Exact Algorithms for Duplication-Transfer-Loss Reconciliation with Non-Binary Gene Trees

Misagh Kordi and Mukul S. Bansal

Abstract—Duplication-Transfer-Loss (DTL) reconciliation is a powerful method for studying gene family evolution in the presence of horizontal gene transfer. DTL reconciliation seeks to reconcile gene trees with species trees by postulating speciation, duplication, transfer, and loss events. Efficient algorithms exist for finding optimal DTL reconciliations when the gene tree is binary. In practice, however, gene trees are often non-binary due to uncertainty in the gene tree topologies, and DTL reconciliation with non-binary gene trees is known to be NP-hard.

In this paper, we present the first exact algorithms for DTL reconciliation with non-binary gene trees. Specifically, we (i) show that the DTL reconciliation problem for non-binary gene trees is fixed-parameter tractable in the maximum degree of the gene tree, (ii) present an exponential-time, but in-practice efficient, algorithm to track and enumerate all optimal binary resolutions of a non-binary input gene tree, and (iii) apply our algorithms to a large empirical data set of over 4700 gene trees from 100 species to study the impact of gene tree uncertainty on DTL-reconciliation and to demonstrate the applicability and utility of our algorithms. The new techniques and algorithms introduced in this paper will help biologists avoid incorrect evolutionary inferences caused by gene tree uncertainty.

Keywords: Gene family evolution, gene-tree/species-tree reconciliation, non-binary trees, gene duplication, horizontal gene transfer, fixed-parameter tractable.

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 [15], [30], reconstruction of ancestral gene content [6], [8], and accurate gene tree and species tree construction [11], [30], [4], [24], [3]. The DTL reconciliation problem has therefore been widely studied, e.g., [13], [10], [22], [29], [8], [1], [27], [2], [25], [20], [9], [7], [17].

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 [29], [23]. Thus, in practice, the goal is often to find an optimal (not necessarily time-consistent) DTL-reconciliation [29], [8], [1], [2], [20], and this problem can be solved in $O(mn)$ time [1], 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], [10] 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 [26]. 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 [5], [11], [18], [31], with the most efficient of these algorithms having an optimal $O(m + n)$ time complexity [31]. However, the corresponding problem for DTL reconciliation has recently been shown to be NP-hard [17], and, to the best of our knowledge, no algorithms, heuristic or otherwise, currently exist for DTL reconciliation

- *Misgah Kordi is with the Department of Computer Science & Engineering at the University of Connecticut, Storrs, USA.* misagh.kordi@uconn.edu
- *Mukul S. Bansal is with the Department of Computer Science & Engineering and the Institute for Systems Genomics at the University of Connecticut, Storrs, USA.* mukul.bansal@uconn.edu

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 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 paper 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.

A preliminary version of this work appeared in the proceedings of ACM-BCB 2016 [16]. The current manuscript substantially expands upon [16] and contains a more detailed exposition, proofs for all lemmas and theorems, additional technical and algorithmic details, additional experimental analyses and results, and several new figures.

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 5. The next section introduces basic definitions and preliminaries. The FPT algorithm is presented in Section 3, the enumeration algorithm in Section 4, and experimental results in Section 6. Concluding remarks appear in Section 7.

2 Definitions and Preliminaries

We follow the basic definitions and notation from [1] and [17]. 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

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 [22] and NOTUNG [27], 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).

$x \leq_T y$ (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 $\mathcal{BR}(T)$ the set of all binary resolutions of a 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.

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* $\mathcal{L}_{G,S}\colon Le(G) \rightarrow Le(S)$ 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.

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) [29], [1] 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 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 $\mathcal{L}\colon Le(G^B) \rightarrow Le(S)$ represents the leaf-mapping from G^B to S, $\mathcal{M}\colon V(G^B) \rightarrow V(S)$ maps each node of G^B to a node of S, the sets Σ, Δ, and Θ partition $I(G^B)$ into speciation, duplication, and transfer nodes respectively, Ξ is a subset of gene tree edges that represent transfer edges, and $\tau\colon \Theta \rightarrow V(S)$ specifies the recipient species for each transfer event, subject to the following constraints:*

1) *If $g \in Le(G^B)$, 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) *$\mathcal{M}(g) \not<_S \mathcal{M}(g')$ and $\mathcal{M}(g) \not<_S \mathcal{M}(g'')$,*
   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 $\mathcal{M}(g) = lca(\mathcal{M}(g'), \mathcal{M}(g''))$ and $\mathcal{M}(g')$ and $\mathcal{M}(g'')$ are incomparable,*
   b) *$g \in \Delta$ only if $\mathcal{M}(g) \geq_S lca(\mathcal{M}(g'), \mathcal{M}(g''))$,*
   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., $\mathcal{M}(g') \leq_S \tau(g)$.*

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 1 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 [29], [28]. Given a DTL-scenario α, one can directly count the minimum number of gene losses, $Loss_\alpha$, in the corresponding reconciliation [1].

