Super-Reconciliation with segmental duplication, transfer and loss events

Maher Mallem[1,2] and Nadia El-Mabrouk[2]

[1] Computer Science Department, ENS Paris-Saclay, Cachan, France
[2] DIRO, Université de Montréal, Montréal, Canada

7 October 2019 - 11 July 2020

1 Introduction

Reconciliation is a long standing modeling of the evolution of gene families, explaining the incongruence between the tree of a given gene family with the phylogenetic tree of the corresponding species by the fact that genes have been subject to events changing their occurrence in genomes, typically gene duplications (D) and gene losses (L) [11]. This modeling allows characterizing genes related through *orthology* (derived from a speciation event) and those related through *paralogy* (derived from a duplication event), with an important implication towards deciphering the function of gene copies. The standard parsimony criteria used to choose among all possible reconciliations is the minimum number of duplications (D-distance) and losses (DL-distance) induced by the reconciliation. This can be computed in linear time using the LCA mapping [11, 19, 21].

Horizontal gene transfer (HGT), largely involved in shaping bacterial gene content, has been included later in the reconciliation analysis of gene families, allowing for the identification of *xenologs*, i.e. genes related through HGT events. In this case, the parsimony problem consists in finding a minimum scenario of duplication loss and transfer events (DTL-distance) explaining a gene tree with respect to a given species tree. The problem of finding a most parsimonious time-consistent DTL scenario has been shown NP-Hard. However the problem becomes polynomial if consistency requirement is relaxed [2, 18].

Although used successfully for many years, one of the major drawbacks of the reconciliation model consists in the fact that gene families are considered to evolve independently one from another. While this assumption is plausible for genes spread apart in the genome, it prevents from explaining the evolution of chromosomal segments, containing a sequence of adjacent genes which evolved together. This is the case of genes organized in operons in bacteria, or paralogons i.e. sets of homologous chromosomal regions - among one or many genomes - sharing the same genes (also called *syntenies*).

While some work has been done for inferring the evolution of syntenies or adjacent genes [3, 10], adjusting the computation of the evolutionary cost to favor co-evolution events - hence grouping individual events into single segmental ones [9] or inferring the minimum number of "duplication episodes" defined as sets of single duplications mapped to the same node in the species tree [16, 6], none of these methods explicitly seeks for an evolutionary scenario minimizing segmental duplication, losses or HGT events. The first attempt to generalize the reconciliation approach to a set of gene trees has been conducted by our group [5]. Given a set of gene families grouped into syntenies, a gene tree for each gene family and a species tree, the *DL Super-Reconciliation* problem was defined as finding an evolutionary scenario of the syntenies in agreement with the individual gene trees, whilst minimizing the number of segmental duplications and losses. We showed that the associated decision problem is NP-hard and we gave an exact exponential-time algorithm to solve it. Alternatively, we showed that accounting for rearrangements in the evolutionary model, while still only

minimizing segmental duplication and loss events, leads to an exact polynomial-time algorithm.

In this work, we generalize the Super-Reconciliation model to handle HGT events, namely we seek for a minimum scenario of segmental duplication, loss, and also transfers explaining the evolution of a set of syntenies from the reconciliation of the set of corresponding gene trees.

2 Preliminary definitions

A *string* or a *sequence* is an ordered set of characters. Given a string $X = x_1 \cdots x_n$, a *substring* of X is a consecutive set of characters from X in the same order as in X (possibly X itself), and a *subsequence* is a set of characters of X in the same order, but not necessarily consecutive in X (X is a substring and a subsequence of X). We also denote by $Set(X) = \{x_1, x_2, ..., x_k\}$ the *range* of X, i.e. the set of all genes contained in X, without any particular order.

All trees are considered rooted. Given a tree T, we denote by $r(T)$ its root, by $V(T)$ its set of nodes and by $\mathcal{L}(T) \subset V(T)$ its leafset. We say that T *is a tree for* $L = \mathcal{L}(T)$. A node v is an *ancestor* of v' if v is on the path from $r(T)$ to v'; v is the *parent* of v' if it directly precedes v' on this path. In this latter case, v' is called the *child* of v. We denote by $E(T)$ the set of edges of T, where an edge is represented by its two terminal nodes (v, v'), with v being the parent of v'. Two nodes v and v' are *separated* in T iff neither one is an ancestor of the other. A node is said to be *unary* if it has a single child and *binary* if it has two children. Given a node v of T, the subtree of T rooted at v is denoted $T[v]$.

A *binary tree* is a tree with all internal (i.e. non-leaf) nodes being binary. If internal nodes have one or two children, then the tree is said *partially binary*.

Creating a unary root consists in creating a new node v, a new edge $(v, r(T))$ and assigning v as the new root of T. *Grafting* a leaf w consists in subdividing an edge (v, v') of T, thereby creating a new node v'' between v and v', then adding a leaf w with parent v''. If W is a rooted tree, *grafting* W *to* T corresponds to grafting a leaf w, then replacing w by the root of W.

The *lowest common ancestor* (LCA) in T of a subset L' of $\mathcal{L}(T)$, denoted $lca_T(L')$, is the ancestor common to all nodes in L' that is the most distant from the root. The restriction $T|_{L'}$ of T to L' is the tree with leafset L' obtained from the subtree of T rooted at $lca_T(L')$ by removing all leaves that are not in L' and all unary nodes. Let T' be a tree such that $\mathcal{L}(T') = L' \subseteq \mathcal{L}(T)$. We say that T *displays* T' iff $T|_{L'}$ is label-isomorphic to T' (i.e., isomorphic with preservation of leaf labels). We also say that T is an *extension* of T'.

Species, gene and synteny trees: The *species tree* S for a set Σ of species represents an ordered set of speciation events that have led to Σ.

A *gene family* is a set Γ of genes where each gene g belongs to a given species $s(g)$ of Σ. If $\Gamma' \subseteq \Gamma$ is a subset of genes, we denote $s(\Gamma') = \{s(g) : g \in \Gamma'\}$.

A *synteny* X is an ordered sequence of genes belonging to a genome $s(X)$. We consider that genes of a synteny all belong to different gene families (tandem

duplications are ignored). More precisely, let $\mathcal{F} = \{\Gamma_1, \Gamma_2, ..., \Gamma_t\}$ be a set of gene families, and $\lambda_{\mathcal{F}} = \{(g, \Gamma) : g \in \Gamma \wedge \Gamma \in \mathcal{F}\}$ be a function. We say that an ordered sequence of genes $X = g_1 g_2 ... g_k$ is a synteny on $\mathcal{F}$ iff $\lambda_{\mathcal{F}}$ is well-defined for all genes of X, $\lambda_{\mathcal{F}}$ is injective, and all genes in X belong to the same species.

