Inferring Microbial Gene Family Evolution Using Duplication-Transfer-Loss Reconciliation: Algorithms and Complexity

Misagh Kordi
University of Connecticut - Storrs, misagh.kordi@uconn.edu

Follow this and additional works at: https://opencommons.uconn.edu/dissertations

Recommended Citation
https://opencommons.uconn.edu/dissertations/2101
Gene families evolve through complex evolutionary events such as speciation, gene duplication, horizontal gene transfer, gene loss, etc., and reconstructing these evolutionary histories is an important problem in evolutionary biology with many important applications. Duplication-Transfer-Loss (DTL) reconciliation is among the most effective and most popular methods for studying gene family evolution, especially in microbes. DTL reconciliation takes as input a gene tree and a species tree and reconciles the two by postulating gene duplication, transfer, and loss events, showing the evolution of that gene family inside the species tree. The DTL reconciliation problem has been extensively studied, but existing problem formulations and algorithms have several limitations that affect the accuracy and applicability of DTL reconciliation in practice.

In this thesis, we focus on addressing two of the most important limitations. The first limitation is that existing algorithms assume a fixed, binary gene tree topology and therefore cannot account for uncertainty in gene tree topologies, a common occurrence in practice. The second limitation is that all transfer events are assumed to be “additive”, i.e., they introduce a new gene into the recipient genome. It is well known, however, that transfer events can also be
“replacing”, i.e., they can replace an existing gene in the recipient genome. To address the first limitation, we devise an extension of DTL reconciliation to non-binary gene trees and show that the resulting problem is NP-hard. We then provide fixed parameter and other exact and heuristic algorithms for this problem, and demonstrate their impact in practice on real and simulated data. For the second limitation, we propose and develop a new extended reconciliation framework, called the DTRL reconciliation framework, which models both additive and replacing transfers, and show that the resulting computational problem is NP-hard.
Inferring Microbial Gene Family Evolution Using
Duplication-Transfer-Loss Reconciliation:
Algorithms and Complexity

Misagh Kordi
M.Sc.Eng., University of Tehran, Tehran, Iran, 2013

A Dissertation
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy
at the
University of Connecticut
2019
APPROVAL PAGE

Doctor of Philosophy Dissertation

Inferring Microbial Gene Family Evolution Using
Duplication-Transfer-Loss Reconciliation:
Algorithms and Complexity

Presented by
Misagh Kordi, M.S.

Major Advisor
__________________________________________
Dr. Mukul Bansal

Associate Advisor
__________________________________________
Dr. Ion Mandoiu

Associate Advisor
__________________________________________
Dr. Sheida Nabavi

Associate Advisor
__________________________________________
Dr. Yufeng Wu

University of Connecticut

2019
Firstly, I would like to express my sincere gratitude to my advisor Prof. Mukul Bansal for the continuous support of my Ph.D study and related research, for his scholarly advice, valuable insights, clarity of thought, patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Wu, Prof. Mandoiu, and Prof. Nabavi, for their insightful comments and encouragement.

Last but not the least, I would like to thank my family: my parents, my brother and my sister, my uncle, and my best friend, Yanis Papavisiliousis, for supporting me spiritually throughout writing this thesis and my life in general.
# TABLE OF CONTENTS

## Chapter 1: Introduction

1.1 Our Contribution .................................................. 3

## Chapter 2: On the Complexity of Duplication-Transfer-Loss Reconciliation with Non-Binary Gene Trees

2.1 Introduction ......................................................... 6

2.2 Definitions and Preliminaries .................................. 9

2.2.1 Reconciliation and DTL-scenarios ......................... 10

2.2.2 Optimal gene tree resolution ............................... 12

2.3 NP-hardness of the OGTR problem ............................ 13

2.3.1 Reduction from minimum 3-set cover .................... 13

2.3.2 Gadget .......................................................... 14

2.3.3 Key insight .................................................... 17

2.3.4 Proof of Claim 2: forward direction .................... 18

2.3.5 Proof of Claim 2: reverse direction .................... 25

2.4 Extension to dated DTL reconciliation ....................... 38

2.5 Conclusion ......................................................... 39

## Chapter 3: Exact Algorithms for Duplication-Transfer-Loss Reconciliation with Non-Binary Gene Trees

3.1 Introduction ......................................................... 43

3.2 Definitions and Preliminaries ................................. 48
3.2.1 Reconciliation and DTL-scenarios ........................................ 49
3.2.2 Optimal gene tree resolution ............................................ 51
3.3 Fixed Parameter Algorithm for OGTR .................................... 53
3.4 Enumeration Algorithm for OGTR-All .................................... 59
3.5 Extension to dated DTL Reconciliation ................................ 71
3.6 Experimental Evaluation .................................................. 72
  3.6.1 Description of the data set ............................................. 72
  3.6.2 Scalability and runtime of the algorithms ......................... 74
  3.6.3 Experimental results .................................................. 75
  3.6.4 Impact on inference of gene family evolution .................... 76
3.7 Conclusion ................................................................. 80

Chapter 4: On Inferring Additive and Replacing Horizontal Gene Transfers Through Phylogenetic Reconciliation 82
4.1 Introduction ............................................................... 82
4.2 Definitions and preliminaries ............................................. 85
  4.2.1 Additive and replacing transfers ..................................... 86
  4.2.2 DTRL Reconciliation .................................................. 88
4.3 NP-hardness of ODTRL ................................................... 94
  4.3.1 Reduction from minimum rSPR distance .......................... 94
  4.3.2 Gadget ................................................................. 96
  4.3.3 Proof of Claim 2: Forward direction ............................... 98
  4.3.4 Proof of Claim 2: Reverse direction ............................... 110
Chapter 1

Introduction

Duplication-Transfer-Loss (DTL) reconciliation is a powerful, well-known technique for studying gene family evolution in microbial species. Microbial gene families evolve primarily through gene duplication, gene loss, and horizontal gene transfer, and DTL reconciliation can infer these evolutionary events through the systematic comparison and reconciliation of gene trees and species trees. Gene trees represent the evolutionary histories of gene families, while species trees represent the evolutionary histories of the corresponding species. For a given gene tree and a species tree, DTL reconciliation shows the evolution of the gene tree inside the species tree, and explicitly infers duplication, transfer, and loss events. Accurate inference of these evolutionary events has many uses in biology, including inference of orthologs, paralogs and xenologs [39,68], reconstruction of ancestral gene content [12,17], and accurate gene tree and species tree construction [5,9,21,56,68]. The DTL reconciliation problem has therefore been widely studied, e.g., [2,3,14,17–19,25,42,49,51,59,63,67].

DTL reconciliation is generally formulated as a parsimony problem where each evolutionary event is assigned a cost and the goal is to find a reconciliation with minimum total
The resulting optimization problem is called the DTL-reconciliation problem. DTL-reconciliations can sometimes be time-inconsistent in the sense that the inferred transfers may induce contradictory constraints on the dates for the internal nodes of the species tree. The problem of finding an optimal time-consistent reconciliation is known to be NP-hard [54, 67]. Thus, in practice, the goal is often to find an optimal (not necessarily time-consistent) DTL-reconciliation [2, 3, 17, 49, 67] and this problem can be solved in $O(mn)$ time [2], where $m$ and $n$ denote the number of nodes in the gene tree and species tree, respectively. Interestingly, the problem of finding an optimal time-consistent reconciliation becomes efficiently solvable [19, 48] in $O(mn^2)$ time if the species tree is fully dated. Thus, the two efficiently solvable formulations, dated and undated, are the two standard formulations of the DTL-reconciliation problem.

There are two important limitations of all existing work on DTL reconciliation. First, both formulations of the DTL-reconciliation problem assume that the input gene tree and species tree are binary. However, gene trees are frequently non-binary. This happens whenever there is insufficient information in the underlying gene sequences to fully resolve gene tree topologies. In such cases, all poorly supported edges in the reconstructed gene trees are collapsed, resulting in non-binary gene trees. Since gene family sequence alignments are often short and have limited information content, non-binary gene trees arise very frequently in practice. When the input consists of a non-binary gene tree, the reconciliation problem seeks a binary resolution of the gene tree that minimizes the reconciliation cost. Many efficient algorithms have been developed for reconciling non-binary gene trees in the context of the simpler Duplication-Loss (DL) reconciliation model [10, 21, 47, 70], with the most efficient of these algorithms having an optimal $O(m + n)$ time complexity [70]. However, the DTL reconciliation model is more
general and significantly more complex than the DL reconciliation model. Consequently, no efficient algorithms exist for DTL reconciliation with non-binary gene trees and the complexity of the problem remains unknown. As a result, DTL reconciliation has remained inapplicable to non-binary gene trees, significantly reducing its utility in practice.

Second, all previous phylogenetic methods for inferring transfer events can be divided into two classes: (i) Those that implicitly assume that all transfers are replacing transfers and that all discordance between gene trees and species trees is due to these replacing transfer events, e.g., [1, 6, 7, 27, 31, 35, 52, 66], and (ii) those based on the Duplication-Transfer-Loss (DTL) reconciliation framework, which model gene duplication and gene loss as additional sources of gene tree/species tree discordance, but implicitly assume that all transfers are additive transfers, e.g., [2, 13, 17, 19, 32, 44, 51, 60, 63–65, 67]. Thus, no existing phylogenetic method models both additive and replacing transfers. And while methods based on DTL reconciliation represent a major advance in the ability to accurately detect transfer events, they are limited by their inability to properly handle replacing transfers.

1.1 Our Contribution

In this thesis we provide new problem formulations for DTL reconciliation that address these limitations, analyze the computational complexities of the new formulations, and provide new algorithms for solving them. Specifically our contributions are as follows:

In chapter 2, we settle the open problem by proving that the DTL-reconciliation problem on non-binary gene trees is, in fact, NP-hard. Our proof is based on a reduction from the minimum 3-set cover problem and applies to both formulations of the DTL-reconciliation problem. An especially desirable feature of our reduction is that it implies NP-hardness for biologically
relevant settings of the event cost parameters, showing that the problem is difficult even for biologically meaningful scenarios. This work was published in [40].

In chapter 3, we present the first, exact algorithms for DTL reconciliation with non-binary gene trees. We show that the DTL-reconciliation problem for non-binary gene trees is fixed-parameter tractable (FPT) in the maximum degree of the gene tree. Crucially, our algorithms also make it possible to distinguish between those aspects of the reconciliation that are highly supported based on all optimal (i.e., minimum cost) resolutions of the gene tree from those that are not. This makes it possible to not only apply DTL-reconciliation to non-binary gene trees, but to also negate the impact of gene tree uncertainty by distinguishing evolutionary inferences that have high support across all optimal resolutions of the given non-binary gene tree from those evolutionary inferences that have low support across the optimal resolutions. Even though our algorithms have exponential time complexity in the worst case, we show that they can be applied efficiently in most cases and can be used to analyze even large gene trees and species trees. This work was published in [44].

In chapter 4, we define and formalize the first phylogenetic reconciliation framework that simultaneously models both additive and replacing transfer events. Our framework builds upon the standard parsimony-based DTL reconciliation model, by explicitly modeling replacing transfer events. Specifically, we formally define the Duplication–Additive-Transfer–Replacing-Transfer–Loss (DTRL) reconciliation model that explicitly models both additive and replacing transfer events, along with gene duplications and losses. As with the underlying DTL reconciliation model, we formulate the DTRL reconciliation problem as one of finding a most parsimonious DTRL reconciliation, i.e., one with smallest total “reconciliation cost”. We prove that the problem of computing a most parsimonious DTRL reconciliation is NP-hard,
using a reduction from the NP-hard minimum rooted Subtree Prune and Regraft (rSPR) distance problem, and in the sections 4.4 co-author of this work perform the very first experiments to study the impact of replacing transfer events on the accuracy of DTL reconciliation itself. Surprisingly, we found that DTL reconciliation is highly robust to the presence of replacing transfer.

In chapter 5, we propose an alternative formulation for the problem of DTL reconciliation. The idea is to consider only those optimal binary resolutions that have some support in the sequence data. More precisely, in addition to the non-binary gene tree and species tree, we take as input a set of bootstrap replicates or samples from the posterior of a Bayesian analysis for the gene tree, and then consider only those binary resolutions of the gene tree whose clades appear in at least one of the given gene tree samples/bootstrap replicates. We applied our algorithm to both empirical and simulated datasets in a thorough experimental study and demonstrate that: results obtained using constrained DTL reconciliation problem with non-binary gene trees are more accurate than results obtained using the traditional (unconstrained) formulation of the problem, the algorithm is highly scalable and efficient, and our approach for identifying a single, most highly supported binary resolution is highly effective at gene tree error correction.
Chapter 2

On the Complexity of Duplication-Transfer-Loss Reconciliation
with Non-Binary Gene Trees

2.1 Introduction

Duplication-Transfer-Loss (DTL) reconciliation is one of the most powerful techniques for studying gene and genome evolution in microbes and other non-microbial species engaged in horizontal gene transfer. DTL reconciliation accounts for the role of gene duplication, gene loss, and horizontal gene transfer in shaping gene families and can infer these evolutionary events through the systematic comparison and reconciliation of gene trees and species trees. Gene trees represent the evolutionary histories of gene families, while species trees represent the evolutionary histories of the corresponding species. Given a gene tree and a species tree, DTL reconciliation shows the evolution of the gene tree inside the species tree, and explicitly infers duplication, transfer, and loss events. Accurate knowledge of gene family evolution has many uses in biology, including inference of orthologs, paralogs and xenologs for functional genomic studies, e.g., [39, 68], reconstruction of ancestral gene content, e.g., [12, 17], and
accurate gene tree and species tree construction, e.g., [5, 9, 21, 58, 68], as well as potential application to error-correcting taxonomic assignments of metagenomic reads. Consequently, the DTL reconciliation problem has been widely studied, e.g., [2, 3, 17, 19, 25, 49, 59, 63, 67].

DTL reconciliation is typically formulated using a parsimony framework where each evolutionary event is assigned a cost and the goal is to find a reconciliation with minimum total cost. The resulting optimization problem is called the DTL-reconciliation problem. DTL-reconciliations can sometimes be time-inconsistent; i.e., the inferred transfers may induce contradictory constraints on the dates for the internal nodes of the species tree. The problem of finding an optimal time-consistent reconciliation is known to be NP-hard [54, 67]. Thus, in practice, the goal is to find an optimal (not necessarily time-consistent) DTL-reconciliation [2, 3, 17, 49, 67] and this problem can be solved in $O(mn)$ time [2], where $m$ and $n$ denote the number of nodes in the gene tree and species tree, respectively. Interestingly, the problem of finding an optimal time-consistent reconciliation actually becomes efficiently solvable [19, 48] in $O(mn^2)$ time if the species tree is fully dated. Thus, these two efficiently solvable formulations, regular and dated, are the two standard formulations of the DTL-reconciliation problem.

Both these formulations of the DTL-reconciliation problem assume that the input gene tree and species tree are binary. However, while relatively accurate species trees can be obtained through the use of well-behaved orthologous gene families or multi-gene species tree reconstruction methods [9, 16, 23], gene tree inference is confounded by the fact that there is often insufficient information in the underlying gene sequences to fully resolve gene tree topologies. As a result, gene trees are frequently non-binary in practice. When the input consists of a non-binary gene tree, the reconciliation problem seeks to find a binary resolution of the gene tree that minimizes the reconciliation cost. Given the prevalence of non-binary gene trees,
many efficient algorithms have been developed for this problem in the context of the simpler Duplication-Loss (DL) reconciliation model [10, 21, 47, 70], with the most efficient of these algorithms having an optimal $O(m + n)$ time complexity [70]. However, the DTL reconciliation model is more general and significantly more complex than the DL reconciliation model. Consequently, no efficient algorithms exist for DTL reconciliation with non-binary gene trees and the complexity of the problem remains unknown. As a result, DTL reconciliation is currently inapplicable to non-binary gene trees, significantly reducing its utility in practice.

In this work, we settle this open problem by proving that the DTL-reconciliation problem on non-binary gene trees is, in fact, NP-hard. Our proof is based on a reduction from the minimum 3-set cover problem and applies to both formulations of the DTL-reconciliation problem. An especially desirable feature of our reduction is that it implies NP-hardness for biologically relevant settings of the event cost parameters, showing that the problem is difficult even for biologically meaningful scenarios. By settling this question, our work will spur the development of both exact (better than brute-force) and efficient approximation and heuristic algorithms for this important problem.

We develop our NP-hardness proof in the context of the regular (undated) DTL-reconciliation formulation, and revisit dated DTL-reconciliation later in Section 3.5. The next section introduces basic definitions and preliminaries, and we present the NP-hardness proof for the optimal gene tree resolution problem in Section 4.3. Concluding remarks appear in Section 5.5.
2.2 Definitions and Preliminaries

We follow the basic definitions and notation from [2]. Given a tree \( T \), we denote its node, edge, and leaf sets by \( V(T) \), \( E(T) \), and \( Le(T) \) respectively. If \( T \) is rooted, the root node of \( T \) is denoted by \( rt(T) \), the parent of a node \( v \in V(T) \) by \( pa_T(v) \), its set of children by \( Ch_T(v) \), and the (maximal) subtree of \( T \) rooted at \( v \) by \( T(v) \). The set of internal nodes of \( T \), denoted \( I(T) \), is defined to be \( V(T) \setminus Le(T) \). We define \( \leq_T \) to be the partial order on \( V(T) \) where \( x \leq_T y \) if \( y \) is a node on the path between \( rt(T) \) and \( x \). The partial order \( \geq_T \) is defined analogously, i.e., \( x \geq_T y \) if \( x \) is a node on the path between \( rt(T) \) and \( y \). We say that \( y \) is an ancestor of \( x \), or that \( x \) is a descendant of \( y \), if \( x \leq_T y \) (note that, under this definition, every node is a descendant as well as ancestor of itself). We say that \( x \) and \( y \) are incomparable if neither \( x \leq_T y \) nor \( y \leq_T x \). Given a non-empty subset \( L \subseteq Le(T) \), we denote by \( lca_T(L) \) the last common ancestor (LCA) of all the leaves in \( L \) in tree \( T \); that is, \( lca_T(L) \) is the unique smallest upper bound of \( L \) under \( \leq_T \). Given \( x, y \in V(T) \), \( x \rightarrow_T y \) denotes the unique path from \( x \) to \( y \) in \( T \). We denote by \( d_T(x, y) \) the number of edges on the path \( x \rightarrow_T y \); note that if \( x = y \) then \( d_T(x, y) = 0 \). Throughout this work, the term tree refers to rooted trees. A tree is binary if all of its internal nodes have exactly two children, and non-binary otherwise. We say that a tree \( T' \) is a binary resolution of \( T \) if \( T' \) is binary and \( T \) can be obtained from \( T' \) by contracting one or more edges. We denote by \( BR(T) \) the set of all binary resolutions of a non-binary tree \( T \). Gene trees may be either binary or non-binary while the species tree is always assumed to be binary. Throughout this work, we denote the gene tree and species tree under consideration by \( G \) and \( S \), respectively. If \( G \) is restricted to be binary we refer to it as \( GB \) and as \( GN \) if it is restricted to be non-binary. We assume that each leaf of the gene tree is labeled with the species from
which that gene was sampled. This labeling defines a leaf-mapping \( \mathcal{L}_{G,S} : \text{Le}(G) \rightarrow \text{Le}(S) \) that maps a leaf node \( g \in \text{Le}(G) \) to that unique leaf node \( s \in \text{Le}(S) \) which has the same label as \( g \). Note that gene trees may have more than one gene sampled from the same species. We will implicitly assume that the species tree contains all the species represented in the gene tree.

2.2.1 Reconciliation and DTL-scenarios

A binary gene tree can be reconciled with a species tree by mapping the gene tree into the species tree. Next, we define what constitutes a valid reconciliation; specifically, we define a Duplication-Transfer-Loss scenario (DTL-scenario) \([2, 67]\) for \( G^B \) and \( S \) that characterizes the mappings of \( G^B \) into \( S \) that constitute a biologically valid reconciliation. Essentially, DTL-scenarios map each gene tree node to a unique species tree node in a consistent way that respects the immediate temporal constraints implied by the species tree, and designate each gene tree node as representing either a speciation, duplication, or transfer event. For any gene tree node, say \( g \), that represents a transfer event, DTL-scenarios also specify which of the two edges \((g, g') \) or \((g, g'') \), where \( g', g'' \) denote the children of \( g \), represents the transfer edge on \( S \), and identify the recipient species of the corresponding transfer.

**Definition 2.2.1 (DTL-scenario).** A DTL-scenario for \( G^B \) and \( S \) is a seven-tuple \( \langle \mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau \rangle \), where \( \mathcal{L} : \text{Le}(G^B) \rightarrow \text{Le}(S) \) represents the leaf-mapping from \( G^B \) to \( S \), \( \mathcal{M} : V(G^B) \rightarrow V(S) \) maps each node of \( G^B \) to a node of \( S \), the sets \( \Sigma, \Delta, \) and \( \Theta \) partition \( I(G^B) \) into speciation, duplication, and transfer nodes respectively, \( \Xi \) is a subset of gene tree edges that represent transfer edges, and \( \tau : \Theta \rightarrow V(S) \) specifies the recipient species for each transfer event, subject to the following constraints:
1. If $g \in \text{Le}(G^B)$, then $\mathcal{M}(g) = \mathcal{L}(g)$.

2. If $g \in I(G^B)$ and $g'$ and $g''$ denote the children of $g$, then,

   (a) $\mathcal{M}(g) \not\leq_S \mathcal{M}(g')$ and $\mathcal{M}(g) \not\leq_S \mathcal{M}(g'')$.

   (b) At least one of $\mathcal{M}(g')$ and $\mathcal{M}(g'')$ is a descendant of $\mathcal{M}(g)$.

3. Given any edge $(g, g') \in E(G^B)$, $(g, g') \in \Xi$ if and only if $\mathcal{M}(g)$ and $\mathcal{M}(g')$ are incomparable.

4. If $g \in I(G^B)$ and $g'$ and $g''$ denote the children of $g$, then,

   (a) $g \in \Sigma$ only if $\mathcal{M}(g) = \text{lca}(\mathcal{M}(g'), \mathcal{M}(g''))$ and $\mathcal{M}(g')$ and $\mathcal{M}(g'')$ are incomparable,

   (b) $g \in \Delta$ only if $\mathcal{M}(g) \geq_S \text{lca}(\mathcal{M}(g'), \mathcal{M}(g''))$.

   (c) $g \in \Theta$ if and only if either $(g, g') \in \Xi$ or $(g, g'') \in \Xi$.

   (d) If $g \in \Theta$ and $(g, g') \in \Xi$, then $\mathcal{M}(g)$ and $\tau(g)$ must be incomparable, and $\mathcal{M}(g')$ must be a descendant of $\tau(g)$, i.e., $\mathcal{M}(g') \leq_S \tau(g)$.

Constraint 1 above ensures that the mapping $\mathcal{M}$ is consistent with the leaf-mapping $\mathcal{L}$.
Constraint 2a imposes on $\mathcal{M}$ the temporal constraints implied by $S$. Constraint 2b implies that any internal node in $G^B$ may represent at most one transfer event. Constraint 3 determines the edges of $T$ that are transfer edges. Constraints 4a, 4b, and 4c state the conditions under which an internal node of $G^B$ may represent a speciation, duplication, and transfer respectively. Constraint 4d specifies which species may be designated as the recipient species for any given transfer event.
DTL-scenarios correspond naturally to reconciliations and it is straightforward to infer the reconciliation of $G^B$ and $S$ implied by any DTL-scenario. Figure 5 shows an example of a DTL-scenario. Given a DTL-scenario $\alpha$, one can directly count the minimum number of gene losses, $Loss_{\alpha}$, in the corresponding reconciliation. For brevity, we refer the reader to [2] for further details on how to count losses in DTL-scenarios.

Let $P_\Delta$, $P_\Theta$, and $P_{loss}$ denote the non-negative costs associated with duplication, transfer, and loss events, respectively. The reconciliation cost of a DTL-scenario is defined as follows.

**Definition 2.2.2** (Reconciliation cost of a DTL-scenario). Given a DTL-scenario $\alpha = (L, M, \Sigma, \Delta, \Theta, \Xi, \tau)$ for $G^B$ and $S$, the reconciliation cost associated with $\alpha$ is given by $R_\alpha = P_\Delta \cdot |\Delta| + P_\Theta \cdot |\Theta| + P_{loss} \cdot Loss_{\alpha}$.

A most parsimonious reconciliation is one that has minimum reconciliation cost.

**Definition 2.2.3** (Most Parsimonious Reconciliation (MPR)). Given $G^B$ and $S$, along with $P_\Delta$, $P_\Theta$, and $P_{loss}$, a most parsimonious reconciliation (MPR) for $G^B$ and $S$ is a DTL-scenario with minimum reconciliation cost.

### 2.2.2 Optimal gene tree resolution

Non-binary gene trees cannot be directly reconciled against a species tree. Thus, given a non-binary gene tree $G^N$, the problem is to find a binary resolution of $G^N$ whose MPR with $S$ has the smallest reconciliation cost.

**Problem 1** (Optimal Gene Tree Resolution (OGTR)). Given $G^N$ and $S$, along with $P_\Delta$, $P_\Theta$, and $P_{loss}$, the Optimal Gene Tree Resolution (OGTR) problem is to find a binary resolution
of $G^N$ such that the MPR of $G^B$ and $S$ has the smallest reconciliation cost among all $G^B \in BR(G^N)$.

An example of a non-binary gene tree and a binary resolution is shown in Figure 5.

### 2.3 NP-hardness of the OGTR problem

We claim that the OGTR problem is NP-hard; specifically, that the corresponding decision problem is NP-Complete. The decision version of the OGTR problem is as follows:

**Problem 2 (D-OGTR).**

**Instance:** $G^N$ and $S$, event costs $P_\Delta$, $P_\Theta$, and $P_{\text{loss}}$, and a non-negative integer $l$.

**Question:** Does there exist a $G^B \in BR(G^N)$ such that the MPR of $G^B$ and $S$ has reconciliation cost at most $l$?

**Theorem 2.3.1.** The D-OGTR problem is NP-Complete.

The D-OGTR problem is clearly in NP. In the remainder of this section we will show that the D-OGTR problem is NP-hard using a poly-time reduction from the decision version of the NP-hard minimum 3-set cover problem [36].

#### 2.3.1 Reduction from minimum 3-set cover

The decision version of minimum 3-set cover can be stated as follows.

**Problem 3 (M3SC).**

**Instance:** Given a set of $n$ elements $U = \{u_1, u_2, \ldots, u_n\}$, a set $A = \{A_1, A_2, \ldots, A_m\}$ of $m$ subsets of $U$ such that $|A_i| = 3$ for each $1 \leq i \leq m$, and a nonnegative integer $k \leq m$. 
Question: Is there a subset of $A$ of size at most $k$ whose union is $U$?

We point out that the M3SC problem as defined above is a slight variation of the traditional minimum 3-set cover problem: In our formulation the subsets of $U$ in $A$ are restricted to have exactly three elements each while the traditional formulation allows for the subsets to have less than or equal to three elements [36]. However, it is easy to establish that the NP-Completeness of the traditional version immediately implies the NP-completeness of our formulation of the M3SC problem.

We will also assume, without any loss of generality, that each element $u_i$ appears in at least two subsets from $A$. Elements that only appear in one subset imply necessary inclusion of that subset and so M3SC instances where an element occurs in a single subset can be trivially reduced to instances where each element appears in at least two subsets from $A$.

Consider an instance $\phi$ of the M3SC problem with $U = \{u_1, u_2, \ldots, u_n\}$, $A = \{A_1, A_2, ..., A_m\}$, and $k$ given. We now show how to transform $\phi$ into an instance $\lambda$ of the D-OGTR problem by constructing $G^N$ and $S$ and setting the three event costs in such a way that there exists a YES answer to the M3SC instance $\phi$ if and only if there exists a YES answer to the D-OGTR instance $\lambda$ with $l = k + 48m - 12n$.

2.3.2 Gadget

Gene tree. We first show how to construct the gene tree $G^N$. Note that each element of $U$ occurs in at least two of the subsets from $A$. We will treat each of the occurrences of an element separately and will order them according to the indices $p$ of the $A_p$’s which contain that element. More precisely, for an element $u_i \in U$, we denote by $x_{i,j}$ the $j^{th}$ occurrence of
$u_i$ in $A$. For instance, if element $u_5$ occurs in the subsets $A_2$, $A_4$, $A_{10}$, and $A_{25}$, then $x_{5,2}$ refers to the occurrence of $u_5$ in $A_4$, while $x_{5,4}$ refers to the occurrence of $u_5$ in $A_{25}$.

Let $c_i$ denote the cardinality of the set $\{A_p : u_i \in A_p, \text{ for } 1 \leq p \leq m\}$. Then, $x_{i,j}$ is well defined as long as $1 \leq i \leq n$ and $1 \leq j \leq c_i$. Each $x_{i,j}$ will correspond to exactly four leaves, $x_{i,j,1}$, $x_{i,j,2}$, $x_{i,j,3}$, and $x_{i,j,4}$ in the gene tree $G^N$. In addition, the leaf set of $G^N$ also contains a special node that we label $start$, provided for orienting the reconciliation.

Thus, $Le(G^N) = \{x_{i,j,1}, x_{i,j,2}, x_{i,j,3}, x_{i,j,4} : 1 \leq i \leq n \text{ and } 1 \leq j \leq c_i\} \cup \{start\}$. The overall structure of $G^N$ is shown in Figure 11(a). As shown, the root node of the gene tree is unresolved and has $3m + 3n + 1$ children consisting of (i) the $start$ node, (ii) the $\sum_{i=1}^{n} c_i = 3m$ leaf nodes, collectively called blue nodes, and (iii) the $3n$ internal nodes labeled $g_i$, $g'_i$, and $g''_i$, for each $1 \leq i \leq n$. These internal nodes represent the $n$ elements in $U$ and the subtrees rooted at those nodes have the structure shown in Figure 11(a). Note that the number of children for each of the internal nodes labeled $g_i$, $g'_i$, and $g''_i$, for $1 \leq i \leq n$, is $c_i$. These nodes may thus be either binary or non-binary. The leaves labeled $x_{i,j,3}$ appear in the node $g'_i$, those labeled $x_{i,j,4}$ appear in $g''_i$, and those labeled $x_{i,j,1}$ or $x_{i,j,2}$ appear in $g_i$. The $x_{i,j,1}$'s also appear in the collection of blue nodes and thus appear twice in the gene tree. Note, also, that all the children of a node $g_i$, for $1 \leq i \leq n$, are themselves internal nodes (and binary) and are labeled as $y_{i,j}$, where $1 \leq j \leq c_i$.

**Species tree.** Next, we show how to construct the species tree $S$. The tree $S$ is binary and consists of $m$ subtrees whose root nodes are labeled $s_1, \ldots, s_m$, each corresponding to a subset from $A$, connected together through a backbone tree as shown in Figure 11(b). The exact structure of this backbone tree is unimportant, as long as each $s_i$ is sufficiently separated from the roots of the rest of the subtrees. For concreteness, we will assume that this backbone
Figure 1: **Construction of non-binary gene tree and species tree.** (a) Structure of the non-binary gene tree $G^N$. (b) Structure of the species tree $S$.

This consists of a “caterpillar” tree as shown Figure 11(b), and that $9m$ extraneous leaves (not present in the gene tree) have been added to this backbone as shown in the figure to ensure that each pair of subtrees is sufficiently separated.
Recall that we use \( x_{i,j} \) to denote the \( j^{th} \) occurrence of \( u_i \) in \( A \). Assuming that \( u_i \in A_p \) and that \( x_{i,j} \) refers to the occurrence of \( u_i \) in \( A_p \), we define \( f(i,p) \) to be \( j \). In other words, if the \( j^{th} \) occurrence of an element \( u_i \) is in the subset \( A_p \), then we assign \( f(i,p) \) to be \( j \). Each \( S_i \) corresponds to the subset \( A_i \) and has the structure depicted in Figure 11(b). In particular, if \( A_i \) contains the three elements \( u_a, u_b, \) and \( u_c \), then \( S_i \) contains the 12 leaves labeled \( x_{a,f(a,i),j}, x_{b,f(b,i),j}, \) and \( x_{c,f(c,i),j}, \) for \( 1 \leq j \leq 4 \).

**Event costs.** We assign the following event costs for problem instance \( \lambda \): \( P_{\Delta} = 2, P_{\Theta} = 4, \) and \( P_{\text{loss}} = 1. \)

Note that the D-OGTR instance \( \lambda \) can be constructed in time polynomial in \( m \) and \( n \).

**Claim 1.** There exists a YES answer to the M3SC instance \( \phi \) if and only if there exists a YES answer to the D-OGTR instance \( \lambda \) with \( l = k + 48m - 12n \).

The remainder of this section is devoted to proving this claim which, in turn, would complete our proof for Theorem 4.3.1. We begin by explaining the main idea of the reduction and describing the association between the instances \( \phi \) and \( \lambda \), and then prove the forward and reverse directions of the claim.

### 2.3.3 Key insight

The main idea behind our reduction can be explained as follows: In the gene tree \( G^N \), subtrees \( G^N(g_i), G^N(g'_i) \) and \( G^N(g''_i) \) correspond to the element \( u_i \), for each \( 1 \leq i \leq n \), while in the species tree the subtree \( S(s_j) \) corresponds to the subset \( A_j \), for each \( 1 \leq j \leq m \). Let \( G^B \) be any binary resolution of \( G^N \). It can be shown that in any MPR of any optimal binary resolution \( G^B \) of \( G^N \) the following must hold: For each \( i \in \{1, \ldots, n\}, g_i \) (along with
\(g'_i\) and \(g''_i\) must map to an \(S(s_j)\) for which \(u_i \in A_j\). Under these restrictions on the mappings, observe that if we were to solve the OGTR problem on \(G^N\) and \(S\) and then choose all those \(A_j\)'s for which the subtree \(S(s_j)\) has at least one of the \(g_i\)'s mapping into it, then the set of chosen \(A_j\)'s would cover all the elements of \(U\).

The source of the optimization is that, due to the specific construction of the gene tree and species tree, it is more expensive (in terms of reconciliation cost) to use more \(S(s_j)\)'s for the mapping. Thus, all the \(g_i\)'s (along with \(g'_i\)'s and \(g''_i\)'s) must map to as few of the subtrees, \(S(s_j)\)'s, as possible. Recall that the OGTR problem optimizes the topology of the binary resolution \(G^B\) in such a way that its MPR with \(S\) has minimum reconciliation cost. Thus, the OGTR problem effectively optimizes the topology of \(G^B\) in a way that minimizes the total number of \(S(s_j)\)'s receiving mappings from the \(g_i\)'s, \(g'_i\)'s, or \(g''_i\)'s, yielding a set cover of smallest possible size. This is the key idea behind our reduction and we develop this idea further in the next two subsections.