Definition 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:*

- *$|d_S(\mathcal{M}(g), \mathcal{M}(g')) - 1| + |d_S(\mathcal{M}(g), \mathcal{M}(g'')) - 1|$, 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 α to be $Loss_\alpha = \sum_{g \in I(G)} Loss_\alpha(g)$.

Let P_Δ, P_Θ, 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 2.3 (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 α is given by $\mathcal{R}_\alpha = P_\Delta \cdot |\Delta| + P_\Theta \cdot |\Theta| + P_{loss} \cdot Loss_\alpha$.*

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

Definition 2.4 (MPR). *Given G^B and S, along with P_Δ, P_Θ, and P_{loss}, a* most parsimonious reconciliation (MPR) *for G^B and S is a DTL-scenario with minimum reconciliation cost.*

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

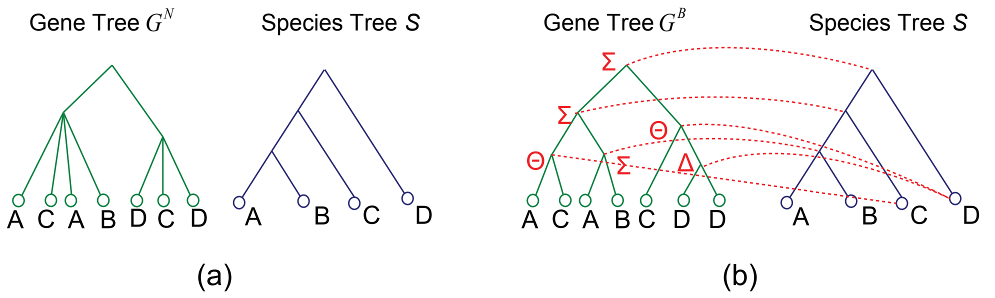


Fig. 1. **DTL reconciliation and OGTR problem.** Part (a) shows a non-binary gene tree G^N with two unresolved nodes and a binary species tree S. Part (b) shows a DTL reconciliation between a possible binary resolution G^B of G^N and species tree S. The dotted arcs show the mapping $\mathcal{M}$ (with 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.

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 1. This yields the following problem.

Problem 1 (OGTR)**.** *Given G^N and S, along with P_Δ, P_Θ, 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 2 (OGTR-All)**.** *Given G^N and S, along with P_Δ, P_Θ, 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 $\mathcal{BR}(G^N)$.*

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_Δ, P_Θ, and P_{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 straightforward 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 [29], [1]. 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 $g \in Le(G)$, $c(g,s) = 0$ 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 $\min_{s \in V(S)} c(rt(G),s)$.

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 [1] 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 $s \in V(S)$,

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

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 $\mathcal{BR}_G(g)$ denote the set of all binary resolutions of the (partial) subtree of G formed by g and its children. Consider any $H \in \mathcal{BR}_G(g)$. 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 $H \in \mathcal{BR}_G(g)$, 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:

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

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:

Algorithm *OGTR-FPT*$(G, S, \mathcal{L})$

```
1:  for each $g \in V(G)$ and $s \in V(S)$ do
2:      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:      Initialize $c(g,\mathcal{L}(g))$ to 0.
5:  for each $g \in I(G)$ in post-order do
6:      if $g$ is a binary node then
7:          for each $s \in V(S)$ in post-order do
8:              Let $\{g', g''\} = Ch_G(g)$.
9:              if $s \in Le(S)$ then
10:                 $c_\Sigma(g,s) = \infty$.
11:                 $c_\Delta(g,s) = P_\Delta + c(g',s) + c(g'',s)$.
12:                 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:                 $c(g,s) = \min\{c_\Sigma(g,s), c_\Delta(g,s), c_\Theta(g,s)\}$.
14:             else
15:                 Let $\{s', s''\} = Ch_S(s)$.
16:                 $c_\Sigma(g,s) = \min\{in(g',s') + in(g'',s''),\ in(g'',s') + in(g',s'')\}$.
17:                 $c_\Delta(g,s) = P_\Delta + \min\{in(g',s) + in(g'',s)\}$.
18:                 If $s \neq rt(S)$, then $c_\Theta(g,s) = P_\Theta + \min\{in(g',s)+out(g'',s),\ in(g'',s)+out(g',s)\}$.
19:                 $c(g,s) = \min\{c_\Sigma(g,s), c_\Delta(g,s), c_\Theta(g,s)\}$.
20:     if $g$ is a non-binary node then
21:         for each $H \in \mathcal{BR}_G(g)$ do
22:             for each $h \in Le(H)$ do
23:                 for each $s \in V(S)$ do
24:                     Initialize $c^H(h,s)$ to $c(h,s)$.
25:             for each $h \in I(H)$ in post-order do
26:                 for each $s \in V(S)$ in post-order do
27:                     Let $\{h', h''\} = Ch_H(h)$.
28:                     if $s \in Le(S)$ then
29:                         $c^H_\Sigma(h,s) = \infty$.
30:                         $c^H_\Delta(h,s) = P_\Delta + c^H(h',s) + c(h'',s)$.
31:                         If $s \neq rt(S)$, then $c^H_\Theta(h,s) = P_\Theta + \min\{in(h',s) + out(h'',s),\ in(h'',s) + out(h',s)\}$.
32:                         $c^H(h,s) = \min\{c^H_\Sigma(h,s), c^H_\Delta(h,s), c^H_\Theta(h,s)\}$.
33:                     else
34:                         Let $\{s', s''\} = Ch_S(s)$.
35:                         $c^H_\Sigma(h,s) = \min\{in(h',s') + in(h'',s''),\ in(h'',s') + in(h',s'')\}$.
36:                         $c^H_\Delta(h,s) = P_\Delta + \min\{in(h',s) + in(h'',s)\}$.
37:                         If $s \neq rt(S)$, then $c^H_\Theta(h,s) = P_\Theta + \min\{in(h',s) + out(h'',s),\ in(h'',s) + out(h',s)\}$.
38:                         $c^H(h,s) = \min\{c^H_\Sigma(h,s), c^H_\Delta(h,s), c^H_\Theta(h,s)\}$.
39:             for each $s \in V(S)$ in post-order do
40:                 if $c^H(g,s) < c(g,s)$ then
41:                     $c(g,s) = c^H(g,s)$.
42: 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 Hs 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 [1]. 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 *OGTR-FPT* correctly solves the OGTR problem, and that it can be implemented to run in time $O(2^{k(\log_2 2k)} \cdot l \cdot n + mn)$.

Theorem 3.1. *The OGTR problem can be solved in* $O(2^{k(\log_2 2k)} \cdot l \cdot n + mn)$ *time.*

Proof. We first prove the correctness of Algorithm *OGTR-FPT* and then analyze its time complexity.

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 $\{g', g''\} = Ch_G(g)$. 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 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 $H \in \mathcal{BR}_G(g)$ (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 $c(g, s) = \min_{H \in \mathcal{BR}_G(g)} c^H(g, s)$ ("for" loop of step 39), as required by the definition of $c(g, s)$.

Induction completes the proof.

Complexity: It has previously been shown [1] that the values *in*$(\cdot, \cdot)$ and *out*$(\cdot, \cdot)$ can be computed in $O(1)$ time per value by computing them incrementally as part of the nested post-order traversal. Details on their computation are omitted (for clarity) from the pseudocode of Algorithm *OGTR-FPT* above, and we refer the reader to [1] for details. For our analysis, we will assume that any particular *in*$(\cdot, \cdot)$ and *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 $O(2^{k(\log_2 2k)} \cdot l \cdot n + mn)$. □

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 programming 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 [2], 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* ρ_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 $\{\rho_i \colon G_i \in \mathcal{OR}(G)\}$. 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 ρ 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 ρ.

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 $\mathcal{R}^{\mathcal{F}}_x(g, s)$ 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 $\{x_i \oplus y_j \colon 1 \leq i \leq a \text{ and } 1 \leq j \leq b\}$.

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 4.1. *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*

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

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