A *synteny family* is a set $\mathcal{X}$ of syntenies. We say that a set $\mathcal{F}$ of gene families are *organized into a set $\mathcal{X}$ of syntenies* iff there is a bijection between the genes of $\mathcal{F}$ and the genes in $\mathcal{X}$ (each gene of $\mathcal{F}$ belongs to exactly one synteny of $\mathcal{X}$).

A tree T is a *gene tree* for a gene family Γ (respec. a *synteny tree* for a synteny family $\mathcal{X}$) if its leafset is in bijection with Γ (respec. $\mathcal{X}$).

Given a gene tree T, the *corresponding synteny tree* is the tree $\tilde{T}$ obtained from T by replacing each leaf of T by the synteny containing the considered gene.

Given a tree T (either gene tree or synteny tree), we extend the mapping s to internal nodes v of T by defining $s(v) = lca_S(\{s(l) : l \in \mathcal{L}(T[v])\})$.

An evolutionary history is represented by a *labeled* tree, where the label of a node is its corresponding event. In the case of gene families, an event is entirely determined by its type, either a duplication, a speciation or a loss. The labels of a gene tree are obtained through reconciliation, as described below.

3 DL Super-Reconciliation

In this section, we recall the important elements about DL Super-Reconciliation established in [5] which will be of later use when dealing with DTL Super-Reconciliation.

3.1 Definition

Before extending the reconciliation concept to a set of gene trees, we need to specify an evolutionary model for syntenies. Here, syntenies are considered to have evolved from a single ancestral synteny through speciations (defined as for single genes), segmental duplications and segmental losses, where:

- a speciation $Spe(X, [1, l])$ acting on a synteny $X = g_1 \cdots g_l$ belonging to a genome $s(X)$ has the effect of reproducing X in the two genomes s_l and s_r children of $s(X)$ in S.
- a (segmental) duplication $Dup(X, [i, j])$ acting on a synteny X belonging to a genome $s(X)$ is an operation that copies a substring $g_i \cdots g_j$ of size $j-i+1$ of $X = g_1 g_2 \cdots g_i \cdots g_j \cdots g_l$ somewhere else into the genome $s(X)$, creating a new *copied synteny* $X' = g'_i \cdots g'_j$ where each g'_k, for $i \leq k \leq j$ belongs to the same gene family as g_k; we say that the copied synteny is *partial* if $[i, j] \neq [1, l]$.
- a (segmental) loss $Loss(X, [i, j])$ acting on a synteny $X = g_1 \cdots g_i \cdots g_j \cdots g_l$ is an operation that removes a substring $g_i \cdots g_j$ of size $j-i+1$ of X, leading to the *truncated synteny* $X' = g_1 \cdots g_{i-1} g_{j+1} \cdots g_l$. A loss is called *full* if X' is the empty string (i.e. all genes of X are removed) and *partial* otherwise. We may denote full loss events as $fLoss$ and partial loss events as $pLoss$.

An evolutionary history of a set of syntenies can thus be represented as a partially binary tree where leaves correspond to extant syntenies and lost syntenies (resulting from full losses), and each internal node v corresponds to an event $\mathcal{E}(X,[i,j])$ with $\mathcal{E} \in \{Spe, Dup, pLoss\}$ (and leaves correspond to either extant genes or $fLoss$ events). Thus, in contrast to a single gene family, a tree representing the evolution of a set of syntenies is not only labeled by the type of event corresponding to each internal node, but also by the segment of the synteny affected by the event.

If $\mathcal{E}$ is:

1. Spe, then v is a binary node with two children corresponding to syntenies Y and Z such that $X = Y = Z$ and $s(Y)$ and $s(Z)$ being the two children of $s(X)$ in S.
2. Dup, then v is a binary node with two children corresponding to syntenies X and $X' = X[i,j]$, where $s(X) = s(X')$.
3. $pLoss$, then v is a unary node with a child corresponding to the truncated synteny $X' = X[1,i-1]X[j+1,l]$, and $s(X) = s(X')$.

DL Super-Reconciliation problem:
Input: A set Σ of species and a species tree S for Σ; a set of gene families $\mathcal{F} = \{\Gamma_1, \ldots, \Gamma_t\}$ organized into a set of syntenies $\mathcal{X}$; a set of gene trees $\mathcal{G} = \{T_1, \ldots, T_t\}$, one for each family of $\mathcal{F}$.
Output: A Super-Reconciliation $R(\mathcal{G}, S)$ of minimum DL cost.

Given a set of gene families $\mathcal{F} = \{\Gamma_1, \Gamma_2, \cdots, \Gamma_t\}$ organized into a set of syntenies $\mathcal{X}$, we define the *precedence graph* $\mathcal{P}$ as the directed graph with n vertices, each corresponding to a gene family of $\mathcal{F}$, such that a directed edge (i,j) between two vertices i and j exists iff there is a synteny $X = x_1 x_2 \cdots x_k$ of $\mathcal{X}$ containing a gene in Γ_i preceding a gene in Γ_j, i.e. there is a pair $1 \leq l_1 < l_2 \leq k$ such that $x_{l_1} \in \Gamma_i$ and $x_{l_2} \in \Gamma_j$.

If $\mathcal{P}$ is acyclic, then $\mathcal{P}$ is a Directed Acyclic Graph (DAG). In this case, there is a topological sorting for $\mathcal{P}$, i.e., a linear ordering X of vertices such that for every directed edge (i,j) in $\mathcal{P}$, i precedes j in X. Verifying if a directed graph is acyclic and finding a topological sorting of a DAG is a classical problem solvable in linear time.

The following lemma gives necessary and sufficient conditions for a set of syntenies to be order consistent and exhibits the set of possible ancestral syntenies.

Lemma 1 (Order consistency condition). *Let $\mathcal{F} = \{\Gamma_1, \Gamma_2, \cdots, \Gamma_t\}$ be a set of gene families organized into a set $\mathcal{X}$ of syntenies. Then $\mathcal{X}$ is order consistent iff the corresponding precedence graph $\mathcal{P}$ is acyclic. In this case, any topological sorting for $\mathcal{P}$ is an order consistent ancestral synteny for $\mathcal{X}$.*

A set of trees on subsets of $\mathcal{X}$ is said *consistent* iff, for any triplet $Trp = \{X_1, X_2, X_3\}$ of disjoint elements of $\mathcal{X}$, all trees containing Trp as a sub-leafset exhibit the same topology for Trp.