2.3.4 Proof of Claim 2: forward direction

Let us assume that we have a YES answer for the M3SC instance \(\phi\). We will show how to create a binary resolution \(G^B\) of \(G^N\) whose MPR with \(S\) has reconciliation cost at most \(k + 48m - 12n\).

We first show how to resolve the subtrees \(G^N(g_i)\), \(G^N(g'_i)\), and \(G^N(g''_i)\), for \(1 \leq i \leq n\). Recall that, for any fixed \(i\), these three subtrees correspond to element \(u_i\) of \(U\). The \(y_{i,j}\)'s in \(G^N(g_i)\) correspond to the different occurrences of element \(u_i\) in the subsets from \(A\). The same holds for the \(x_{i,j,3}\)'s in \(G^N(g'_i)\) and the \(x_{i,j,4}\)’s in \(G^N(g''_i)\).
Suppose a solution to instance $\phi$ consists of the $k$ subsets $A_{r(1)}, A_{r(2)}, \ldots, A_{r(k)}$. Since every element in $U$ must be covered by at least one of these $k$ subsets, we can designate a covering subset for each element $u_i \in U$, $1 \leq i \leq n$, chosen arbitrarily from among those subsets in the solution that contain $u$. Suppose that element $u_i$ is assigned the covering subset $A_{j}$ (so we must have $u_i \in A_{j}$ and $A_{j} \in \{A_{r(1)}, A_{r(2)}, \ldots, A_{r(k)}\}$). The subtree $G^N(g_i)$ will then be resolved as follows: The $y_{i,j}$ corresponding to the occurrence of $u_i$ in $A_{j}$, i.e., $y_{i,f(i,j)}$, will be separated out as one of the two children of $g_i$. The other child of $g_i$ will be the root of an arbitrary caterpillar tree on all the remaining $y_{i,j}$'s in $G^N(g_i)$. This is depicted in Figure 2(d).

The subtrees $G^N(g_{i}')$ and $G^N(g_{i}'')$ are resolved similarly, except that in $G^N(g_{i}')$ the leaf node $x_{i,f(i,j),3}$ is separated out and in $G^N(g_{i}'')$ the leaf node $x_{i,f(i,j),4}$ is separated out. Thus, the resolution of $G^N(g_i), G^N(g_{i}')$, and $G^N(g_{i}'')$ is done based on the assigned covering subset of element $u_i$. This is repeated for all $i$, where $1 \leq i \leq n$.

Next, we show how to resolve the root node of $G^N$ to obtain $G^B$. The start node will become an outgroup to the rest of $G^B$. The backbone of the rest of $G^B$ consists of an arbitrary caterpillar tree on $k$ “leaf” nodes as shown in Figure 2(a). These $k$ nodes are labeled $h_{r(1)}, \ldots, h_{r(k)}$ and are the root nodes of $k$ subtrees. Each of the $k$ subtrees corresponds to one of the subsets $A_{r(1)}, A_{r(2)}, \ldots, A_{r(k)}$. In particular, subtree $G^B(h_{r(i)})$, for $1 \leq i \leq k$ corresponds to the subset $A_{r(i)}$. Each of the blue nodes and the subtrees rooted at the $g_i$'s, $g_i'$'s, and $g_i''$'s, for $1 \leq i \leq n$ will be included in one of these $k$ subtrees. Specifically, the subtree $G^B(h_{r(j)})$ will include all those $g_i$'s, $g_i'$'s, and $g_i''$'s for which the covering subset of the corresponding $u_i$ is $A_{r(j)}$. Since there may be $0, 1, 2$, or $3$ $i$'s for which the covering subset of $u_i$ is $A_{r(j)}$, the sizes of different $G^B(h_{r(j)})$ subtrees may vary. The structure of $G^B(h_{r(j)})$ when there are $3$ $i$'s is depicted in Figure 2(b). The structure of $G^B(h_{r(j)})$ when there are only
1 or 2 such \(i\)'s is similar and is the induced subtree, on the relevant \(i\)'s, of the full subtree for all 3 \(i\)'s. As shown in the figure, note that each subtree \(G^B(h_{r(j)})\) also includes at least three blue nodes, corresponding to the three elements in \(A_{r(j)}\). These three blue nodes are included even for cases where there are fewer than 3 \(i\)'s. Thus, when there are 0 such \(i\)'s, which can happen when the size of the minimum set cover for instance \(\phi\) is less than \(k\), the subtree \(G^B(h_{r(j)})\) consists of the three blue nodes.

This results in the assignment of all \(g_i\)'s, \(g'_i\)'s, and \(g''_i\)'s, for \(1 \leq i \leq n\) to one of the subtrees \(G^B(h_{r(j)})\), for \(1 \leq j \leq k\). As discussed above, 3\(k\) out of the 3\(m\) blue nodes also get assigned in this process. The remaining 3\(m - 3k\) of the blue nodes are organized into an arbitrary caterpillar tree and added to the subtree \(G^B(h_{r(k)})\) as shown in Figure 2(c).

This finishes our description of \(G^B\). The next lemma follows directly from this construction of \(G^B\).

**Lemma 1.** Gene tree \(G^B\) is a binary resolution of \(G^N\).

**Proof.** From the construction of \(G^B\) from \(G^N\) above, it is easy to verify that all edges (or, more accurately, clusters) in \(G^N\) also appear in \(G^B\). By construction, \(G^B\) is also binary. Thus, \(G^B\) is a binary resolution of \(G^N\). \(\blacksquare\)

Next, we show how to construct a DTL-scenario for \(G^B\) and \(S\) with cost at most \(k + 48m - 12n\).

**DTL-scenario for \(G^B\) and \(S\).** All leaves of the gene tree, \(G^B\), map to the corresponding leaves on the species tree \(S\). Consider the depiction of \(G^B\) as shown in Figure 2. For each \(i\) such that \(1 \leq i \leq k - 1\), \(h_{r(i)}\) and \(pa(h_{r(i)})\) map to \(s_i\). The node \(pa(h_{r(i)})\) represents a transfer event and \(h_{r(i)}\) a speciation event. Finally, \(h_{r(k)}\) maps to \(s_k\) and represents a speciation event.
For each internal node $a$ in subtree $B$, if only one child of $a$ is a leaf node then $a$ has the same mapping as its unique leaf-child. If both children of $a$ are leaf nodes, then it has the same mapping as any one of them. Thus, all internal nodes of $B$ are transfer nodes.

For each $i$, consider subtree $G_B(h_{r(i)})$. For each element $j$ represented in that subtree, $g'_j$ and $g''_j$ are all transfer nodes and map to leaves $x_{j,v,3}$ and $x_{j,v,4}$ on $S(s_i)$, respectively. Consider any internal node $a$ in the subtrees $G^B(g'_j)$ and $G^B(g''_j)$. If only one child of $a$ is a leaf node then $a$ has the same mapping as its unique leaf-child. If both children of $a$ are leaf nodes, then it has the same mapping as any one of them. Thus, all internal nodes of $G^B(g'_j)$ and $G^B(g''_j)$ are transfers. In the subtree $G^B(g_j)$, each node labeled $y_{..}$ is a speciation node and maps to the LCA of the mapping of its two children. Consider any other internal node $a$ in the subtree $G^B(g_j)$. If only one child of $a$ is a $y_{..}$ node then $a$ has the same mapping as its unique $y_{..}$-child. If both children of $a$ are $y_{..}$ nodes, then it has the same mapping as any one of them. Thus, all nodes along the spine of $G^B(g_j)$ are transfers. Furthermore, $pa(g_j)$ is a duplication node, while $pa(g'_j)$ and $pa(g''_j)$ are both speciation nodes.

The root of $G^B$, maps to the start node on the species tree $S$ and is a transfer node. All other nodes of $G^B$ are speciation nodes. We denote the resulting DTL-scenario for $G^B$ and $S$ by $\alpha$. It is not difficult to verify that $\alpha$ is a valid DTL-scenario.

The following two lemmas help bound the cost of the reconciliation implied by $\alpha$.

**Lemma 2.** Under DTL-scenario $\alpha$, the reconciliation cost of any subtree $G^B(g_j)$, $G^B(g'_j)$, or $G^B(g''_j)$, for $1 \leq j \leq n$, with $S$ is $(c_j - 1) \times P_\emptyset$.

**Proof.** Based on the reconciliation implied by $\alpha$, each internal node along the spine of any subtree $G^B(g_j)$, $G^B(g'_j)$, or $G^B(g''_j)$, for $1 \leq j \leq n$, is a transfer node. Note that each of the
nodes in $G^B(g_j)$ labeled $y_j$. is a speciation node and the subtrees rooted at the $y_j$’s do not invoke any losses. Thus, none of the subtrees $G^B(g_j)$, $G^B(g'_j)$, or $G^B(g''_j)$, for $1 \leq j \leq n$, invoke any duplications or losses. Since the number of internal nodes along the spines of each of $G^B(g_j)$, $G^B(g'_j)$, or $G^B(g''_j)$, for $1 \leq j \leq n$, is $c_j - 1$, the lemma follows.

Recall that, since there may be 0, 1, 2, or 3 $i$’s for which the covering subset of $u_i$ is $A_{r(j)}$, the sizes of different $G^B(h_{r(j)})$ subtrees may vary. The next two lemmas shows that, under $\alpha$, the reconciliation cost of any subtree $G^B(h_{r(j)})$ behaves predictably. The next lemma applies to all $G^B(h_{r(j)})$ where $1 \leq j \leq k - 1$. We separate out the case of $j = k$ as a separate lemma since all the unassigned blue nodes get attached to $G^B(h_{r(k)})$.

Lemma 3. For each $j$, $1 \leq j \leq k - 1$, the total reconciliation cost of subtree $G^B(h_{r(j)})$ with $S$ under DTL-scenario $\alpha$ is as follows:

1. If there exist exactly three distinct subtrees $g_a$, $g_b$, and $g_c$, where $1 \leq a, b, c \leq n$, within subtree $G^B(h_{r(j)})$, then the reconciliation cost is $12 \times (c_a + c_b + c_c - 3) + 9$.

2. If there exist exactly two distinct subtrees $g_a$ and $g_b$, where $1 \leq a, b \leq n$, within subtree $G^B(h_{r(j)})$, then the reconciliation cost is $12 \times (c_a + c_b - 2) + 9$.

3. If there exists exactly one subtree $g_a$, where $1 \leq a \leq n$, within subtree $G^B(h_{r(j)})$, then the reconciliation cost is $12 \times (c_a - 1) + 9$.

4. If there do not exist any subtrees of the form $g_a$, where $1 \leq a \leq n$, within subtree $G^B(h_{r(j)})$, then the reconciliation cost is 9.

Proof. Consider the first case of the lemma. Based on Lemma 2.3.4, the reconciliation cost of any subtree $G^B(g_i)$, $G^B(g'_i)$, $G^B(g''_i)$, for each $1 \leq i \leq n$, with $S$ is $P_{\Theta} \times (c_i - 1)$. Thus, the
total reconciliation cost contributed by all such subtrees is \( P_\Theta \times 3 \times (c_a + c_b + c_c - 3) \), which is \( 12 \times (c_a + c_b + c_c - 3) \). Also, as shown in Figure 3, nodes \( x, y, \) and \( z \) are duplication nodes that each also invoke one loss, and all the other nodes of \( G^B(h_{r(j)}) \) are speciations without any losses. Thus, the total reconciliation cost of \( G^B(h_{r(j)}) \) under DTL-scenario \( \alpha \) is \( 12 \times (c_a + c_b + c_c - 3) \) plus the cost of three duplications and three losses, which is \( 12 \times (c_a + c_b + c_c - 3) + 9 \).

For the other cases, note that for each set of "missing" subtrees \( g_i, g'_i, \) and \( g''_i \), for \( i \in \{a, b, c\} \), the reconciliation of \( G^B(h_{r(j)}) \) with \( S \) invokes two additional losses for the missing \( g'_i \) and \( g''_i \), and one less duplication for the missing \( g_i \). Since \( P_{loss} = 1 \) and \( P_\Delta = 2 \), there is no net change on the total additive cost of 9. Thus, in cases 2, 3, and 4, the total cost is the sum of the reconciliation costs for the subtrees \( g_i, g'_i, \) and \( g''_i \) that are in \( G^B(h_{r(j)}) \), plus the additive cost of 9.

\[ \text{Lemma 4.} \quad \text{The total reconciliation cost of subtree} \ G^B(h_{r(k)}) \text{ with} \ S \text{ under DTL-scenario} \ \alpha \text{ is the same as given in Lemma 3 but with an additional additive cost of} \ 4 \times (3m - 3k). \]

\[ \text{Proof.} \quad \text{The proof for this lemma proceeds identically to that of Lemma 3, depending on whether} \ G^B(h_{r(k)}) \text{ falls under case 1, 2, 3, or 4. However,} \ G^B(h_{r(k)}) \text{ contains a additional subtree of} \ (3m - 3k) \text{ unassigned blue nodes (see Figure 2) and there is an additional cost associated with that subtree. As shown in Figure 2c, this subtree introduces} \ 3m - 3k \text{ additional internal nodes to} \ G^B(h_{r(k)}). \text{ Under DTL-scenario} \ \alpha, \text{ each of these} \ 3m - 3k \text{ internal nodes is a transfer node (and there are no duplications or losses). This contributes an additive reconciliation cost of} \ P_\Theta \times (3m - 3k) \text{ to the reconciliation cost of} \ G^B(h_{r(k)}). \]

Thus, the reconciliation cost of any subtree \( G^B(h_{r(j)}) \) depends only on the total reconciliation cost of the subtrees \( G^B(g_i), G^B(g'_i), \text{ and} \ G^B(g''_i) \), for each \( 1 \leq i \leq n, \) within \( G^B(h_{r(j)}) \)
plus an additive cost of 9. In addition, there is an added cost of $4 \times (3m - 3k)$ for the subtree $G^B(h_{r(k)})$.

The following lemma implies the forward direction of Claim 2.

Lemma 5. Any MPR of $G^B$ with $S$ must have reconciliation cost at most $k + 48m - 12n$.

Proof. Since $\alpha$ is a valid DTL-scenario, an MPR of $G^B$ with $S$ cannot have reconciliation cost more than that implied by $\alpha$. Thus, it suffices to show that the DTL-scenario $\alpha$ has a reconciliation cost of exactly $k + 48m - 12n$. The total reconciliation cost under $\alpha$ is the sum of the reconciliation costs for each subtree $G^B(h_{r(j)})$, for $1 \leq j \leq k$, and the reconciliation cost implied by the backbone of $G^B$ that connects these $k$ subtrees.

Consider the $k$ $G^B(h_{r(j)})$’s. Note that there are exactly $n$ $g_i$’s, $g'_i$’s and $g''_i$’s distributed among these $k$ subtrees. Thus, by Lemmas 3 and 4, the total reconciliation cost of the $k$ subtrees is $12 \times \sum_{i=1}^{n} (c_i - 1) + 9 \times k + 4 \times (3m - 3k)$. Since $\sum_{i=1}^{n} c_i = 3m$, this evaluates to $48m - 12n - 3k$.

Now consider the backbone of $G^B$ that connects the $k$ $G^B(h_{r(j)})$’s (see Figure 2). According to DTL-scenario $\alpha$, for each $j \in \{1, \ldots, k - 1\}$, the node $pa(h_{r(j)})$ is a transfer node. In addition, the root node of $G^B$ is also a transfer node. Moreover, according to the mapping defined by $\alpha$, this backbone does not invoke any losses. Thus, the backbone contributes a total of $P_{\Theta} \times k$, which is $4k$, to the total reconciliation cost.

The total reconciliation cost of $G^B$ with $S$ under DTL-scenario $\alpha$ is thus $48m - 12n - 3k + 4k$, which is $k + 48m - 12n$. \qed
2.3.5 Proof of Claim 2: reverse direction

Conversely, let us assume that we have a YES answer for the OGTR instance $\lambda$ with $l = k + 48m - 12n$. We will show that there exists a solution of size at most $k$ for the set cover instance $\phi$. We first characterize the structure of optimal resolutions and their most parsimonious reconciliations.

Lemma 6. For any optimal binary resolution $G^B$ of $G^N$, all MPRs of $G^B$ with $S$ must satisfy the following:

1. Each node in $I(G^B)$ maps to either the start node or to a node in the subtree $S(s_j)$, for some $j \in \{1, \ldots, m\}$.

2. Each subtree $G^B(g_i), G^B(g_i'), G^B(g_i'')$, where $1 \leq i \leq n$, has at least $(c_i - 1)$ transfer nodes.

Proof. Part (1). Suppose there exists a minimum-cost DTL-scenario $\alpha$ for $G^B$ and $S$ such that, under $\alpha$, there exists a node in $I(G^B)$ that does not map to the start node or to a node in the subtree $S(s_j)$, for any $j \in \{1, \ldots, m\}$. We will show how to construct an alternative DTL-scenario $\beta$ with lower reconciliation cost, leading to a contradiction.

Note that the set $V(S) \setminus (\bigcup_{i=1}^{m} V(S(s_i)) \cup \text{start})$ consists of three types of nodes: (i) the set of extra leaves added to each species tree branch (9 per branch), (ii) the set of internal nodes created by adding the extra leaves, and (iii) the rest of the nodes (each representing a branching point in the induced species tree without the added extra leaves). We will refer to these as extra-leaf node, extra nodes, and backbone nodes, respectively. Note that, by the definition of DTL scenarios, none of the nodes of $I(G^B)$ can map to an extra-leaf node. They may, however, map to extra nodes or backbone nodes. We will first show how to modify $\alpha$ into a
new DTL-scenario $\alpha'$ with the same or lower reconciliation cost such that no node of $I(G^B)$ maps to an extra node.

*Modifying mappings to extra nodes.* Suppose $I(G^B)$ contains nodes that map to extra nodes under the DTL-scenario $\alpha$. Let $a$ denote such a node. If there is more than one such node of $G^B$, then $a$ is chosen to be a node that does not have any descendants that map to extra nodes. Let $b$ denote the node of $S$ to which $a$ maps. Let $c$ denote the closest descendant of $b$ that is not an extra node (or an extra-leaf node). Thus, $c$ must either be an $s_i$, for $1 \leq i \leq m$, or a backbone node. Note that, by definition, $a$ cannot be a speciation node. However, it may be a duplication or a transfer, yielding the following two cases.

**Case 1.** $a$ is a duplication: Since no descendant of $a$ maps to an extra node, we can change its mapping from $b$ to $c$. The node $a$ still remains a duplication node, and this change does not create any additional duplications, transfers, or losses. In fact, the number of losses is reduced by at least one since there are no longer any losses of the duplicated lineage along the path from $b$ to $c$.

**Case 2.** $a$ is a transfer: As in the previous case, since no descendant of $a$ maps to an extra node, we can change its mapping from $b$ to $c$. The node $a$ remains a transfer node, and this change does not create any additional duplications, transfers, or losses. Note that, if the node $pa(a)$ exists and maps either to $b$ or an ancestor of $b$, then there is no reduction in the number of losses. And similarly, if the node $pa(a)$ does not exist or does not map either to $b$ or to an ancestor of $b$, then the number of losses reduces by at least one.

Thus, in both cases, there is no increase in the reconciliation cost. We can apply this procedure iteratively to each node $a$ in $G^B$ that maps to an extra node, resulting in a new DTL-scenario $\alpha'$ that has either the same or lower reconciliation cost, and in which none of the nodes
of $G^B$ map to an extra node. If the reconciliation cost of $\alpha'$ is smaller than that of $\alpha$, then we have a contradiction and the proof finishes. If the two costs are the same, one of the following two cases must hold: (i) There were no nodes in $I(G^B) \setminus \{rt(G^B)\}$ that mapped to an extra node under $\alpha$ (and thus $\alpha' = \alpha$, or (ii) all the candidate $a$'s were transfer events and moreover, each $a$ has a parent $pa(a)$ that maps to a node along the path from $b$ to $rt(S)$. In either case, there must be at least one node in $I(G^B) \setminus \{rt(G^B)\}$ that maps to a backbone node under $\alpha'$.

Next, we show how to further modify DTL-scenario $\alpha'$ into DTL-scenario $\beta$ by modifying the mappings to the backbone nodes.

**Modifying mappings to backbone nodes.** Let $a$ be a node from $I(G^B)$ that maps to a backbone node under DTL-scenario $\alpha'$. If there is more than one such node of $G^B$, then $a$ is chosen to be a node that does not have any descendants that map to backbone nodes. Let $b$ denote the backbone node of $S$ to which $a$ maps. We now have three cases depending on whether $a$ is a speciation, duplication, or transfer.

**Case 1.** $a$ is a speciation: In this case, one child of $a$ must map to a node in subtree $S(s_i)$ and the other child to a node in the subtree $S(s_j)$, where $1 \leq i, j \leq m$, and $i \neq j$. Moreover $s_i$ and $s_j$ must both be descendants of $b$. We will change the mapping to $a$ from $b$ to $s_i$. The node $a$ now becomes a transfer node and the DTL-scenario remains valid. With this change, the number of transfers increases by 1, and the number of losses decreases by at least 9 (since there is one fewer loss at each of the extra nodes along the path from $b$ to $s_i$). Thus, overall, the reconciliation cost decreases by at least $9 \times P_{\text{loss}} - 1 \times P_{\Theta}$, which is 5.

**Case 2.** $a$ is a duplication: In this case, one child of $a$ must map to a node in subtree $S(s_i)$ and the other child to a node in the subtree $S(s_j)$, where $1 \leq i, j \leq m$, and $i$ may be the same as $j$. Moreover $s_i$ and $s_j$ must both be descendants of $b$. We will change the mapping to $a$ from
The node $a$ now becomes either a transfer node, if $i \neq j$, or remains a duplication node if $i = j$, and the DTL-scenario remains valid. With this change, the number of losses decreases by at least 9 (since there is one fewer loss at each of the extra nodes along the path from $b$ to $s_i$), while the number of transfers may increase by one with a corresponding decrease in one duplication. Thus, overall, the reconciliation cost decreases by at least $9 \times P_{\text{loss}} - 1 \times (P_{\Theta} - P_{\Delta})$, which is 7.

**Case 3.** $a$ is a transfer: In this case, one child of $a$ must map to a node in subtree $S(s_i)$ and the other child to a node in the subtree $S(s_j)$, where $1 \leq i, j \leq m$ and $i \neq j$, such that $s_i$ is a descendant of $b$ while $s_j$ is neither a descendant nor an ancestor of $b$. We will change the mapping to $a$ from $b$ to $s_i$. The node $a$ remains a transfer node and the DTL-scenario remains valid. In this case, if the node $pa(a)$ exists and maps either to $b$ or an ancestor of $b$, then there is no reduction in the number of losses. But if the node $pa(a)$ does not exist or does not map either to $b$ or to an ancestor of $b$, then the number of losses, and the reconciliation cost, reduces by at least 9.

We can apply this procedure iteratively to each node $a$ in $G^B$ that maps to a backbone node, resulting in a new DTL-scenario $\beta$ that has reconciliation cost no greater than that of $\alpha$. In particular, if any of the $a$’s are duplications or speciations, then the new DTL-scenario $\beta$ has a cost smaller than that of $\alpha$ and we have a contradiction. Similarly, if any of the $a$’s are transfers such that their parent node does not map to $b$ or its ancestor, then $\beta$ must have cost smaller than that of $\alpha$. Therefore, assume that none of the $a$’s is a speciation or duplication, and that the parent of any given $a$ maps to $b$ or its ancestor. Under this assumption, as we iterate through all the candidate $a$’s we eventually reach an $a$ for which $pa(a)$ is $rt(G^B)$. If $rt(G^B)$ maps to the start node then, we are done, since then updating $a$’s mapping will reduce
the reconciliation cost by at least 9. Otherwise, if \( rt(G^B) \) maps to either \( b \) or its ancestor, then we can update the mapping of \( rt(G^B) \) to be the same as the mapping of \( a \) (i.e., to \( s_i \)). With this change, \( rt(G^B) \) becomes a transfer node, irrespective of its previous event-type, and the DTL-scenario remains valid. This would result in a reduction of at least \( 9 - P_\Theta = 5 \) in the reconciliation cost.

Thus, the reconciliation cost under \( \beta \) would be strictly smaller than the reconciliation cost under \( \alpha \), leading to a contradiction.

Part (2). Consider any \( g_i' \), for \( 1 \leq i \leq n \). \( G^B(g_i') \) contain \( c_i \) leaves and \( (c_i - 1) \) internal nodes, and each of the \( c_i \) leaves maps to a different subtree \( S(s_j) \), for \( 1 \leq j \leq m \). We will show that all \( (c_i - 1) \) internal nodes of \( G^B(g_i') \) must be transfers. Suppose not. Then there must be an internal node \( a \) in \( G^B(g_i') \) that is not a transfer node. Without loss of generality assume that \( a \) is such that all of its internal node descendants are transfers. By the part (1) of this lemma, we know that each node of \( G^B \) maps either to a node in \( S(s_j) \), for \( 1 \leq j \leq m \) or to the start node. Now, since each leaf node maps to a different \( S(s_j) \), for \( 1 \leq j \leq m \), the two children of \( a \) must also map to two different subtrees \( S(s_j) \), for \( 1 \leq j \leq m \). Therefore, if \( a \) is either a speciation or duplication, it must map to a node that is neither in one of the \( S(s_j) \)’s nor the start node, which is a contradiction.

The proof for \( g_i'' \) is identical to the one for \( g_i' \). For \( g_i \), observe that there are \( c_i \) of the \( y_{i,j} \)’s and each of the \( y_{i,j} \)’s contains exactly two leaves that both map to the same subtree \( S(s_j) \), for \( 1 \leq j \leq m \). Moreover, the two leaves of each distinct \( y_{i,j} \) both map to a distinct subtree \( S(s_j) \), for \( 1 \leq j \leq m \). Thus, each of the \( y_{i,j} \)’s must themselves map to distinct subtrees \( S(s_j) \), for \( 1 \leq j \leq m \). Based on this observation, the proof for \( g_i \) also follows along the same lines as the proof for \( g_i' \).
For the next few lemmas we need the following two definitions:

**Definition 2.3.1** (Most recent Ancestral Transfer). Given a DTL-scenario $\alpha$ for $G^B$ and $S$, and any node $a \in V(G^B)$, we define the Most Recent Ancestral Recipient node of $a$, denoted $MRAR(a)$, to be the first node $x$ along the path from $a$ to $rt(G^B)$ that $(pa(x), x) \in \Xi$ (i.e., $x$ is the recipient of a transfer event). Note that not all $a \in V(G^B)$ have an MRAR node.

**Definition 2.3.2** (Canonical optimal resolution and MPR). Consider an optimal resolution $G^B$ of $G^N$ and an MPR, represented by DTL-scenario $\alpha$, of $G^B$ with $S$. We say that $G^B$ and the MPR implied by $\alpha$ are both canonical if the node $rt(G^B)$ maps to the start node in $S$.

Not all optimal resolutions $G^B$ and their MPRs are canonical. However, as we show next, any given optimal resolution $G^B$ and its MPR $\alpha$ that are not canonical can be converted into a canonical resolution $G^B'$ and canonical MPR $\alpha'$, without any change in reconciliation cost.

**Lemma 7.** Consider an optimal binary resolution $G^B$ of $G^N$ along with its MPR with $S$, represented by DTL-scenario $\alpha$. If $G^B$ and its MPR $\alpha$ are not canonical, then it is possible to efficiently compute a canonical optimal resolution $G^B'$ and a canonical MPR, $\alpha'$ of $G^B'$ with $S$.

**Proof.** Since $G^B$ and its MPR $\alpha$ are not canonical, it follows from Lemma 6(1) that $rt(G^B)$ must map to $S(s_i)$, for some $i \in \{1, \ldots, m\}$. We will show how to create an alternative binary resolution $G^B'$ of $G^N$ and an MPR $\alpha'$ of $G^B'$ with $S$, with the same reconciliation cost such that $rt(G^B')$ maps to the start node. Since $rt(G^B)$ does not map to the start node, the start node must have an MRAR. We perform a subtree-prune-and-regraft operation on $G^B$ as follows: We prune the subtree $G^B(MRAR(start))$ and regraft it above the root of the remainder of $G^B$, thereby creating a new root node in the resulting tree. Thus, the resulting

---

30
tree, $G^{B'}$, has a root node whose children are the roots of the subtrees $G^{B}(MRAR(start))$ and $G^{B}(rt(G^{B}))) \setminus G^{B}(MRAR(start))$. The DTL-scenario $\alpha'$ for $G^{B'}$ and $S$ is identical to that for $G^{B}$ and $S$, except that, the edge from $rt(G^{B'})$ to $G^{B}(rt(G^{B}))) \setminus G^{B}(MRAR(start))$ is designated as a transfer edge, and $rt(G^{B'})$ is assigned the same mapping as that for $MRAR(start)$ in $G^{B}$. The resulting DTL-scenario remains valid and has the same reconciliation cost as the original since we simply remove the transfer edge $(pa(MRAR(start)), MRAR(start))$ in $G^{B}$ and replace it with another. Observe that $rt(G^{B'})$ must now map to the $start$ node resulting in a canonical binary resolution and its canonical MPR. Also observe that this construction has time complexity linear in the size of $G^{B}$.

Lemma 8. Given any canonical optimal binary resolution $G^{B}$ of $G^{N}$ and a canonical MPR of $G^{B}$ with $S$, each node in $V(G^{B})$ that maps to a node of $S(s_{j})$, for any $1 \leq j \leq m$, must have an MRAR node.

Proof. For contradiction, suppose there exists an $S(s_{j})$, where $1 \leq j \leq m$, such that at least one of the nodes of $G^{B}$ that maps to $S(s_{j})$ doesn’t have an MRAR. Since $G^{B}$ and its given MPR are canonical, $rt(G^{B})$ must map to the $start$ node. Consider all those nodes of $G^{B}$ that map to $S(s_{j})$ but do not have any ancestors that map to $S(s_{j})$. From Lemma 6(1), it follows that all such nodes must be recipients of transfer events. Since all other nodes of $G^{B}$ that map to $S(s_{j})$ must descend from one such node in $G^{B}$, the lemma follows.

Lemma 9. Consider any subtree $S(s_{j})$, for $1 \leq j \leq m$, of the species tree, and consider its three leaf nodes with labels of the form $x_{1}, \ldots, 1$. There are exactly three blue nodes in the gene tree that must map to these three leaf nodes of $S(s_{j})$. Let these three blue nodes be denoted by
Given any canonical optimal binary resolution $G^B$ of $G^N$ and a canonical MPR of $G^B$ with $S$, if there are no nodes $g_i$, $g_i'$, or $g_i''$, for any $i \in \{1, \ldots, n\}$, that map to a node of $S(s_j)$, then the MRAR’s for $a$, $b$, and $c$ must all be distinct and must map to nodes of $S(s_j)$.

**Proof.** By Lemma 2.3.5 we know that each blue node has an MRAR in $G^B$. Note that each of these MRAR nodes must map to a node of the subtree $S(s_j)$ to which its blue node maps. We therefore separate our analysis into three cases: (i) There is only one node that is an MRAR for $a$, $b$, and $c$, (ii) there are two nodes that are MRAR’s for $a$, $b$, and $c$, and (iii) there are three nodes that are MRAR’s for $a$, $b$, and $c$. If case (iii) holds, then we are done, since each of those three MRAR’s must map to a node of $S(s_j)$. We will consider each of these three cases:

**Case 1.** If $a$, $b$, and $c$ have the same MRAR, say $x$, then $x \geq \text{lca}_{G^B}(a, b, c)$. Let $y$ denote $\text{lca}_{G^B}(a, b, c)$. Since there are no MRAR’s within the subtree $G^B(y)$, each node of $G^B(y)$ must map to $S(s_j)$, and $y$ and $x$ must both map to $s_j$. Without loss of generality, let $z$ denote $\text{lca}_{G^B}(a, b)$ such that $z < y$. Observe that, since none of the $g_i$’s, $g_i'$’s or $g_i''$’s map to $S(s_j)$, $a$, $b$, and $c$ are the only leaves of $G^B(y)$ that map to leaves in $S(s_j)$. This implies that all of the internal nodes along the paths from $y$ to $a$, $b$, and $c$, except for nodes $y$ and $z$ must be transfer nodes. The observation also implies that subtree $G^B(y)$ must induce at least 9 losses in $S(s_j)$. Furthermore, each node along the path from $x$ to $y$ must itself be a transfer node for the same reason.

We will now show how to create an alternative DTL-scenario $\alpha'$ with smaller reconciliation cost than $\alpha$, leading to a contradiction. We update the mappings of all internal nodes along the path from $a$ to $x$ (including $x$) to be the mapping of $a$, all nodes along the path from $b$ to $z$ (not including $z$) to be the mapping of $b$, and the mapping of all internal nodes along the path from $c$ to $y$ (not including $y$) to be the mapping of $c$. The resulting DTL-scenario remains valid,
and only introduces two additional transfer nodes, \( y \) and \( z \), and no additional losses. This is because all existing transfer nodes on the paths remain valid transfer nodes, and changing the mapping of the \( MRAR \) node does not lead to any increase in the number of losses (only the recipient node of the transfer event changes). Since this update decreases the number of losses by 9, the new DTL-scenario \( \alpha' \) must have a reconciliation cost that is lower than the original \( \alpha \) by \( 9 \times P_{\text{loss}} - 2 \times P_\Theta = 1 \). A contradiction.

Case 2. If there are two nodes that are \( MRAR \)’s for \( a \), \( b \), and \( c \), then two of the blue nodes, say \( a \) and \( b \) must have the same \( MRAR \). Let \( x \) denote the \( MRAR \) of \( a \) and \( b \), \( y \) denote \( \text{lca}_{G_B}(a, b) \), and \( x' \) denote \( MRAR(c) \). Then, \( x \geq y \), and each node along the paths from \( a \) to \( y \) and \( b \) to \( y \) must map to \( S(s_j) \). Note that the subtree \( G_B(y) \) must invoke at least 6 losses in \( S(s_j) \). We will show that, in spite of the relative arrangement of \( a \), \( b \), \( c \), \( y \), \( x \), and \( x' \), all internal nodes along the paths from \( a \) to \( y \) (not including \( y \)), \( b \) to \( y \) (not including \( y \)), \( c \) to \( x' \) (including \( x' \)), and \( y \) to \( x \) (including \( x \) unless \( x = y \)) must be transfer nodes.