Let $N(G(g_1)) = \{h^1_1, h^1_2, \ldots, h^1_{|N(G(g_1))|}\}$ and $N(G(g_2)) = \{h^2_1, h^2_2, \ldots, h^2_{|N(G(g_2))|}\}$. Note that $N(G(g)) = N(G(g_1)) \cup N(G(g_2)) = \{h^1_1, h^1_2, \ldots, h^1_{|N(G(g_1))|}, h^2_1, h^2_2, \ldots, h^2_{|N(G(g_2))|}\}$. Consider any $\rho \in \mathcal{ORV}(G(g), s)$, and let H denote the particular binary resolution of $G(g)$ corresponding to ρ. Let $\rho = \langle r_{a(1)}(h^1_1), r_{a(2)}(h^1_2), \ldots, r_{a(|N(G(g_1))|)}(h^1_{|N(G(g_1))|}), r_{b(1)}(h^2_1), r_{b(2)}(h^2_2), \ldots, r_{b(|N(G(g_2))|)}(h^2_{|N(G(g_2))|})\rangle$, where $a(1), \ldots, a(|N(G(g_1))|)$ are the specific resolution numbers for the non-binary nodes $h^1_1, \ldots, h^1_{|N(G(g_1))|}$, 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 $h^2_1, \ldots, h^2_{|N(G(g_2))|}$, respectively, corresponding to $H(g_2)$. Finally, let ρ_1 and ρ_2 be the resolution vectors for $H(g_1)$ and $H(g_2)$, respectively; i.e., $\rho_1 = \langle r_{a(1)}(h^1_1), r_{a(2)}(h^1_2), \ldots, r_{a(|N(G(g_1))|)}(h^1_{|N(G(g_1))|})\rangle$ and $\rho_2 = \langle r_{b(1)}(h^2_1), r_{b(2)}(h^2_2), \ldots, r_{b(|N(G(g_2))|)}(h^2_{|N(G(g_2))|})\rangle$.

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 α. Under α, suppose g_1 map to node $s_1 \in V(S)$ and g_2 map 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 $\rho_1 \in \mathcal{ORV}(G(g_1), s_1)$ and $\rho_2 \in \mathcal{ORV}(G(g_2), s_2)$, otherwise α would not be an MPR. This proves that $\rho \in \bigcup_{\langle s_1, s_2\rangle \in \mathcal{F}(g,s)} \mathcal{ORV}(G(g_1), s_1) \otimes \mathcal{ORV}(G(g_2), s_2)$.

To prove the converse, consider any $\langle s_1, s_2\rangle \in \mathcal{F}(g, s)$. By the definition of $\mathcal{F}(G, S)$, there exists some $\rho \in \mathcal{ORV}(G(g), s)$ such that there exists an MPR α 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 $\rho_1 \in \mathcal{ORV}(G(g_1), s_1)$ and $\rho_2 \in \mathcal{ORV}(G(g_2), s_2)$. Now, consider any $\rho'_1 \in \mathcal{ORV}(G(g_1), s_1)$ and $\rho'_2 \in \mathcal{ORV}(G(g_2), s_2)$, and let $\rho' = \rho'_1 \oplus \rho'_2$. Observe that, since $c^{\rho'_1}(g_1, s_1) = c^{\rho_1}(g_1, s_1)$, $c^{\rho'_2}(g_2, s_2) = c^{\rho_2}(g_2, s_2)$, 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 $\rho' = \rho'_1 \oplus \rho'_2$ such that $\rho'_1 \in \mathcal{ORV}(G(g_1), s_1)$ and $\rho'_2 \in \mathcal{ORV}(G(g_2), s_2)$, where $\langle s_1, s_2\rangle \in \mathcal{F}(g, s)$, we must have $\rho' \in \mathcal{ORV}(G(g), s)$, proving the converse. □

Lemma 4.2. *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*

$$\mathcal{ORV}(G(g), s) = \bigcup_{\langle s_1, s_2, \ldots, s_p\rangle \in \mathcal{F}(g,s)} \quad \bigcup_{r \in \mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g,s)} \mathcal{ORV}(G(g_1), s_1) \otimes \mathcal{ORV}(G(g_2), s_2) \otimes \ldots \otimes \mathcal{ORV}(G(g_p), s_p) \otimes r.$$

Proof. This proof follows along the lines of the proof for Lemma 4.1 above. We will first show that if $\rho \in \mathcal{ORV}(G(g), s)$ then $\rho \in \bigcup_{\langle s_1, s_2, \ldots, s_p\rangle \in \mathcal{F}(g,s)} \bigcup_{r \in \mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g,s)} \mathcal{ORV}(G(g_1), s_1) \otimes \mathcal{ORV}(G(g_2), s_2) \otimes \ldots \otimes \mathcal{ORV}(G(g_p), s_p) \otimes r$, and then the converse.

Let $N(G(g_i)) = \{h^i_1, h^i_2, \ldots, h^i_{|N(G(g_i))|}\}$. Note that $N(G(g)) = \cup_{2 \le i \le p} N(G(g_i)) \cup \{g\} = \{h^1_1, h^1_2, \ldots, h^1_{|N(G(g_1))|}, \ldots, h^p_1, h^p_2, \ldots, h^p_{|N(G(g_p))|}, g\}$. Consider any $\rho \in \mathcal{ORV}(G(g), s)$, and let H denote the particular binary resolution of $G(g)$ corresponding to ρ. Let $\rho_i = \langle r_{a_i(1)}(h^i_1), r_{a_i(2)}(h^i_2), \ldots, r_{a_i(|N(G(g_i))|)}(h^i_{|N(G(g_i))|})\rangle$ be the resolution vector for $H(g_i)$, where $a_i(1), \ldots, a_i(|N(G(g_1))|)$ are the specific resolution numbers for the non-binary nodes $h^i_1, \ldots, h^i_{|N(G(g_i))|}$, respectively, corresponding to $H(g_i)$. Then, $\rho = (\bigoplus_{1 \le i \le p} \rho_i) \oplus r$, where r 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 α. Under α, suppose g_i maps to node $s_i \in V(S)$, for $1 \le i \le p$. Then, by definition, $\langle s_1, s_2, \ldots, s_p\rangle \in \mathcal{F}(g, s)$ and $r \in \mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g, s)$. Moreover, for each $i \in \{1, \ldots, p\}$, we must have $\rho_i \in \mathcal{ORV}(G(g_i), s_i)$, otherwise α would not be an MPR. This proves that $\rho \in \bigcup_{\langle s_1, s_2, \ldots, s_p\rangle \in \mathcal{F}(g,s)} \bigcup_{r \in \mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g,s)} \mathcal{ORV}(G(g_1), s_1) \otimes \mathcal{ORV}(G(g_2), s_2) \otimes \ldots \otimes \mathcal{ORV}(G(g_p), s_p) \otimes r$.