Lemma 2 (Tree consistency condition). *Let $\mathcal{G} = \{T_1, T_2, \cdots, T_t\}$ be a set of gene trees for a set of gene families organized into a set $\mathcal{X}$ of syntenies, and let S be the species tree. If a Super-Reconciliation $R(\mathcal{G}, S)$ exists, then the set of corresponding synteny trees $\{\tilde{T}_1, \tilde{T}_2, \cdots \tilde{T}_t\}$ is consistent.*

The consistency problem of rooted trees has been widely studied. The BUILD algorithm [1, 4, 14, 17] can be used to test, in polynomial-time, whether a collection of rooted trees is consistent, and if so, construct a compatible, not necessarily fully resolved supertree, i.e. a tree displaying them all. Then the set of all compatible minimally resolved supertrees - which may be exponential in the number of genes - can be determined.

Theorem 1. *Let $\mathcal{G} = \{T_1, T_2 \cdots, T_t\}$ be a set of trees for a set of families organized in an order-consistent set of syntenies $\mathcal{X}$, and S be the species tree. Let $\tilde{\mathcal{G}} = \{\tilde{T}_1, \tilde{T}_2 \cdots, \tilde{T}_t\}$ be the set of synteny trees corresponding to those in $\mathcal{G}$. If $\tilde{\mathcal{G}}$ is a consistent set of trees then:*

1. *A Super-Reconciliation $R(\mathcal{G}, S)$ is an extension of a supertree for $\tilde{\mathcal{G}}$;*
2. *Any supertree is the "backbone" of a Super-Reconciliation. Namely, for any supertree $\tilde{T}$ for $\tilde{\mathcal{G}}$, there is a Super-Reconciliation $R(\mathcal{G}, S)$ which is an extension of $\tilde{T}$.*

3.2 Results

The general method is the following:

- Loop on compatible supertrees
- Apply dynamic programming on each supertree T in order to determine the min. cost of a super-reconciliation based on T

From Theorem 1 a natural algorithm for the Super-Reconciliation problem follows:

1. Explore the space of all order consistent ancestral syntenies A for $\mathcal{X}$;
2. Explore the space of all supertrees $\tilde{T}$ for $\tilde{\mathcal{G}}$;
3. Find a Super-Reconciliation of minimum cost which is an extension of $\tilde{T}$ with A as an ancestral synteny;
4. Select the Super-Reconciliations leading to the minimum cost.

What remains then is to explain how to deal with step 3. more precisely. It consists in solving the following problem:

SMALL-PHYLOGENY FOR SYNTENIES problem (SPFS):
Input: A set Σ of species and a species tree S for Σ; a set of gene families $\mathcal{F} = \{\Gamma_1, \ldots, \Gamma_t\}$ organized into a set of syntenies $\mathcal{X}$; a set of gene trees $\mathcal{G} = \{T_1, \ldots, T_t\}$, one for each family of $\mathcal{F}$; a supertree $\tilde{T}$ with leaves labeled with the syntenies in $\mathcal{X}$ and consistent with $\mathcal{G}$.

Output: A Super-Reconciliation $R(\mathcal{G}, S)$ of minimum DL cost which is an extension of $\tilde{T}$.

As in the *Reconciliation* problem the optimal mapping of species is the LCA one, and the optimal binary node labeling can be inferred from it.

Lemma 3. *Let $R(\mathcal{G}, S)$ be a solution of (SPFS) (i.e. a Super-Reconciliation of minimum cost which is an extension of $\tilde{T}$). Let $v \in V(\tilde{T})$ and let v' be the node corresponding to v in $R(\mathcal{G}, S)$. Then $s(v') = lca_S(s(\mathcal{L}(\tilde{T}[v])))$.*

Proof. See Appendix. □

Lemma 4. *Let $R(\mathcal{G}, S)$ be a solution of (SFPS). Let $v \in V(\tilde{T})$ be an internal node of $\tilde{T}$ and let v' be its corresponding node in $R(\mathcal{G}, S)$. Moreover let v_l and v_r be the children of v. If $s(v_l)$ and $s(v_r)$ are separated in S, then v' is a speciation, and otherwise v' is a duplication.*

Proof. See Appendix. □

Now only the losses remain to be optimized. This is handled with dynamic programming. For $v \in V(\tilde{T})$, define $d(v, X)$ as the minimum number of segmental duplications and losses induced by a synteny assignment on $\tilde{T}[v]$ with X being the assignment at v.

Theorem 2. *Let v be a node of $\tilde{T}$, X be a synteny and $\mathcal{S}(X)$ be the set of subsequences of X.*

- *If v is a leaf, then $d(v, X) = 0$ if X is the extant synteny corresponding to leaf v, and $+\infty$ otherwise;*
- *If v is a speciation with children v_l and v_r, then,*

$$\begin{aligned} d(v, X) = & min_{(X_l \in \mathcal{S}(X))}(D^T(X, X_l) + d(v_l, X_l)) + \\ & min_{(X_r \in \mathcal{S}(X))}(D^T(X, X_r) + d(v_r, X_r)); \end{aligned}$$

- *If v is a duplication node with children v_l and v_r, then*

$$d(v, X) = 1 + min \begin{cases} min_{(X_l \in \mathcal{S}(X))}(D^T(X, X_l) + d(v_l, X_l)) + \\ \quad min_{(X_r \in \mathcal{S}(X))}(D^P(X, X_r) + d(v_r, X_r)), \\ min_{(X_l \in \mathcal{S}(X))}(D^P(X, X_l) + d(v_l, X_l)) + \\ \quad min_{(X_r \in \mathcal{S}(X))}(D^T(X, X_r) + d(v_r, X_r)) \end{cases}$$

Corollary 1. *The* Small-Phylogeny for Syntenies *problem can be solved in time $\mathcal{O}(t2^{t \log t + 2t} n)$, where t is the number of gene families present in the input and n is the number of syntenies.*

4 DTL Super-Reconciliation

This section gathers the contributions made during the internship.

4.1 Definition

We extend the DL Super-Reconciliation to horizontal transfers. A (segmental) horizontal transfer $Tran(u, u')(X, [i, j])$ of source node u and destination node u' acting on a synteny X belonging to a genome $s(X)$ is an operation that copies a substring $g_i \cdots g_j$ of size $j - i + 1$ of $Y \stackrel{\text{def}}{=} synteny(u) = g_1 g_2 \cdots g_i \cdots g_j \cdots g_l$ from the genome $s(Y)$ and includes it somewhere into the genome $s(X)$, creating a new *copied synteny* $Y' = g'_i \cdots g'_j$ where each g'_k, for $i \leq k \leq j$ belongs to the same gene family as g_k; we say that the copied synteny is *partial* if $[i, j] \neq [1, l]$.