Consider the path \( a \) to \( y \). Suppose there is an internal node, say \( z \), where \( z \neq y \), along this paths that is not a transfer node. Then \( z \) must be a speciation or duplication node. Let \( z' \) denote the child of \( z \) that is not on the \( a \) to \( y \) path. Since \( z \) maps to \( S(s_j) \), so must \( z' \), and \( z' \) must therefore have at least one leaf descendant that maps to \( S(s_j) \). The node \( c \) is the only possible candidate for this leaf descendant. Thus, the path from \( z' \) to \( c \) cannot contain any transfer edges. This implies that \( x' \geq z \), which is a contradiction, since \( MRAR(a) = x \) and \( MRAR(c) \neq x \). A completely analogous argument also establishes that each node except \( y \) along the path from \( y \) to \( x \) must be a transfer node. Finally, consider the path \( c \) to \( x' \). As before, suppose there is an internal node, say \( z \), along this paths that is not a transfer node. Then \( z \) must be a speciation or duplication node. Let \( z' \) denote the child of \( z \) that is not on the \( c \) to \( x' \) path. Since \( z \) maps to
$S(s_j)$, so must $z'$, and $z'$ must therefore have at least one leaf descendant that maps to $S(s_j)$. $a$ and $b$ are the only two possible candidates for this leaf descendant. Note, however, that any path from $z'$ to $a$ or $b$ must go through the node $x$ (since $MRAR(a) = MRAR(b) = x$ and $MRAR(c) = x'$). Thus, the path from $z'$ to $a$ or $b$ travels through a transfer edge, implying that $z'$ cannot have either $a$ or $b$ as descendants, a contradiction. This proves that all internal nodes along the paths from $a$ to $y$ (not including $y$), $b$ to $y$ (not including $y$), $c$ to $x'$ (including $x'$), and $y$ to $x$ (including $x$, unless $x = y$) must be transfer nodes.

We will now show how to create an alternative DTL-scenario $\alpha'$ with smaller reconciliation cost than $\alpha$, leading to a contradiction. We update the mappings of all internal nodes along the path from $a$ to $x$ (including $y$) to be the mapping of $a$, all nodes along the path from $b$ to $y$ (not including $y$) to be the mapping of $b$, and all nodes along the path from $c$ to $x'$ (including $x'$) to be the mapping $c$. The resulting DTL-scenario remains valid, and only introduces one additional transfer node, $y$, and no additional losses. This is because all existing transfer nodes on the paths remain valid transfer nodes, and changing the mapping of the two $MRAR$ nodes does not lead to any increase in the number of losses (only the recipient node for the transfer event changes). Since this update decreases the number of losses by at least 6, the new DTL-scenario $\alpha'$ must have a reconciliation cost that is lower than the original by at least $\alpha$ by $6 \times P_{loss} - 1 \times P_\Theta = 2$. A contradiction.

The next lemma places a lower bound on the reconciliation cost of any optimal binary resolution $G^B$ of $G^N$.

**Lemma 10.** For any canonical optimal binary resolution $G^B$ of $G^N$ and a canonical MPR of $G^B$ with $S$, if the nodes $g_i$ and $g'_i$ and $g''_i$, for each $i \in \{1, \ldots, n\}$, map to exactly $k$
distinct subtrees $S(s_j)$, for $1 \leq j \leq m$, then the reconciliation cost of $G^B$ with $S$ is at least $k + 48m - 12n$.

Proof. From Lemma 6(1) we know that each of the subtrees $g_i$ and $g'_i$ and $g''_i$ has $c_i - 1$ transfer nodes. This contributes a total of $3 \times (3m - n)$ transfer edges. Similarly, from Lemma 2.3.5, we know that all nodes, labeled $x_{i,j}$, for any $i \in \{1, \ldots, n\}$ that map to subtrees $S(s_j)$ other than the $k$ chosen ones, must have a distinct MRAR. This contributes another $(3m - 3k)$ transfer edges. Also, from Lemma 6(1), it follows that all of the nodes of $G^B$ that map to the $k$ chosen $S(s_j)$’s, must have at least one MRAR, giving a total of $k$ additional transfers. The total reconciliation cost due to these transfers is $4 \times 3(3m - n) + 4(3m - 3k) + 4k$, which is $48m - 12n - 8k$. To complete the proof it suffices to show that the remainder of the reconciliation cost is at least $9k$.

Specifically, we will show that, for each of the $k$ chosen subtrees $S(s_j)$, the nodes of $G^B$ that map to $S(s_j)$ contribute an average additional cost of at least $9$ through either losses, duplications, or uncounted transfers. Note that the nodes $g'(i)$ and $g''(i)$ may each prevent a single loss event. We will initially ignore the presence of the $g'(i)$’s and $g''(i)$’s when counting losses for any given $S(s_j)$, but we will reduce the total number of losses obtained from our analysis by $2n$ later.

We first consider those $S(s_j)$ that have a mapping from one or more $g'_i$ or $g''_i$, but not from $g(i)$, and calculate the minimum additional cost induced. Let $S(s_j)$ be a subtree that has mappings from one or more $g'_i$ or $g''_i$, but not from $g(i)$. We distinguish 3 cases, depending on whether there are one, two, three distinct MRAR’s for the three blue nodes, denoted $a$, $b$, and $c$.  

\[ 35 \]
Case 1: If \(a, b,\) and \(c\) share the same MRAR, then this MRAR node must map to \(s_j\) and must induce 9 losses along the paths from the MRAR to \(a, b,\) and \(c\) (since there are no \(g_i\)'s and we ignore \(g'(i)'s\) and \(g''(i)'s\) when counting losses).

Case 2: If \(a, b,\) and \(c\) have two distinct MRAR's then two of the blue nodes, say \(a\) and \(b\) must share an MRAR, denoted \(x\). The paths from \(x\) to \(a\) and \(b\) must thus induce 6 losses (since there are no \(g_i\)'s and we ignore \(g'(i)'s\) and \(g''(i)'s\) when counting losses). Also, since we have only counted one MRAR (transfer event) per \(S_j\) in the analysis above, there is one additional MRAR in this case, giving an additional cost of 4 for its transfer event. The total additional cost in this case is thus 10, which is greater than 9.

Case 3: If \(a, b,\) and \(c\) have three distinct MRAR's then we consider two further cases: In the first case, suppose that one of the \(g'_i\)'s or \(g''_i\)'s that map to \(S(s_j)\) have an MRAR that is different than the three MRAR's for \(a, b,\) and \(c\). This means that there are at least 4 distinct MRAR's that map to \(S(s_j)\), only one of which has been counted before. This yields an additional cost of 12 for the remaining three transfers, and we again have a cost of at least 9. In the second case, there are only three MRAR's for \(a, b,\) and \(c,\) and the \(g'_i\)'s and \(g''_i\)'s. There must thus be shared MRAR, denoted \(x\) for one of the blue nodes, say \(a,\) and a \(g'_i\) or \(g''_i\). The path from \(x\) to \(a\) must induce at least one loss (since there are no \(g_i\)'s). Thus, in this case we have two uncounted MRAR's (transfers) and at least one additional loss, yielding an additional cost of at least 9.

Thus, the nodes of \(G^B\) that map to an \(S(s_j)\) that has a mapping from one or more \(g'_i\) or \(g''_i,\) but not from \(g(i),\) contribute at least an additional cost of 9.

We now consider all other \(S(s_j),\) i.e. all \(S(s_j)\)'s that have mappings from one or more \(g_i\)'s. Observe that for each \(g_i\) that maps to \(S_j,\) the nodes of \(G^B\) mapping to \(S(s_j)\) must either induce an additional duplication event or an additional transfer event. This contributes a cost
of at least 2 for each \( g_i \), thus contributing at least \( 2n \) overall. Let \( S(s_j) \) be a subtree that has mappings from at least one \( g(i) \). The computation of contributed loss costs due to \( S(s_j) \) is analogous to that shown above (cases 1, 2, and 3, with only minor variation) and again shows that the nodes of \( G^B \) that map to an \( S(s_j) \) that has a mapping from at least one \( g(i) \), contribute at least an additional cost of 9.

The total additional cost over all the \( k \) \( S(s_j) \)'s is thus at least \( 9k \), plus at least \( 2n \) for the duplications or additional transfers caused by the \( g_i \)'s, and minus at most \( 2n \) for the losses prevented by the \( g_i' \)s and \( g_i'' \)s, i.e., \( 9k \). This completes the proof.

The following lemma establishes the reverse direction of Claim 2.

**Lemma 11.** If there exists an optimal binary resolution of \( G^N \) such that its MPR with \( S \) has reconciliation cost at most \( k + 48m - 12n \), then there exists a solution of size at most \( k \) for the M3SC instance \( \phi \).

**Proof.** Consider an optimal binary resolution \( G^B \) such that its MPR with \( S \) has reconciliation cost at most \( k + 48m - 12n \). We will assume that both \( G^B \) and its MPR are canonical. (If not, we can use the efficient constructive procedure from the proof of Lemma 2.3.5 to create a canonical resolution and a canonical MPR with the same reconciliation cost.) We can obtain a solution for the M3SC instance as follows: Choose the set \( A_j \) to be in the set cover, for \( j \in \{1, \ldots, m\} \), if and only if the subtree \( S(s_j) \) has a mapping from at least one of the \( g_i \)'s, \( g_i' \)'s, or \( g_i'' \)'s, for \( i \in \{1, \ldots, n\} \).

We first show that this yields a valid set cover. From Lemma 6(2) it follows that \( g_i, g_i' \), or \( g_i'' \), for any given \( i \in \{1, \ldots, n\} \), can only map to a subtree \( S(s_j) \), for \( j \in \{1, \ldots, m\} \) that contains leaves with labels of the form \( x_{i_1} \ldots, \) i.e., at least one leaf in the subtree \( G^B(g_i) \),
$G^B(g'_i)$, or $G^B(g''_i)$ must map to that $S(s_j)$. The subtree $S(s_j)$ contains leaves with labels of the form $x_{i,.}$, if and only if the set $A_j$ in the M3SC instance $\phi$ contains element $u_i$. Finally, since $g_i$, $g'_i$, and $g''_i$, for each $i \in \{1, \ldots, n\}$ must map to an $S(s_j)$, for some $j \in \{1, \ldots, m\}$, it follows that the chosen $A_j$’s would cover all the elements $u_1, u_2, \ldots, u_n$.

We now show that the size of the resulting solution for the M3SC instance $\phi$ has size at most $k$. Suppose, for contradiction, that the size is $k'$, where $k' > k$. This means that there must be $k'$ subtrees $S(s_j)$, where $j \in \{1, \ldots, m\}$, that receive mappings from at least one of the $g_i$’s, $g'_i$’s, or $g''_i$’s, for $i \in \{1, \ldots, n\}$. However, from Lemma 10, we know that the MPR of $G^B$ with $S$ must then have a cost of at least $k' + 48m - 12n$, which is strictly greater than $k + 48m - 12n$. A contradiction. Thus, there must be at most $k$ subtrees $S(s_j)$, where $j \in \{1, \ldots, m\}$, that receive mappings from at least one of the $g_i$’s, $g'_i$’s, or $g''_i$’s, for $i \in \{1, \ldots, n\}$, completing the proof.

### 2.4 Extension to dated DTL reconciliation

An alternative model of DTL reconciliation has been proposed when the internal nodes of the species tree can be fully ordered in time [19]. We refer to this model as the *Dated-DTL* reconciliation model. Dated-DTL reconciliation makes use of the total order on the species nodes to ensure that the reconstructed optimal reconciliation is time-consistent. A key feature of this model is that it subdivides the species tree into different time slices [19] and then restricts transfer events to only occur within the same time slice.

We show how to assign divergence times to each node of the species tree. Observe that all subtrees $S(s_i)$, for each $i \in \{1 \ldots m\}$, have identical structure. All nodes at the same level in each $S(s_i)$ are assigned the same divergence time across all the subtrees. The *start* node
is assigned to be at the same level as the other leaves of $S$. The rest of the nodes in $S$ may be assigned arbitrary divergence times respecting the topology of $S$. Under this divergence time assignment, it can be shown that there exists an optimal resolution of the gene tree for which an MPR exists that only invokes transfer events that respect the timing constraints of the dated species tree as required by the dated-DTL reconciliation model. This implies that, for our gadget, any optimal resolution of the gene tree under the undated DTL reconciliation model has the same minimum reconciliation cost as the dated-DTL reconciliation model.

**Theorem 2.4.1.** The OGTR problem under the dated-DTL reconciliation model is NP-hard.

**Proof.** Consider the DTL-scenario $\alpha$ described in Section 4.3.3 to prove the forward direction of the proof. Note that all transfer events invoked by $\alpha$ occur within the same time-slice of the dated species tree described above, as required by the dated-DTL reconciliation model. Thus, even for the dated case, any MPR has cost at most $k + 48m - 12n$. Moreover, since the reconciliation cost under dated-DTL reconciliation cannot be smaller than that under DTL reconciliation, Claim 2 must also apply under dated-DTL reconciliation. This completes the proof. \(\square\)

### 2.5 Conclusion

In this work, we have shown that the OGTR problem, i.e., the problem of reconciling non-binary gene trees with binary species trees under the DTL reconciliation model, is NP-hard. Our reduction applies to both the undated and dated formulations of DTL-reconciliation and, furthermore, shows that the problem is NP-hard even for a biologically meaningful event cost assignment of 1, 2, and 4 for losses, duplications, and transfers, respectively. The uncertainty
about its complexity has prevented the development of algorithms for the OGTR problem. This work will spur the development of effective exact, approximate, and heuristic algorithms for this problem, making it possible to apply the powerful DTL reconciliation framework to non-binary gene trees.
Figure 2: **Resolution of $G^N$ into $G^B$.** (a) The structure of the backbone of the gene tree $G^B$. (b) Structure of the subtree $h_{r(j)}$ for any $j \in \{1, \ldots, k\}$. (c) The two possible structures of the subtree with root $B$ in $h_{r(j)}$. For any $j \in \{1, \ldots, k - 1\}$, this subtree is as shown at the top of part (c) while, for $j = k$, it is as shown at the bottom and includes all the “remaining” $3m - 3k$ blue nodes. (d) The resolution of the $g_i$’s, $g''_i$’s, $g'''_i$’s. In the figure, $u_a$, $u_b$, and $u_c$ represent the three elements in $A_{r(j)}$, with $u = f(a, r(j))$, $w = f(b, r(j))$, and $z = f(c, r(j))$. In part (d), if the covering subset of element $u_i$ is $A_p$, then $v$ represents $f(i, p)$. The labels inside the blue boxes represent blue nodes.
Figure 3: **Mapping of subtree** $G^B(h_{r(j)})$ **to** $S(s_{r(j)})$. As the figure shows, nodes $x$, $y$, and $z$ are duplication nodes that each invoke one loss. All the other nodes of $G^B(h_{r(j)})$ are speciation nodes without any losses.

Figure 4: **DTL reconciliation and OGTR problem.** Part (a) shows a non-binary gene tree $G^N$ and binary species tree $S$. Part (b) shows a DTL reconciliation between a possible binary resolution $G^B$ of $G^N$ and species tree $S$. The dotted arcs show the mapping $\mathcal{M}$ (with the leaf mapping being specified by the leaf labels on the gene tree), and the label at each internal node of $G^B$ specifies the type of event represented by that node. This reconciliation invokes two transfer events.
Chapter 3

Exact Algorithms for Duplication-Transfer-Loss Reconciliation

with Non-Binary Gene Trees

3.1 Introduction

Duplication-Transfer-Loss (DTL) reconciliation is a powerful, well-known technique for studying gene family evolution in microbial species. Microbial gene families evolve primarily through gene duplication, gene loss, and horizontal gene transfer, and DTL reconciliation can infer these evolutionary events through the systematic comparison and reconciliation of gene trees and species trees. Specifically, given a gene tree and a species tree, DTL reconciliation shows the evolution of the gene tree inside the species tree, and explicitly infers duplication, transfer, and loss events. Accurate inference of these evolutionary events has many uses in biology, including inference of orthologs, paralogs and xenologs [39, 68], reconstruction of ancestral gene content [12, 17], and accurate gene tree and species tree construction [5, 9, 21, 56, 68]. The DTL reconciliation problem has therefore been widely studied, e.g., [2, 3, 14–19, 25, 42, 49, 51, 59, 63, 67].
DTL reconciliation is generally formulated as a parsimony problem, where each evolutionary event is assigned a cost and the goal is to find a reconciliation with minimum total cost. The resulting optimization problem is called the DTL-reconciliation problem. DTL-reconciliations can sometimes be time-inconsistent in the sense that the inferred transfers may induce contradictory constraints on the dates for the internal nodes of the species tree. The problem of finding an optimal time-consistent reconciliation is known to be NP-hard [54, 67]. Thus, in practice, the goal is often to find an optimal (not necessarily time-consistent) DTL-reconciliation [2, 3, 17, 49, 67], and this problem can be solved in $O(mn)$ time [2], where $m$ and $n$ denote the number of nodes in the gene tree and species tree, respectively. Interestingly, the problem of finding an optimal time-consistent reconciliation becomes efficiently solvable [19, 48] in $O(mn^2)$ time if the species tree is fully dated. Thus, the two efficiently solvable formulations, dated and undated, are the two standard formulations of DTL-reconciliation.

Both formulations of the DTL-reconciliation problem assume that the input gene tree and species tree are binary. However, gene trees are frequently non-binary. This happens whenever there is insufficient information in the underlying gene sequences to fully resolve gene tree topologies. In such cases, all poorly supported edges in the reconstructed gene trees are collapsed, resulting in non-binary gene trees. Since gene family sequence alignments are often short and have limited information content, non-binary gene trees arise very frequently in practice [61]. When the input consists of a non-binary gene tree, the reconciliation problem seeks a binary resolution of the gene tree that minimizes the reconciliation cost. Many efficient algorithms have been developed for reconciling non-binary gene trees in the context of the simpler Duplication-Loss (DL) reconciliation model [10, 21, 47, 70], with the most efficient of these algorithms having an optimal $O(m + n)$ time complexity [70]. However, the corresponding
problem for DTL reconciliation has recently been shown to be NP-hard [42], and, to the best of our knowledge, no algorithms, heuristic or otherwise, currently exist for DTL reconciliation with non-binary gene trees.\(^1\) As a result, DTL reconciliation is currently inapplicable to non-binary gene trees, significantly reducing its utility in practice.

Our Contribution. In this work, we present the first, exact algorithms for DTL reconciliation with non-binary gene trees. Crucially, our algorithms also make it possible to distinguish between those aspects of the reconciliation that are highly supported based on all optimal (i.e., minimum cost) resolutions of the gene tree from those that are not. This makes it possible to not only apply DTL-reconciliation to non-binary gene trees, but to also negate the impact of gene tree uncertainty by distinguishing evolutionary inferences that have high support across all optimal resolutions of the given non-binary gene tree from those evolutionary inferences that have low support across the optimal resolutions. Even though our algorithms have exponential time complexity in the worst case, we show that they can be applied efficiently in most cases and can be used to analyze even large gene trees and species trees. Specifically, our contributions are as follows:

1. We show that the DTL-reconciliation problem for non-binary gene trees is fixed-parameter tractable (FPT) in the maximum degree of the gene tree. Our FPT algorithm runs in \(O(2^{k \log_2 2k} \cdot l \cdot n + mn)\) time for undated DTL-reconciliation, where \(m\) denotes the size of the gene tree, \(n\) the size of the species tree, \(k\) the maximum number of children for any node in the gene tree, and \(l\) the total number of non-binary nodes, and can be easily extended to dated DTL-reconciliation with only a slight increase

\(^1\)While some of the existing software packages for DTL-reconciliation do allow for the use of non-binary gene trees, e.g., CoRe-PA [51] and NOTUNG [63], they either assume that the gene tree is actually non-binary (i.e., do not try to resolve it) or just resolve the gene tree to minimize the simpler duplication-loss reconciliation cost (i.e., do not consider transfer events).
in time complexity. Since the time complexity is exponential only in the maximum degree and not in the number of non-binary nodes, this FPT algorithm is applicable to a large fraction of non-binary gene trees that arise in practice, even for large gene families.

2. We present an algorithm to track and enumerate all optimal binary resolutions of an unresolved input gene tree. As we show later, unresolved gene trees often have a very large number of optimal resolutions, and enumeration of optimal resolutions is therefore necessary for properly handling gene tree uncertainty. The enumeration algorithm accounts for the fact that the same resolution may have many different most parsimonious reconciliations, and also makes use of a special optimization to improve efficiency.

3. We implemented our algorithms for undated DTL-reconciliation and applied them to a large empirical data set of over 4700 gene families from 100 broadly sampled species to study the impact of gene tree uncertainty on DTL-reconciliation and to demonstrate the applicability and utility of our algorithms. We observed that the vast majority of the gene trees became non-binary when poorly supported edges were collapsed, that a large fraction of the non-binary gene trees had small maximum degree, and that the non-binary gene trees generally had a very large number of optimal reconciliations. Our FPT and enumeration algorithms could both quickly reconcile all gene trees with \( k \leq 8 \), which constituted the majority of the gene trees in the data set.

4. We study the impact of gene tree uncertainty on the inference of gene family evolution. We observed that even though unresolved gene trees often have a very large number of optimal binary resolutions, these optimal resolutions tend to be significantly more
similar to one another than to randomly selected binary resolutions. This result is important because it shows that a significant amount of new phylogenetic information can be extracted even when there is phylogenetic uncertainty by optimally resolving unresolved gene trees by DTL reconciliation and considering all optimal resolutions. We also directly measured the impact of uncertainty due to multiple optimal resolutions on the robustness of the inferred DTL reconciliation and observed that the vast majority of the nodes in the input gene trees are assigned a consistent (single) event and consistent (single) mapping to the species tree across all optimal resolutions. This implies that many aspects of gene family evolution can be confidently inferred despite the presence of multiple optimal resolutions.

The new techniques and algorithms introduced in this chapter make it possible to not only apply DTL-reconciliation to non-binary gene trees but also to systematically calculate and negate the impact of gene tree uncertainty on reconciliation accuracy and will help biologists avoid incorrect evolutionary inferences caused by gene tree uncertainty.

We develop our algorithms in the context of the undated DTL reconciliation problem. Extension to dated DTL reconciliation is straight-forward and is discussed in Sections 3.5. The next section introduces basic definitions and preliminaries. The FPT algorithm is presented in Section 3.3, the enumeration algorithm in Section 3.4, and experimental results in Section 5.4. Concluding remarks appear in Section 5.5.
3.2 Definitions and Preliminaries

We follow some basic definitions and notation from chapter 2. Given a tree \( T \), we denote its node, edge, and leaf sets by \( V(T) \), \( E(T) \), and \( Le(T) \) respectively. If \( T \) is rooted, the root node of \( T \) is denoted by \( rt(T) \), the parent of a node \( v \in V(T) \) by \( pa_T(v) \), its set of children by \( Ch_T(v) \), and the (maximal) subtree of \( T \) rooted at \( v \) by \( T(v) \). The set of internal nodes of \( T \), denoted \( I(T) \), is defined to be \( V(T) \setminus Le(T) \). We define \( \leq_T \) to be the partial order on \( V(T) \) where \( x \leq_T y \) if \( y \) is a node on the path between \( rt(T) \) and \( x \). The partial order \( \geq_T \) is defined analogously, i.e., \( x \geq_T y \) if \( x \) is a node on the path between \( rt(T) \) and \( y \). We say that \( y \) is an ancestor of \( x \), or that \( x \) is a descendant of \( y \), if \( x \leq_T y \) (note that, under this definition, every node is a descendant as well as ancestor of itself). We say that \( x \) and \( y \) are incomparable if neither \( x \leq_T y \) nor \( y \leq_T x \). Given a non-empty subset \( L \subseteq Le(T) \), we denote by \( lca_T(L) \) the last common ancestor (LCA) of all the leaves in \( L \) in tree \( T \). Given \( x, y \in V(T) \), \( x \rightarrow_T y \) denotes the unique path from \( x \) to \( y \) in \( T \). We denote by \( d_T(x, y) \) the number of edges on the path \( x \rightarrow_T y \); note that if \( x = y \) then \( d_T(x, y) = 0 \). Throughout this work, the term tree refers to rooted trees. A tree is binary if all of its internal nodes have exactly two children, and non-binary otherwise. An internal edge is an edge whose end points are both internal nodes in the tree. An internal edge \( (x, pa_T(x)) \) in tree \( T \) can be contracted by removing \( (x, pa_T(x)) \) and creating new edges joining \( pa_T(x) \) with \( Ch_T(x) \), thereby yielding a new tree distinct from \( T \). We say that a tree \( T' \) is a binary resolution of \( T \) if \( T' \) is binary and \( T \) can be obtained from \( T' \) by contracting some (zero or more) internal edges. We denote by \( BR(T) \) the set of all binary resolutions of a non-binary tree \( T \). Given any node \( x \) from \( T \), we define the \textit{out-degree} of \( x \) to be the total number of children of \( x \).
A species tree is a tree that depicts the evolutionary relationships of a set of species. Given a gene family from a set of species, a gene tree is a tree that depicts the evolutionary relationships among the sequences encoding only that gene family in the given set of species. Thus, the nodes in a gene tree represent genes. Gene trees may be either binary or non-binary while the species tree is always assumed to be binary. Throughout this work, we denote the gene tree and species tree under consideration by \( G \) and \( S \), respectively. If \( G \) is restricted to be binary we refer to it as \( G^B \) and as \( G^N \) if it is restricted to be non-binary. We assume that each leaf of the gene tree is labeled with the species from which that gene was sampled. This labeling defines a leaf-mapping \( L_{G,S}: \text{Le}(G) \rightarrow \text{Le}(S) \) that maps a leaf node \( g \in \text{Le}(G) \) to that unique leaf node \( s \in \text{Le}(S) \) that has the same label as \( g \). Note that gene trees may have more than one gene sampled from the same species, and that the species tree must contain all species represented in the gene tree.

### 3.2.1 Reconciliation and DTL-scenarios

A binary gene tree can be reconciled with a species tree by mapping the gene tree into the species tree. Next, we define what constitutes a valid reconciliation; specifically, we define a Duplication-Transfer-Loss scenario (DTL-scenario) \[2,67\] for \( G^B \) and \( S \) that characterizes the mappings of \( G^B \) into \( S \) that constitute a biologically valid reconciliation. Essentially, DTL-scenarios map each gene tree node to a unique species tree node and designate each gene tree node as representing either a speciation, duplication, or transfer event.

**Definition 3.2.1** (DTL-scenario). A DTL-scenario for \( G^B \) and \( S \) is a seven-tuple \( \langle L, M, \Sigma, \Delta, \Theta, \Xi, \tau \rangle \), where \( L: \text{Le}(G^B) \rightarrow \text{Le}(S) \) represents the leaf-mapping from \( G^B \) to \( S \), \( M: V(G^B) \rightarrow V(S) \) maps each node of \( G^B \) to a node of \( S \), the sets \( \Sigma, \Delta, \) and \( \Theta \) partition
$I(G^B)$ into speciation, duplication, and transfer nodes respectively, $\Xi$ is a subset of gene tree edges that represent transfer edges, and $\tau: \Theta \to V(S)$ specifies the recipient species for each transfer event, subject to the following constraints:

1. If $g \in Le(G^B)$, then $M(g) = L(g)$.

2. If $g \in I(G^B)$ and $g'$ and $g''$ denote the children of $g$, then,

   (a) $M(g) \not<_S M(g')$ and $M(g) \not<_S M(g'')$.

   (b) At least one of $M(g')$ and $M(g'')$ is a descendant of $M(g)$.

3. Given any edge $(g, g') \in E(G^B)$, $(g, g') \in \Xi$ if and only if $M(g)$ and $M(g')$ are incomparable.

4. If $g \in I(G^B)$ and $g'$ and $g''$ denote the children of $g$, then,

   (a) $g \in \Sigma$ only if $M(g) = \text{lca}(M(g'), M(g''))$ and $M(g')$ and $M(g'')$ are incomparable,

   (b) $g \in \Delta$ only if $M(g) \geq_S \text{lca}(M(g'), M(g''))$.

   (c) $g \in \Theta$ if and only if either $(g, g') \in \Xi$ or $(g, g'') \in \Xi$.

   (d) If $g \in \Theta$ and $(g, g') \in \Xi$, then $M(g)$ and $\tau(g)$ must be incomparable, and $M(g')$

       must be a descendant of $\tau(g)$, i.e., $M(g') \leq_S \tau(g)$.

DTL-scenarios correspond naturally to reconciliations and it is straightforward to infer
the reconciliation of $G^B$ and $S$ implied by any DTL-scenario. Figure 5 shows an example
of a DTL-scenario. For a discussion on some of the limitations of this DTL reconciliation
framework, we refer the reader to [65, 67]. Given a DTL-scenario $\alpha$, one can directly count
the minimum number of gene losses, $\text{Loss}_\alpha$, in the corresponding reconciliation [2].
Definition 3.2.2 (Losses). Given a DTL-scenario $\alpha = \langle \mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau \rangle$ for $G$ and $S$, let $g \in V(G)$ and $\{g’, g''\} = Ch(g)$. The number of losses $Loss_\alpha(g)$ at node $g$, is defined to be:

- $|d_S(M(g), M(g')) - 1| + |d_S(M(g), M(g'')) - 1|$, if $g \in \Sigma$,
- $d_S(M(g), M(g')) + d_S(M(g), M(g''))$, if $g \in \Delta$, and
- $d_S(M(g), M(g'')) + d_S(\tau(g), M(g'))$ if $(g, g') \in \Xi$.

We define the total number of losses in the reconciliation corresponding to the DTL-scenario $\alpha$ to be $Loss_\alpha = \sum_{g \in V(G)} Loss_\alpha(g)$.

Let $P_\Delta$, $P_\Theta$, and $P_{loss}$ denote the non-negative costs associated with duplication, transfer, and loss events, respectively. The reconciliation cost of a DTL-scenario is defined as follows.

Definition 3.2.3 (Reconciliation cost). Given a DTL-scenario $\alpha = \langle \mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau \rangle$ for $G^B$ and $S$, the reconciliation cost associated with $\alpha$ is given by $R_\alpha = P_\Delta \cdot |\Delta| + P_\Theta \cdot |\Theta| + P_{loss} \cdot Loss_\alpha$.

A most parsimonious reconciliation is one that has minimum reconciliation cost.

Definition 3.2.4 (MPR). Given $G^B$ and $S$, along with $P_\Delta$, $P_\Theta$, and $P_{loss}$, a most parsimonious reconciliation (MPR) for $G^B$ and $S$ is a DTL-scenario with minimum reconciliation cost.

3.2.2 Optimal gene tree resolution

Non-binary gene trees cannot be directly reconciled against a species tree. Thus, given a non-binary gene tree $G^N$, the problem is to find a binary resolution $G^B$ of $G^N$ such that an MPR of $G^B$ with $S$ has smallest reconciliation cost. An example of a non-binary gene tree and a binary resolution is shown in Figure 5. This yields the following problem.
Figure 5: **DTL reconciliation and OGTR problem.** Part (a) shows a non-binary gene tree $G^N$ with two unresolved nodes and a binary species tree $S$. Part (b) shows a DTL reconciliation between a possible binary resolution $G^B$ of $G^N$ and species tree $S$. The dotted arcs show the mapping $\mathcal{M}$ (with the leaf mapping being specified by the leaf labels on the gene tree), and the label at each internal node of $G^B$ specifies the type of event represented by that node. This reconciliation invokes two transfer events and one duplication event.

**Problem 4 (OGTR).** Given $G^N$ and $S$, along with $P_\Delta$, $P_\Theta$, and $P_{loss}$, the Optimal Gene Tree Resolution (OGTR) problem is to find a binary resolution $G^B$ of $G^N$ such that an MPR of $G^B$ and $S$ has the smallest reconciliation cost among all $G^B \in \mathcal{BR}(G^N)$.

Since there may be more than one optimal binary resolution of $G^N$, a more useful formulation of the problem is to find all optimal resolutions of $G^N$.

**Problem 5 (OGTR-All).** Given $G^N$ and $S$, along with $P_\Delta$, $P_\Theta$, and $P_{loss}$, the All Optimal Gene Tree Resolutions (OGTR-All) problem is to compute the set $\mathcal{OR}(G^N)$ of all optimal binary resolutions of $G^N$ such that, for any $G^B \in \mathcal{OR}(G^N)$, an MPR of $G^B$ and $S$ has the smallest reconciliation cost among all gene trees in $\mathcal{BR}(G^N)$. 

52
3.3 Fixed Parameter Algorithm for OGTR

Note that the number of resolutions of an unresolved gene tree is exponential in both the number of non-binary nodes and their maximum out-degree. Thus, any algorithm that is exponential only in the maximum out-degree is a tremendous improvement over the naïve algorithm for the OGTR problem. We present an FPT algorithm for the OGTR problem that is exponential only in the maximum out-degree of the gene tree. Our algorithm takes as input a non-binary gene tree $G^N$, species tree $S$, and event costs $P_\Delta$, $P_\Theta$, and $P_{\text{loss}}$, and outputs an optimal binary resolution $G^B$ of $G^N$ along with the optimal reconciliation cost.

A key challenge with designing such an FPT algorithm for DTL reconciliation of non-binary gene trees is that different unresolved (non-binary) nodes in the gene tree can not be resolved optimally locally, without consideration of how other unresolved nodes are resolved. Thus, a straight-forward solution to the OGTR problem would involve considering all possible resolutions of the given gene tree, reconciling each resolution with the species tree, and choosing the resolution that gives the minimum reconciliation cost. As mentioned in the paragraph above, such a solution would have complexity exponential in both the number of non-binary nodes and their maximum out-degree.