To prove the converse, consider any $\langle s_1, s_2, \ldots, s_p\rangle \in \mathcal{F}(g, s)$ and $r \in \mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g, s)$. By the definitions of $\mathcal{F}(G, S)$ and $\mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g, s)$, there exists some $\rho \in \mathcal{ORV}(G(g), s)$ such that there exists an MPR α of the corresponding resolution, under the constraint that g maps to s, in which g_i maps to s_i for each $i \in \{1, \ldots, p\}$. As shown in the first part of this proof, we must have $\rho_i \in \mathcal{ORV}(G(g_i), s_i)$, for each $i \in \{1, \ldots, p\}$. Now, consider any $\rho'_i \in \mathcal{ORV}(G(g_i), s_i)$, where $1 \le i \le p$, and any $r \in \mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g, s)$, and let $\rho' = (\bigoplus_{1 \le i \le p} \rho'_i) \oplus r$. 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 $\rho' = (\bigoplus_{1 \le i \le p} \rho'_i) \oplus r$ such that $\rho'_i \in \mathcal{ORV}(G(g_i), s_i)$ for each $i \in \{1, \ldots, p\}$, $\langle s_1, s_2, \ldots, s_p\rangle \in \mathcal{F}(g, s)$, and $r \in \mathcal{R}^{\mathcal{F}}_{\langle s_1, s_2, \ldots, s_p\rangle}(g, s)$, we must have $\rho' \in \mathcal{ORV}(G(g), s)$, proving the converse. □

The next lemma shows how to compute $\mathcal{ORV}(G)$ based on the previously computed sets $\mathcal{ORV}(G, \cdot)$.

Lemma 4.3. *Let A be the set $\{s \in V(S)\colon c(rt(G), s) = \min_{s' \in V(S)} c(rt(G), s')\}$. Then, $\mathcal{ORV}(G) = \bigcup_{s \in A} \mathcal{ORV}(G, s)$.*

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

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

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

Lemma 4.4. *Given any binary node $g \in V(G)$, let $g', g'' \in V(G)$ be such that $g = lca_G(\{g', g''\})$, $g', g'' \neq g$, 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 $\langle s', s'' \rangle$ such that there exists an optimal resolution $H \in \mathcal{BR}(G(g))$, and a most parsimonious reconciliation (MPR) of H with S in which g' maps to s' and g'' maps to s''. Then, $\mathcal{ORV}(G(g), s) = \bigcup_{\langle s', s'' \rangle \in X} \mathcal{ORV}(G(g'), s') \otimes \mathcal{ORV}(G(g''), s'')$.*

Proof. The proof of this lemma is almost identical to the proof for Lemma 4.1. We will first show that if $\rho \in \mathcal{ORV}(G(g), s)$ then $\rho \in \bigcup_{\langle s', s'' \rangle \in X} \mathcal{ORV}(G(g'), s') \otimes \mathcal{ORV}(G(g''), s'')$, 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 ρ. Let $\rho = \langle r_{a(1)}(h'_1), r_{a(2)}(h'_2), \ldots, r_{a(|N(G(g'))|)}(h'_{|N(G(g'))|}),$ $r_{b(1)}(h''_1), r_{b(2)}(h''_2), \ldots, r_{b(|N(G(g''))|)}(h''_{|N(G(g''))|}) \rangle$, 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 ρ' and ρ'' 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 α. Under α, suppose g' maps to node $s' \in V(S)$ and g_2 maps to node $s'' \in V(S)$. Then, by definition, $\langle s', s'' \rangle \in X$. Moreover, we must have $\rho' \in \mathcal{ORV}(G(g'), s')$ and $\rho'' \in \mathcal{ORV}(G(g''), s'')$, otherwise α would not be an MPR. This proves that $\rho \in \bigcup_{\langle s', s'' \rangle \in X} \mathcal{ORV}(G(g'), s') \otimes \mathcal{ORV}(G(g''), s'')$.

To prove the converse, consider any $\langle s', s'' \rangle \in X$. By the definition of X, there exists some $\rho \in \mathcal{ORV}(G(g), s)$ such that there exists an MPR α 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 $c^{\nu'}(g', s') = c^{\rho'}(g', s')$, $c^{\nu''}(g'', s'') = c^{\rho''}(g'', s'')$, we must have $c^{\nu}(g, s) = c^{\rho}(g, s)$. This implies that $\nu \in \mathcal{ORV}(G(g), s)$. Thus, we have shown that, given any $\nu = \nu' \oplus \nu''$ such that $\nu' \in \mathcal{ORV}(G(g'), s')$ and $\nu'' \in \mathcal{ORV}(G(g''), s'')$, where $\langle s', s'' \rangle \in X$, we must have $\nu \in \mathcal{ORV}(G(g), s)$, proving the converse. □

The enumeration algorithm is based on Lemmas 4.1 through 4.4 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 $\mathcal{ORV}(G(g), s)$ based on Lemmas 4.1 through 4.4. A more precise description of the algorithm follows:

Algorithm *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 $\mathcal{F}(g, s)$ and $\mathcal{ORV}(G(g), s)$ to $\emptyset$.
4:  Initialize $\mathcal{ORV}(G)$ 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 $\mathcal{F}(g, s)$.
13:             Compute $\mathcal{ORV}(G(g), s)$ according to the equation of Lemma 4.1
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 $H \in \mathcal{BR}_G(g)$ do
18:                 Compute $c^H(g, s)$ as in Algorithm OGTR-FPT.
19:                 if $c^H(g, s) \le c(g, s)$ then
20:                     $c(g, s) = c^H(g, s)$.
21:                     Update $\mathcal{F}(g, s)$.
22:                     Let $r$ be the resolution number corresponding to resolution $H$.
23:                     Set $\mathcal{ORV}(G(g), s) = \bigcup_{\langle s_1, s_2, \ldots, s_p \rangle \in \mathcal{F}(g,s)}$
                        $\mathcal{ORV}(G(g_1), s_1) \otimes \mathcal{ORV}(G(g_2), s_2) \otimes \ldots \otimes$
                        $\mathcal{ORV}(G(g_p), s_p) \otimes r$.
24: Let $A = \{s \in V(S)\colon c(rt(G), s) = \min_{s' \in V(S)} c(rt(G), s')\}$.
25: for each $s \in A$ do
26:     Set $\mathcal{ORV}(G) = \bigcup_{s \in A} \mathcal{ORV}(G, s)$.
27: Return $\mathcal{ORV}(G)$.
```