Since an internal node is only labeled by one event, each internal node can only be **the destination of at most one horizontal transfer**. As a side note, this does not mean that a species can receive at most one transfer. The "unchanged" child of the transfer can be itself a transfer node for that same species, and so on.

To recap: in a DTL super-reconciliation, every internal node v is labeled by an event $\mathcal{E} \in \{Spe, Dup, pLoss\} \uplus \{Tran(u, u'), u' \text{ node of } \tilde{T}, u' \text{ child of } v\}$. If $\mathcal{E}$ is:

1. *Spe*, then v is a binary node with two children corresponding to syntenies Y and Z such that $X = Y = Z$ and $s(Y)$ and $s(Z)$ being the two children of $s(X)$ in S.
2. *Dup*, then v is a binary node with two children corresponding to syntenies X and $X' = X[i, j]$, where $s(X) = s(X')$.
3. *pLoss*, then v is a unary node with a child corresponding to the truncated synteny $X' = X[1, i-1]X[j+1, l]$, and $s(X) = s(X')$.
4. $Tran(u, u')$ with u a node of $\tilde{T}$, u' a child of v and $Y \stackrel{\text{def}}{=} synteny(u)$, then v is a binary node with two children corresponding to syntenies X and $Y' \stackrel{\text{def}}{=} synteny(u') = Y[i, j]$, where $s(Y) = s(Y')$.

The cost of a DTL super-reconciliation is defined as the number of duplications, losses and horizontal transfers. The definition of the problem is the following:

DTL Super-Reconciliation problem:
Input: A set Σ of species and a species tree S for Σ; a set of gene families $\mathcal{F} = \{\Gamma_1, \ldots, \Gamma_t\}$ organized into a set of syntenies $\mathcal{X}$; a set of gene trees $\mathcal{G} = \{T_1, \ldots, T_t\}$, one for each family of $\mathcal{F}$.
Output: A Super-Reconciliation $R(\mathcal{G}, S)$ of minimum DTL cost.

About order consistency and tree consistency, Lemma 1, Lemma 2 and Theorem 1 apply as in the DL Super-Reconciliation.

It has not been proven yet that the general DTL Super-Reconciliation problem is NP-hard (albeit it is likely to be). However *time-consistent* DTL Super-Reconciliation can be proven NP-hard from the NP-hardness of *time-consistent* DTL reconciliation.

Proposition 1. *If time consistency is required, then the* Super-Reconciliation *problem is NP-hard for the Dup, fLoss, pLoss and Tran cost.*

Proof. From the DTL reconciliation problem, let Σ be the set of species; S the species tree for Σ; Γ the set of genes; T the gene tree.

We define:

- $\mathcal{F} = \{\Gamma\}$
- $\mathcal{X}$ the set of genes Γ (= set of unary syntenies)
- $\mathcal{G} = \{T\}$

We claim that there is a bijection from the set of valid reconciliations of T and S to the set of valid super-reconciliations of $\mathcal{G}$ and S. Because there is a unique gene family in $\mathcal{F}$ and there cannot two genes of the same gene family in a synteny, all the syntenies in a valid super-reconciliation of $\mathcal{G}$ and S have to contain a single gene, which put such super-reconciliations in bijection with the regular reconciliations.

Hence if we have a reconciliation $\mathcal{R}(T, S)$ and we define the super-reconciliation $\mathcal{R}(\mathcal{G}, S)$ by taking $\mathcal{R}(T, S)$ identifying the genes as (unary) syntenies, then $\mathcal{R}(T, S)$ is of minimal cost in DTL Reconciliation(Σ, S, T) if and only if $\mathcal{R}(\mathcal{G}, S)$ is of minimal cost in DTL Super-Reconciliation$(\Sigma, S, \mathcal{F}, \mathcal{X}, \mathcal{G})$. Since the time-consistent DTL reconciliation problem is NP-hard, so is the time-consistent DTL super-reconciliation problem. □

4.2 Computing the minimum cost

As seen previously with Theorem 2, dynamic programming was the way to deal with DL super-reconciliation in [5]. However horizontal transfers prevent dynamic cost computing methods from working. More precisely: if, as described on the left tree in Figure 1, we wanted to compute the cost of a horizontal transfer on node v from node u (and with destination node u'), then the naive way Theorem 2 could be extended would be the following:

$$\begin{aligned} d(v, X) = 1 + \min_{(X_{v'} \in \mathcal{S}(X))}(D^T(X_v, X_{v'}) + d(v', X_{v'})) \\ + \min_{(X_{u'} \in \mathcal{S}(X))}(D^P(X_u, X_{u'}) + d(u', X_{u'})) \end{aligned}$$

where X_α denotes $synteny(\alpha)$ for a node α in $\mathcal{R}(\mathcal{G}, S)$.

The cost of the transfer on node v would be determined by $D^P(X_u, X_{u'})$, with u not in the subtree of v, which could rapidly lead to a deadlock.

One way to circumvent this issue is to decide in advance the transfer locations and transform the partially labeled supertree into another partially labeled supertree with no transfer label but with the same optimal cost. While the whole set of transfer mappings would have to be scanned in the worst case, this could allow us to reinvest some of the results that had been established for the DL super-reconciliation problem.

It is worth noting that u' being a child of v is irrelevant for computing the cost of the transfer. Instead if u' were somehow a child of u, then all the information needed to compute the cost of this transfer would be locally accessible and a dynamic approach would be made possible. This is the objective of the transform detailed below.

The transform

Let $\tilde{T}$ be a supertree compatible with the input trees. Suppose that the nodes labeled $Tran(...)$ and the transfer sources have already been decided. Let $\Theta = \{(source_1, dest_1), (source_2, dest_2), \ldots\}$ be the set of transfers.

If (u, u') is a transfer (source u, destination u'), then the proposed procedure is the one described below and illustrated in Figure 1:

Procedure $f_{(u,u')}$***:***

1. create a node $\tilde{u}$ as the new parent of u
2. remove the edge between v and u' and have $\tilde{u}$ as the new parent of u'
3. put the label Dup on $\tilde{u}$
4. remove v
5. remove (u, u') in the set of transfers
6. if v was a source node in some other transfers: replace v with v' in these transfers

Let l_Θ be a list of the transfers in Θ in some order. The transform $\tau(l_\Theta)$ consists in applying this procedure to all the transfers in l_Θ successively. In other words if: $l_\Theta = [(u_1, u'_1), \ldots, (u_{|\Theta|}, u'_{|\Theta|})]$, then: $\tau(l_\Theta) \stackrel{\text{def}}{=} f_{(u_{|\Theta|}, u'_{|\Theta|})} \circ \ldots \circ f_{(u_1, u'_1)}$. Then the resulting partially labeled super-reconciliation has interesting properties, which will be detailed in the next subsection.