Our algorithm overcomes this difficulty by using a dynamic programming approach built upon the classical dynamic programming algorithm used for DTL reconciliation of binary gene trees [2, 67]. By utilizing dynamic programming, we are able to efficiently account for the interdependence between different resolutions of the various unresolved nodes, without having to explicitly consider all possible resolutions of the gene tree.
Classical dynamic programming algorithm for binary gene trees. Given any \( g \in I(G) \) and \( s \in V(S) \), let \( c_{\Sigma}(g, s) \) denote the cost of an optimal reconciliation of \( G(g) \) with \( S \) such that \( g \) maps to \( s \) and \( g \in \Sigma \). The terms \( c_{\Delta}(g, s) \) and \( c_{\Theta}(g, s) \) are defined similarly for \( g \in \Delta \) and \( g \in \Theta \), respectively. Given any \( g \in V(G) \) and \( s \in V(S) \), define \( c(g, s) \) to be the cost of an optimal reconciliation of \( G(g) \) with \( S \) such that \( g \) maps to \( s \). Note that, for \( g \in I(G) \), \( c(g, s) = \min\{c_{\Sigma}(g, s), c_{\Delta}(g, s), c_{\Theta}(g, s)\} \). The dynamic programming algorithm for binary gene trees performs a nested post-order traversal of the gene tree and species tree, computing the value \( c(g, s) \) for each \( g \in I(G) \) and \( s \in V(S) \). To initialize the dynamic programming table, we set, for each \( g \in \text{Le}(G) \), \( c(g, s) = 0 \) if \( s = \mathcal{M}(g) \), and \( c(g, s) = \infty \) otherwise. Once all the \( c(\cdot, \cdot) \) values are computed, the minimum reconciliation of \( G \) and \( S \) is simply \( \min_{s \in V(S)} c(\text{rt}(G), s) \).

The values of \( c_{\Sigma}(g, s) \), \( c_{\Delta}(g, s) \), and \( c_{\Theta}(g, s) \), for any \( g \in I(G) \) and \( s \in V(S) \), can be computed based on the previously computed values of \( c(\cdot, \cdot) \). Further details on how these values are computed appear in [2] as well as in the pseudocode below. Note that, to help compute \( c_{\Sigma}(g, s) \), \( c_{\Delta}(g, s) \), and \( c_{\Theta}(g, s) \), we also define, for each \( g \in V(G) \) and \( s \in V(S) \),

\[
\text{in}(g, s) = \min_{x \in V(S)} \{P_{\text{loss}} \cdot d_S(s, x) + c(g, x)\}, \text{ and} \\
\text{out}(g, s) = \min_{x \in V(S) \text{ incomparable to } s} c(g, x).
\]

Extension to non-binary gene trees. To allow for non-binary gene trees, we extend this dynamic programming approach as follows: During the nested post-order traversal of the gene tree and species tree, if the current gene tree node, \( g \), is binary the algorithm proceeds as before. But if \( g \) is non-binary then the algorithm considers all possible resolutions of \( g \) to compute the
minimum value of $c(g, s)$, for each $s \in V(S)$, over all resolutions of $g$. Specifically, let $BR_G(g)$ denote the set of all binary resolutions of the (partial) subtree of $G$ formed by $g$ and its children. Consider any $H \in BR_G(g)$. Note that (i) $H$ is rooted at $g$, (ii) the leaf set of $H$ is $Ch_G(g)$, and (iii) $I(H) \setminus \{g\}$ consists of new nodes that do not occur in $G$. Since $H$ is binary and the values $c(\cdot, \cdot)$ have already been computed for all its leaf nodes, we can use the dynamic programming algorithm for binary gene trees to compute the value of $c(g, s)$, for each $s \in V(S)$, for the given $H$. We denote this value by $c^H(g, s)$. The algorithm considers all possible binary resolutions $H \in BR_G(g)$, computing the values $c^H(g, s)$, for each $s \in V(S)$. The final value of $c(g, s)$, for each $s \in V(S)$ is then set to:

$$c(g, s) = \min_{H \in BR_G(g)} c^H(g, s).$$

To keep track of which binary resolution of non-binary node $g$ yields the final value of $c(g, s)$, we also record a best binary resolution $H$ for each $s \in V(S)$. Once all $c(g, \cdot)$ values are computed, the dynamic programming algorithm proceeds as usual with its post order traversal of $G$. A more precise description of the algorithm follows:

**Input:** $OGTR-FPT(G, S, L)$

1: for each $g \in V(G)$ and $s \in V(S)$ do
2: \hspace{1em} Initialize $c(g, s)$, $c_{\Sigma}(g, s)$, $c_{\Delta}(g, s)$, and $c_{\Theta}(g, s)$ to $\infty$.
3: for each $g \in Le(G)$ do
4: \hspace{1em} Initialize $c(g, \mathcal{L}(g))$ to $0$.
5: for each $g \in I(G)$ in post-order do
6: \hspace{1em} if $g$ is a binary node then
7: \hspace{2em} for each $s \in V(S)$ in post-order do
8: \hspace{3em} Let $\{g', g''\} = Ch_G(g)$.
9: \hspace{2em} if $s \in Le(S)$ then
10: \hspace{3em} $c_{\Sigma}(g, s) = \infty$.
11: \hspace{3em} $c_{\Delta}(g, s) = P_\Delta + c(g', s) + c(g'', s)$.
12: \hspace{3em} if $s \neq rt(S)$, then $c_{\Theta}(g, s) = P_\Theta + \min\{in(g', s) + out(g'', s), in(g'', s) + out(g', s)\}$.
13: \hspace{3em} $c(g, s) = \min\{c_{\Sigma}(g, s), c_{\Delta}(g, s), c_{\Theta}(g, s)\}$.  

55
else
Let \( \{s', s''\} = Ch_{S}(s) \).
\[
\Sigma(g, s) = \min \{in(g', s') + in(g'', s''), \, in(g'', s') + in(g', s'')\}.
\]
\[
c_{\Delta}(g, s) = P_{\Delta} + \min \{in(g', s') + in(g'', s)\}.
\]
If \( s \neq rt(S) \), then \( c_{\Theta}(g, s) = P_{\Theta} + \min \{in(g', s) + out(g'', s), \, in(g'', s) + out(g', s)\} \).
\[
c(g, s) = \min \{c_{\Sigma}(g, s), c_{\Delta}(g, s), c_{\Theta}(g, s)\}.
\]
if \( g \) is a non-binary node then

for each \( H \in BR_{G}(g) \) do

for each \( h \in Le(H) \) do

for each \( s \in V(S) \) do

\[\begin{align*}
\text{Initialize } c_{H}(h, s) \text{ to } c(h, s). \\
\text{for each } h \in I(H) \text{ in post-order do} \\
\text{for each } s \in V(S) \text{ in post-order do} \\
\text{Let } \{h', h''\} = Ch_{H}(h). \\
\text{if } s \in Le(S) \text{ then} \\
c_{H}^{s}(h, s) = \infty. \\
c_{H}^{h}(h, s) = P_{\Delta} + c_{H}(h', s) + c(h'', s). \\
\text{if } s \neq rt(S), \text{ then} \\
\begin{align*}
&c_{H}(h, s) = P_{\Theta} + \min \{in(h', s) + out(h'', s), \, in(h', s) + out(h'', s)\}. \\
&c_{H}(h, s) = c_{\Sigma}(h, s), c_{\Delta}(h, s), c_{\Theta}(h, s) \}.
\end{align*}
\end{align*} \]
\]
else

Let \( \{s', s''\} = Ch_{S}(s) \).
\[
\Sigma(h, s) = \min \{in(h', s') + in(h'', s''), \, in(h'', s') + in(h', s'')\}. \\
\]
\[
\Delta(h, s) = P_{\Delta} + \min \{in(h', s) + in(h'', s)\}. \\
\]
If \( s \neq rt(S) \), then \( c_{\Theta}(h, s) = P_{\Theta} + \min \{in(h', s) + out(h'', s), \, in(h', s) + out(h'', s)\}. \\
\]
\[
c_{H}(h, s) = \min \{c_{\Sigma}(h, s), c_{\Delta}(h, s), c_{\Theta}(h, s) \}.
\]
for each \( s \in V(S) \) in post-order do

if \( c_{H}(g, s) < c(g, s) \) then
\[
c(g, s) = c_{H}(g, s).
\]

Return \( \min_{s \in V(S)} c(rt(G), s) \).

In the pseudocode above, steps 1 through 19 implement the dynamic programming algorithm for binary gene trees, while steps 20 through 41 implement our algorithmic extension to non-binary gene trees as described previously.
Note that, while the above pseudocode only outputs the minimum reconciliation cost, it can be easily adapted to record the optimal $H$s in the dynamic programming table and output an optimal binary resolution of $G$ by backtracking, without any change in its time complexity. Note also, that the time complexity of this pseudocode can be reduced by a factor of $n$ by computing and maintaining the values of $in(\cdot, \cdot)$ and $out(\cdot, \cdot)$ efficiently within the nested post-order traversals, as shown in [2]. These additional steps are omitted here in the interest of clarity.

Let $m$ and $n$ denote the number of leaves in $G$ and $S$, respectively. Let $k$ denote the maximum out-degree of any node in $G$, and $l$ denote the total number of non-binary nodes in $V(G)$. Next, we show that Algorithm \textit{OGTR-FPT} correctly solves the OGTR problem, and that it can be implemented to run in time $O(2^{k \log_2 2k} \cdot l \cdot n + mn)$.

\textbf{Theorem 3.3.1.} The OGTR problem can be solved in $O(2^{k \log_2 2k} \cdot l \cdot n + mn)$ time.

\textit{Proof.} We first prove the correctness of Algorithm \textit{OGTR-FPT} and then analyze its time complexity.

\textit{Correctness:} It suffices to show that the value $c(g, s)$, for each $g \in V(G)$ and $s \in V(S)$, is computed correctly. Note that, for each $g \in Le(G)$, the value $c(g, s)$, for any $s \in V(S)$, is correctly initialized. These values form the base case of our inductive argument. Suppose $g \in I(G)$. We will assume (our inductive hypothesis), that all values $c(h, x)$, for each $h \in V(G(g)) \setminus \{g\}$ and $x \in V(S)$, have been correctly computed. There are now two cases, depending on whether $g$ is a binary node or non-binary node.
Case 1: $g$ is binary. Let $\{g', g''\} = \text{Ch}_G(g)$. By the inductive hypothesis, $c(g', x)$ and $c(g'', x)$ have been computed correctly for each $x \in V(S)$. Observe that the values $c_\Sigma(g, s)$, $c_\Delta(g, s)$, and $c_\Theta(g, s)$ are computed in accordance with Definition 5.2.1 (in steps 10 through 12 if $s$ is a leaf node, and in steps 16 through 18 otherwise), based on the values $c(\cdot, \cdot)$ correctly computed previously. Thus, the value of $c(g, s)$ is computed correctly as well (steps 13 and 19).

Case 2: $g$ is non-binary. Let $g_1, \ldots, g_p$ denote the $p$ children of $g$. By the inductive hypothesis, the value $c(g_i, s)$ has been computed correctly for each $i \in \{1, \ldots, p\}$ and $s \in V(S)$. The value $c(g, s)$ is defined to be the minimum reconciliation cost of any binary resolution of $G(g)$, under the constraint that $g$ maps to $s$. Algorithm $OGTR-FPT$ explicitly considers every possible resolution of node $g$ by considering all trees $H \in BR_G(g)$ (step 21). Since $H$ is binary and its leaves $(g_1, \ldots, g_p)$ already have the correctly computed values of $c(\cdot, \cdot)$, the algorithm computes the cost $c^H(h, s)$, for each newly created binary node $h$ (including node $g$) and each $s \in V(S)$, using the same steps proved correct in Case 1 above (steps 22 through 38). The final value of $c(g, s)$, for each $s \in V(S)$ is then set to $c(g, s) = \min_{H \in BR_G(g)} c^H(g, s)$ (“for” loop of step 39), as required by the definition of $c(g, s)$.

Induction completes the proof.

**Complexity:** It has previously been shown [2] that the values $\text{in}(\cdot, \cdot)$ and $\text{out}(\cdot, \cdot)$ can be computed in $O(1)$ time per value by computing them incrementally as part of the nested postorder traversal. Details on their computation are omitted (for clarity) from the pseudocode of Algorithm $OGTR-FPT$ above, and we refer the reader to [2] for details. For our analysis, we will assume that any particular $\text{in}(\cdot, \cdot)$ and $\text{out}(\cdot, \cdot)$ value is computable in $O(1)$ time.

Steps 1 through 4 of the algorithm are related to initialization and take $O(mn)$ time. Consider the block of Steps 8 through 19 that handles binary nodes. This block is executed $O(mn)$
times by the ‘for’ loops of Steps 5 and 7. Each step within this block requires $O(1)$ time and
the total time complexity of Steps 5 through 19 is thus $O(mn)$.

Now, consider the block of Steps 22 through 41 that handles non-binary nodes. This block
is executed a total of $O(l \times |\mathcal{BR}_G(g)|)$ times through the ‘for’ loops of Steps 5 and 21. For any
non-binary node $g$, its number of children is bounded above by $k$. The total number of trees in
$\mathcal{BR}_G(g)$, for any $g$, is thus $O((2k - 3)!!)$, which is $O(2^k \cdot (k - 1)!))$. Consider the sequence of
Steps 22 through 24. A single execution of this sequence requires $O(|V(H)| \cdot n)$ time, which
is $O(kn)$. Similarly, consider the sequence of Steps 25 through 38. A single execution of this
sequence also requires $O(kn)$ time. Finally, consider the sequence of Steps 39 through 41.
A single execution of this sequence requires $O(m)$ time. Thus, the total time complexity of
Steps 22 through 41 (together with the ‘for’ loops of Steps 5 and 21) is $O(2^k \cdot k! \cdot l \cdot n)$, which
is $O(2^{k \log_2 2k} \cdot l \cdot n)$.

The overall time complexity of the algorithm is thus $O(2^{k \log_2 2k} \cdot l \cdot n + mn)$.

3.4 Enumeration Algorithm for OGTR-All

Ordinarily, enumeration of optimal solutions in a dynamic programming framework is a
straightforward task, easily accomplished by repeated backtracking through the dynamic pro-
gramming table. In the case of the OGTR-All problem, however, this task is complicated by the
fact that the same optimal resolution can have many different optimal DTL-reconciliations [3],
which means that the same resolution can be counted and enumerated multiple times as part
of different reconciliations. As a result, enumeration of optimal resolutions, and also uniform
random sampling, becomes more challenging.
Furthermore, since the number of optimal resolutions can be very large (exponential in the number of non-binary nodes and their maximum out-degree), the worst case time complexity of any algorithm for the OGTR-All problem must also be exponential in both the number of non-binary nodes and their maximum out-degree.

**Additional definitions and notation.** Given a non-binary gene tree $G$, binary species tree $S$, and $g \in V(G)$, let $N(G(g))$ be the set of all non-binary nodes in the subtree $G(g)$. Note that $l = |N(G)|$. We will assume that, given any non-binary node $h \in N(G)$, the possible resolutions of $h$ have each been assigned a resolution number. Specifically, let $r_i(h)$ denote the $i^{th}$ resolution of $h$.

Recall that $\mathcal{OR}(G)$ denotes the set of all optimal resolutions of $G$ (with respect to $S$ and the given event costs). Each binary resolution $G_i \in \mathcal{OR}(G)$ is associated with a resolution vector $\rho_i$ that specifies the resolution numbers for all nodes in $N(G)$, corresponding to the specific resolution $G_i$. Specifically, given $G_i \in \mathcal{OR}(G)$, suppose $h_1, \ldots, h_{|N(G)|}$ denote the elements of $N(G)$ (i.e., all non-binary nodes in subtree $G$) ordered according to a post-order traversal of $G$, then $\rho_i = \langle r_{b(1)}(h_1), r_{b(2)}(h_2), \ldots, r_{b(|N(G)|)}(h_{|N(G)|}) \rangle$, where $b(1), \ldots, b(|N(G)|)$ are the specific resolution numbers for the nodes $h_1, \ldots, h_{|N(G)|}$, respectively, corresponding to $G_i$. We define the set of all optimal resolution vectors of $G$, denoted $\mathcal{ORV}(G)$, to be the set $\{\rho_i : G_i \in \mathcal{OR}(G)\}$. We further extend the $\mathcal{OR}(G)$ notation and define $\mathcal{OR}(G(g), s)$ to be the set of all optimal resolutions of $G(g)$ under the constraint that $g$ maps to $s \in V(S)$. The notation $\mathcal{ORV}(G)$ is extended analogously to $\mathcal{ORV}(G(g), s)$. Note that if $G(g)$ does not contain any non-binary nodes, i.e., $N(G(g)) = \emptyset$, then both $\mathcal{OR}(G(g), s)$ and $\mathcal{ORV}(G(g), s)$ are empty sets, for any $s \in V(S)$. 
Given $g \in V(G)$, $s \in V(S)$, and $H \in \mathcal{BR}(G)$, we previously defined $c^H(g, s)$ to be the value $c(g, s)$ computed on the specific binary resolution $H$ of $G$. We extend this notation as follows: Given any $g \in V(G)$, $g' \in V(G(g))$, and a resolution vector $\rho$ corresponding to a specific binary resolution of the subtree $G(g)$, we define $c^\rho(g', s)$ to be the value $c(g', s)$ computed on the specific binary resolution of $G(g)$ corresponding to $\rho$.

Given any $g \in V(G)$, if $g$ has $p$ children (where $2 \leq p \leq k$) denoted $g_1, g_2, \ldots, g_p$, then we say that the vector $\langle s_1, s_2, \ldots, s_p \rangle$ is feasible under the constraint that $g$ maps to node $s \in V(S)$, if there exists an optimal resolution $H \in \mathcal{BR}(G(g))$ and a most parsimonious reconciliation (MPR) of $H$ with $S$ in which, under the constraint that $g$ maps to $s$, $g_i$ maps to $s_i$ for each $i \in \{1, \ldots, p\}$. We define the feasible set of $g$ and $s$, denoted $\mathcal{F}(g, s)$, to be the set of all vectors $\langle s_1, s_2, \ldots, s_p \rangle$ that are feasible under the constraint that $g$ maps to node $s$. Observe that, if $g$ is non-binary, then each vector $x$ in the set $\mathcal{F}(g, s)$ corresponds to one or more resolutions of $g$ from $\mathcal{ORV}(G(g), s)$. We denote by $\mathcal{R}_x^\mathcal{F}(g, s)$ the set of all those resolutions of $g$ seen in $\mathcal{ORV}(G(g), s)$ that correspond to vector $x \in \mathcal{F}(g, s)$.

Finally, given two vectors $x = \langle m_1, m_2, \ldots, m_p \rangle$ and $y = \langle n_1, n_2, \ldots, n_q \rangle$, we define $x \oplus y$ to be the concatenated vector $\langle m_1, m_2, \ldots, m_p, n_1, n_2, \ldots, n_q \rangle$. Given two sets $X = \{x_1, x_2, \ldots, x_a\}$ and $Y = \{y_1, y_2, \ldots, y_b\}$, where each $x_i$, for $1 \leq i \leq a$, and $y_j$, for $1 \leq j \leq b$, is a vector, we define $X \otimes Y$ to be the set $\{x_i \oplus y_j : 1 \leq i \leq a \text{ and } 1 \leq j \leq b\}$.

Note that the set $\mathcal{ORV}(G(g), s)$ consists of exactly all those resolutions of $G(g)$ whose MPR with $S$ has cost $c(g, s)$ when $g$ is constrained to map to $s$. Our goal is to compute the set $\mathcal{OR}(G)$, or equivalently, the set $\mathcal{ORV}(G)$. Our enumeration algorithm uses the same nested post-order traversal as the FPT algorithm, described previously, to compute the set $\mathcal{ORV}(G(g), s)$ alongside the value of $c(g, s)$, for each $g \in V(G)$ and $s \in V(S)$. 
The first two of the next four lemmas show how the set $\mathcal{ORV}(G(g), s)$ can be computed using the previously computed $\mathcal{ORV}(\cdot, \cdot)$ sets.

**Lemma 12.** Given any binary node $g \in V(G)$, if $g_1$ and $g_2$ denote its two children and $s_1, s_2 \in V(S)$ refer to the mappings of $g_1$ and $g_2$, respectively, then

$$\mathcal{ORV}(G(g), s) = \bigcup_{(s_1, s_2) \in \mathcal{F}(g, s)} \mathcal{ORV}(G(g_1), s_1) \otimes \mathcal{ORV}(G(g_2), s_2).$$

**Proof.** We will first show that if $\rho \in \mathcal{ORV}(G(g), s)$ then $\rho \in \bigcup_{(s_1, s_2) \in \mathcal{F}(g, s)} \mathcal{ORV}(G(g_1), s_1) \otimes \mathcal{ORV}(G(g_2), s_2)$, and then the converse.

Let $N(G(g_1)) = \{h_1^1, h_2^1, \ldots, h_{|N(G(g_1))|}^1\}$ and $N(G(g_2)) = \{h_1^2, h_2^2, \ldots, h_{|N(G(g_2))|}^2\}$. Note that $N(G(g)) = N(G(g_1)) \cup N(G(g_2)) = \{h_1^1, h_2^1, \ldots, h_{|N(G(g_1))|}^1, h_1^2, \ldots, h_{|N(G(g_2))|}^2\}$.

Consider any $\rho \in \mathcal{ORV}(G(g), s)$, and let $H$ denote the particular binary resolution of $G(g)$ corresponding to $\rho$. Let $\rho = \langle r_{a(1)}(h_1^1), r_{a(2)}(h_2^1), \ldots, r_{a(|N(G(g_1))|)}(h_{|N(G(g_1))|}^1), r_{b(1)}(h_1^2), r_{b(2)}(h_2^2), \ldots, r_{b(|N(G(g_2))|)}(h_{|N(G(g_2))|}^2) \rangle$, where $a(1), \ldots, a(|N(G(g_1))|)$ are the specific resolution numbers for the non-binary nodes $h_1^1, \ldots, h_{|N(G(g_1))|}^1$, respectively, corresponding to $H(g_1)$, and $b(1), \ldots, b(|N(G(g_2))|)$ are the specific resolution numbers for the non-binary nodes $h_1^2, \ldots, h_{|N(G(g_2))|}^2$, respectively, corresponding to $H(g_2)$. Finally, let $\rho_1$ and $\rho_2$ be the resolution vectors for $H(g_1)$ and $H(g_2)$, respectively; i.e.,

$$\rho_1 = \langle r_{a(1)}(h_1^1), r_{a(2)}(h_1^1), \ldots, r_{a(|N(G(g_1))|)}(h_{|N(G(g_1))|}^1) \rangle$$

and

$$\rho_2 = \langle r_{b(1)}(h_1^2), r_{b(2)}(h_2^2), \ldots, r_{b(|N(G(g_2))|)}(h_{|N(G(g_2))|}^2) \rangle.$$

Consider any MPR of $H$ with $S$ under the constraint that $g$ (the root of $H$) maps to $s$. Let this MPR be denoted by $\alpha$. Under $\alpha$, suppose $g_1$ maps to node $s_1 \in V(S)$ and $g_2$ maps to node $s_2 \in V(S)$. Then, by definition, $\langle s_1, s_2 \rangle \in \mathcal{F}(g, s)$. Moreover, we must have $\rho_1 \in$
Given any non-binary node \( ORV(G(g_1), s_1) \) and \( ORV(G(g_2), s_2) \), otherwise \( \alpha \) would not be an MPR. This proves that \( \rho \in \bigcup_{\langle s_1, s_2 \rangle \in F(g,s)} ORV(G(g_1), s_1) \otimes ORV(G(g_2), s_2) \).

To prove the converse, consider any \( \langle s_1, s_2 \rangle \in F(g,s) \). By the definition of \( F(G,S) \), there exists some \( \rho \in ORV(G(g), s) \) such that there exists an MPR \( \alpha \) of the corresponding resolution, under the constraint that \( g \) maps to \( s \), in which \( g_1 \) maps to \( s_1 \) and \( g_2 \) maps to \( s_2 \). As shown in the first part of this proof, we must have \( \rho_1 \in ORV(G(g_1), s_1) \) and \( \rho_2 \in ORV(G(g_2), s_2) \). Now, consider any \( \rho'_1 \in ORV(G(g_1), s_1) \) and \( \rho'_2 \in ORV(G(g_2), s_2) \), and let \( \rho' = \rho'_1 \oplus \rho'_2 \). Observe that, since \( c^p\rho'_1(g_1, s_1) = c^p\rho_1(g_1, s_1) \) and \( c^p\rho'_2(g_2, s_2) = c^p\rho_2(g_2, s_2) \), we must have \( c^p\rho'(g, s) = c^p\rho(g, s) \). This implies that \( \rho' \in ORV(G(g), s) \). Thus, we have shown that, given any \( \rho' = \rho'_1 \oplus \rho'_2 \) such that \( \rho'_1 \in ORV(G(g_1), s_1) \) and \( \rho'_2 \in ORV(G(g_2), s_2) \), where \( \langle s_1, s_2 \rangle \in F(g,s) \), we must have \( \rho' \in ORV(G(g), s) \), proving the converse.

Lemma 13. Given any non-binary node \( g \in V(G) \), if \( g_1, g_2, \ldots, g_p \) denote its \( p \) children and \( s_1, s_2, \ldots, s_p \in V(S) \) refer to the mappings of \( g_1, g_2, \ldots, g_p \), respectively, then

\[
ORV(G(g), s) = \bigcup_{\langle s_1, s_2, \ldots, s_p \rangle \in F(g,s)} \bigcup_{r \in R^F_{\langle s_1, s_2, \ldots, s_p \rangle}(g,s)} \bigotimes_{i=1}^{p} ORV(G(g_i), s_i) \oplus \bigotimes_{i=1}^{p} ORV(G(g_p), s_p) \oplus r.
\]

Proof. This proof follows along the lines of the proof for Lemma 12 above. We will first show that if \( \rho \in ORV(G(g), s) \) then \( \rho \in \bigcup_{\langle s_1, s_2, \ldots, s_p \rangle \in F(g,s)} \bigcup_{r \in R^F_{\langle s_1, s_2, \ldots, s_p \rangle}(g,s)} ORV(G(g_1), s_1) \otimes \bigotimes_{i=1}^{p} ORV(G(g_i), s_i) \otimes r \) and then the converse.

Let \( N(G(g_i)) = \{ h^1_{i_1}, h^2_{i_2}, \ldots, h^i_{N(G(g_i))} \} \). Note that \( N(G(g)) = \bigcup_{2 \leq i \leq p} N(G(g_i)) \cup \{ g \} = \{ h^1_{1}, h^2_{1}, \ldots, h^1_{N(G(g_1))}, h^p_{1}, h^2_{p}, \ldots, h^p_{N(G(g_p))}, g \} \). Consider any
\( \rho \in \mathcal{ORV}(G(g), s) \), and let \( H \) denote the particular binary resolution of \( G(g) \) corresponding to \( \rho \). Let \( \rho_i = \{r_{a_i(1)}(h_1^i), r_{a_i(2)}(h_2^i), \ldots, r_{a_i(\vert N(G(g_i)) \vert)}(h_{\vert N(G(g_i)) \vert}^i)\} \) be the resolution vector for \( H(g_i) \), where \( a_i(1), \ldots, a_i(\vert N(G(g_i)) \vert) \) are the specific resolution numbers for the non-binary nodes \( h_1^i, \ldots, h_{\vert N(G(g_i)) \vert}^i \), respectively, corresponding to \( H(g_i) \). Then, by definition, \( \alpha \) proves that \( \rho \) is the resolution number for (the non-binary) node \( g \).

Consider any MPR of \( H \) with \( S \) under the constraint that \( g \) (the root of \( H \)) maps to \( s \). Let this MPR be denoted by \( \alpha \). Under \( \alpha \), suppose \( g_i \) maps to node \( s_i \in V(S) \), for \( 1 \leq i \leq p \). Then, by definition, \( (s_1, s_2, \ldots, s_p) \in \mathcal{F}(g, s) \) and \( r \in \mathcal{R}_\mathcal{F}(s_1, s_2, \ldots, s_p)(g, s) \). Moreover, for each \( i \in \{1, \ldots, p\} \), we must have \( \rho_i \in \mathcal{ORV}(G(g_i), s_i) \), otherwise \( \alpha \) would not be an MPR. This proves that \( \rho \in \bigcup_{(s_1, s_2, \ldots, s_p) \in \mathcal{F}(g, s)} \mathcal{R}_\mathcal{F}(s_1, s_2, \ldots, s_p)(g, s) \). Moreover, for each \( i \in \{1, \ldots, p\} \), let \( \rho_i \) be an MPR of \( G(g_i), s_i \). Now, consider any \( \rho_i' \in \mathcal{ORV}(G(g_i), s_i) \), where \( 1 \leq i \leq p \), and any \( r \in \mathcal{R}_\mathcal{F}(s_1, s_2, \ldots, s_p)(g, s) \), and let \( \rho' = (\bigoplus_{1 \leq i \leq p} \rho_i') \oplus r \). Observe that, since \( c^\rho_i(g_i, s_i) = c^{\rho_i}(g_i, s_i) \), and since \( r \) must be a resolution of \( g \) seen in \( \mathcal{ORV}(G(g), s) \), we must have \( c^\rho(g, s) = c^\rho(g, s) \). This implies that \( \rho' \in \mathcal{ORV}(G(g), s) \). Thus, we have shown that, given any \( \rho' = (\bigoplus_{1 \leq i \leq p} \rho_i') \oplus r \) such that \( \rho_i' \in \mathcal{ORV}(G(g_i), s_i) \) for each \( i \in \{1, \ldots, p\} \), \( (s_1, s_2, \ldots, s_p) \in \mathcal{F}(g, s) \), and \( r \in \mathcal{R}_\mathcal{F}(s_1, s_2, \ldots, s_p)(g, s) \), we must have \( \rho' \in \mathcal{ORV}(G(g), s) \), proving the converse. \( \square \)
The next lemma shows how to compute $\text{ORV}(G)$ based on the previously computed sets $\text{ORV}(G, \cdot)$.

**Lemma 14.** Let $A$ be the set $\{s \in V(S) : c(\text{rt}(G), s) = \min_{s' \in V(S)} c(\text{rt}(G), s')\}$. Then, $\text{ORV}(G) = \bigcup_{s \in A} \text{ORV}(G, s)$.

**Proof.** Consider any $\rho \in \text{ORV}(G)$. Then, $\rho$ is the resolution vector for an optimal resolution, say $H$, of $G$. Consider any MPR $\alpha$ of $H$ with $S$. The root of $H$ must map to some specific node $s' \in V(S)$ according to $\alpha$. Thus, since $H$ is an optimal resolution and $\alpha$ an MPR, we must have $s' \in A$ and, therefore, $\rho \in \bigcup_{s \in A} \text{ORV}(G, s)$.

Conversely, consider any $\rho \in \bigcup_{s \in A} \text{ORV}(G, s)$ and let $H$ denote the resolution of $G$ corresponding to $\rho$. There must be an MPR $\alpha$ of $H$ with $S$ that maps $\text{rt}(H)$ to some node $s' \in A$. Thus, by definition of $A$, $c^\rho(\text{rt}(G), s') = \min_{s \in V(S)} c(\text{rt}(G), s)$. Consequently, $\rho \in \text{ORV}(G)$, completing the proof. \qed

The previous three lemmas are sufficient to derive the enumeration algorithm. The next lemma, shows how to economize the computation so that the set $\text{ORV}(G(g), s)$ need not be computed for all $g \in V(G)$.

**Lemma 15.** Given any binary node $g \in V(G)$, let $g', g'' \in V(G)$ be such that $g = \text{lca}_G(\{g', g''\})$, $g', g'' \neq g$, and $N(G(g)) = N(G(g')) \cup N(G(g''))$. Under the constraint that $g$ maps to node $s \in V(S)$, let $X$ denote the set of all vectors $(s', s'')$ such that there exists an optimal resolution $H \in \text{BR}(G(g))$, and a most parsimonious reconciliation (MPR) of $H$ with $S$ in which $g'$ maps to $s'$ and $g''$ maps to $s''$. Then, $\text{ORV}(G(g), s) = \bigcup_{(s', s'') \in X} \text{ORV}(G(g'), s') \otimes \text{ORV}(G(g''), s'')$.  

65
Proof. The proof of this lemma is almost identical to the proof for Lemma 12. We will first show that if \( \rho \in \mathcal{ORV}(G(g), s) \) then \( \rho \in \bigcup_{(s', s'') \in X} \mathcal{ORV}(G(g'), s') \otimes \mathcal{ORV}(G(g''), s'') \), and then the converse.

Consider any \( \rho \in \mathcal{ORV}(G(g), s) \), and let \( H \) denote the particular binary resolution of \( G(g) \) corresponding to \( \rho \). Let \( \rho = \langle r_{a(1)}(h'_1), r_{a(2)}(h'_2), \ldots, r_{a(|N(G(g'))|)}(h'_{|N(G(g'))|}) \rangle, \)
\( r_{b(1)}(h''_1), r_{b(2)}(h''_2), \ldots, r_{b(|N(G(g''))|)}(h''_{|N(G(g''))|}) \rangle \), where \( a(1), \ldots, a(|N(G(g'))|) \) are the specific resolution numbers for the non-binary nodes \( h'_1, \ldots, h'_{|N(G(g'))|} \), respectively, corresponding to \( H(g') \), and \( b(1), \ldots, b(|N(G(g''))|) \) are the specific resolution numbers for the non-binary nodes \( h''_1, \ldots, h''_{|N(G(g''))|} \), respectively, corresponding to \( H(g'') \). Finally, let \( \rho' \) and \( \rho'' \) be the resolution vectors for \( H(g') \) and \( H(g'') \), respectively; i.e.,
\( \rho' = \langle r_{a(1)}(h'_1), r_{a(2)}(h'_2), \ldots, r_{a(|N(G(g'))|)}(h'_{|N(G(g'))|}) \rangle \) and \( \rho'' = \langle r_{b(1)}(h''_1), r_{b(2)}(h''_2), \ldots, r_{b(|N(G(g''))|)}(h''_{|N(G(g''))|}) \rangle \).

Consider any MPR of \( H \) with \( S \) under the constraint that \( g \) (the root of \( H \)) maps to \( s \). Let this MPR be denoted by \( \alpha \). Under \( \alpha \), suppose \( g' \) maps to node \( s' \in V(S) \) and \( g_2 \) maps to node \( s'' \in V(S) \). Then, by definition, \( (s', s'') \in X \). Moreover, we must have \( \rho' \in \mathcal{ORV}(G(g'), s') \) and \( \rho'' \in \mathcal{ORV}(G(g''), s'') \), otherwise \( \alpha \) would not be an MPR. This proves that \( \rho \in \bigcup_{(s', s'') \in X} \mathcal{ORV}(G(g'), s') \otimes \mathcal{ORV}(G(g''), s'') \).