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 4.4. Next, we first show how to compute the sets $\mathcal{F}(g, s)$ (Steps 27 and 21 from Algorithm *OGTR-Enumerate*) and then show how Lemma 4.4 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 *OGTR-FPT*). If g is binary, as in Step 27 of Algorithm *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 $in(g',\cdot)$ and $out(g',\cdot)$ (resp. $in(g'',\cdot)$ and $out(g'',\cdot)$) 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 *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 $in(g',s') + in(g'',s'')$ (and not from the other choice) in Step 16 , and that the value of $c_\Theta(g,s)$ is obtained from $in(g',s) + out(g'',s)$ (and not from the other choice) in Step 18 of Algorithm *OGTR-FPT*. Now, let A be the set $\{x \in V(S(s')) \colon P_{loss} \cdot d_S(s',x) + c(g',x) = in(g',s')\}$, B be the set $\{x \in V(S(s'')) \colon P_{loss} \cdot d_S(s'',x) + c(g'',x) = in(g'',s'')\}$, C be the set $\{x \in V(S(s)) \colon P_{loss} \cdot d_S(s,x) + c(g',x) = in(g',s)\}$, and D be the set $\{x \in V(S)$ incomparable to $s \colon c(g'',x) = out(g'',s)\}$. 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 *OGTR-FPT*. Let $\mathcal{F}^H(g,s)$ denote the value of $\mathcal{F}(g,s)$ computed for a particular resolution, $H \in \mathcal{BR}_G(g)$, of g. As we consider all the different ways of resolving the node g (i.e., the different $H \in \mathcal{BR}_G(g)$) 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 $\mathcal{F}(g,s)$ is then simply equal to $\cup_{H \in A} \mathcal{F}^H(g,s)$. 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 $H \in \mathcal{BR}_G(g)$, not all of the nodes $g_1, g_2, \ldots, g_p$ will be children of g in H. Thus, to compute the set $\mathcal{F}^H(g,s)$, for any given $H \in \mathcal{BR}_G(g)$, 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 4.4. Given any $g \in I(G)$ and $s \in V(S)$, the computation of $\mathcal{ORV}(G(g),s)$ is one of the most computationally intensive steps of Algorithm *OGTR-Enumerate*. Lemma 4.4 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 $g' <_G g$ 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 $g', g'' <_G g$ 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 4.4 makes it possible to skip the computation of $\mathcal{ORV}(G(g),s)$ 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 4.4 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 $\mathcal{ORV}(G(g),s)$, for each $s \in V(S)$, for *category-1* and *category-3* nodes (and also for $rt(G)$), as shown in Lemmas 4.2 and 4.4. Note that the set X, as defined in Lemma 4.4, can be computed similarly to how $\mathcal{F}(g,s)$ is computed for non-binary nodes g, as described in the previous paragraph.