Remark 1. While a transform is primarily meant to be utilized on super-reconciliations, applying it to the supertree $\tilde{T}$ is a convenient shortcut: then it only has to be computed once for the whole set of super-reconciliations which are an extension of $\tilde{T}$.

Proposition 2. *A transform takes $\mathcal{O}(|\Theta| \cdot \alpha(|\Theta|))$ operations, where α is the inverse Ackermann function.*

Proof. A transform consists in applying the procedure described above for each transfer in Θ. We show that each procedure takes $\mathcal{O}(\alpha(|\Theta|))$ operations. We

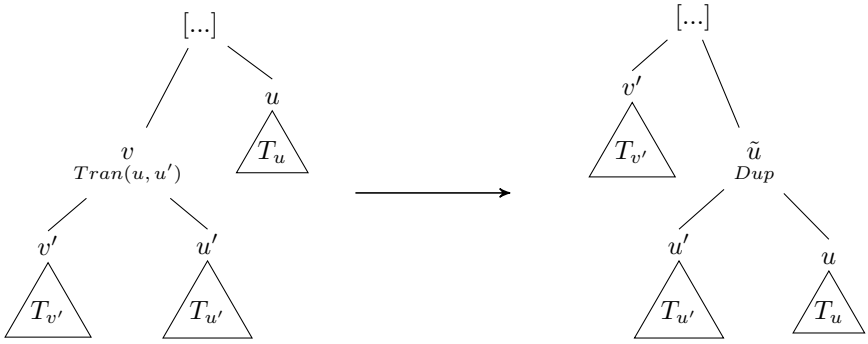

Fig. 1. Procedure $f_{(u,u')}$ for an internal node v of label $Tran(u, u')$

use an union-find data structure on the nodes of $\tilde{T}$. Then in step 6. when v is replaced with v', we add v' to the equivalence class of v in the union-find instead of renaming the $\mathcal{O}(|\Theta|)$ transfers where v appears. As a result from using an union-find, accessing and merging nodes take an additional $\mathcal{O}(\alpha(|\Theta|))$ factor and steps 1. to 5. take $\mathcal{O}(\alpha(|\Theta|))$ operations instead of $\mathcal{O}(1)$. However the whole procedure still takes $\mathcal{O}(\alpha(|\Theta|))$ operations as we wanted. □

Remark 2. Let $\mathcal{R}(\mathcal{G}, S)$ be a super-reconciliation. If $\mathcal{R}' \stackrel{\text{def}}{=} \tau(l_\Theta)(\mathcal{R})$ for some l_Θ list of the transfers in Θ, it must be noted that $\mathcal{R}'$ is not compatible with $\mathcal{G}$ in most cases. Rather $\mathcal{R}'$ is compatible with a set of trees $\mathcal{G}'$, which is obtained from $\mathcal{G}$ by "applying" $\tau(l_\Theta)$ to each tree in $\mathcal{G}$ - this is presented more formally in the appendix. Because we only want to determine the cost of the transformed super-reconciliations, there is no need to construct $\mathcal{G}'$ and to give more details about it in this work.

Note that the resulting supertree may be different depending on the order of the transfers in l_Θ. The non-commutative procedures are notably those with the same source node: then the order of the appended subtrees along the branch would be impacted. However no matter the order of the procedures, we will show that the resulting supertree will have the desired properties.

We define $N(\tilde{T}, \Theta)$ the set of the partially labeled supertrees which can obtained by applying a transform from $\tilde{T}$ with the starting transfer set Θ.

Dealing with pre-labeled supertrees

Let $\tilde{T}_{DL} \in N(\tilde{T}, \Theta)$. Let Δ be the set of nodes that were labeled *Dup* during the transform that led to $\tilde{T}_{DL}$ from $\tilde{T}$. Let $\mathcal{G}_{\mathcal{DL}}$ be the set of gene trees obtained from $\mathcal{G}$ with the same transform that led to $\tilde{T}_{DL}$ from $\tilde{T}$.

We define a variant of (SPFS) which includes a set Δ of pre-labeled internal nodes as *Dup*.

Δ - SMALL-PHYLOGENY FOR SYNTENIES problem (Δ-SPFS):
Input: A set Σ of species and a species tree S for Σ; a set of gene families $\mathcal{F} = \{\Gamma_1, \ldots, \Gamma_t\}$ organized into a set of syntenies $\mathcal{X}$; a set of gene trees $\mathcal{G} = \{T_1, \ldots, T_t\}$, one for each family of $\mathcal{F}$; a supertree $\tilde{T}$ with leaves labeled with the syntenies in $\mathcal{X}$ and consistent with $\mathcal{G}$; a set Δ of internal nodes of $\tilde{T}$.
Output: A Super-Reconciliation $R(\mathcal{G}, S)$ of minimum DL cost which is an extension of $\tilde{T}$ and with the nodes corresponding to those in Δ labeled as *Dup*.

As a slight variant of (SPFS), Lemma 3 (LCA mapping of species) and Lemma 4 (optimal node labeling) do not apply directly. However it turns out that the addition of a set Δ of pre-labeled nodes does not invalidate the proofs of these lemmas (proofs of lemmas 3 and 4 given in the Appendix). Hence the optimal mapping of species is still the LCA one, and the optimal labeling of the non pre-labeled internal nodes is determined in the same way.

Lemma 5. *Let $R(\mathcal{G}, S)$ be a solution of (Δ-SPFS). Let $v \in V(\tilde{T})$ and let v' be the node corresponding to v in $R(\mathcal{G}, S)$. Moreover let v_l and v_r be the children of v. Then:*

1. *$s(v') = lca_S(s(\mathcal{L}(\tilde{T}[v])))$.*
2. *For all internal nodes v not in Δ: if $s(v_l)$ and $s(v_r)$ are separated in S, then v' is a speciation, and otherwise v' is a duplication.*

Then as with (SPFS) only the losses would remain to be optimized, which would be handled by the algorithm described in Theorem 2. This would effectively solve (Δ-SPFS) with **the same time complexity as (SPFS)** (see Corollary 1). If $\widehat{\tilde{T}_{DL}}$ is the supertree $\tilde{T}_{DL}$ without its pre-labeling, $(\Sigma, S, \mathcal{F}, \mathcal{X}, \mathcal{G}_{DL}, \widehat{\tilde{T}_{DL}}, \Delta)$ is a valid input for (Δ-SPFS). Hence all that remains to be proved is a connection between the cost of the DTL super-reconciliations for $\mathcal{G}$ based on $\tilde{T}$ and the cost of the DL super-reconciliations for $\mathcal{G}_{DL}$ based on the partially labeled supertree $\tilde{T}_{DL}$.