To prove the converse, consider any \( (s', s'') \in X \). By the definition of \( X \), there exists some \( \rho \in \mathcal{ORV}(G(g), s) \) such that there exists an MPR \( \alpha \) of the corresponding resolution, under the constraint that \( g \) maps to \( s \), in which \( g' \) maps to \( s' \) and \( g'' \) maps to \( s'' \). As shown in the first part of this proof, we must have \( \rho' \in \mathcal{ORV}(G(g'), s') \) and \( \rho'' \in \mathcal{ORV}(G(g''), s'') \). Now, consider any \( \nu' \in \mathcal{ORV}(G(g'), s') \) and \( \nu'' \in \mathcal{ORV}(G(g''), s'') \), and let \( \nu = \nu' \oplus \nu'' \). Observe that, since \( c'(g', s') = c'(g', s'), \ c''(g'', s'') = c''(g'', s'') \), we must have \( c'(g, s) = c'(g, s) \).
This implies that $\nu \in ORV(G(g), s)$. Thus, we have shown that, given any $\nu = \nu' \oplus \nu''$ such that $\nu' \in ORV(G(g'), s')$ and $\nu'' \in ORV(G(g''), s'')$, where $(s', s'') \in X$, we must have $\nu \in ORV(G(g), s)$, proving the converse.

The enumeration algorithm is based on Lemmas 12 through 15 and follows along the lines of Algorithm $OGTR-FPT$ described earlier. Essentially, in addition to computing the values $c(g, s)$, for each $g \in V(G)$ and $s \in V(S)$, as described in the Algorithm $OGTR-FPT$, the enumeration algorithm also computes the sets $ORV(G(g), s)$ based on Lemmas 12 through 15. A more precise description of the algorithm follows:

**Input:** $OGTR-Enumerate(G, S, \mathcal{L})$

1. for each $g \in V(G)$ and $s \in V(S)$ do
2. Initialize $c(g, s)$, to $\infty$.
3. Initialize $F(g, s)$ and $ORV(G(g), s)$ to $\emptyset$.
4. Initialize $ORV(G)$ to $\emptyset$.
5. for each $g \in Le(G)$ do
6. Initialize $c(g, \mathcal{L}(g))$ to 0.
7. for each $g \in I(G)$ in post-order do
8. if $g$ is a binary node then
9. Let $Ch_{G}(g) = \{g_{1}, g_{2}\}$.
10. for each $s \in V(S)$ in post-order do
11. Compute $c(g, s)$ as in Algorithm $OGTR-FPT$.
12. Compute $F(g, s)$.
13. Compute $ORV(G(g), s)$ according to the equation of Lemma 12.
14. if $g$ is a non-binary node then
15. Let $\{g_{1}, \ldots, g_{p}\} = Ch_{G}(g)$.
16. for each $s \in V(S)$ in post-order do
17. for each resolution $H \in BR_{G}(g)$ do
18. Compute $c^{H}(g, s)$ as in Algorithm $OGTR-FPT$.
19. if $c^{H}(g, s) \leq c(g, s)$ then
20. $c(g, s) = c^{H}(g, s)$.
21. Update $F(g, s)$.
22. Let $r$ be the resolution number corresponding to resolution $H$.
23. Set $ORV(G(g), s) = \bigcup_{(s_{1}, s_{2}, \ldots, s_{p}) \in F(g, s)} ORV(G(g_{1}), s_{1}) \otimes ORV(G(g_{2}), s_{2}) \otimes \ldots \otimes ORV(G(g_{p}), s_{p}) \otimes r$.
24. Let $A = \{s \in V(S) : c(rt(G), s) = \min_{s' \in V(S)} c(rt(G), s')\}$.
25. for each $s \in A$ do
26. Set $ORV(G) = \bigcup_{s \in A} ORV(G, s)$.
27. Return $ORV(G)$. 

67
For simplicity, the pseudocode above does not describe how to compute the sets \( \mathcal{F}(g, s) \) and does not make use of the optimization of Lemma 15. Next, we first show how to compute the sets \( \mathcal{F}(g, s) \) (Steps 27 and 21 from Algorithm \textit{OGTR-Enumerate}) and then show how Lemma 15 can be used to reduce computational requirements and speed up the algorithm.

**Computing \( \mathcal{F}(g, s) \).** For any given \( g \in I(G) \) and \( s \in V(S) \), the set \( \mathcal{F}(g, s) \) can be computed during the same nested post-order traversal (dynamic programming algorithm) used to compute the value \( c(g, s) \) (as in Algorithm \textit{OGTR-FPT}). If \( g \) is binary, as in Step 27 of Algorithm \textit{OGTR-FPT}, with \( g' \) and \( g'' \) denoting its two children, then \( \mathcal{F}(g, s) \) can be computed by keeping track of all mappings of \( g' \) (resp. \( g'' \)) that result in the values \( \text{in}(g', \cdot) \) and \( \text{out}(g', \cdot) \) (resp. \( \text{in}(g'', \cdot) \) and \( \text{out}(g'', \cdot) \)) used in the computation of \( c(g, s) \). For example, suppose we wish to compute \( \mathcal{F}(g, s) \) while computing \( c(g, s) \) in Step 19 of Algorithm \textit{OGTR-FPT}, and suppose that \( c_\Theta(g, s) = c_\Sigma(g, s) = c(g, s) \) while \( c_\Delta(g, s) > c(g, s) \). Furthermore, suppose that the value of \( c_\Sigma(g, s) \) is obtained from \( \text{in}(g', s') + \text{in}(g'', s'') \) (and not from the other choice) in Step 16, and that the value of \( c_\Theta(g, s) \) is obtained from \( \text{in}(g', s) + \text{out}(g'', s) \) (and not from the other choice) in Step 18 of Algorithm \textit{OGTR-FPT}. Now, let \( A \) be the set \( \{ x \in V(S(s')) : P_{\text{loss}} \cdot d_S(s', x) + c(g', x) = \text{in}(g', s') \} \), \( B \) be the set \( \{ x \in V(S(s'')) : P_{\text{loss}} \cdot d_S(s'', x) + c(g'', x) = \text{in}(g'', s'') \} \), \( C \) be the set \( \{ x \in V(S(s)) : P_{\text{loss}} \cdot d_S(s, x) + c(g', x) = \text{in}(g', s) \} \), and \( D \) be the set \( \{ x \in V(S) \text{ incomparable to } s : c(g'', x) = \text{out}(g'', s) \} \). Then, \( \mathcal{F}(g, s) = (A \times B) \cup (C \times D) \), where \( \times \) denotes cross product (and, to be consistent with the definition of \( \mathcal{F}(g, s) \), results in a vector).

The set \( \mathcal{F}(g, s) \) can be computed similarly if \( g \) is a non-binary node by leveraging the computation of \( c(g, s) \) in Steps 20 through 41 in Algorithm \textit{OGTR-FPT}. Let \( \mathcal{F}^H(g, s) \) denote the value of \( \mathcal{F}(g, s) \) computed for a particular resolution, \( H \in \mathcal{B} \mathcal{R}_G(g) \), of \( g \). As we consider...
all the different ways of resolving the node \( g \) (i.e., the different \( H \in BR_G(g) \)) in Step 21 of Algorithm \( OGTR-FPT \), we will keep track of all those \( H \) that yield the optimal cost, i.e., for which \( c^H(g,s) = c(g,s) \). Let \( A \) denote this set of optimal \( H \)'s. Observe that the set \( F(g,s) \) is then simply equal to \( \bigcup_{H \in A} F^H(g,s) \). Note that, if \( g_1, g_2, \ldots, g_p \) denote the \( p \) children of the non-binary node \( g \) in \( G \), then, given any \( H \in BR_G(g) \), not all of the nodes \( g_1, g_2, \ldots, g_p \) will be children of \( g \) in \( H \). Thus, to compute the set \( F^H(g,s) \), for any given \( H \in BR_G(g) \), one must store and propagate the information on optimal mappings of \( g_1, g_2, \ldots, g_p \) upwards during the nested post-order traversal of \( H \) and \( S \) (Steps 25 and 26 of Algorithm \( OGTR-FPT \)). This can be done along similar lines as for the case of binary \( g \), described above, and further details are left to the reader.

**Optimization using Lemma 15.** Given any \( g \in I(G) \) and \( s \in V(S) \), the computation of \( ORV(G(g),s) \) is one of the most computationally intensive steps of Algorithm \( OGTR-Enumerate \). Lemma 15 makes it possible to limit the nodes \( g \in I(G) \) for which these values must be computed. Any node \( g \in I(G) \) can be classified into one of three categories depending on the distribution of non-binary nodes in \( G(g) \): If \( g \) is non-binary, i.e., \( g \in N(G) \), then \( g \) belongs to category-1. If \( g \) is binary and there exists \( g' \prec_G g \) such that \( N(G(g)) = N(G(g')) \) then \( g \) belongs to category-2. Finally, all binary nodes that do not belong to category-2 are assigned to category-3. Note that category-3 consists precisely of all those binary nodes \( g \in I(G) \) for which there exist two distinct nodes \( g', g'' \prec_G g \) such that \( N(G(g)) = N(G(g')) \cup N(G(g'')) \) and neither \( N(G(g')) \) nor \( N(G(g'')) \) is an empty set. Lemma 15 makes it possible to skip the computation of \( ORV(G(g),s) \) for all category-2 nodes (except for the root node, if it belongs to category-2). If the total number of non-binary nodes is relatively low then category-2 comprises a large fraction of the nodes of \( I(G) \), and
Lemma 15 results in a noticeable speed-up. Algorithm $OGTR$-Enumerate can be easily extended to label each node $g \in I(G)$ with its category and then only compute $ORV(G(g), s)$, for each $s \in V(S)$, for category-1 and category-3 nodes (and also for $rt(G)$), as shown in Lemmas 13 and 15. Note that the set $X$, as defined in Lemma 15, can be computed similarly to how $F(g, s)$ is computed for non-binary nodes $g$, as described in the previous paragraph.

**Theorem 3.4.1.** Algorithm $OGTR$-Enumerate correctly solves the $OGTR$-All problem.

**Proof.** Algorithm $OGTR$-Enumerate computes the values of $c(g, s)$ as shown in Algorithm $OGTR$-FPT. Thus, by the proof of Theorem 3.3.1, all $c(g, s)$ values are computed correctly.

The sets $ORV(G(g), s)$, for each $g \in V(G)$ and $s \in V(S)$, are computed in accordance with Lemmas 12 and 13 in Steps 13 and 23. Finally, the set $ORV(G(g), s)$ is computed according to Lemma 14 in Steps 24 through 26. The correctness of Algorithm $OGTR$-Enumerate follows.

\[\square\]

**A note on time complexity.** Observe that the total number of binary resolutions of $G$ is $O(2^lk \log 2k)$. Thus, the $OGTR$-All problem can be trivially solved in time $O(2^{l \times k \log 2k} \cdot mn)$ by generating all possible binary resolutions of $G$ and computing their reconciliation costs. The worst case time complexity of Algorithm $OGTR$-Enumerate is actually even worse than the complexity of this brute-force solution, since the sizes of the sets $F(g, s)$ and $ORV(G(g), s)$, for a given $g \in V(G)$ and $s \in V(S)$ can be $O(n^k)$ and $O(2^{lk \log 2k})$, respectively, in the worst case. However, by utilizing the dynamic programming structure of the problem, our algorithm avoids considering many suboptimal resolutions and becomes dramatically more efficient than the brute-force algorithm in practice. In fact, in our experimental analysis we observed that the size of $F(g, s)$, for any $g \in V(G)$ and $s \in V(S)$, is usually very small and effectively
constant. Furthermore, we found that usually only a small fraction of the possible resolutions at each non-binary node are optimal. This explains why, despite the worse-than-brute-force worst-case time complexity, our enumeration algorithm is only slightly slower than the FPT algorithm in practice in most cases.

3.5 Extension to dated DTL Reconciliation

The FPT and enumeration algorithms described above for undated DTL reconciliation can be applied to dated DTL reconciliation as well. Dated DTL reconciliation assumes that the internal nodes of the species tree can be fully ordered in time [19], and uses the total order on the species nodes to ensure that the reconstructed optimal reconciliation is time-consistent. A key feature of this model is that it subdivides the species tree into time slices [19] and then restricts transfer events to occur within the same time slice. The dynamic programming algorithm for dated DTL reconciliation proceeds in the same manner as for the (undated) DTL reconciliation problem, with a nested post-order traversal of the gene tree and species tree, but requires \( O(mn^2) \) time due to the additional sub-division of the species tree edges into time-slices [19]. Our FPT can be directly adapted to dated DTL reconciliation by substituting the dynamic programming algorithm for binary DTL reconciliation with the dynamic programming algorithm for binary dated DTL reconciliation, with a corresponding factor of \( n \) increase in time complexity.

**Theorem 3.5.1.** The OGTR problem with dated DTL reconciliation can be solved in

\[
O(2^{k(\log_2 2k)} \cdot \ln^2 + mn^2) \text{ time.}
\]
Proof. This proof is along the same lines as the time complexity proof of Theorem 3.3.1. Details are omitted.

Likewise, our enumeration algorithm can also be directly adapted to dated DTL reconciliation with a corresponding increase in run time.

3.6 Experimental Evaluation

To assess the performance and impact of our algorithms in practice, we implemented the FPT and enumeration algorithms and applied them to a biological data set of over 4700 gene trees from a broadly sampled set of 100, predominantly prokaryotic, species [17]. This is one of the largest data sets ever to be analyzed using (binary) DTL reconciliation and we use it here to demonstrate the feasibility of applying our exact algorithms to large gene trees and species trees and to assess the impact of using unresolved gene trees for DTL reconciliation.

3.6.1 Description of the data set

Data set. The data set consists of 4736 maximum likelihood gene trees constructed using PhyML [26]. All trees are binary and unrooted and range in size (number of leaves) from a minimum of 3 to a maximum of 1007, with a mean size of 35.1. To create rooted gene trees, we rooted each tree optimally so as to minimize the DTL reconciliation cost of that rooted binary gene tree, i.e., we chose, among all possible rootings of an initial binary gene tree, one that minimizes the reconciliation cost with the species tree. We fixed these rootings for the remainder of the analysis. To create non-binary gene trees, we followed the standard phylogenetic practice of collapsing all branches with weak bootstrap support [22]. Specifically, we chose two bootstrap support cutoffs: 80% and 50%. A bootstrap cutoff of 80% is a commonly
Figure 6: **Experimental results.** (a) Number of gene trees (cumulative) plotted against their maximum out-degrees for the 80% and 50% cutoffs. (b) Number of gene trees (cumulative) plotted against the percentage of their internal nodes that are non-binary. (c) Average running time (in seconds, on a log scale) of the FPT and enumeration algorithms on gene trees. (d) Average reduction in reconciliation cost. (e) Number of optimal resolutions, on average, for the gene trees with maximum out-degrees 3 through 8, for 50% and 80% bootstrap cutoffs. (f) Percent increase in the number of internal nodes of the strict consensus trees of all optimal resolutions for the gene trees compared to the strict consensus for the original bootstrap replicates for the same gene trees.
used threshold for collapsing weak branches in phylogenetics, while the 50% value represents a more relaxed threshold where only branches with lower than 50% confidence are collapsed.

**Basic statistics.** Figure 6(a) shows the distribution of the maximum out-degrees (number of children) for all gene trees in the data set. As the figure shows, for the 80% and 50% cutoffs, only 336 and 919 gene trees, respectively, remain binary. The figure also shows that for the majority of the gene trees in the data set the maximum out-degree is 8 or smaller (65.03% and 53.99% for the 50% and 80% bootstrap cutoffs, respectively). These results suggest that our FPT and enumeration algorithms should be applicable to a large fraction of gene trees that arise in practice. The results also show, somewhat surprisingly, that many gene trees have very large degree, even for the more relaxed 50% cutoff. Indeed, the maximum observed out-degrees were 951 and 989 for the 50% and 80% cut-offs, respectively. In addition, as Figure 6(b) shows, the total fraction of unresolved nodes in each gene tree can vary widely across gene trees but is generally between 5% and 25%.

### 3.6.2 Scalability and runtime of the algorithms

We applied our FPT and enumeration algorithms to both the 80% bootstrap cutoff and 50% bootstrap cutoff gene trees and observed that all gene trees whose maximum out-degree was 8 or smaller could be reconciled efficiently. Thus, for either bootstrap cutoff value, both our algorithms could be applied to the majority of the gene trees in the data set. As Figure 6(c) shows, gene trees whose maximum out-degree was 6 or smaller could be reconciled virtually instantaneously using the FPT algorithm and in under a minute using the enumeration algorithm, while gene trees with maximum out-degree 8 required, on average, less than 12 minutes using the FPT algorithm and less than 40 minutes using the enumeration algorithm. We point
out that the size of the gene tree by itself does not have a significant impact on the running time of the FPT or enumeration algorithms (as also suggested by their time complexities); the total number of unresolved nodes and their out-degrees have a larger impact. Gene trees with out-degrees 9 or greater can also be handled by the FPT algorithm, but can require substantially longer run times. For the enumeration algorithm we found that memory becomes a bottleneck beyond out-degree 8. All our analyses were run using a single core on a 3.4 GHz machine with an Intel Quad core processor and 8 GB of RAM.

3.6.3 Experimental results

Impact on reconciliation cost.

We measured the impact of optimal resolution on DTL-reconciliation by reconciling the optimally resolved gene trees and comparing their reconciliation costs against those of the original binary gene trees. Following [5, 17], we used costs 1, 2, and 3 for losses, duplications, and transfers, respectively. As Figure 6(d) shows, the average reduction using the 80% (50%) bootstrap cutoff gene trees was 6.04% (4.9%) for the gene trees with maximum out-degree 3 and increased to 18.86% (15.7%) for the gene trees with maximum out-degree 8. This shows that the original reconciliation can be significantly altered during optimal resolution, especially as the maximum out-degree increases.

Number of optimal resolutions. We used the enumeration algorithm to compute all optimal resolutions for the 80% bootstrap cutoff and 50% bootstrap cutoff gene tree data sets. As Figure 6(e) shows, the number of optimal resolutions, on average, for the 80% (50%) cutoff gene trees varies from a low of 4.64 (3.63) for the gene trees with maximum out-degree 3 to a high of 630590 (553060) for the gene trees with maximum out-degree 8. It is worth noting
that several of the gene trees with out-degrees 7 or 8 had on the order of millions of optimal resolutions. Interestingly, as Figure 6(e) also suggests, we noticed that the number of optimal resolutions does not keep increasing exponentially with increasing out-degree.

### 3.6.4 Impact on inference of gene family evolution

We performed additional analyses on the generated sets of multiple optimal resolutions to study the impact of multiple optima on the inference of gene family evolutionary histories.

**Strict consensus of optimal resolutions.** A standard technique to account for differences in candidate phylogenies is to compute the strict consensus tree of all candidate topologies (e.g., bootstrap replicates) [50]. Each branch in the strict consensus tree is a phylogenetic relationship that is conflict-free (universally supported) across all candidate topologies. Thus, the more resolved the strict consensus tree the better. We computed, for all gene trees with maximum out-degree no more than 8, strict consensus trees of all optimal resolutions obtained using our enumeration algorithm and compared them against the original unresolved gene trees (80% and 50% bootstrap cutoff) used for the analysis.2 This is illustrated in Figure 7. The goal of this analysis is to determine if considering only the optimal resolutions yields more conflict-free phylogenetic information than in the original data set. As Figure 6(f) shows, when using 80% bootstrap cutoffs, there is, on average, a 21% increase in the number of conflict-free phylogenetic relationships, increasing from an average of 10% for out-degree 3 gene trees to about 47% for out-degree 8 gene tree. We also observed about a 10% average increase even with the 50% bootstrap gene trees. The increase in conflict-free phylogenetic information is smaller for the 50% bootstrap gene trees because those gene trees are already more resolved than the

---

2For gene trees that had more than 20,000 optimal resolutions, we chose 20,000 samples uniformly at random for computing the strict consensus.
Figure 7: **Strict consensus analysis.** This figure depicts the steps in our strict consensus analysis and illustrates how the strict consensus of all optimal resolutions may be more resolved (i.e., have more internal nodes) than the input non-binary gene tree. The first step in the analysis is to compute all optimal binary resolutions of the input non-binary gene tree with respect to the input species tree $S$. The second step is to compute the strict consensus of all the optimal binary resolutions. The nodes shaded blue on the strict consensus tree correspond to the nodes that were originally non-binary in the input gene tree. As the figure shows, some of the non-binary nodes in the input gene tree may resolve as binary nodes in the strict consensus tree, while some others may remain non-binary but with reduced out-degree.
corresponding 80% cutoff gene trees, so there is less to resolve. This result is important because it shows that a significant amount of new phylogenetic information can be extracted even when there is phylogenetic uncertainty by optimally resolving unresolved gene trees by DTL reconciliation and considering all possible optimal resolutions.

Figure 8: **Stability of mapping and event assignments across optimal resolutions.** The plot in part (a) shows the fraction of binary and non-binary nodes from the input non-binary gene trees that are assigned the same mapping to the species tree at least a certain fraction of times across a randomly chosen sample of 100 optimal binary resolutions of that input gene tree. Plot (b) shows the fraction of binary and non-binary nodes from the input non-binary gene tree that are assigned the same event type at least a certain fraction of times across a randomly chosen sample of 100 optimal binary resolutions of that input gene tree. Note that the analysis also accounts for multiple optimal reconciliations, and the results shown here consider 100 optimal reconciliations, sampled uniformly at random, for each optimal binary resolution.

**Impact on reconciliation.** To assess the impact of the multiple optimal resolutions on the ability to perform meaningful DTL reconciliation, we computed DTL reconciliations for the optimal
resolutions of each non-binary gene tree with maximum out-degree between 3 and 8 (inclusive) and measured how often the gene tree nodes in the original (non-binary) gene tree are assigned the same mapping across all the optimal resolutions and the same event across all the optimal resolutions. This is illustrated in Figure S1 in the supplement. For this analysis, we used the non-binary gene trees obtained using the 80% bootstrap cutoff threshold. Since the number of optimal resolutions can be extremely large for many gene trees, we used sampling for computational efficiency; specifically, for each non-binary gene tree with more than 100 optimal resolutions, we sampled 100 optimal resolutions uniformly at random for the analysis. Furthermore, since there can be multiple optimal reconciliations for any given optimal resolution [3], we computed 100 optimal DTL-reconciliations, sampled uniformly at random from the space of all optimal reconciliations, for each optimal resolution (using the algorithm described in [3]). Thus, for each input non-binary gene tree, we generated up to 10,000 DTL reconciliations across its optimal resolutions.

We observed that event and mapping assignments are highly conserved across the optimal resolutions for each gene tree. The input non-binary gene trees have a total of 12,124 internal nodes, of which 8,647 are binary and 3,477 non-binary. For the gene nodes that were originally binary in the input gene trees, 88% have a fully conserved event assignment across all 100 sampled optimal resolutions and their multiple optimal reconciliations. Likewise, 70% of the gene nodes that were originally binary have a fully conserved mapping assignment to the species tree. Mappings and events are slightly less conserved for the nodes that were originally non-binary in the input gene trees. Among these non-binary nodes, 59% have a fully conserved event assignment across all 100 sampled optimal resolutions and their multiple optimal reconciliations, and 46% have a fully conserved mapping assignment to the species tree.
Further details appear in Figure 8. These results are striking and show that most aspects of the reconciliation are conserved across all optimal resolutions for the non-binary gene trees, even after accounting for uncertainty in the optimal reconciliations themselves. Since a fair number of gene nodes did not have a fully conserved mapping assignment, we further computed the number of optimal mappings for each internal gene tree node. As Figure S2 in the supplement shows, for the roughly 30% of the binary nodes and 54% of non-binary nodes that do not have a fully conserved mapping assignment, the majority of these nodes have at most 2 or 3 optimal mapping assignments. Overall, our reconciliation analysis shows that DTL reconciliation can be meaningfully applied even to non-binary gene trees to infer the evolutionary histories of their gene families.

**Software availability.** An implementation of our software is available as part of the RANGER-DTL software package [2], available at http://compbio.engr.uconn.edu/software/RANGER-DTL.

3.7 Conclusion

In this work, we have presented exact algorithms for DTL-reconciliation of non-binary gene trees and have shown how to address the problem of gene tree uncertainty in DTL-reconciliation. The algorithms and techniques developed in this chapter make it possible to not only apply DTL-reconciliation to non-binary gene trees, but to also negate the impact of gene tree uncertainty by distinguishing evolutionary inferences that have high support from those that have low support across all optimal resolutions of the gene tree. In short, these algorithms and techniques help address a major gap in biologists’ ability to apply DTL reconciliation to real data. As our experiments with real data demonstrate, despite their exponential
worst-case time complexities, our algorithms are applicable to a large fraction of non-binary
gene trees that arise in practice. We further observed that even though unresolved gene trees
often have a very large number of optimal binary resolutions, these optimal resolutions tend
to be significantly more similar to one another than to randomly selected binary resolutions.
Moreover, when reconciled with the species tree, the vast majority of the nodes in the input
gene trees are assigned a consistent (single) event and consistent (single) mapping across all
optimal resolutions. This implies that many aspects of gene family evolution can be confidently
inferred despite the presence of multiple optimal resolutions.

Our experimental results also demonstrate that many gene trees that arise in practice have
very high degree, making their reconciliation computationally infeasible using the FPT and
enumeration algorithms. A useful direction for future research would be to design efficient
heuristics or approximation algorithms that could be used to reconcile high-degree gene trees.
Chapter 4

On Inferring Additive and Replacing Horizontal Gene Transfers

Through Phylogenetic Reconciliation

4.1 Introduction

The transfer of genetic information between organisms that are not in a direct ancestor-descendant relationship, called horizontal gene transfer or simply transfer for short, is a crucial process in microbial evolution. The problem of detecting transfer events has been extensively studied and many different methods have been developed for the problem; see, e.g., [69] for a review. The two most widely used classes of methods are those based on atypical sequence composition and those based on phylogenetic discordance. Sequence composition methods look for atypical dinucleotide frequencies, codon usage biases, or other sequence features that might indicate instances of horizontally acquired genes, but are only effective at short evolutionary time scales and are unable to accurately identify the donors and recipients of transfer events [24, 69]. Phylogenetic methods rely on the fact that horizontal transfers leave tell-tale phylogenetic signatures in the topologies of the transferred genes. These methods construct
gene trees for individual gene families and compare them to known species phylogenies to in-
fer possible transfer events. It is well-understood that when a gene is horizontally transferred,
it may either add itself as a new gene to the recipient genome, resulting in an additive transfer,
or replace an existing homologous gene, resulting in a replacing transfer [15, 37]. Yet, there
do not currently exist any phylogenetic methods that simultaneously model both these types of
transfers. This limitation not only affects the applicability and accuracy of these methods but
also makes it difficult to distinguish between additive and replacing transfers.

Phylogenetic methods for inferring transfer events can be divided into two classes: (i) Those that implicitly assume that all transfers are replacing transfers and that all discordance
between gene trees and species trees is due to these replacing transfer events, e.g., [1, 6, 7,
27, 31, 35, 52, 66], and (ii) those based on the Duplication-Transfer-Loss (DTL) reconcilia-
tion framework, which model gene duplication and gene loss as additional sources of gene
tree/species tree discordance, but implicitly assume that all transfers are additive transfers,
e.g., [2, 13, 17, 19, 32, 44, 51, 60, 63–65, 67]. Thus, no existing phylogenetic method models
both additive and replacing transfers. And while methods based on DTL reconciliation repre-
sent a major advance in the ability to accurately detect transfer events, they are limited by their
inability to properly handle replacing transfers.
Our contribution. In this work, we define and formalize the first phylogenetic reconciliation framework that simultaneously models both additive and replacing transfer events. Our framework builds upon the standard parsimony-based DTL reconciliation model [2, 67], by explicitly modeling replacing transfer events. Specifically, we formally define the Duplication–Additive-Transfer–Replacing-Transfer–Loss (DTRL) reconciliation model that explicitly models both additive and replacing transfer events, along with gene duplications and losses. As with the underlying DTL reconciliation model, we formulate the DTRL reconciliation problem as one of finding a most parsimonious DTRL reconciliation, i.e., one with smallest total “reconciliation cost”. We prove that the problem of computing a most parsimonious DTRL reconciliation is NP-hard, using a reduction from the NP-hard minimum rooted Subtree Prune and Regraft (rSPR) distance problem, and perform the very first experiments to study the impact of replacing transfer events on the accuracy of DTL reconciliation itself. Surprisingly, we found that DTL reconciliation is highly robust to the presence of replacing transfer. Our experimental results show that, even though the problem of inferring optimal DTRL reconciliations is NP-hard, it should be possible to design effective heuristics for the problem based on the simpler, and efficiently solvable, DTL reconciliation model.

We note that the problem of integrating replacing transfers with DTL reconciliation has also been recently, and independently, studied by Hasic and Tannier in an unpublished manuscript [28]. That manuscript proves that the problem of inferring replacing transfers through phylogenetic reconciliation is NP-hard when the species tree is dated. However, the results in that manuscript are largely complementary to the current work. Specifically, we provide the first formalization of the DTRL reconciliation framework, our proof of NP-hardness is not only

1Note that in the current work we use the undated version of DTL reconciliation, where the species tree is undated, but one could easily use the dated version of DTL reconciliation [19] as well.
completely different but applies to the undated version of the problem where the species tree is undated, we provide the first experimental results on the impact of replacing transfer on conventional DTL reconciliation.

The remainder of the manuscript is organized as follows: Basic definitions, preliminaries, and a formal description of the DTRL reconciliation model appear in the next section. The NP-hardness proof appears in Section 4.3, and experimental results on the effect of replacing transfers on DTL reconciliation are described in Section 4.4. Concluding remarks appear in Section 4.5.

Note: Section 4.4 is based on the work which was done by co-authors.

4.2 Definitions and preliminaries

We follow basic definitions and notation from [2]. Given a rooted tree $T$, we denote its node, edge, and leaf sets by $V(T)$, $E(T)$, and $Le(T)$ respectively. The root node of $T$ is denoted by $rt(T)$, the parent of a node $v \in V(T)$ by $pa_T(v)$, its set of children by $Ch_T(v)$, and the (maximal) subtree of $T$ rooted at $v$ by $T(v)$. The set of internal nodes of $T$, denoted $I(T)$, is defined to be $V(T) \setminus Le(T)$. We define $\leq_T$ to be the partial order on $V(T)$ where $x \leq_T y$ if $y$ is a node on the path between $rt(T)$ and $x$. The partial order $\geq_T$ is defined analogously, i.e., $x \geq_T y$ if $x$ is a node on the path between $rt(T)$ and $y$. We say that $y$ is an ancestor of $x$, or that $x$ is a descendant of $y$, if $x \leq_T y$ (note that, under this definition, every node is a descendant as well as ancestor of itself). We say that $x$ and $y$ are incomparable if neither $x \leq_T y$ nor $y \leq_T x$.

Given a non-empty subset $L \subseteq Le(T)$, we denote by $lca_T(L)$ the last common ancestor (LCA) of all the leaves in $L$ in tree $T$; that is, $lca_T(L)$ is the unique smallest upper bound of $L$ under $\leq_T$. Given $x, y \in V(T)$, $x \rightarrow_T y$ denotes the unique path from $x$ to $y$ in $T$. We denote by
$\text{dist}_T(x, y)$ the number of edges on the path $x \rightarrow_T y$; note that if $x = y$ then $\text{dist}_T(x, y) = 0$.

Given a set $L \subseteq \text{Le}(T)$, let $T'$ be the minimal rooted subtree of $T$ with leaf set $L$. We define the leaf induced subtree of $T$ on leaf set $L$, denoted $T'[L]$, to be the tree obtained from $T'$ by successively removing each non-root node of degree two and adjoining its two neighbors. A tree is binary if all of its internal nodes have exactly two children. Throughout this work, the term tree refers to rooted binary trees.

A species tree is a tree that depicts the evolutionary relationships of a set of species. Given a gene family from a set of species, a gene tree is a tree that depicts the evolutionary relationships among the sequences encoding only that gene family in the given set of species. Thus, the nodes in a gene tree represent genes. Throughout this work, we denote the gene tree and species tree under consideration by $G$ and $S$, respectively. We assume that each leaf of the gene tree is labeled with the species from which that gene (sequence) was obtained. This labeling defines a leaf-mapping $\mathcal{L}_{G, S} : \text{Le}(G) \rightarrow \text{Le}(S)$ that maps a leaf node $g \in \text{Le}(G)$ to that unique leaf node $s \in \text{Le}(S)$ which has the same label as $g$. Note that gene trees may have more than one gene from the same species. The species tree contains at least all the species represented in the gene tree.

4.2.1 Additive and replacing transfers

When a gene is horizontally transferred, there are two possibilities for how it may incorporate itself into the recipient genome. The first possibility is that the transferred gene inserts itself to the recipient genome without overwriting any existing genes, thereby creating a new gene locus for itself. The second possibility is that the transferred gene replaces an existing homologous copy of itself, preserving the total number of genes in the recipient genome.
Figure 9: Additive and replacing transfers. This figure shows the evolution of two gene families inside the same species tree. Both gene families exist in the root of the species tree and evolve according to the topology of the species tree without any gene duplications or losses. Gene family 1 is affected by a replacing transfer event, as shown in the figure by the upper orange (dashed) arrow. Gene family 2 is affected by an additive transfer event, as shown by the lower orange (dashed) arrow. The topologies of the resulting gene trees for these two gene families are shown.
**Definition 4.2.1** (Additive transfer).

An additive transfer is a horizontal gene transfer that inserts itself into the recipient genome through the addition of a new gene locus.

**Definition 4.2.2** (Replacing transfer).

A replacing transfer is a horizontal gene transfer that inserts itself into the recipient genome by replacing a homologous gene at an existing gene locus.

Note that additive transfers result in an increase in the total number of genes in the recipient genome, while replacing transfers do not. We also point out that replacing transfers can only happen if the recipient genome already contains a homologous copy of the gene being transferred. Figure 9 illustrates how additive and replacing transfer events impact the resulting gene tree topology.

