_{j,1}, v_{j,2}) = r(S_{v_j})$. This shows that there are at least $m - k$ subtrees $D_{i,j}$, and hence arcs (v_i, v_j) such that N has a path from $r(S_{v_i})$ to $r(S_{v_j})$. $\square$

We are done with the proof, since $A' = A \setminus \hat{A}$ is a feedback arc set of H by Claim 5, and $|A'| = |A| - |\hat{A}| \leq m - (m - k) = k$. $\square$

We have shown that that H has a feedback arc set of size k if and only D is S-base-reconcilable using $K = 2m + k$ transfers. By Lemma 6, H has a feedback arc set of size k if and only if the relation graph $R(D)$ is S-base-consistent using K transfers. Therefore we get the following.

Theorem 4. *The TMSTC problem is NP-hard, even if the input relation graph R has a corresponding least-resolved DS-tree that is binary.*