Let $Rec_{\tilde{T},\Theta}$ be the set of valid DTL super-reconciliations $\mathcal{R}$ for $\mathcal{G}$ based on $\tilde{T}$ such that the transfers are exactly those indicated in Θ. Given a set $\mathcal{G}'$of gene trees and a partially labeled supertree $\tilde{T}'$ consistent with $\mathcal{G}'$, let $Rec_{\tilde{T}',\mathcal{G}'}$ be the set of DTL valid super-reconciliations $\mathcal{R}$ for $\mathcal{G}'$ based on $\tilde{T}'$ which abide by the already labeled nodes and *do not add any extra horizontal transfer*. (For the sake of simplicity, we will define the special case $Rec_{\tilde{T},\mathcal{G}} \stackrel{\text{def}}{=} Rec_{\tilde{T},\Theta}$).

Lemma 6. *There exists $h : Rec_{\tilde{T},\Theta} \rightarrow Rec_{\tilde{T}_{DL},\mathcal{G}_{DL}}$ bijection such that:*

$$\forall \mathcal{R} \in Rec_{\tilde{T},\Theta}, cost_{DTL}(\mathcal{R}) = cost_{DL}(h(\mathcal{R}))$$

Proof. Let $\tilde{T} = \tilde{T}^{(0)}, \tilde{T}^{(1)}, \ldots, \tilde{T}^{(|\Theta|)} = \tilde{T}_{DL}$ be the consecutive partially labeled trees which were obtained during the transform that led to $\tilde{T}_{DL}$ from $\tilde{T}$. Similarly, let $\mathcal{G} = \mathcal{G}^{(0)}, \mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(|\Theta|)} = \mathcal{G}_{DL}$ be the corresponding sets of gene trees.

We prove that: $\forall 0 \leq i \leq |\Theta|-1$, there exists $h^{(i)} : Rec_{\tilde{T}^{(i)},\mathcal{G}^{(i)}} \rightarrow Rec_{\tilde{T}^{(i+1)},\mathcal{G}^{(i+1)}}$ bijection such that:

$$\forall \mathcal{R} \in Rec_{\tilde{T}^{(i)},\mathcal{G}^{(i)}}, cost_{DTL}(\mathcal{R}) = cost_{DTL}(h^{(i)}(\mathcal{R}))$$

Let $i \in [0..|\Theta| - 1]$, let $\mathcal{R} \in Rec_{\tilde{T}^{(i)},\mathcal{G}^{(i)}}$. Let (u_i, u'_i) be the transfer involved at the $(i+1)^{th}$ iteration in the transform. Then $\tilde{T}^{(i+1)}$ was obtained from $\tilde{T}^{(i)}$ with procedure $f_{(u_i,u'_i)}$. We propose $h^{(i)} \stackrel{\text{def}}{=} f_{(u_i,u'_i)}$ where the same procedure is applied, but on the super-reconciliations instead.

- Besides u_i, v_i and $\tilde{u}_i$, nothing is changed in the super-reconciliation. Plus the event $Tran(u_i)$ on v in $\tilde{T}^{(i)}$ and the event Dup on $\tilde{u}$ in $\tilde{T}^{(i+1)}$ infer the same constraints on the species and the syntenies: $s(X_{u_i})$ $(= s(X_{\tilde{u}_i})) = s(X_{u'_i})$, and $X_{u'_i}$ has to be a subsequence of X_{u_i} $(= X_{\tilde{u}_i})$. Hence $h^{(i)}(\mathcal{R}) \in Rec_{\tilde{T}^{(i+1)}}$.
- As a Dup label and a $Tran(\ldots)$ label have the same cost, we also infer that $cost_{DTL}(\mathcal{R}) = cost_{DTL}(h^{(i)}(\mathcal{R}))$.
- It remains to prove that $h^{(i)}$ is invertible, which comes from the fact procedure $f_{(u_i,u'_i)}$ is invertible.

Then with $h \stackrel{\text{def}}{=} h^{(|\Theta-1|)} \circ \ldots \circ h^{(0)}$, we have h a bijection such for any super-reconciliation $\mathcal{R}$ in $Rec_{\tilde{T},\Theta}$: $cost_{DTL}(\mathcal{R}) = cost_{DTL}(h(\mathcal{R}))$. And because there is no transfer label in any super-reconciliation in $Rec_{\tilde{T}_{DL},\mathcal{G}_{DL}}$, we have: $cost_{DTL}(h(\mathcal{R})) = cost_{DL}(h(\mathcal{R}))$.

□

Hence by Lemma 6 the minimal cost of a DTL super-reconciliation on $\tilde{T}$, given an enforced set Θ of horizontal transfers, can be computed in exponential time by computing a pre-labeled supertree $\tilde{T}_{DL}$ with the transform and solving (Δ-SPFS) on it.

The minimum cost algorithm

In order to solve DTL Super-Reconciliation, we propose the following method:

1. Loop on all valid mappings Θ of the supertree $\tilde{T}$
2. Use the transform and compute the minimum cost relative to the set Θ in the way described in the previous subsection
3. Take the minimum cost found

Looping on all possible transfer sets Θ adds an additional exponential factor to an already exponential procedure, which still results in an exponential algorithm in the end.

Corollary 2. *Given $\tilde{T}$ a supertree compatible with the set of gene trees $\mathcal{G}$, the minimum DTL cost of a super-reconciliation which is an extension of $\tilde{T}$ can be computed in time $\mathcal{O}(n^2\alpha(n) \cdot t \cdot 2^{n \log n + t \log t + 2t})$, where t is the number of gene families present in $\mathcal{G}$ and n is the number of syntenies.*

Proof. As explained in the previous subsection, by Lemma 5 (Δ-SPFS) can be solved with the same time complexity as (SPFS), which is $\mathcal{O}(t2^{t \log t + 2t} n)$ by Corollary 1. Plus by Proposition 2 every transform takes $\mathcal{O}(|\Theta| \cdot \alpha(|\Theta|))$ operations. Because each node of the supertree can be the destination of at most one horizontal transfer, we have: $|\Theta| = \mathcal{O}(n)$. And since the valid transfer sets Θ can be as many as $\mathcal{O}(2^{n \log n})$, we get the complexity proposed in the corollary. □

5 Discussion

We proposed a way to extend the DL Super-Reconciliation to horizontal transfers. This came as a natural extension of the label-based definition used in [5]. We proposed a cost-preserving transform which removed the transfer relations and thus allowed us to reuse the dynamic programming approach developed in [5], at the cost of dealing with partially-labeled super-reconciliations and supertrees. This led to an exponential-time algorithm effectively computing the minimum DTL cost that can be obtained out of a supertree $\tilde{T}$ compatible with the input data. While we proved that *time-consistent* Super-Reconciliation is NP-hard, the time complexity of the general case remains an open problem.