### 4.2.2 DTRL Reconciliation

The new *Duplication–Additive-Transfer–Replacing-Transfer–Loss* (DTRL) Reconciliation model is based upon the well-studied parsimony-based DTL reconciliation framework [2, 67] (which implicitly assumes that all transfer events are additive). However, the introduction of replacing transfers into the model poses several challenges, as we describe below, and the DTL reconciliation framework must therefore be substantially extended to allow for replacing transfers. Specifically, to fully specify a DTRL reconciliation, we must (i) account for hidden duplication or transfer events that do not label any node of the gene tree, and (ii) include in the reconciliation those gene lineages that have been lost (i.e., are no longer visible on the gene tree) but which played a role in the evolution of that gene family by participating in transfer events. We elaborate on these below.
**Hidden events.** Unlike the DTL reconciliation model, where each speciation, duplication, or transfer event required by the reconciliation can be assigned to an individual gene tree node, a most-parsimonious DTRL reconciliation may postulate duplication and transfer events (additive or replacing) that cannot be assigned to any node on the gene tree. Such hidden events may be required for most-parsimonious DTRL reconciliation but are invisible on the gene tree either because only descendants from one of the loci resulting from a duplication or additive transfer event survive in the gene family or because they appear on an invisible lineage. The reason hidden events can occur in optimal DTRL reconciliations is that one of the loci resulting from the hidden event is subsequently used (and overwritten) by one or more replacing transfers. This phenomenon is illustrated in Figure 10.

**Invisible gene lineages and augmented gene trees.** To properly recover replacing transfer events and correctly count the number of losses, it is necessary to postulate and account for those gene lineages that are no longer visible on the gene tree but which played a role in the evolution of that gene family by participating in replacing transfer events. Such invisible gene lineages can result from duplication, speciation, or transfer events, but become invisible because no descendants survive in the extant gene family. If these lineages do not participate in any transfer events that impacted the rest of the gene tree, then they can be safely ignored, but otherwise they must be accounted for if replacing transfers are to be recovered accurately and the number of losses counted correctly. We account for invisible lineages by augmenting the input gene tree with additional edges/subtrees, resulting in an augmented gene tree, and showing the DTRL reconciliation for this entire augmented gene tree. Figure 10 shows an example of an augmented gene tree and illustrates why it is important to consider invisible gene lineages.
The DTRL reconciliation model takes as input a rooted gene tree and a rooted species tree and defines a framework for reconciling the gene tree with the species tree by postulating duplication, additive transfer, replacing transfer, and gene loss events. The reconciliation creates an augmented gene tree, maps each augmented gene tree node to a unique species tree node, respecting the temporal constraints implied by the species tree topology, and designates each augmented gene tree node as representing either a speciation, duplication, additive transfer, or replacing transfer event. For any gene tree node, say $g$, that represents a transfer event, the reconciliation also specifies which of the two edges $(g, g')$ or $(g, g'')$, where $g', g''$ denote the children of $g$, represents the transfer edge and identifies the recipient species of the corresponding transfer. If $g$ represents a replacing transfer event, the reconciliation also identifies the specific gene lineage that was lost as a result of that replacing transfer.

Next, we define what constitutes a valid DTRL reconciliation.

**Definition 4.2.3 (DTRL-reconciliation).**

A DTRL-reconciliation for $G$ and $S$ is a ten-tuple

$\langle L, G', M, \Sigma, \Delta, \Theta_A, \Theta_R, \Xi, \tau, \lambda \rangle$, where $L \colon Le(G) \rightarrow Le(S)$ represents the leaf-mapping from $G$ to $S$, $G'$ represents the augmented gene tree, $M \colon V(G') \rightarrow V(S)$ maps each node of $G'$ to a node of $S$, the sets $\Sigma$, $\Delta$, $\Theta_A$ and $\Theta_R$ partition $I(G')$ into speciation, duplication, additive transfer, and replacing transfer nodes, respectively, $\Xi$ is a subset of $E(G')$ that represents transfer edges (additive or replacing), $\tau : \Theta_A \cup \Theta_R \rightarrow V(S)$ specifies the recipient species for each transfer event, and $\lambda : \Theta_R \rightarrow Le(G') \setminus Le(G)$ is an injective function that associates each replacing transfer event with a lost gene in the augmented gene tree, subject to the following constraints:

**Augmented gene tree constraint**
1. \( G = G'[Le(G)] \).

**Mapping constraints**

2. If \( g \in Le(G) \), then \( \mathcal{M}(g) = \mathcal{L}(g) \).

3. If \( g \in I(G') \) and \( g' \) and \( g'' \) denote the children of \( g \), then,

   (a) \( \mathcal{M}(g) \not\preceq \mathcal{M}(g') \) and \( \mathcal{M}(g) \not\preceq \mathcal{M}(g'') \).

   (b) At least one of \( \mathcal{M}(g') \) and \( \mathcal{M}(g'') \) is a descendant of \( \mathcal{M}(g) \).

**Event constraints**

4. Given any edge \((g, g') \in E(G')\), \((g, g') \in \Xi\) if and only if \( \mathcal{M}(g) \) and \( \mathcal{M}(g') \) are incomparable.

5. If \( g \in I(G') \) and \( g' \) and \( g'' \) denote the children of \( g \), then,

   (a) \( g \in \Sigma \) only if \( \mathcal{M}(g) = \text{lca}(\mathcal{M}(g'), \mathcal{M}(g'')) \) and \( \mathcal{M}(g') \) and \( \mathcal{M}(g'') \) are incomparable,

   (b) \( g \in \Delta \) only if \( \mathcal{M}(g) \geq \text{lca}(\mathcal{M}(g'), \mathcal{M}(g'')) \),

   (c) \( g \in \Theta_A \cup \Theta_R \) if and only if either \((g, g') \in \Xi\) or \((g, g'') \in \Xi\).

   (d) If \( g \in \Theta_A \cup \Theta_R \) and \((g, g') \in \Xi\), then \( \mathcal{M}(g) \) and \( \tau(g) \) must be incomparable, and \( \mathcal{M}(g') \) must be a descendant of \( \tau(g) \), i.e., \( \mathcal{M}(g') \leq \tau(g) \).

**Replacing transfer constraint**

6. If \( g \in \Theta_A \cup \Theta_R \), then \( g \in \Theta_R \) if and only if \( \mathcal{M}(\lambda(g)) = \tau(g) \).
Note: This definition allows any invisible leaf node \( g \) (i.e., \( g \in \text{Le}(G') \setminus \text{Le}(G) \)) to map to a leaf node of \( S \), say \( s \in \text{Le}(S) \). However, gene \( g \) is not actually present in species \( s \) (otherwise it would not be invisible). Instead, \( M(g) = s \) indicates that \( g \) existed in a predecessor species of \( s \) represented along the edge \((pa(s), s) \in E(S)\).

In the definition above, Constraint 1 specifies that the augmented gene tree, \( G' \), must be consistent with the topology of the input gene tree \( G \). Constraint 2 above ensures that the mapping \( M \) is consistent with the leaf-mapping \( L \). Constraint 3a imposes on \( M \) the temporal constraints implied by \( S \), and Constraint 3b implies that any internal node in \( G' \) may represent at most one transfer event. Constraint 4 determines the edges of \( T \) that are transfer edges. Constraints 5a, 5b, and 5c state the conditions under which an internal node of \( G' \) may represent a speciation, duplication, and (additive or replacing) transfer respectively. Constraint 5d specifies which species may be designated as the recipient species for any given transfer event. Finally, constraint 6 specifies that a transfer event is labeled as a replacing transfer if and only if there exists a unique invisible leaf node in \( G' \) that represents the gene that is “replaced” by that replacing transfer.

While duplications, additive transfers, and replacing transfers are directly specified by any DTRL-reconciliation, losses are not. However, given a DTRL-reconciliation, the minimum number of losses can be computed along the same lines as in the DTL reconciliation model [2], but with an adjustment to account for invisible lineages and replacing transfers. The adjustment is required to account for the implicit loss of a gene that occurs at each invisible leaf in the augmented gene tree \( G' \). Some of these losses are due to replacing transfers, but those that are not must be counted as gene losses.
Definition 4.2.4 (Losses). Given a DTRL-reconciliation \( \alpha = \langle L, G', M, \Sigma, \Delta, \Theta_A, \Theta_R, \Xi, \tau, \lambda \rangle \) for \( G \) and \( S \), let \( g \in I(G') \) and \( \{ g', g'' \} = Ch(g) \).

The number of losses \( \text{Loss}_\alpha(g) \) at node \( g \), is defined to be:

- \( |\text{dist}_G(M(g), M(g')) - 1| + |\text{dist}_G(M(g), M(g'')) - 1| \), if \( g \in \Sigma \).
- \( \text{dist}_G(M(g), M(g')) + \text{dist}_G(M(g), M(g'')) \), if \( g \in \Delta \).
- \( \text{dist}_G(M(g), M(g'')) + \text{dist}_G(\tau(g), M(g')) \) if \( (g, g') \in \Xi \).

The number of implicit losses at invisible leaves of \( G' \) (i.e., for the set \( \text{Le}(G') \setminus \text{Le}(G) \)) is defined to be \( |\text{Le}(G') \setminus \text{Le}(G)| - |\Theta_R| \).

The total number of losses in the DTRL-reconciliation \( \alpha \) is defined to be

\[
\text{Loss}_\alpha = |\text{Le}(G') \setminus \text{Le}(G)| - |\Theta_R| + \sum_{g \in I(G)} \text{Loss}_\alpha(g).
\]

In the DTRL reconciliation framework, each evolutionary event other than speciation is assigned a positive cost. Let \( P_\Delta \), \( P_{\Theta_A} \), \( P_{\Theta_R} \), and \( P_{\text{loss}} \) denote the gene duplication, additive transfer, replacing transfer, and gene loss costs, respectively. The reconciliation cost of a given DTRL-reconciliation is defined as follows.

Definition 4.2.5 (Reconciliation cost). Given a DTRL-reconciliation \( \alpha = \langle L, G', M, \Sigma, \Delta, \Theta_A, \Theta_R, \Xi, \tau, \lambda \rangle \), the reconciliation cost for \( \alpha \) is the total cost of all events invoked by \( \alpha \). In other words, the reconciliation cost of \( \alpha \) is

\[
|\Delta| \times P_\Delta + |\Theta_A| \times P_{\Theta_A} + |\Theta_R| \times P_{\Theta_R} + \text{Loss}_\alpha \times P_{\text{loss}}.
\]

The goal is to find a DTRL-reconciliation that has minimum reconciliation cost. More formally:
**Definition 4.2.6** (ODTRL problem). Given $G$ and $S$, along with $P_\Delta$, $P_{\Theta_A}$, $P_{\Theta_R}$, and $P_{\text{loss}}$, the Optimal DTRL-Reconciliation Problem (ODTRL) problem is to find a DTRL-reconciliation for $G$ and $S$ with minimum reconciliation cost.

### 4.3 NP-hardness of ODTRL

We claim that the ODTRL problem is NP-hard and that the corresponding decision problem is NP-Complete. The decision version of the ODTRL problem is as follows:

**Problem 6** (D-DTRL).

**Instance:** $G$ and $S$, along with event costs $P_\Delta$, $P_{\Theta_A}$, $P_{\Theta_R}$, and $P_{\text{loss}}$, and a non-negative integer $l$.

**Question:** Does there exist a DTRL-reconciliation for $G$ and $S$ with reconciliation cost at most $l$?

**Theorem 4.3.1.** The D-DTRL problem is NP-Complete.

The D-DTRL problem is clearly in NP. In the remainder of this section we will show that the D-DTRL problem is NP-hard using a poly-time reduction from the decision version of the NP-hard minimum rooted Subtree Prune and Regraft (rSPR) Distance problem [8].

#### 4.3.1 Reduction from minimum rSPR distance

We begin by defining an rSPR operation and define the decision version of the minimum rSPR distance problem.

**Definition 4.3.1** (rSPR operation [8]). Let $T$ be a rooted binary tree and let $e = \{u, v\}$ be an edge of $T$ where $u$ is the vertex that is in the path from the root of $T$ to $v$. Let $T'$ be the rooted
binary tree obtained from $T$ by deleting $e$ and then adjoining a new edge $f$ between $v$ and the component $C_u$ that contains $u$ in one of the following two ways:

- Creating a new vertex $u'$ which subdivides an edge in $C_u$, and adjoining $f$ between $u'$ and $v$. Then, either suppressing the degree-two vertex $u$ or, if $u$ is the root of $T$, deleting $u$ and the edge incident with $u$, making the other end-vertex of this edge the new root.

- Creating a new root vertex $u'$ and a new edge between $u'$ and the original root. Then adjoining $f$ between $u'$ and $v$ and suppressing the degree-two vertex $u$.

We say that $T'$ has been obtained from $T$ by a single rooted subtree prune and regraft (rSPR) operation.

**Definition 4.3.2** (rSPR distance). Given two trees $T$ and $T'$ with identical leaf sets, the rSPR distance between $T$ and $T'$, denoted $d_{rSPR}(T, T')$, is defined to be the minimum number of rSPR operations required to transform $T$ into $T'$.

The minimum rSPR distance problem is to find the rSPR distance between two trees. Its decision version can be stated as follows:

**Problem 7** (D-rSPR problem).

**Instance:** Two trees $T$ and $T'$ with identical leaf sets, and a non-negative integer $k$.

**Question:** Is $d_{rSPR}(T, T') \leq k$?

The D-rSPR problem is known to be NP-Complete [8]. Consider any instance $\rho$ of the D-rSPR problem with trees $T$ and $T'$ on the same leaf set of size $n$ (i.e., $Le(T) = Le(T')$ and $n = |Le(T)|$), and non-negative integer $k$. We will show how to transform $\rho$ into an instance $\delta$ of the D-DTRL problem by constructing $G$, $S$, and assigning the four event costs.
$P_\Delta$, $P_{R R}$, $P_{R A}$, and $P_{Loss}$, such that there exists a YES answer to the D-rSPR problem on $\rho$ if and only if there exists a YES answer to the D-DTRL instance $\delta$ with reconciliation cost at most $l = 10n + 5k - 4$.

### 4.3.2 Gadget

We assume that the leaf set of $T$ and $T'$ is $\{t_1, t_2, \ldots, t_n\}$. We also assume that the internal nodes of $T$ are labeled $\{z_1, z_2, \ldots, z_{n-1}\}$, as depicted in Figure 11(a). Next, we first show how to construct the species tree $S$, then the gene tree $G$, and then assign event costs.

**Species tree.** The species tree $S$, is composed of two subtrees denoted $S_l$ and $S_r$ and ten extraneous leaf nodes (which are not represented in the gene tree). The root of subtree $S_l$ is a child of $rt(S)$. The other subtree, $S_r$, is connected to $rt(S)$ through a path to which the ten extraneous leaves are connected. This is shown in Figure 11(b). The subtree $S_l$ is identical to tree $T'$. Subtree $S_r$ is a modified version of tree $T$, obtained as follows: We first perform a post-order traversal of tree $T$ and number each node according to its position in the ordering, e.g, the left-most leaf node in $T$ would be labeled with a 1, while $rt(T)$ would be assigned the number $2n - 1$. Next, for each edge $(pa(t), t) \in E(T)$, if the number associated with $t$ is $i$, we attach a subtree $((x_i, u_{2i-1}), u_{2i})$; to edge $(pa(t), t)$. Thus, $2n - 2$ subtrees are attached in all. Finally, we delete all the original leaf nodes $\{t_1, t_2, \ldots, t_n\}$ from $T$ and binarize the remaining tree by suppressing all non-root nodes of degree two. The resulting tree is $S_r$. This modification is depicted in Figure 11.

**Gene tree.** Gene tree $G$ consists of two main subtrees, denoted $G_l$ and $G_r$. Subtree $G_l$ is obtained from species tree subtree $S_r$ by removing all leaf nodes labeled with prefix $x$ and then suppressing all non-root nodes of degree two. Subtree $G_r$ is obtained by modifying $T$
as follows: We consider again the post-order numbering of the nodes of $T$ and, for each edge $(pa(t), t) \in E(T)$, if the number associated with $t$ is $i$, we attach a leaf labeled $x_i$ to edge $(pa(t), t)$. The new internal node created in attaching leaf $x_i$ to the tree is denoted $y_i$. This construction is depicted in Figure 11(c).

Observe that each internal node of $T$ has a corresponding node in $G_r$. We label these corresponding nodes of $G_r$ as $Z_G = \{z'_1, z'_2, \ldots, z'_{n-1}\}$, where node $z'_i \in I(G_r)$ corresponds to node $z_i \in I(T)$ for $1 \leq i \leq n - 1$. We also define the following three subsets of $V(G)$: $Y_G = \{y_1, y_2, \ldots, y_{2n-2}\}$, $X_G = \{x_1, x_2, \ldots, x_{2n-2}\}$, and $T_G = \{t_1, \ldots, t_n\}$. Note that $I(G_r) = Y_G \cup Z_G$.

**Event costs.** Event costs are assigned as follows: $P_{\Delta} = 4$, $P_{\Theta_A} = 6$, $P_{\Theta_R} = 5$, and $P_{\text{loss}} = 3$. This completes our construction of instance $\delta$ of the D-DTRL problem. Note that $G$ and $S$ can be both constructed in time polynomial in $n = |Le(T)|$.

**Claim 2.** There exists a YES answer to the D-rSPR problem on $\rho$ if and only if there exists a YES answer to the D-DTRL instance $\delta$ with reconciliation cost $l \leq 10n + 5k - 4$.

The main idea behind this reduction can be explained briefly as follows. Each rSPR operation on instance $\rho$ corresponds to exactly one replacing transfer event on gene tree $G$ from instance $\delta$. Based on the structure of gene tree $G$ and species tree $S$, we will be able to show that for each rSPR operation there is at least one way to get a valid corresponding replacing transfer.

The correctness of Theorem 4.3.1 follows immediately from Claim 2.
4.3.3 Proof of Claim 2: Forward direction

Assuming we have a YES answer for the \( rSPR \) instance \( \rho \), we will show how to construct a DTRL-reconciliation \( \alpha \) for instance \( \delta \) with reconciliation cost at most \( 10n + 5k - 4 \).

Suppose \( d_{rSPR}(T, T') = k' \), where \( k' \leq k \). Then, based on the close association between \( rSPR \) distances and maximum-agreement forests [8], we know that \( d_{rSPR}(T, T') = m(T, T') \), where \( m(T, T') \) is the size of a maximum-agreement forest for \( T \) and \( T' \). In particular, there exist \( k' \) rooted, vertex-disjoint subtrees of \( T \), denoted \( T_1, \ldots, T_{k'} \) with leaf sets \( L_1, \ldots, L_{k'} \), respectively, such that \( T[L_i] = T'[L_i] \) for all \( i \in \{1, \ldots, k'\} \), and \( L_1 \cup \ldots \cup L_{k'} = Le(T) \). These \( k' \) subtrees from the maximum-agreement forest correspond to the \( k' \) subtrees that are pruned and regrafted to transform \( T \) into \( T' \) through \( rSPR \) operations. In other words, there exist \( k' \) nodes, denoted \( \mathcal{P} = \{p_1, \ldots, p_{k'}\} \) in \( V(T) \), corresponding to the roots of the \( k' \) subtrees \( T_1, \ldots, T_{k'} \), respectively, that identify the edges that will be cut in the \( k' \) \( rSPR \) operations. For brevity, we refer the reader to [8] for a definition of maximum-agreement forests and for proofs of the preceding statements.

The following observation states three simple facts about the set of nodes \( \mathcal{P} \).

**Observation 1.** Let \( t \in V(T) \) and \( Ch(t) = \{t', t''\} \).

1. If \( t, t' \in \mathcal{P} \), then \( t'' \notin \mathcal{P} \).

2. If \( t', t'' \in \mathcal{P} \), then \( t \notin \mathcal{P} \). Moreover, the set \( (\mathcal{P} \setminus \{t'\}) \cup t \) must also correspond to a valid maximum-agreement forest for \( T \) and \( T' \).

3. \( |\mathcal{P}| = k' \leq k \leq n - 2 \).
Parts (1) and (2) in the above observation follow directly from the definition of a maximum-agreement forest. Part (3) follows from the fact that the maximum rSPR distance between any two rooted trees with \( n \) leaves is bounded above by \( n - 2 \) [62].

**Notation:** Note that both leaf nodes and internal nodes of \( T \) have corresponding nodes in the gene tree subtree \( G_r \). We denote by \( P' = \{p_1', \ldots, p_{k'}'\} \) the nodes corresponding to \( P = \{p_1, \ldots, p_{k'}\} \) in \( G_r \).

Next, we show how to construct the augmented gene tree \( G' \) and reconcile \( G' \) with \( S \) such that the total reconciliation cost is no more than \( 10n + 5k - 4 \). We begin by showing how to reconcile \( G \) with \( S \) and then show how to augment \( G \) into \( G' \) and complete the reconciliation. It is worth noting that we start out with \( P' \) as initialized above, but change its composition as we proceed with defining the reconciliation; however, we will always maintain \( |P'| = k' \).

**Reconciliation of \( G \) and \( S \).** We begin by defining a useful edit operation for reconciliations.

**Definition 4.3.3** (Switch-recipient operation). *Given a partial reconciliation of \( G \) and \( S \), and a node \( g \in V(G) \) that is labeled as a (replacing or additive) transfer event, let \( g' \) and \( g'' \) denote the two children of \( g \) such that \( (g, g') \) is the transfer edge. A switch-recipient operation on \( g \), denoted \( \text{SR}(g) \), modifies the partial reconciliation by setting \( (g, g'') \) to be the transfer edge, removing edge \((g, g')\) from the set \( \Xi \), and updating the mappings \( M(g) \) and \( \tau(g) \) to be \( M(g') \). Note that the partial reconciliation of \( G \) and \( S \) need not remain a valid DTRL reconciliation after this operation.*

The leaf-to-leaf mapping from \( G \) to \( S \) is defined by the leaf labels. To define the remainder of the reconciliation, we first perform a post-order traversal of \( G_l \) and map each internal node \( a \in I(G_l) \) to the species node \( \text{lca}(M(b), M(c)) \), where \( b, c \) denote the two children of \( a \), and
assign $a$ to be a speciation event. Next, we perform a post-order traversal of $G_r$ and map each internal node $a \in I(G_r)$, where $b$ and $c$ denote its two children, as described below.

Observe that $I(G_r) = Y_G \cup Z_G$, that $X_G \cap P' = \emptyset$, that every node from $Y_G$ has exactly one child in $X_G$, and that every node from $Z_G$ has both its children from $Y_G$.

1. If $a \in Y_G$ and $b \in X_G$ then:

   (a) If $c \notin P'$, then $a$ maps to $M(c)$ and represents a replacing transfer event with $(a, b)$ representing the transfer edge and $\tau(a) = M(b)$.

   (b) If $c \in P'$, then $a$ maps to $M(c)$ and represents a replacing transfer event with edge $(a, b)$ representing the transfer edge and $\tau(a) = M(b)$. We also update $P'$ to be $(P' \setminus \{c\}) \cup \{a\}$.

2. If $a \in Z_G$ and $b, c \in Y_G$, then:

   (a) If $a, b, c \notin P'$, then $a$ maps to $\text{lca}(M(b), M(c))$ and represents a speciation event.

   (b) If $a, b \notin P'$ and $c \in P'$, then $a$ maps to $M(b)$ and represents a replacing transfer event and edge $(a, c)$ represents the transfer edge with $\tau(a) = M(c)$. By Case 1 above, we know that every node of $Y_G$ represents a replacing transfer event, and so $c$ must also represent a replacing transfer event. If $M(b)$ and $M(c)$ are comparable in $S$, i.e., $M(c) \leq_S M(b)$ or $M(b) \leq_S M(c)$, then we perform the switch-recipient operation $SR(c)$ (which, as we prove later, makes $M(b)$ and $M(c)$ incomparable).

   (c) If $a, c \notin P'$ and $b \in P'$, then this case is analogous to the previous case.

   (d) If $a \notin P'$ and $b, c \in P'$, then
• If $\mathcal{M}(b)$ and $\mathcal{M}(c)$ are incomparable in $S$, then $a$ maps to $\mathcal{M}(b)$ and represents a replacing transfer event with edge $(a, c)$ representing the transfer edge and $\tau(a) = \mathcal{M}(c)$.

• If $\mathcal{M}(b)$ and $\mathcal{M}(c)$ are comparable in $S$, then we perform the switch-recipient operation $\text{SR}(c)$. Observe that nodes $b$ and $c$ must represent replacing transfer events. We also update $\mathcal{P}'$ to be $\mathcal{P}' = (\mathcal{P}' \setminus \{b\}) \cup \{a\}$.

(e) If $a \in \mathcal{P}'$ and $b, c \notin \mathcal{P}'$, then $a$ maps to $\text{lca}(\mathcal{M}(b), \mathcal{M}(c))$ and represents a speciation event.

(f) If $a, b \in \mathcal{P}'$ and $c \notin \mathcal{P}'$, then $a$ maps to $\mathcal{M}(c)$ and represents a replacing transfer with edge $(a, b)$ representing the transfer edge and $\tau(a) = \mathcal{M}(b)$. If $\mathcal{M}(b)$ and $\mathcal{M}(c)$ are comparable in $S$ then we perform the switch-recipient operation $\text{SR}(b)$ (recall that $b$ must represent a replacing transfer event).

(g) If $a, c \in \mathcal{P}'$ and $b \notin \mathcal{P}'$, then this case is analogous to the previous case.

(h) If $a, b, c \in \mathcal{P}'$, then, as we prove later in Lemma 16, this case cannot arise in any optimal solution.

Finally, $rt(G)$ maps to $rt(S_r)$ and represents an additive transfer event with edge $(rt(G), rt(G_r))$ representing the transfer edge and $\tau(rt(G)) = \mathcal{M}(rt(G_r))$.

Next, we prove some useful properties of the reconciliation described above, show how to augment $G$ into $G'$ and “complete” the reconciliation, and prove that the completed DTRL reconciliation is valid.

**Lemma 16.** Suppose $a \in Z_G$, with children $b$ and $c$, then at no point in the post-order traversal of $G_r$, as described above, can $a, b$, and $c$ be in the set $\mathcal{P}'$ simultaneously.
Proof. Assume, for contradiction, that \( a, b, c \in \mathcal{P}' \) at some point during the post-order traversal. Let \( a' \) denote the node corresponding to \( a \) in the tree \( T \). Suppose \( a' \notin \mathcal{P} \). Then, \( a \notin \mathcal{P}' \) at the beginning of the post-order traversal. Observe that \( a \) cannot be added to \( \mathcal{P}' \) unless the post-order traversal is exactly at node \( a \) and both \( b, c \in \mathcal{P}' \) at that time. If \( a \) is added to \( \mathcal{P}' \) at this step, then one of \( b \) or \( c \) will be removed from \( \mathcal{P}' \) and will never be added back at any later time. Thus, if \( a' \notin \mathcal{P} \) then \( a, b, c \notin \mathcal{P}' \) at any point during the post-order traversal. Consequently, under our assumption, we must have \( a' \in \mathcal{P} \).

We will now show that there must exist a node \( l \in \text{Le}(T'(a')) \) such that no node along the path from \( a' \) to \( l \), except for \( a' \) itself, is in \( \mathcal{P} \). Consider the two children \( u \) and \( v \) of \( a' \) in \( T \). By part 1 of Observation 1, we know that at most one of \( u \) or \( v \) can be in the set \( \mathcal{P} \). Without loss of generality we may therefore assume that \( u \notin \mathcal{P} \). Now, if \( u \in \text{Le}(T) \), then we are done. Therefore, suppose \( u \notin \text{Le}(T) \) and let \( u' \) and \( u'' \) denote the two children of \( u \) in \( T \). There are now two possible cases:

1. \( v \in \mathcal{P} \): In this case, it is not possible that both \( u' \) and \( u'' \) are in the set \( \mathcal{P} \). This is because if \( a', v, u', u'' \in \mathcal{P} \), then \( \mathcal{P} \setminus \{a'\} \) would yield a valid solution for the D-rSPR problem instance \( \rho \), implying \( d_{rSPR}(T, T') = k' - 1 \), which is a contradiction.

2. \( v \notin \mathcal{P} \): In this case, if \( v \in \text{Le}(T) \), then we have proved our claim. Therefore, assume \( v \notin \text{Le}(T) \) and let \( v' \) and \( v'' \) denote the two children of \( v \). Now, it is not possible that \( a', u', u'', v', v'' \) are simultaneously in the set \( \mathcal{P} \). Otherwise, \( \mathcal{P} \setminus \{a'\} \) would yield a valid solution for the D-rSPR problem instance \( \rho \), implying \( d_{rSPR}(T, T') = k' - 1 \), which is a contradiction.
By applying this argument inductively from $a$ towards the leaves of $T$, it follows that there exists a node $l \in Le(T(a'))$ such that no node along the path from $a'$ to $l$, except for $a'$ itself, is in $\mathcal{P}$.

Finally, consider the path from $l$ to $a$ in $G_r$. This path in $G_r$ consists of nodes corresponding to the $l$ to $a'$ path in $T$, along with a subset of nodes from $Y_G$. Observe that, before the post-order traversal of $G_r$, $\mathcal{P}'$ is initialized to $\mathcal{P}$ and so none of the nodes along the $l$ to $a$ path in $G_r$, except for node $a$ is in $\mathcal{P}'$. Furthermore, during the post-order traversal of $G_r$, the current node is added to $\mathcal{P}'$ only if both children of the current node are in $\mathcal{P}'$ at that time. Thus, no node along the path from $l$ to $a$ in $G_r$, except for node $a$ can ever be added to the set $\mathcal{P}'$, and so $a, b,$ and $c$ cannot simultaneously be in $\mathcal{P}'$ at any time during the post-order traversal.

\[\square\]

**Lemma 17.** In the constructed reconciliation of $G$ and $S$, $\mathcal{M}(z) \in V(S_l)$ for all $z \in Z_G$.

**Proof.** Observe that each node of $T_G$ maps to a node from $S_l$, and that each $z \in Z_G$ has both children from $Y_G$. To prove that $\mathcal{M}(z) \in V(S_l)$, for all $z \in Z_G$, it suffices to prove that, for each $y \in Y_G$, $\mathcal{M}(y) \in V(S_l)$ when $\mathcal{M}(y)$ is first assigned during the post-order traversal of $G_r$. This is because, per case (2) of the post-order traversal, the mapping $\mathcal{M}(z)$ is assigned based on the initial mapping assignment of the two children of $z$, and while the mapping of one of the children of $z$ may be subsequently be changed through a switch-receptient operation, the mapping of $z$ remains unchanged.

There are two possible cases:

Case 1: consider any $y \in Y_G$ such that $y$ does not have a child from $Z_G$. In this case, one child of $y$ must be in $T_G$ and the other in $X_G$. Since all nodes of $T_G$ map to $S_l$, by case (1)
of the post-order traversal we know that the initial mapping assignment for $y$ must also be to a node in $S_l$.

Case 2: consider any $y \in Y_G$ that has a child from $Z_G$. In this case, one child of $y$ must be in $Z_G$ and the other in $X_G$. Under a simple inductive argument, we may assume that the child of $y$ that is from $Z_G$ maps to a node of $S_l$. Under this assumption, case (1) of the post-order traversal applies and the initial mapping assignment for $y$ would therefore be to a node of $S_l$.

A simple inductive argument now immediately establishes that, for each $y \in Y_G$, $M(y) \in V(S_l)$ when $M(y)$ is initially assigned during the post-order traversal of $G_r$. \hfill \square

The next two lemmas helps establish that the assigned transfer events and speciation events are valid.

**Lemma 18.** In the constructed reconciliation of $G$ and $S$, if $g \in V(G)$ represents a replacing or additive transfer event then $M(g)$ and $\tau(g)$ must be incomparable.

**Proof.** Observe that if $g \in \Theta_A \cup \Theta_R$, then $g \in \{rt(G_r)\} \cup Y_G \cup Z_G$. We therefore have the following three cases:

1. $g = rt(G)$. In this case, based on the constructed reconciliation, $\tau(g) = M(rt(G_r))$ and $M(g) = M(rt(G_l))$. Note that $rt(G_r) \in Z_G$ and so, by Lemma 17, $rt(G_r)$ must map to a node in $V(S_l)$. Similarly, based on the constructed reconciliation, $rt(G_l)$ and $rt(G)$ both map to $S_r$. Thus, $M(rt(G))$ and $\tau(g)$ are incomparable.

2. $g \in Y_G$. Let $g'$ and $g''$ denote the two children of $g$. We know that $g' \in Z_G \cup T_G$ and $g'' \in X_G$. We know that all nodes of $X_G$ map to nodes of $S_r$, all nodes of $T_G$ map to nodes of $S_l$, and, by Lemma 17, all nodes of $Z_G$ map to nodes of $S_l$. Thus, $g'$ must map
to a node of $S_l$ and $g''$ must map to a node of $S_r$. Thus, in the initial mapping assignment of $g$, $M(g) \in V(S_l)$ while $\tau(g) \in V(S_r)$. Later, if a switch-recipient operation is performed on $g$, we would get $M(g) \in V(S_r)$ while $\tau(g) \in V(S_l)$. In either case, $M(g)$ and $\tau(g)$ are incomparable.

3. $g \in Z_G$. Let $g'$ and $g''$ denote the two children of $g$. We know that $g'$ and $g''$ are both in $Y_G$. Based on the case above, we know that each node from $Y_G$ has one child mapping to $S_l$ and the other child mapping to $S_r$. According to case 2 of the post-order traversal, if $M(g')$ and $M(g'')$ are comparable (so both map to either $S_l$ or both to $S_r$) then a switch-recipient operation is performed on one of the children of $g$, say $g'$, which would change the mapping of $g'$ from either $S_l$ to $S_r$ or vice versa. Thus, $g'$ and $g''$ are either incomparable to begin with or are made incomparable through a switch-recipient operation. Finally, the mapping of $M(g)$ is assigned to be the mapping of one of $g'$ or $g''$, with $\tau(g)$ assigned to be the mapping of the other child. $M(g)$ and $\tau(g)$ must therefore be incomparable.

For the next lemma we need the following definition.

**Definition 4.3.4 (Base Leaf Set).** Given the reconciliation of $G$ and $S$ as defined earlier, along with the set $P'$, we define the base leaf set of a node $g \in V(G)$ in $G$, denoted $BLe_G(g)$, to be $\{l \in Le(G(g)) \mid \text{none of the nodes, except possibly } g, \text{ on the path from } g \text{ to } l \text{ is in } P'\}$. We also define $BLe_S(g)$, for $g \in V(G)$, to denote the corresponding set of leaf nodes from $S$.

Note that, based on the proof of Lemma 16, it follows that $|BLe_G(g)| \geq 1$ for any $g \in V(G)$. 

105
Lemma 19. In the constructed reconciliation of $G$ and $S$, if $g \in I(G)$ represents a speciation event and $g', g''$ denote the two children of $g$, then $M(g')$ and $M(g'')$ must be incomparable in $S$.

Proof. Based on the constructed reconciliation, if $g \in I(G)$ represents a speciation event then either $g \in I(G_l)$ or $g \in Z_G$. We consider these two cases separately.

1. $g \in I(G_l)$. In this case, $M(g)$ maps to $lca(M(g'), M(g''))$, and, based on the topologies of $G$ and $S$, $M(g')$ and $M(g'')$ must be siblings in $S$. Thus, $M(g')$ and $M(g'')$ must be incomparable in $S$.

2. $g \in I(G_r)$. In this case, based on cases 2(a) and 2(e) of the post-order traversal, we must have $g \in Z_G, g', g'' \notin P'$ and $g', g'' \in Y_G$. Now, observe that for any node $y \in Y_G$, where $y \notin P'$, if $y'$ is the child of $y$ that is from $Z_G \cup T_G$, then $y' \notin P'$ and $M(y) = M(y')$. Also observe that if a node $z \in Z_G$ is not in $P'$, then it follows from the proof of Lemma 16 that at most one of its two children, denoted $y', y''$, can be in $P'$. Furthermore, if $y' \in P'$, then $M(z) = M(y'')$, while if $y', y'' \notin P'$, then $M(z) = lca(M(y'), M(y''))$.

Continuing in this fashion towards the leaves of $G$, it follows that $M(g') = lca(BLe_S(g'))$ and $M(g'') = lca(BLe_S(g''))$. Since $g$ is a speciation node it also follows that $BLe_G(g) = BLe_G(g') \cup BLe_G(g'')$ and $M(g) = lca(M(g'), M(g''))$. Consider the induced subtrees $G[BLe_G(g)], G[BLe_G(g')]$, and $G[BLe_G(g'')]$. Since none of the edges in these induced subtrees is in $P'$, these subtrees must be isomorphic to the induced subtrees $S[BLe_S(g)], S[BLe_S(g')]$, and $S[BLe_S(g'')]$, respectively. Thus, since $G(g')$ and $G(g'')$ are disjoint subtrees, so must $S(M(g')$ and $S(M(g''))$, completing the proof.
Lemma 20. In the constructed reconciliation of $G$ and $S$, there is at most one gene copy in each node (or edge) of $S$.

Proof. This follows directly from the fact that all internal nodes in $V(G) \setminus \{rt(G)\}$ represent either speciation or replacing transfer events. □

We now show how to create the augmented tree $G'$ based on gene tree $G$ and construct a complete DTRL reconciliation. We begin by initializing $G'$ to be the same as $G$, with each node of $G'$ having the same event and mapping assignment as in the reconciliation of $G$. We then perform a post-order traversal of $G'$ and for each node that represents a replacing transfer event, say $g$, we will augment $G'$ by adding a new leaf node, denoted $\bar{u}$, connected to $G'$ through a new internal node denoted $u$. This augmentation happens through the $Add_G$ operation defined below.

Definition 4.3.5 (Add operation). Given $G'$, $S$, and a node $g \in V(G')$ that is a replacing transfer event, let $g'$ and $g''$ denote the two children of $g$ such that $(g, g')$ is the transfer edge and $s' = M(g')$. Note that $s' \in V(S) \setminus \{rt(S)\}$ and so it must have a sibling, which we denote by $s''$. Let $u \in V(G')$ be a node such that $M(u) \in V(S(s''))$, $M(pa(u)) >_S s''$ and $M(u)$ has minimum distance to the node $s''$ among all options for $u$. The operation $Add_{G'}(g)$ modifies $G'$ by (i) adding a new node $u$ subdividing the edge $(u, pr(u))$ (or as new root of $G'$ in the case that $rt(G') = u$), (ii) adding an edge connecting $u$ to a new leaf node denoted $\bar{u}$, (iii) assigning to $u$ a mapping of $pr(s')$ and event type speciation, and (iv) assigning to $\bar{u}$ a mapping of $s'$.  

107
Lemma 21. For any \( g \in G' \) where \( g \in \Theta_R \), the operation \( Add_{G'}(g) \) can be successfully applied.

Proof. Suppose \( g \) has children \( g' \) and \( g'' \), with \((g, g') \in \Xi\). Based on the constructed reconciliation, if node \( g \in \Theta_R \), then \( g \in V(G'_r) \setminus \{rt(G'_r)\} \). Consequently, \( M(g') \in V(S_l) \cup V(S_r) \setminus \{rt(S_l)\} \setminus \{rt(S_r)\} \). Thus, if \( s' = M(g') \), then \( s' \) must have a sibling, say \( s'' \).

Now, since each leaf \( y \in \text{Le}(S_r) \cup \text{Le}(S_l) \), has a mapping from a node in \( G' \), there must be at least one node \( u \) that maps to a node in \( V(S(s'')) \) and for which \( M(pa(u)) \succ_S s'' \). Thus, \( Add_{G'}(g) \) can be successfully applied. \( \square \)

Lemma 22. The final augmented gene tree \( G' \) is a valid DTRL-reconciliation.

Proof. From Lemmas 16 through 19 we know that the mapping and event assignments on \( G \) were valid, and from Lemma 21 we know that each \( Add \) operation itself can be successfully applied. To show that \( G' \) is a valid DTRL-reconciliation it therefore suffices to establish the following: (i) the new internal nodes added through the \( Add \) operations have valid mapping and event assignments, (ii) the parent of each newly added internal node continues to have a valid event and mapping assignment (carried over from \( G \)), and (iii) each replacing transfer event on \( G' \) is associated with a unique lost gene on \( G' \).

Consider any new internal node \( u \) added to \( G \) through a \( Add \) operation. By the definition of an \( Add \) operation, if \( u \) maps to node \( s \) in the \( S \), then one child of \( u \) maps to a node from \( V(S(s'')) \), and the other child of \( u \) maps to node \( s' \), where \( s' \) and \( s'' \) denote the two children of \( s \). Thus, both the mapping assignment and event assignment (of speciation) for \( u \) are valid.

Now, consider the edge \((v, u) \in E(G')\), where \( v = pa(u) \), on which a new internal node \( u \) is added through an \( Add \) operation. Let \( s = M(u), s', s'' \in Ch(s) \), and, consistent with
the definition of an Add operation, \( M(u) \in V(S(s'')) \). Observe that, since \( M(u) < M(v) \), \( v \) could only have been a speciation node in \( G \). Moreover, from the definition of an Add operation we know that \( M(v) \geq_S s \). However, node \( v \) could not map to \( s \), since then the sibling of \( u \) in \( G' \) would map to a node from \( V(S(s')) \). But then, \( s' \) could not have been the recipient of a replacing transfer event, a contradiction. Thus, \( M(v) >_S s \), and so \( v \) remains a valid speciation event in \( G' \) with a valid mapping.

Finally, since an Add operation is performed for each replacing transfer node \( g \) in \( V(G) \) and \( Add_{G'}(g) \) adds a corresponding lost gene copy to the gene tree, each replacing transfer event on \( G' \) is associated with a unique lost gene on \( G' \).  

**Lemma 23.** If \( G' \) denotes the final augmented gene tree, the constructed reconciliation of \( G' \) and \( S \) does not have any gene losses.

**Proof.** From Lemma 20 we know that each node (edge) on \( S \) has at most one gene copy. We also know that each leaf node of the species tree node \( a \in Le(S(S_r)) \cup Le(S(S_l)) \) has a corresponding gene in \( G' \). Thus, if there was ever a loss of a gene copy along any edge of the species tree, it would have to be compensated for by either a gene duplication event or an additive transfer event to ensure that all species descended from that edge still have a copy of the gene. Since the constructed reconciliation of \( G' \) and \( S \) does not have any gene duplications and the only additive transfer does not affect edges of \( S_r \) or \( S_l \), there can not be any losses in the constructed reconciliation.  

The following lemma establishes the forward direction of claim 2.

**Lemma 24.** If there exists a YES answer to the D-rSPR problem on \( \rho \) then there exists a YES answer to the D-DTRL instance \( \delta \) with reconciliation cost at most \( 10n + 5k - 4 \).
Proof. Lemma 22 shows that the constructed reconciliation of $G'$ an $S$ is a valid DTRL-reconciliation, and Lemmas 22 and 23 imply that this reconciliation does not have any losses or duplications. Furthermore, if $|P| \leq k$ then, based on our construction and on Observation 1, $|P'| \leq k$. Thus, the constructed reconciliation of $G'$ and $S$ has at most $2n + k - 2$ nodes that represent replacing transfers, with at most $k$ replacing transfers corresponding to the nodes of $P'$ and exactly $2n - 2$ replacing transfers corresponding to the set $X_G$. Finally, $rt(G)$ represents an additive transfer event. Thus, the reconciliation cost of $G'$ and $S$ is at most $(2n + k - 2) \cdot P_{\Theta_R} + P_{\Theta_A}$ which is $10n + 5k - 4$.  

4.3.4 Proof of Claim 2: Reverse direction

Conversely, we now assume that we have a YES answer to the D-DTRL instance $\delta$ with reconciliation cost at most $10n + 5k - 4$, and will show that there must then exist a solution of size at most $k$ to the D-rSPR instance $\rho$. In this proof, we will first characterize the structure of any optimal DTRL-reconciliation of $G$ and $S$, and then show that this structure implies the existence of a specific set of evolutionary events.

The next three lemmas identify basic properties of any optimal DTRL-reconciliation of $G$ and $S$ and follow easily based on the construction of the gadget. Specifically, the first lemma follows directly from the close correspondence between the topologies of $G_l$ and $S_r$, the second lemma follows from the presence of the 10 extraneous leaves on the path from $rt(S)$ to $S_r$, and the third lemma follows easily from the specific construction of the nodes in $Y_G$ in the gene tree gadget.

**Lemma 25.** Given any optimal DTRL-reconciliation for $G$ and $S$, any internal node $g \in I(G_l)$ must map to $lca_S(L(G(g)))$ and represent a speciation event.
Lemma 26. Given any optimal DTRL-reconciliation for $G$ and $S$, no node of $G$ maps to $rt(S)$.

Lemma 27. Given any optimal DTRL-reconciliation for $G$ and $S$, each node $y \in Y_G$ must represent a replacing transfer event.

The next lemma shows that in any optimal DTRL-reconciliation of $G$ and $S$, the number of gene copies present in any node (or edge) or the species tree is at most 1.

Lemma 28. Given any optimal DTRL-reconciliation for $G$ and $S$, there does not exist any node of $S$ with more than one gene copy.

Proof. Suppose for contradiction that at least one such node $a \in V(S)$ exists. Without loss of generality we can assume that $a$ is the first such node in a post-order traversal of $S$. Since, each leaf node of $S$ has at most one gene copy, $a$ must be an internal node. Thus, let $a'$ and $a''$ denote the two children of $a$. By our assumption, both $a'$ and $a''$ have at most one gene copy, while $a$ has at least 2. Thus, there must be at least one loss along the edge $(a, a')$ and at least one loss along the edge $(a, a'')$. We will show how to modify this current DTRL reconciliation and reduce the total reconciliation cost. For simplicity, we will assume that $a$ has exactly two gene copies, but the proof easily generalizes to greater than two gene copies.

We will modify the current DTRL-reconciliation as follows: Instead of incurring one loss at each of the two children edges of $a$, we move this loss upwards to the edge $(pa(a), a)$, reducing the number of gene copies in $a$ by 1. At least one of $a'$ or $a''$, say $a'$, must have inherited its single gene copy from the surviving gene lineage. Thus, the gene lineage entering $a'$ would be unaffected by the loss of the other copy in $a$. The other child $a''$ may have received its copy from the deleted lineage, and so may be affected by the loss at $a$. This can be resolved by invoking a replacing transfer event to replace the gene lineage coming into $a''$ from $a$ with
the desired gene lineage. Such a replacing transfer can always be added, if it does not already exist, at the parent node of the node from $G$ that maps to $a''$ (or to its closest descendant if no node maps directly to $a''$).

We apply this modification iteratively towards the root of $S$, until the first (or highest) node along this path with the additional gene copy is reached. The source of the additional gene copy at this node must be either a gene duplication or additive transfer event on the gene tree. By removing the extra gene copy at this node, we therefore also reduce the number of gene duplications or additive gene transfers by 1. Overall, during this iterative process, we reduce the number of losses at each iteration by 2, add at most one replacing transfer event per iteration, and replace at least one duplication or additive transfer event by a speciation event during the last iteration. Based on our assigned event costs, this results in a net reduction in the total reconciliation cost. Since the initial DTRL-reconciliation of $G$ and $S$ was optimal, this is a contradiction. Thus, there cannot be any nodes in $S$ with more than one gene copy.

The following corollary follows immediately based on the proof of the previous lemma.

**Corollary 4.3.1.** Given any optimal DTRL-reconciliation for $G$ and $S$, there does not exist any node in $G$ that represents a duplication event.

**Lemma 29.** There are no gene losses in any optimal DTRL-reconciliation for $G$ and $S$.

**Proof.** By Lemma 28 we know that each node of $S$ has at most one gene copy. We also know that each leaf node of the species tree node $a \in \text{Le}(S(S_r)) \cup \text{Le}(S(S_l))$ has a corresponding gene in $G'$. Thus, if there was ever a loss of a gene copy along any edge of the species tree, it would have to be compensated for by either a gene duplication event or an additive transfer.
event to ensure that all species descended from that edge still have a copy of the gene. By Corollary 4.3.1 we know that \( G \) does not have any duplication nodes in any optimal DTRL-reconciliation. Furthermore, since any node of \( S \) has at most one gene copy (Lemma 28), any additive transfer event not the root of \( G \) would either be preceded by a gene loss in the recipient lineage or would be immediately followed by a gene loss so as not to have more than one gene copy in any node of \( S \). Thus, it would be possible to substitute any such additive transfer with a replacing transfer event and reducing the number of gene losses. However, this would lead to a DTRL-reconciliation with lower reconciliation cost, a contradiction. Thus, since the only additive transfer may occur at the root of the gene tree, and there are no gene duplications, there cannot be any gene losses in any optimal DTRL-reconciliation of \( G \) and \( S \).

\( \square \)

**Lemma 30.** Given any optimal DTRL-reconciliation for \( G \) and \( S \), then there is exactly one node that represent additive transfer.

**Proof.** By the proof of Lemma 29 above, we know that the only possible additive transfer node is \( rt(G) \). It therefore suffices to prove that \( G \) must have at least one additive transfer event. By Lemma 26 we know that no node of \( G \) maps to \( rt(S) \), and by Lemma 25 we know that node \( rt(G_l) \) maps to a node of \( V(S_r) \). Without an additive transfer event bringing a copy of the gene to nodes of \( S_l \), the number of gene copies in nodes of \( S_l \) would be zero, a contradiction. \( \square \)

The following lemma establishes the reverse direction of claim 2.

**Lemma 31.** Given any optimal DTRL-reconciliation for \( G \) and \( S \) with cost at most \( 10n + 5k - 4 \), there exists a solution for the D-rSPR instance \( \rho \) of size at most \( k \).
Proof. Based on Lemmas 29 and 30 and Corollary 4.3.1, we know that any optimal DTRL-reconciliation of $G$ and $S$ must invoke exactly one additive transfer, no duplications, and no losses. Thus, since the total reconciliation cost is at most $10n + 5k - 4$, the total number of replacing transfers can be no more than $2n - 2 + k$. Now, by Lemma 27 we know that each of the $2n - 2$ nodes in $Y_G$ must be replacing transfers. Thus, the number of nodes of $Z_G$ that are replacing transfers is at most $k$, and the number of nodes of $Z_G$ that represent speciation events is at least $n - 1 - k$ (since $|Z_G| = n - 1$).

Observe that, according to our gadget, the original tree $T$ from the D-rSPR instance $\rho$ corresponds to subtree $G_r$ of the gene tree and tree $T'$ corresponds to subtree $S_l$ of the species tree. Also observe that if a node from $Z_G$ represents a speciation event then it must map to a node from $S_l$. Therefore, there exist at most $k$ internal nodes of $S_l$ that are recipients of replacing transfer events (since $S_l$ has exactly $n - 1$ internal nodes). Note that the corresponding transfer events on $G$ must all be from $G_r$, and let $A$ denote the set of these corresponding transfer nodes from $G_r$.

Now, consider the forest $F_S$ created from $S_l$ by cutting all edges that connect the at most $k$ nodes that are recipients of replacing transfer events to the rest of $S_l$. Likewise, consider the forest $F_G$ created from $G_r$ by first removing all nodes from $X_G$ and collapsing all nodes with only one child (i.e., all nodes of $Y_G$ are collapsed), and then cutting all edges that connect the nodes of $A$ to the rest of the tree. It is not hard to argue that the two forests $F_S$ and $F_G$ must be identical, which provides a solution of size at most $k$ for the D-rSPR problem on $T$ and $T'$.

$\square$
4.4 Experimental Analysis

There do not currently exist any algorithms or heuristics to compute DTRL reconciliations, and it is not even known how algorithms for computing optimal DTL reconciliations perform when confronted with gene trees that have been affected by both additive and replacing transfers. Therefore, we first focused on answering two fundamental questions: (i) How is the accuracy of DTL reconciliation affected by the presence of replacing horizontal gene transfers? (ii) How well does DTL reconciliation perform at inferring replacing transfer events?

To answer these questions, we used the recently developed simulation framework SaGePhy [45] to stochastically evolve gene trees inside a given species tree under a model that allows for gene duplications, additive transfers, replacing transfers, and gene losses. Using this simulation framework we created a large number of gene trees with varying rates of evolutionary events, computed optimal DTL reconciliations for the gene/species tree pairs, and evaluated the accuracy of the inferred reconciliations by comparing them to the true evolutionary histories of those gene trees. To compute optimal DTL reconciliations we employed the widely-used RANGER-DTL [2, 4] software package.

Simulated datasets. We used our new simulation framework to generate a large number of gene/species tree pairs, with varying rates of evolutionary events. Specifically, we generated 100 species trees, each containing 100 leaves and of height 1, using a birth-death process. Next, inside each of the species trees, we generated three different gene trees using low, medium, and high rates of duplication, additive transfer, replacing transfer, and loss events, resulting in three sets of 100 gene trees. To generate the low DTRL gene trees, we used duplication, additive transfer, replacing transfer, and loss rates of 0.133, 0.133, 0.133, and 0.266, respectively; for
the medium DTRL gene trees we used rates of 0.3, 0.3, 0.3, and 0.6, respectively; and for the high DTRL gene trees we used rates of 0.6, 0.6, 0.6, and 1.2, respectively. Thus, the total transfer rate was twice the duplication rate, with an equal rate of additive and replacing transfers, and the loss rate was assigned to be equal to the sum of the duplication and additive transfer rates. These duplication, transfer, and loss rates are based on rates observed in real data and capture both datasets with lower rates of these events and datasets with a very high rate of these events [5].

For the low DTRL gene trees, the average gene tree leaf set size was 96.11, with an average of 2.37 additive transfers, 2.65 replacing transfers, and 2.19 duplication events per gene tree. For the medium DTRL gene trees, the average gene tree leaf set size was 94.75, with an average of 5.09 additive transfers, 5.01 replacing transfers, and 5.00 duplication events per gene tree. For the high DTRL gene trees, the average gene tree leaf set size was 110.22, with an average of 9.52 additive transfer events, 9.42 replacing transfer events, and 10.39 duplication events per gene tree.

4.4.1 Impact of replacing transfers on DTL reconciliation

We evaluated the accuracy of DTL reconciliation in inferring the evolutionary event and species tree mapping for each internal node in the simulated gene trees. We computed a single optimal reconciliation for each gene tree using RANGER-DTL 2.0 [4] with default parameters and compared the computed reconciliation against the true evolutionary history of that gene tree. We observed very high accuracy for inferring the correct event type (speciation, duplication, or transfer) at each gene tree node. For instance, for the low DTRL gene trees, 99.67%, 96.35% and 96.22% of the gene tree nodes labeled as speciation, duplication, and transfer,
respectively, in the computed reconciliations were inferred correctly. Even for the high DTRL gene trees, these percentages remained very high at 95.69%, 87.49%, and 95.25%, respectively. These results are shown in Figure 12(a).

Looking at the accuracy of mapping inference, we found that 99.09%, 97.11%, and 92.15% of all internal nodes were assigned the correct species node mapping for the low, medium, and high DTRL gene trees, respectively. Detailed results are shown in Figure 12(b).

We compared these results for event and mapping accuracy with results obtained on gene trees simulated with the same overall rates of duplication, transfer, and loss events but in which all simulated transfers were additive transfers (no replacing transfers). We found that the numbers were nearly identical, showing that the presence of replacing transfers does not negatively affect the accuracy of DTL reconciliation itself. For example, for the high DTL gene trees, the percentage of speciation, duplication, and transfer nodes assigned the correct event type was 95%, 81%, and 95%, respectively, and 91% of all nodes were assigned the correct mapping. Note, however, that DTL reconciliation cannot distinguish between additive and replacing transfers, and both types of transfer events are simply inferred as “transfers”.

**Accuracy of inferring replacing transfers.** Next, we performed additional analysis to study if there was any discrepancy in the accuracies of inferring the correct event type (transfer) or mapping for additive transfers and those for replacing transfers. For the low DTRL gene trees, we found that additive transfers were assigned the correct event type 97.05% of the time and the correct mapping 89.45% of the time, while for replacing transfers these numbers were 95.47% and 85.28%, respectively. Likewise, for the medium DTRL gene trees, additive transfers were assigned the correct event type 95.87% of the time and the correct mapping 87.03% of the time, while for replacing transfers these numbers were 93.01% and 81.04%, respectively. For
high DTRL gene trees, these numbers were 95.38% and 75.53% for the additive transfers and 95.12% and 74.52% for the replacing transfers. Overall, this shows that replacing transfers are inferred and mapped with accuracy comparable to that of additive transfers. These results are shown in Parts (c) and (d) of Figure 12.

These results are highly significant and suggest that to design a good heuristic for the DTRL reconciliation problem, it would suffice to first use DTL reconciliation to identify transfer events and then classify that set of transfer events as being either replacing or additive.

4.5 Conclusion

Accurate detection of both replacing and additive transfer events is crucial for understanding horizontal gene transfer in microbes and understanding microbial evolution in general. In this work, we address this problem by formalizing and experimentally studying the DTRL reconciliation framework that simultaneously models gene duplication, loss, and both additive and replacing transfer. Our framework builds upon the traditional DTL reconciliation model and extends it substantially to properly model replacing transfers. We prove that the underlying computational problem is NP-hard, and our proof establishes a close relationship between the rSPR distance problem and DTRL reconciliation. Our experimental results show that DTL reconciliation, which assumes all transfers are additive, is surprisingly robust to the presence of replacing transfer, and suggest that it should be possible to design effective heuristics for the DTRL reconciliation problem based on DTL reconciliation. Going forward, it will be important to develop such improved heuristics for DTRL reconciliation.

An alternative, perhaps complementary, approach for estimating optimal DTRL reconciliations is to make use of available gene order information for the extant species in the analysis.
to classify each transfer event inferred through DTL reconciliation as being either additive or
replacing based on genomic context. However, the applicability of such an approach is limited
since it requires the use of complete genomic information and, due to genome rearrangements,
can only be used for closely related sets of species. It therefore remains important to develop
algorithms for distinguishing between additive and replacing transfers based only on patterns
of phylogenetic incongruence without relying on gene order information.
Figure 10: **Hidden events and augmented gene trees.** Parts (b) and (c) of the figure show two alternative DTRL reconciliations for the gene tree \( G \) and species tree \( S \) shown in Part (a). Each reconciliation shows the augmented gene tree \( G' \), the event type for each internal node in the augmented gene tree, and the red arcs show the mapping for each node of \( G' \) not in \( Le(G) \) (the mapping for each leaf node of \( G \) is implicitly defined by its leaf label). The bold orange edges represent transfer edges. The reconciliation in Part (b) invokes an additive transfer event and a loss event. For this reconciliation in Part (b), \( G' \) is the same as \( G \). The reconciliation in Part (c) invokes a replacing transfer event, a hidden gene duplication event (marked by the blue star), and a loss event. The invisible lineage replaced by the replacing transfer event is shown by the purple dotted line in \( G' \).
Figure 11: This figure illustrates the construction of species tree $S$ (Part (b)) and gene tree $G$ (Part (c)) for D-DTRL problem instance $\delta$ based on trees $T$ and $T'$ (Part (a)) in the input instance $\rho$ of the D-rSPR problem.
Figure 12: Accuracy of DTL reconciliation in the presence of replacing transfers. Part (a) shows the fraction of internal nodes across all low DTRL, medium, DTRL, and high DTRL gene trees, whose event types, speciation, duplication, or transfer, are inferred correctly through DTL reconciliation. Part (b) shows the fraction of internal nodes across all low DTRL, medium, DTRL, and high DTRL gene trees, whose mappings are inferred correctly through DTL reconciliation. Part (c) shows the fraction of additive transfer nodes and replacing transfer nodes across all low DTRL, medium, DTRL, and high DTRL gene trees, that are correctly inferred as transfer events by DTL reconciliation. Part (d) shows the fraction of additive transfer nodes and replacing transfer nodes across all low DTRL, medium, DTRL, and high DTRL gene trees, that are mapped correctly by DTL reconciliation. For each DTRL rate, results are averaged across 100 datasets.

Figure 13: Input gene tree and species tree for illustrating the heuristic.
Chapter 5

Handling Gene Tree Uncertainty in Duplication-Transfer-Loss Reconciliation: Algorithms and Applications

5.1 Introduction

Understanding the evolution of gene families is fundamental to understanding how genes and genomes evolve. One of the most useful techniques for studying gene family evolution is phylogenetic reconciliation, which involves the comparison of a gene tree, i.e., a reconstructed evolutionary tree for the gene family of interest, with the corresponding species tree. Phylogenetic reconciliation shows how the gene family (gene tree) evolved inside the species tree and also specifies the evolutionary events, such as gene duplications and losses, that shaped the evolution of that gene family. In case of microbial gene families, three of the most important evolutionary events are gene duplications, losses, and horizontal gene transfer, giving rise to the Duplication-Transfer-Loss (DTL) reconciliation problem. Given its importance to understanding microbial evolution, the DTL reconciliation problem has been widely studied [2, 3, 17–19, 25, 44, 49, 51, 57, 59, 60, 63–65, 67].
It is well known that gene trees can be very hard to reconstruct accurately, and there is often considerable uncertainty in gene tree topologies, e.g., [5, 53]. However, the accuracy of phylogenetic reconciliation depends on the accuracy of the gene tree used, e.g., [5, 57]. One solution for this problem is to use a probabilistic framework for phylogenetic reconciliation and integrate over the uncertainty in gene tree topologies, e.g., [60, 65]. However, such probabilistic frameworks have high computational requirements and typically require the estimation of several parameters that can be hard to estimate. As a result, DTL reconciliations are frequently computed using a parsimony framework, e.g., [20, 29, 30, 34, 38, 55], where costs are assigned to gene duplication, transfer, and loss events and the goal is to find a reconciliation that has the smallest reconciliation cost (sum of costs for all invoked events) [2, 3, 17–19, 25, 44, 49, 51, 57, 59, 63, 67]. To address the problem of gene tree error, several methods have been developed for error-correcting gene trees based on the parsimonious DTL reconciliation model [5, 17, 57]. These methods compute an error-corrected gene tree that is often much more accurate than a gene tree constructed using sequence data alone. However, gene trees computed using these error-correction methods still contain errors and are typically not identical to the true gene tree topologies [5, 57]. As such, there continues to be considerable uncertainty in gene tree topologies even after gene tree error correction.

A standard approach for handling gene tree uncertainty in a parsimony-based reconciliation framework is to collapse all weakly supported edges in the reconstructed gene tree, resulting in a non-binary gene tree, and then reconcile this non-binary gene tree with the species tree [11, 33, 41, 43, 44, 46, 71]. To properly reconcile a non-binary gene tree with a species tree, one must compute all optimal binary resolutions of the non-binary gene tree that result in lowest reconciliation cost with the species tree. However, the problem of computing optimal gene tree
resolutions under DTL reconciliation has been shown to be NP-hard, even for generating only one optimal binary resolution of a given non-binary gene tree [43]. More recently, an exact exponential-time algorithms to generate all optimal binary resolutions has been developed for this problem [41, 44] that can handle non-binary gene trees with small maximum out-degrees (up to 10). Overall, there are two major limitations of DTL reconciliation with non-binary gene trees. First, the time complexity of any algorithm to compute all optimal binary resolutions is, necessarily, exponential in the number of non-binary nodes in the given non-binary gene trees and in their out-degrees. And second, the problem formulation ignores sequence information and can therefore consider gene tree resolutions for which there is no sequence support, leading to over-fitting to the species tree and potentially incorrect inferences. Thus, existing solutions for DTL reconciliation with non-binary gene trees are both computationally infeasible and prone to yielding erroneous reconciliations due to over-fitting.

**Our contributions.** In this work, we propose an alternative formulation for the problem of DTL reconciliation with non-binary gene trees that addresses both these limitations. The idea is to consider only those optimal binary resolutions that have some support in the sequence data. More precisely, in addition to the non-binary gene tree and species tree, we take as input a set of bootstrap replicates or samples from the posterior of a Bayesian analysis for the gene tree, and then consider only those binary resolutions of the gene tree whose clades appear in at least one of the given gene tree samples/bootstrap replicates. This constraint not only makes the problem of DTL reconciliation with non-binary gene trees more tractable, but also ensures that the generated binary resolutions are biologically meaningful. Moreover, consideration of sample/bootstrap “support” allows for the ordering of optimal binary resolution by their overall support values, providing a way to distinguish between binary resolutions that are otherwise
equally optimal. We point out that the idea of using bootstrap replicates or other gene tree samples to constrain search space has been applied successfully to gene tree error-correction before [17, 57, 65]. However, it has never been used for DTL reconciliation with non-binary gene trees.

We present a self-adaptive algorithm for the constrained DTL reconciliation problem with non-binary gene trees that produces an ordered list of binary resolutions ordered by average support value. Our algorithm is self-adaptive in that it can automatically increase or decrease the search space by considering only those clades that appear in at least a certain fraction of the sampled gene trees (by default, the considered clades should appear in at least one of the sampled gene trees). This self-adaptability is required because, even with the constraints imposed by the gene tree samples/bootstrap replicates, the number of optimal resolutions can grow exponentially in the degree and number of non-binary nodes in the given non-binary gene tree. By dynamically increasing or decreasing the minimum support value required for the clades considered, the algorithm is guaranteed to be very efficient even on very large and highly non-binary gene trees while still maintaining its accuracy. Since the final number of optimal resolutions computed by the algorithm can be very large, the ordering of binary resolutions by average support value makes it possible to identify a set of only the most highly supported binary resolutions, in lieu of considering all optimal binary constrained resolutions. We also explicitly consider the option of outputting just a single, most highly supported estimate of the gene tree. This effectively yields an error-corrected version of the gene tree, and we find that our approach yields remarkably accurate point estimates of gene tree topologies.

We applied our algorithm to both empirical and simulated datasets in a thorough experimental study and demonstrate that: (i) results obtained using constrained DTL reconciliation
problem with non-binary gene trees are more accurate than results obtained using the traditional (unconstrained) formulation of the problem, (ii) the self-adaptive algorithm is highly scalable and efficient and can be easily applied to large genome-scale data sets with very large gene trees, (iii) by taking the consensus across a set of optimal binary resolutions of the gene tree, one can effectively distinguish between correct and incorrect clades in the gene tree, and (iv) our approach for identifying a single, most highly supported binary resolution is highly effective at gene tree error correction and often results in more accurate estimates of gene tree topologies than the state-of-the-art gene tree error-correction method TreeFix-DTL [5] at a fraction of the runtime.

To summarize, the new problem formulation and algorithms described in this work make it possible to systematically handle gene tree uncertainty in DTL reconciliation in a computationally efficient and biologically meaningful manner. Furthermore, we find that our approach for point estimation of gene tree topologies, i.e., gene tree error-correction yields highly accurate gene trees. The remainder of this paper is organized as follows. The next section introduces basic definitions, preliminaries, and problem formulations. Algorithmic details appear in Section 5.3 and experimental results in Section 5.4. Concluding remarks appear in Section 5.5.

5.2 Definitions and Preliminaries

We follow basic definitions and notation from [2] and [44]. Given a tree $T$, we denote its node, edge, and leaf sets by $V(T)$, $E(T)$, and $Le(T)$ respectively.

If $T$ is rooted, the root node of $T$ is denoted by $rt(T)$, the parent of a node $v \in V(T)$ by $pa_T(v)$, its set of children by $Ch_T(v)$, and the (maximal) subtree of $T$ rooted at $v$ by $T(v)$. The set of internal nodes of $T$, denoted $I(T)$, is defined to be $V(T) \setminus Le(T)$. For a rooted tree $T$,
we define \( \leq_T \) to be the partial order on \( V(T) \) where \( x \leq_T y \) if \( y \) is a node on the path between \( rt(T) \) and \( x \). The partial order \( \geq_T \) is defined analogously, i.e., \( x \geq_T y \) if \( x \) is a node on the path between \( rt(T) \) and \( y \). We say that \( y \) is an ancestor of \( x \), or that \( x \) is a descendant of \( y \), if \( x \leq_T y \) (note that, under this definition, every node is a descendant as well as ancestor of itself). We say that \( x \) and \( y \) are incomparable if neither \( x \leq_T y \) nor \( y \leq_T x \). Given a non-empty subset \( L \subseteq \text{Le}(T) \), we denote by \( \text{lca}_T(L) \) the last common ancestor (LCA) of all the leaves in \( L \) in tree \( T \).

A rooted tree is binary if all of its internal nodes have exactly two children, and non-binary otherwise. An internal edge is an edge whose end points are both internal nodes in the tree. An internal edge \((x, \text{pa}_T(x))\) in tree \( T \) can be contracted by removing \((x, \text{pa}_T(x))\) and creating new edges joining \( \text{pa}_T(x) \) with \( \text{Ch}_T(x) \), thereby yielding a new tree distinct from \( T \). We say that a tree \( T' \) is a binary resolution of \( T \) if \( T' \) is binary and \( T \) can be obtained from \( T' \) by contracting some (zero or more) internal edges. We denote by \( \mathcal{BR}(T) \) the set of all binary resolutions of a rooted non-binary tree \( T \). Given any node \( x \) from \( T \), we define the out-degree of \( x \) to be the total number of children of \( x \).