Further work on the current method would involve step 1. of the algorithm, i.e. not having to loop on all possible horizontal transfer sets in order to find the minimum DTL cost. This is the main roadblock to finding an FPT algorithm. A lot of the work done by the algorithm seems to be redundant, and as such it is likely to find more efficient ways to go through the set of possible cost values for a super-reconciliation. While an FPT algorithm may never be found for the general problem, there might be breakthroughs to be found with sub-problems such as requiring time consistency or entries with dated input trees. Such variants would have a significant impact on the set of valid horizontal transfer sets. Plus in the light of the complexity results with DTL Reconciliation, where *time-consistent* DTL Reconciliation is NP-hard but then becomes polynomial when time consistency is dropped, such sub-problems have a chance to change the complexity of the problem.

References

1. Aho, A.V., Sagiv, Y., Szymanski, T.G., Ullman, J.D.: Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM Journal on Computing **10**(3), 405–421 (1981)
2. Bansal, M.S., Alm, E.J., Kellis, M.: Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. Bioinformatics **28**(12), i283–i291 (2012)
3. Bérard, S., Gallien, C., Boussau, B., Szollosi, G., Daubin, V., Tannier, E.: Evolution of gene neighborhoods within reconciled phylogenies. Bioinformatics **28**(18), i382-i388 (2012)
4. Constantinescu, M., Sankoff, D.: An efficient algorithm for supertrees. Journal of Classification **12**(1), 101–112 (1995)
5. Delabre, M., El-Mabrouk, N., Huber, K., Lafond, M., Moulton, V., Noutahi, E., Sautie Castellanos, M.: Evolution through segmental duplications and losses: A super-reconciliation approach. Algorithms for Molecular Biology (2019)
6. Dondi, R., Lafond, M., Scornavacca, C.: Reconciling multiple genes trees via segmental duplications and losses. Algorithms for Molecular Biology (14) (2019)
7. Doyon, J., *et al.*, C.S.: An efficient algo. for gene/species trees parsimonious reconciliation with losses, duplications and transfers. pp. 93-108. RECOMB-CG (2010)
8. Doyon, J., Ranwez, V., Daubin, V., Berry, V.: Models, algorithms and programs for phylogeny reconciliation. Briefings in bioinformatics **12**(5), 392-400 (2011)
9. Duchemin, W.: Phylogeny of dependencies and dependencies of phylogenies in genes and genomes. Theses, Université de Lyon (Dec 2017), https://tel.archives-ouvertes.fr/tel-01779517
10. *et al.*, W.D.: DeCoSTAR: Reconstructing the ancestral organization of genes or genomes using reconciled phylogenies. Genome Biol. Evol. (2017)
11. Goodman, M., Czelusniak, J., Moore, G.W., Romero-Herrera, A.E., Matsuda, G.: Fitting the Gene Lineage into its Species Lineage, a Parsimony Strategy Illustrated by Cladograms Constructed from Globin Sequences. Systematic Biology **28**(2), 132–163 (06 1979). https://doi.org/10.1093/sysbio/28.2.132, https://doi.org/10.1093/sysbio/28.2.132
12. Hallett, M., Lagergren, J.: Efficient algorithms for lateral gene transfer problems. pp. 149-156. RECOMB-CG (2001)
13. Mirkin, B., Muchnik, I., Smith, T.F.: A biologically meaningful model for comparing molecular phylogenies. Tech. rep. (1995)
14. Ng, M.P., Wormald, N.C.: Reconstruction of rooted trees from subtrees. Discrete applied mathematics **69**(1-2), 19–31 (1996)
15. Ovadia, Y., Fielder, D., Conow, C., Libeskind-Hadas, R.: The cophylogeny reconstruction problem is NP-complete. J. Comput. Biol. **18**(1), 59-65 (2011). https://doi.org/10.1089/cmb.2009.0240
16. Paszek, J., Gorecki, P.: Efficient algorithms for genomic duplication models. IEEE/ACM Trans Comput Biol Bioinform. (2017)
17. Semple, C.: Reconstructing minimal rooted trees. Discrete Applied Mathematics **127**(3), 489–503 (2003)
18. Tofigh, A., Hallett, M., Lagergren, J.: Simultaneous identification of duplications and lateral gene transfers. IEEE/ACM Trans.Comput.BiolBioinform. **8**(2), 517-535 (2011). https://doi.org/10.1109/TCBB.2010.14
19. Zhang, L.: On a mirkin-muchnik-smith conjecture for comparing molecular phylogenies. J.Comput.Biol. **4**(2), 177–187 (1997)

20. Zhang, L.: On a mirkin-muchnik-smith conjecture for comparing molecular phylogenies. Journal of Computational Biology **4**, 177–187 (1997)
21. Zmasek, C.M., Eddy, S.R.: A simple algorithm to infer gene duplication and speciation events on a gene tree. Bioinformatics **17**, 821– 828 (2001)

Appendix

Proof of Lemma 3 (from [5]):

Proof. First, observe that the statement is clearly true for the leaves. Assume that the statement is false. Now, let v be a node of $\tilde{T}$ such that its corresponding node v' does not satisfy the statement - moreover, choose v to be a minimal node with this property (meaning that for the children v_l and v_r of v, the corresponding nodes v'_l and v'_r in $R(\mathcal{G}, S)$ satisfy $s(v'_l) = lca_S(s(\mathcal{L}(\tilde{T}[v_l]))$ and $s(v'_r) = lca_S(s(\mathcal{L}(\tilde{T}[v_r])))$. Note that v must exist, since the statement is true for the leaves.

Now, we may assume that $s(v') \neq lca_S(s(v'_l), s(v'_r))$, as otherwise v' satisfies the lemma. Thus in S, there are at least k edges on the path from $s(v')$ to $lca_S(s(v'_l), s(v'_r))$, where here $k > 0$. It is not hard to verify that in this case, v' must be a duplication node, according to the definition of a reconciliation. This implies that there are at least k full losses on the path from v' to v'_l and at least k full losses on the path from v' to v'_r. Consider the Super-Reconciliation R' that is identical to $R(\mathcal{G}, S)$, with the exception that $s(v') = lca_S(s(v'_l), s(v'_r))$. Then the $2k$ losses on the paths between v' and v'_l and between v' and v'_r are not needed anymore, although if v' is not the root, k losses become necessary on the path between v' and w', where w' is the node corresponding to the parent w of v in $\tilde{T}$. Remapping v' cannot increase the number of duplications, and so we have saved k losses.

It remains to argue that the number of partial losses remains the same. But this is easy to see. We keep the same synteny assignment at nodes v', v'_l and v'_r (and w' if v' is not the root) as in $R(\mathcal{G}, S)$. If v' was a segmental duplication in $R(\mathcal{G}, S)$, we set v' to be a segmental duplication in R' as well. The number of partial losses on the paths between v' and v'_l, v'_r (and w') therefore remains the same as in $R(\mathcal{G}, S)$. □

Proof of Lemma 4 (from [5]):

Proof. Let v'_l and v'_r be the nodes corresponding to v_l and v_r, respectively, in $R(\mathcal{G}, S)$. First, if $s(v_l)$ and $s(v_r)$ are not separated, then by Lemma 3, $s(v'_l)$ and $s(v'_r)$ are not separated, hence it is not possible for v' to be a speciation. Therefore v' must be a duplication.

Suppose instead that $s(v_l)$ and $s(v_r)$ are separated in S, but that v' is labeled by a duplication event $Dup(X, [i, j])$, where X is the synteny assigned at v'. On the path from v' to v'_l, there may be some *pLoss* events and some nodes that were grafted owing to full losses. We may assume that all full loss events, if any, have occurred before the *pLoss* events on this path (i.e., nodes grafted from full losses are closer to v'). This is without loss of generality, as this does not change the resulting synteny in v'_l. We shall make the same assumption with the path from v' to v'_r. Now, by Lemma 3, $s(v') = lca_S(s(v_l), s(v_r))$. Because v' is a duplication, the two children w_l, w_r of v' in $R(\mathcal{G}, S)$ must satisfy $s(w_l) = s(w_r) = s(v')$.

Since $s(v'_l) \neq s(v') \neq s(v'_r)$, we have that $\{w_l, w_r\} \cap \{v'_l, v'_r\} = \emptyset$, and therefore w_l and w_r were grafted on $\tilde{T}$ due to full losses. If we label v' as a speciation $Spe(X, [1, |X|])$, these two full losses are not needed anymore, and by doing so we have one duplication less and two full losses less. Let Y_l and Y_r be the two syntenies that were assigned at w_l and w_r in $R(\mathcal{G}, S)$, respectively. Then $Y_l = X$ and $Y_r = X[i, j]$ or vice-versa (assume the former, without loss of generality). Suppose that w_r was an ancestor of v'_r in $R(\mathcal{G}, S)$, again without loss of generality. The substring $X[i, j]$ can be obtained from X by adding at most two partial losses on the path from v' to v'_r. The rest of the reconciliation can remain the same. To sum up, we have removed one duplication and two full losses, and inserted at most two partial losses to reproduce the effect of the segmental duplication. This contradicts that $R(\mathcal{G}, S)$ is a reconciliation of minimum cost. □

Applying a transform $\tau(l_\Theta)$ to a gene tree g in $\mathcal{G}$:

Let $l_\Theta = [(u_1, u'_1), \ldots, (u_{|\Theta|}, u'_{|\Theta|})]$. The goal is to keep the tree consistency after every procedure $f_{(u_i, u'_i)}$ of the transform for $1 \leq i \leq n$. Let $\tilde{T} = \tilde{T}^{(0)}, \tilde{T}^{(1)}, \ldots, \tilde{T}^{(|\Theta|)}$ be the consecutive partially labeled trees which were obtained during the transform.

Let $1 \leq i \leq n$. We consider the procedure $f_{(u_i, u'_i)}$ on $\tilde{T}^{(i-1)}$. Let g_{i-1} be tree g after step $i-1$. Then the procedure applied to g_{i-1} is the following:

- If there is no synteny of g in $T_{u'_i}$ the subtree of u'_i in $\tilde{T}^{(i-1)}$:
  - do nothing
- else :
  - $u'_{i,g} \stackrel{\text{def}}{=} lca_{g_{i-1}}(\{l \in \mathcal{L}(g_{i-1}) | l \text{ in the subtree } T_{u'_i}\})$
  - $u_{i,g} \stackrel{\text{def}}{=} lca_{g_{i-1}}(\{l \in \mathcal{L}(g_{i-1}) | lca_{\tilde{T}^{(i-1)}}(u, l) = min_{l' \in \mathcal{L}(g_{i-1})} lca_{\tilde{T}^{(i-1)}}(u, l')\})$
  - apply $f_{(u_{i,g}, u'_{i,g})}$ to g_{i-1}

Inverse procedure $f^{-1}_{(v,u')}$:

1. create a node $\tilde{v}$ as the new parent of v
2. remove the edge between u and u' and have $\tilde{v}$ as the new parent of u'
3. put the label $Tran(u, u')$ on $\tilde{v}$
4. remove u
5. remove (v, u') in the set of transfers
6. if u was a source node in some other transfers: replace u with $\tilde{u}$ in these transfers

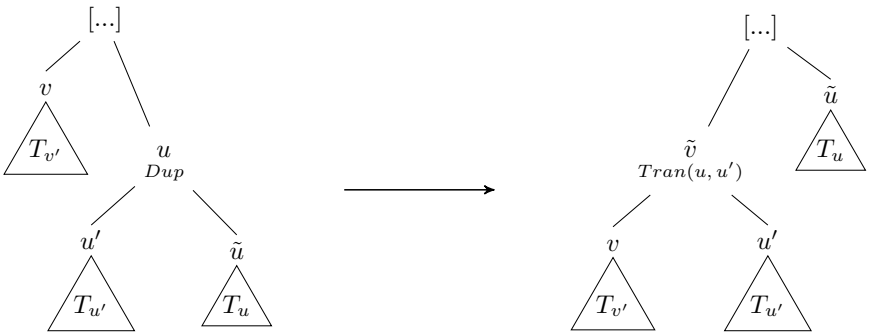

Fig. 2. Procedure $f^{-1}_{(v,u')}$ for an internal node u of label Dup