For a rooted tree \( T \) each node \( v \in V(T) \), the clade \( C_T(v) \) is defined to be the set of all leaf nodes in \( T(v) \); i.e. \( C_T(v) = \text{Le}(T(v)) \). We denote the set of all clades of a rooted tree \( T \) by \( \text{Clade}(T) \). This concept can be extended to unrooted trees as follows. Suppose \( T \) is an unrooted tree. Each edge \((u, v) \in E(T)\) defines a partition of the leaf set of \( T \) into two disjoint subsets \( \text{Le}(T_u) \) and \( \text{Le}(T_v) \), where \( T_u \) is the subtree containing node \( u \) and \( T_v \) is the subtree containing node \( v \), obtained when edge \((u, v)\) is removed from \( T \). We call \( \text{Le}(T_u) \) and \( \text{Le}(T_v) \) the clusters of \( T \) induced by edge \((u, v)\), and denote the set of all clusters in an unrooted tree \( T \) by \( \text{Cluster}(T) \).
In this work, we will consider both rooted and unrooted trees. However, unless otherwise specified, the term tree refers to a rooted tree.

A species tree is a tree that depicts the evolutionary relationships of a set of species. Given a gene family from a set of species, a gene tree is a tree that depicts the evolutionary relationships among the sequences encoding only that gene family in the given set of species. Thus, the nodes in a gene tree represent genes. Gene trees may be either binary or non-binary while the species tree is always assumed to be binary. Throughout this work, we denote the gene tree and species tree under consideration by $G$ and $S$, respectively. If $G$ is restricted to be binary we refer to it as $G^B$ and as $G^N$ if it is restricted to be non-binary. We assume that each leaf of the gene tree is labeled with the species from which that gene was sampled. This labeling defines a leaf-mapping $L_{G,S}: Le(G) \rightarrow Le(S)$ that maps a leaf node $g \in Le(G)$ to that unique leaf node $s \in Le(S)$ that has the same label as $g$. Note that gene trees may have more than one gene sampled from the same species, and that the species tree must contain all species represented in the gene tree.

### 5.2.1 Reconciliation and DTL-scenarios

A binary gene tree can be reconciled with a species tree by mapping the gene tree into the species tree. Next, we define what constitutes a valid reconciliation; specifically, we define a Duplication-Transfer-Loss scenario (DTL-scenario) [2,67] for $G^B$ and $S$ that characterizes the mappings of $G^B$ into $S$ that constitute a biologically valid reconciliation. Essentially, DTL-scenarios map each gene tree node to a unique species tree node and designate each gene tree node as representing either a speciation, duplication, or transfer event.
**Definition 5.2.1** (DTL-scenario). A DTL-scenario for $G^B$ and $S$ is a seven-tuple $(L, M, \Sigma, \Delta, \Theta, \Xi, \tau)$, where $L : \text{Le}(G^B) \rightarrow \text{Le}(S)$ represents the leaf-mapping from $G^B$ to $S$, $M : V(G^B) \rightarrow V(S)$ maps each node of $G^B$ to a node of $S$, the sets $\Sigma$, $\Delta$, and $\Theta$ partition $I(G^B)$ into speciation, duplication, and transfer nodes respectively, $\Xi$ is a subset of gene tree edges that represent transfer edges, and $\tau : \Theta \rightarrow V(S)$ specifies the recipient species for each transfer event, subject to the following constraints:

1. If $g \in \text{Le}(G^B)$, then $M(g) = L(g)$.

2. If $g \in I(G^B)$ and $g'$ and $g''$ denote the children of $g$, then,

   (a) $M(g) \not< S M(g')$ and $M(g) \not< S M(g'')$.

   (b) At least one of $M(g')$ and $M(g'')$ is a descendant of $M(g)$.

3. Given any edge $(g, g') \in E(G^B)$, $(g, g') \in \Xi$ if and only if $M(g)$ and $M(g')$ are incomparable.

4. If $g \in I(G^B)$ and $g'$ and $g''$ denote the children of $g$, then,

   (a) $g \in \Sigma$ only if $M(g) = \text{lca}(M(g'), M(g''))$ and $M(g')$ and $M(g'')$ are incomparable.

   (b) $g \in \Delta$ only if $M(g) \geq S \text{lca}(M(g'), M(g''))$.

   (c) $g \in \Theta$ if and only if either $(g, g') \in \Xi$ or $(g, g'') \in \Xi$.

   (d) If $g \in \Theta$ and $(g, g') \in \Xi$, then $M(g)$ and $\tau(g)$ must be incomparable, and $M(g')$ must be a descendant of $\tau(g)$, i.e., $M(g') \leq S \tau(g)$.

DTL-scenarios correspond naturally to reconciliations and it is straightforward to infer the reconciliation of $G^B$ and $S$ implied by any DTL-scenario. Given a DTL-scenario $\alpha$, one can
directly count the minimum number of gene losses, \( \text{Loss}_\alpha \), in the corresponding reconciliation [2].

**Definition 5.2.2 (Losses).** Given a DTL-scenario \( \alpha = \langle \mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau \rangle \) for \( G \) and \( S \), let \( g \in V(G) \) and \( \{g', g''\} = \text{Ch}(g) \). The number of losses \( \text{Loss}_\alpha(g) \) at node \( g \), is defined to be:

- \( |d_S(\mathcal{M}(g), \mathcal{M}(g')) - 1| + |d_S(\mathcal{M}(g), \mathcal{M}(g'')) - 1| \), if \( g \in \Sigma \),
- \( d_S(\mathcal{M}(g), \mathcal{M}(g')) + d_S(\mathcal{M}(g), \mathcal{M}(g'')) \), if \( g \in \Delta \), and
- \( d_S(\mathcal{M}(g), \mathcal{M}(g'')) + d_S(\tau(g), \mathcal{M}(g')) \) if \( (g, g') \in \Xi \).

We define the total number of losses in the reconciliation corresponding to the DTL-scenario \( \alpha \) to be \( \text{Loss}_\alpha = \sum_{g \in I(G)} \text{Loss}_\alpha(g) \).

Let \( P_\Delta \), \( P_\Theta \), and \( P_{\text{loss}} \) denote the non-negative costs associated with duplication, transfer, and loss events, respectively. The reconciliation cost of a DTL-scenario is defined as follows.

**Definition 5.2.3 (Reconciliation cost).** Given a DTL-scenario \( \alpha = \langle \mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau \rangle \) for \( G^B \) and \( S \), the reconciliation cost associated with \( \alpha \) is given by \( R_\alpha = P_\Delta \cdot |\Delta| + P_\Theta \cdot |\Theta| + P_{\text{loss}} \cdot \text{Loss}_\alpha \).

A most parsimonious reconciliation is one that has minimum reconciliation cost.

**Definition 5.2.4 (MPR).** Given \( G^B \) and \( S \), along with \( P_\Delta \), \( P_\Theta \), and \( P_{\text{loss}} \), a most parsimonious reconciliation (MPR) for \( G^B \) and \( S \) is a DTL-scenario with minimum reconciliation cost.

5.2.2 Optimal gene tree resolution

To reconcile a non-binary gene tree \( G^N \) with the species tree, one must find a binary resolution \( G^B \) of \( G^N \) such that an MPR of \( G^B \) with \( S \) has smallest reconciliation cost. Moreover,
since there may be more than one optimal binary resolution of $G^N$, the desired formulation of the problem is to find all optimal resolutions of $G^N$. This leads to the following computational problem [44].

**Problem 8** (OGTR-All). Given $G^N$ and $S$, along with $P_\Delta$, $P_\Theta$, and $P_{\text{loss}}$, the All Optimal Gene Tree Resolutions (OGTR-All) problem is to compute the set $\mathcal{OR}(G^N)$ of all optimal binary resolutions of $G^N$ such that, for any $G^B \in \mathcal{OR}(G^N)$, an MPR of $G^B$ and $S$ has the smallest reconciliation cost among all gene trees in $\mathcal{BR}(G^N)$.

The OGTR-All problem is known to be NP-hard [43] (even for computing a single optimal resolution), and existing algorithms are limited to solving instances in which the maximum out-degree in $G^N$ is small [44].

### 5.2.3 Constrained optimal gene tree resolution

In addition to its very high computational time complexity, which greatly limits its applicability, the OGTR-All problem ignores sequence information and is therefore prone to over-fitting the gene tree to the species tree. Both these limitations can be addressed by constraining the set of binary resolutions of $G^N$ that can be considered. Specifically, the goal is to allow all binary resolutions that are supported by the sequence data, and disallow those that are unsupported. To achieve this goal we define a constrained version of the OGTR-All problem, in which, in addition to $G^N$ and $S$, we take as input a set of unrooted gene trees that define constraints on the set of binary resolutions of $G^N$. The set of unrooted gene trees used should represent a sample of gene tree topologies supported by the sequence data and can be easily obtained by either computing bootstrap replicates or sampling from the posterior distribution in a Bayesian analysis.
More formally, let $B = \{B_1, B_2, \ldots, B_b\}$ denote a sample of $b$ unrooted gene trees. Then, we define the cluster set $B$ to be

$$
\text{Cluster}(B) = \bigcup_{i=1}^{b} \text{Cluster}(B_i).
$$

This set of clusters is used to define the constrained set of binary resolutions as follows.

**Definition 5.2.5** (Constrained binary resolution). Given $B$ and a non-binary tree $T$, we say that $T'$ is a constrained binary resolution of $T$ (with respect to $B$), if $T' \in BR(T)$ and $\text{Clade}(T') \subseteq \text{Cluster}(B)$. We denote by $CBR(T)$ the set of all constrained binary resolutions of a rooted non-binary tree $T$.

We can now state the constrained optimal gene tree resolution problem.

**Problem 9** (C-OGTR). Given $G^N$, $S$, and $B$, along with $P_\Delta$, $P_\Theta$, and $P_{\text{loss}}$, the All Constrained Optimal Gene Tree Resolutions (C-OGTR) problem is to compute the set $\text{COR}(G^N)$ of all optimal constrained binary resolutions of $G^N$ such that, for any $G^B \in \text{COR}(G^N)$, an MPR of $G^B$ and $S$ has the smallest reconciliation cost among all gene trees in $CBR(G^N)$.

Note: To ensure that a solution always exists to the C-OGTR problem, we include the original binary gene tree from which $G^N$ is obtained in the set $B$. This ensures, that a constrained binary resolution of $G^N$ always exists.

We also define a variant of the problem above that only seeks to find a single optimal reconciliation with highest average clade support.

**Problem 10** (C-OGTR-Best). Given $G^N$, $S$, and $B$, along with $P_\Delta$, $P_\Theta$, and $P_{\text{loss}}$, the Best Constrained Optimal Gene Tree Resolutions (C-OGTR-Best) problem is to compute a tree $G^B \in CBR(G^N)$ such that the total number of occurrences in $B$ of all clades in $G^B$ is the largest among all trees in $CBR(G^N)$.
Observe that one can easily define variants of the \textit{C-OGTR} and \textit{C-OGTR-Best} problems by restricting \textit{Cluster}(\mathcal{B}) to only contain those clusters that are present in at least a certain number, \textit{minSup}, of the samples in \mathcal{B}. Such variants will be considered later in this manuscript. Furthermore, recall that our solution for the \textit{C-OGTR} problem actually outputs a list of optimal binary resolutions sorted by average support value (i.e., by the total number of occurrences in \mathcal{B} of all clades in that binary resolution). Thus, by solving the \textit{C-OGTR} problem we also implicitly solve the \textit{C-OGTR-Best} problem. However, we still distinguish between the two problem formulations since \textit{C-OGTR-Best} represents a problem formulation of independent interest and, furthermore, it can be solved more efficiently than the \textit{C-OGTR} problem.

5.3 Algorithmic Details

Our algorithms for the \textit{C-OGTR} and \textit{C-OGTR-Best} problems are built directly upon the dynamic programming algorithms for the \textit{OGTR} and \textit{OGTR-All} problems described in [44]. In the following, we first give a high-level description of the main dynamic programming framework used to solve \textit{OGTR} and \textit{OGTR-All} [44], and then describe the specific changes required to solve the \textit{C-OGTR} and \textit{C-OGTR-Best} problems.

5.3.1 Overview of the dynamic programming framework

The dynamic programming algorithms for the \textit{OGTR} and \textit{OGTR-All} problems described in [44] are built upon the efficient dynamic programming programming framework for the DTL reconciliation problem with binary gene trees [2]. Given any \( g \in V(G) \) and \( s \in V(S) \), where \( G \) is binary, let \( c(g, s) \) denote the cost of an optimal reconciliation of the subtree \( G(g) \) with \( S \) under that constraint that \( g \) maps to \( s \). The dynamic programming algorithm for binary gene
trees perform a nested post-order traversal of $G$ and $S$, and computes the value of $c(g, s)$ for each $g \in V(G)$ and $s \in V(S)$. To initialize the dynamic programming table, we set for each $g \in \text{Le}(G)$, $c(g, s) = 0$ if $s = \mathcal{M}(g)$, and $c(g, s) = \infty$ otherwise. The value of $c(g, s)$, for any $g \in I(G)$, can then be efficiently computed based on the values $c(g', \cdot)$ and $c(g'', \cdot)$, where $g'$ and $g''$ are the two children of $g$, previously computed. Once all the $c(\cdot, \cdot)$ value are computed, the minimum reconciliation of $G$ and $S$ is simply $\min_{s \in V(S)} c(\text{rt}(G), s)$. Further details appear in [2].

Algorithms for the $OGTR$ and $OGTR-All$ problems use the same dynamic programming framework. If a node $g \in G^N$ is binary then the computation proceeds as before, while if $g \in G_n$ is non-binary then they first consider all possible binary resolutions of node $g$ and assign $c(g, s)$ to be the minimum value over all these possible binary resolutions. Our algorithms for $C-OGTR$ and $C-OGTR-Best$ follow the same framework, except that the possible binary resolutions considered at each non-binary node are limited to those that can be constructed from the clusters available in $\text{Cluster}(\mathcal{B})$.

We point out that the task of keeping track of and enumerating all optimal binary resolutions within this dynamic programming framework is non-trivial and is complicated by the fact that the same optimal resolutions can have many different optimal reconciliations, which means the same resolution can counted and enumerated multiple times as part of different reconciliations. Our algorithm for the $C-OGTR$ problem inherits the algorithmic solutions developed for keeping track of all optimal resolutions under the $OGTR-All$ problem [44]. Also, note that even with the constraint on possible binary resolutions, the total number of optimal binary resolutions computed by the $C-OGTR$ problem can be exponential in the number of non-binary nodes in $G^N$ and in its maximum out-degree.
Next, we show how our algorithms generate all permitted resolutions for any non-binary gene tree node.

### 5.3.2 Generating permitted resolutions

Starting with the sample $B = \{B_1, B_2, \ldots, B_b\}$ of $b$ unrooted gene trees, the algorithms first compute the set $\text{Cluster}(B) = \bigcup_{i=1}^b \text{Cluster}(B_i)$ and assign to each cluster $x \in \text{Cluster}(B)$ a support value equal to the fraction of trees from $B$ that contain cluster $x$. Note that we always include the original binary tree $G$ from which $G^N$ is obtained as one of the trees in $B$. Then, for each cluster $x \in \text{Cluster}(B)$, we identify all pairs of clusters $(y, z)$, where $y, z \in \text{Cluster}(B)$, such that $y \cap z = \emptyset$ and $y \cup z = x$. Once these preprocessing steps are completed, the algorithms can easily construct all possible permitted resolutions for any non-binary node $g \in G^N$ as follows: First, find the cluster $\text{Le}(G^N(g))$ in $\text{Cluster}(B)$ and identify all pairs $(y, z)$ associated with that cluster. For each candidate pair $(y, z)$, check if the partition of $\text{Le}(G^N(g))$ implied by $(y, z)$ is consistent with a partitioning of the children $\text{Ch}_{G^N}(g)$ of $g$; if it is, then binarize $g$ by assigning the roots of the clades corresponding to clusters $y$ and $z$ as its two children and recurse on these two smaller clades until no more non-binary nodes remain.

It is easy to see that this recursive process is guaranteed to generate all permitted binary resolutions at node $g$. Moreover, since we include the original binary tree $G$ from which $G^N$ is obtained as one of the trees in $B$, there is guaranteed to be at least one permitted binary resolution for each non-binary node of $G^N$. 
5.3.3 Self-adaptive approach for limiting search space

As mentioned above, despite the restriction on permitted resolutions imposed by $B$, the total number of optimal binary resolutions can be exponential in the number of non-binary nodes of $G^N$ as well as its maximum out-degree. To address this limitation, our algorithm employs a novel self-adaptive approach to limit the number of binary resolutions considered at each non-binary node. To describe the self-adaptive approach, we need some additional definitions and notation. We first define an upper bound, denoted $U$, on the total number of binary resolutions considered by the algorithm during any step in its execution. For example, for all the experimental results presented in the next section, we assigned $U = 25000$. We also define the following:

$$\text{Cluster}(B, \text{minSup}) = \{ x \in \text{Cluster}(B) \mid x \text{ appears in at least minSup trees from } B \}.$$ 

Finally, define $N(g, \text{minSup})$ to be the number of distinct binary resolutions of the non-binary node $g \in G^N$ permitted by the cluster set $\text{Cluster}(B, \text{minSup})$. For each non-binary node $g \in G^N$ independently, the algorithms compute a value for $\text{minSup}$ for which $N(g, \text{minSup}) \leq U$ but $N(g, \text{minSup} - 1) > U$. This can be accomplished efficiently through a binary-search in the range $[1, |B|]$. Thus, at each non-binary node of the gene tree, we limit the total number of resolutions considered to at most $U$ of the most highly supported ones.

5.3.4 Ordering of binary resolutions by average clade support

In addition to its use for limiting the number of possible resolutions at each non-binary node, the upper bound $U$ is also used to bound the total number of resolutions considered at the subtree rooted at each node of the gene tree. In other words, our algorithm for C-OGTR
always limits the total number of resolutions of the subtree $G^N(g)$ stored at any node $g$ of the
gene tree to $U$. In particular, at each node $g \in G^N$ the algorithm only stores up to the $U$ best
(in terms of average clade support) resolutions for the subtree $G^N(g)$ encountered during the
search, ordered by their average clade support. We denote this ordered list of the $U$ best resolu-
tions for the subtree $G^N(g)$ by $\mathcal{ORV}(g)$ (for optimal resolution vector). Note that each resolu-
tion stored in $\mathcal{ORV}(g)$ also has an associated average clade support value stored along with it.

Next, we describe how each $\mathcal{ORV}(\cdot)$ is computed as part of the bottom-up dynamic program-
ing traversal of $G^N$. We first need some additional notation. Given any binary or non-binary
node $g \in G^N$, define the set of nearest non-binary descendants of $g$, denoted $\mathcal{N}(g)$, to be $\{h \in
V(G^N(g)) \setminus \{g\} | h \text{ is non-binary and no other non-binary nodes exist on the path from } g \text{ to } h\}$. Note that $\mathcal{N}(g)$ may be empty.

Consider any binary or non-binary node $g \in G^N$. If all nodes in the subtree $G^N(g)$ are
binary then there is only one resolution possible (i.e., the current resolution). If $\mathcal{N}(g) = \emptyset$ but $g$ itself is non-binary then we apply the self-adaptive approach described above and compute
up to $U$ binary resolutions of $G^N(g)$. These resolutions are then sorted according to decreasing
average clade support (based on the trees in $B$) and stored as $\mathcal{ORV}(g)$. If $\mathcal{N}(g) \neq \emptyset$ and $g$ is
binary, then $\mathcal{ORV}(g)$ can be computed by suitably combining the vectors $\mathcal{ORV}(h)$, for each $h \in \mathcal{N}(g)$, already computed in previous steps of the algorithm. Observe that each combi-
nation of resolutions from the $\mathcal{ORV}(h)$’s, across all $h \in \mathcal{N}(g)$, yields a permitted resolution
for the subtree $G^N(g)$. Since each $\mathcal{ORV}(h)$ is in sorted order and each resolution is associ-
ated with its average clade support value, computing the $U$ best resolutions for $G^N(g)$, i.e.,
computing $\mathcal{ORV}(g)$, can be accomplished by performing a merge-like procedure (from merge
sort) on the $\mathcal{ORV}(h)$’s to identify just the $U$ best resolutions for $G^N(g)$. The remaining case,
where $\mathcal{N}(g) \neq \emptyset$ and $g$ is non-binary can be handled similarly by considering the sorted list of permitted resolutions for node $g$ together with the $\mathcal{ORV}(h)$’s.

5.3.5 Algorithm optimization for $C$-$OGTR$-$Best$

Our solution for the $C$-$OGTR$ problem implicitly solves the $C$-$OGTR$-$Best$ problem since the computed optimal resolutions are ordered by their clade supports. However, to only compute the “best”, i.e., highest average clade support, resolution, we need not maintain the $\mathcal{ORV}(\cdot)$ vectors and only need to save the best resolution corresponding to each subproblem $c(g, s)$.

5.4 Experimental Evaluation

5.4.1 Preliminaries

Data sets used in the analysis

To evaluate our new approach we used a large simulated data set of 1200 gene tree/species tree pairs on 50 taxa used in [5] to evaluate the accuracy of the state-of-the-art gene tree error-correction method $TreeFix-DTL$. These 1200 gene trees represent 12 categories (each with 100 gene trees) that capture a wide range of evolutionary scenarios. Specifically, the data sets represent all combinations of (i) low, medium, and high rates of duplication, transfer, and loss events, (ii) two different sequence mutation rates (rate 5 and rate 10), and (iii) normal (333 amino acids) and short (173 amino acids) sequence lengths; further details on these data sets are available in [5]. For each of the 1200 gene tree/species tree pairs in this data set, we have available the true (simulated) gene tree and species tree, the reconstructed maximum likelihood
gene tree (constructed using RAxML on sequence data simulated down the true gene tree), and 100 bootstrap replicates computed during the execution of RAxML.

In addition to the simulated data set above, we also used a real biological data set of over 4700 gene trees from 100 predominantly prokaryotic species [17]. This data set was used in [44] to study the scalability of the \textit{OGTR-ALL} problem formulation and algorithm. Here we use this biological data set for the same purpose, demonstrating that \textit{C-OGTR} is far more scalable than \textit{OGTR-ALL} and easily applicable to the entire data set.

**Experimental setup**

For all runs of \textit{OGTR-All}, \textit{C-OGTR}, \textit{C-OGTR-Best}, and \textit{TreeFix-DTL}, we used event costs of 1, 2, and 3, for losses, duplications, and transfers, respectively. To create the non-binary gene trees for \textit{OGTR-All}, \textit{C-OGTR}, and \textit{C-OGTR-Best}, we used two different bootstrap cutoff values: 50\% and 90\%. The cutoff values were applied to the reconstructed RAxML gene trees, and the resulting trees were used as input to \textit{OGTR-All}, \textit{C-OGTR}, and \textit{C-OGTR-Best}. The 100 bootstrap replicates were used to define the corresponding set \(B\). Observe that using higher bootstrap cutoff values results in more non-binary (i.e., more unresolved) gene trees as more edges are collapsed. For the real biological data set, which we used to evaluate scalability, we used 500 bootstrap replicates per gene tree and a 90\% support value cutoff value.

We focused our experimental analysis on comparing the scalability, runtime, and accuracy of \textit{C-OGTR} (and \textit{C-OGTR-Best}) against \textit{OGTR-All} and \textit{TreeFix-DTL}, and on demonstrating the utility of enumerating multiple optimal resolutions, rather than just a single “best” resolution, using \textit{C-OGTR}. To measure gene tree accuracy, we used the normalized Robinson-Foulds
distance (NRFD) against the true gene tree; for any reconstructed gene tree, the NRFD shows the percentage of splits in that gene tree that do not appear in the corresponding true gene tree.

**Basic statistics on data sets**

For the 12 simulated data sets, the average leaf set size of the low, medium, and high DTL gene trees was 52.33, 70.37, and 91.26, respectively. Upon collapsing weakly supported edges with the 90% cutoff threshold, we found that the average number of non-binary nodes and average of maximum out-degrees across all 12 simulated data sets were 11.07 and 10.6, respectively, with the highest averages obtained for the sequence length 173, rate-10, high DTL datasets at 13.8 and 20.58, respectively. For the gene trees obtained after collapsing edges at the 50% bootstrap cutoff threshold, we found that the average number of non-binary nodes and average of maximum out-degrees across all 12 simulated data sets were a more modest 4.5 and 4.78, respectively, with the highest averages obtained for the sequence length 173, rate-10, high DTL datasets at 7.42 and 6.97, respectively.

For the real data set of 4736 gene trees, we found that 4419 became non-binary at a 90% bootstrap cutoff threshold. For these 4419 non-binary gene trees, the average leaf set size was 36.1, and the average number of non-binary nodes and average out-degrees were 3.35 and 21.14.
Figure 14: **Limited applicability of OGTR-All.** Number of gene trees, for each of the 12 simulated datasets, on which OGTR-All could be applied (i.e., with maximum out-degree less than 9). Results are shown for both 50% and 90% bootstrap cutoff values. C-OGTR and C-OGTR-Best were applicable on all 100 gene trees in each simulated dataset for both cutoff values.
Figure 15: **Runtime comparison against OGTR-All.** Average runtime (in seconds, on a log scale) for *OGTR-All* and *C-OGTR* over the simulated datasets for both 50% and 90% bootstrap cutoff values. For each mutation rate, sequence length, and bootstrap cutoff value, runtimes are averaged over the corresponding Low, Medium, and High DTL datasets. Runtimes are only shown for the subset of gene trees on which *OGTR-All* was applicable.
5.4.2 Results

Running time and scalability

As expected, we found that our new \textit{C-OGTR} (and \textit{C-OGTR-Best}) problem formulation is not only far more scalable but also dramatically faster than the \textit{OGTR-All} problem formulation. As Figure 14 shows, for the simulated data sets, \textit{OGTR-All} could only be applied to about a quarter of the cutoff 90%, sequence length 173 gene trees and only to about half of the cutoff 90% gene trees overall. In contrast, both \textit{C-OGTR} and \textit{TreeFix-DTL} were applicable to all 100 gene trees in each of the 12 data sets for both 90% and 50% cutoff thresholds. Furthermore, as Figure 15 shows, even restricting only to the gene trees to which \textit{OGTR-All} could be applied, \textit{C-OGTR} was dramatically faster than \textit{OGTR-All} on all datasets, with an overall average running time of 431 seconds for \textit{OGTR-All} and 22 seconds for \textit{C-OGTR} (on a laptop computer with a 3.1 GHz Intel Core i7 processor and 16 GB of RAM, using a single core). \textit{TreeFix-DTL} is also far slower than \textit{C-OGTR}, requiring an average of over an hour for each simulated data set [5].

Finally, on the 4419 non-binary gene trees in the real biological data set, \textit{C-OGTR} required an average of only 56 seconds per gene tree, demonstrating the applicability of \textit{C-OGTR} and \textit{C-OGTR-Best} to large genome-scale analyses.

Accuracy of inferred gene tree resolutions

We compared the accuracy of \textit{C-OGTR-Best} to \textit{OGTR-All} on the subset of simulated data sets on which \textit{OGTR-All} could be applied. Since, the \textit{OGTR-All} formulation offers no way to differentiate between different optimal resolutions, we chose one optimal resolution at random for our analysis. As Figure 16 shows, \textit{C-OGTR-Best} results in significantly more accurate gene
tree resolutions than OGTR-All across all 12 simulated data sets, and both C-OGTR-Best and OGTR-All trees are substantially more accurate than the base RAxML gene trees. The average normalized Robinson-Foulds distances (NRFD) for RAxML, OGTR-All, and C-OGTR-Best across all 12 simulated data sets are 10.61%, 7.11%, and 4.64%, respectively.

We also compared the accuracy of C-OGTR-Best to the state-of-the-art gene tree error-correction method TreeFix-DTL. As Figure 17 shows, C-OGTR-Best results in more accurate gene tree resolutions than TreeFix-DTL across 5 of the 12 simulated data sets and comes close to matching the accuracy of TreeFix-DTL for the other 7 data sets. This is particularly significant because C-OGTR-Best is orders of magnitude faster than TreeFix-DTL. Interestingly, the data sets on which C-OGTR-Best tends to outperform TreeFix-DTL are the ones that correspond to medium and high DTL. The average normalized Robinson-Foulds distances for RAxML, TreeFix-DTL, and C-OGTR-Best across all 12 simulated data sets are 11.61%, 5.57%, and 5.95%, respectively.

**Impact of enumerating multiple optimal resolutions**

To explore the impact of enumerating multiple optimal resolutions through C-OGTR instead of simply using the “best” resolution computed through C-OGTR-Best, we computed the false positive and false negative branch rates for the strict consensus of all C-OGTR gene trees and for the C-OGTR-Best gene tree. Figures 18 and 19 show the normalized false positive and false negative rates, respectively, for the 12 simulated data sets with a 90% cutoff threshold. As Figure 18 shows, the strict consensus of all C-OGTR gene trees results in a significantly lower false positive rate compared to the C-OGTR-Best gene trees across each of the 12 simulated data sets, with an overall average of 4.74% versus 5.95%, respectively. This suggests that the
Figure 16: **Accuracy of C-OGTR-Best versus OGTR-All.** Error rates in terms of the normalized Robinson-Foulds distance (NRFD) are shown for gene trees inferred using RAxML, OGTR-All, and C-OGTR-Best on the 12 simulated datasets. All results are for bootstrap cutoff value 90%. For OGTR-All a randomly selected optimal resolution was used.

C-OGTR problem formulation can be effective at distinguishing between correct and incorrect gene tree edges. Unsurprisingly, as Figure 19 shows, this improvement in the false positive rate comes at the expense of an increased false negative rate, with the average normalized false negative rate over all 12 data sets for C-OGTR and C-OGTR-Best being 10.39% and 5.95%, respectively.
Figure 17: **Accuracy of C-OGTR-Best versus TreeFix-DTL.** Error rates in terms of the normalized Robinson-Foulds distance (NRFD) are shown for gene trees inferred using RAxML, C-OGTR-Best, and TreeFix-DTL on the 12 simulated datasets. All results are for bootstrap cutoff value 90%.

### 5.5 Discussion and Conclusion

In this work, we provided new problem formulations, C-OGTR and C-OGRT-Best, and associated efficient algorithms for the problem of DTL reconciliation with non-binary gene trees that address the limitations of existing approaches and also serve as powerful tools for gene tree error-correction. Our experimental results on simulated and real biological data sets demonstrate the significantly improved accuracy and scalability of our new algorithms, as well as their potential for use for gene tree error-correction.
Figure 18: **False positive rate for C-OGTR versus C-OGTR-Best.** False positive branch rates are shown for the strict consensus of all gene tree resolutions generated by C-OGTR as well as for the C-OGTR-Best gene tree, averaged across all 100 gene trees for each of the 12 simulated datasets. All results are for bootstrap cutoff value 90%.
Figure 19: **False negative rate for C-OGTR versus C-OGTR-Best.** False negative branch rates are shown for the strict consensus of all gene tree resolutions generated by C-OGTR as well as for the C-OGTR-Best gene tree, averaged across all 100 gene trees for each of the 12 simulated datasets. All results are for bootstrap cutoff value 90%.
A number of interesting directions remain to be explored. For example, it would be useful to study the impact of using multiple optimal resolutions, generated by \textit{C-OGTR}, on DTL reconciliation by distinguishing between highly and weakly supported aspects of the reconciliation. Similarly, a particularly interesting observation about our algorithm for \textit{C-OGTR-Best} is that it tends to outperform the best existing gene tree error-correction methods on data sets with higher rates of evolutionary events (such as duplications, transfers, and losses). This aspect of \textit{C-OGTR-Best} is worth exploring and developing further.
Chapter 6

Conclusions

In this thesis we resolved an open question about the computational complexity of computing optimal Duplication-Transfer-Loss reconciliations with non-binary gene trees and provided new Fixed parameter, exact, and heuristic algorithms for this problem. We also proposed a new Duplication-Additive-Transfer-Replacing-Transfer-Loss (DTRL) reconciliation framework and proved that the underlying computational problem is NP-hard.

The DTL reconciliation problem has been studied extensively [2, 3, 14, 17–19, 25, 42, 49, 51, 59, 63, 67]. However, all previous works had some limitations that affect accuracy and generality of the framework in practice. My Ph.D. research focused on the two most important limitations, (1) assumption that the gene tree is binary and (2) assumption that all transfer events are additive.

We first provide new problem formulations for DTL reconciliation for non-binary gene tree and analyzed the computational complexities of the new formulations. In particular, we proved that this problem is NP-hard.
Next, we provided new fixed parameter, exact, and heuristic algorithms for this problem. Specifically, we defined the Optimal Gene Tree Resolution OGTR and OGTR-All problems which seek to find one and all optimal binary resolutions of non-binary gene tree that minimize the DTL reconciliation cost, respectively. We show OGTR is fixed-parameter tractable (FPT) in the maximum degree of the gene tree. In addition, based on the FPT algorithm we develop an algorithm to track and enumerate all optimal binary resolutions OGTR-All of an unresolved input gene tree. With these new algorithms and techniques, for the first time, it was possible to apply DTR-reconciliation on the non-binary gene tree and also calculate an effect of gene trees uncertainly on reconciliation accuracy. However, both FPT and exact algorithms, ignored sequence information and considered the gene tree topologies that are not supported by sequence. To overcome this over-fitting problem, we have proposed an alternative formulation for optimal resolution problem. Specifically, we defined C-OGTR and C-OGTR-Best, which uses bootstrap replicates to constrain search space and find all binary resolutions that are supported by the sequence data, and disallow those that are unsupported, and also order the optimal binary resolution by their overall support values. Comparing results of C-OGTR-Best and OGTR-All on simulated data shows, that not only C-OGTR-Best is faster and applicable on more gene trees, but also more accurate in inferring gene tree topology.

Finally we have proposed a new Duplication-Transfer-Replacing Transfer-Loss(DTRL) reconciliation framework that explicitly models both additive and replacing transfer events, along with gene duplications and losses, and investigate the complexity of computing optimal DTRL reconciliation. Specifically, we introduced the DTRL reconciliation framework and proved that the underlying computational problem is NP-hard. Also, we perform the first experimental study to evaluate the effect of replacing transfer events on the accuracy of the
traditional DTL reconciliation model (which assumes that all transfers are additive). Our sim-
ulation shows, although the problem of computing optimal DTRL reconciliation is NP-hard,
it should be possible to design effective heuristics algorithm based on simpler and solvable
DTL-reconciliation problem.
Bibliography