Theorem 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 3.1, all $c(g,s)$ values are computed correctly. The sets $\mathcal{ORV}(G(g),s)$, for each $g \in V(G)$ and $s \in V(S)$, are computed in accordance with Lemmas 4.1 and 4.2 in Steps 13 and 23. Finally, the set $\mathcal{ORV}(G(g),s)$ is computed according to Lemma 4.3 in Steps 24 through 26. The correctness of Algorithm *OGTR-Enumerate* follows. □

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 $\mathcal{F}(g,s)$ and $\mathcal{ORV}(G(g),s)$, 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 $\mathcal{F}(g,s)$, 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.

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 [10], 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* [10] 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 [10]. 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 5.1. *The OGTR problem with dated DTL reconciliation can be solved in $O(2^{k(\log_2 2k)} \cdot ln^2 + mn^2)$ time.*

Proof. This proof is along the same lines as the time complexity proof of Theorem 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.

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 [8]. 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.

6.1 Description of the data set

Data set. The data set consists of 4736 maximum likelihood gene trees constructed using PhyML [14]. 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 [12]. Specifically, we chose two bootstrap support cutoffs: 80% and 50%. A bootstrap cutoff of 80% is a commonly 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 2(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 2(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%.

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 2(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.

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 [8], [3], we used costs 1, 2, and 3 for losses, duplications, and transfers, respectively. As Figure 2(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 2(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 2(e) also suggests, we noticed that the number of optimal resolutions does not keep increasing exponentially with increasing out-degree.

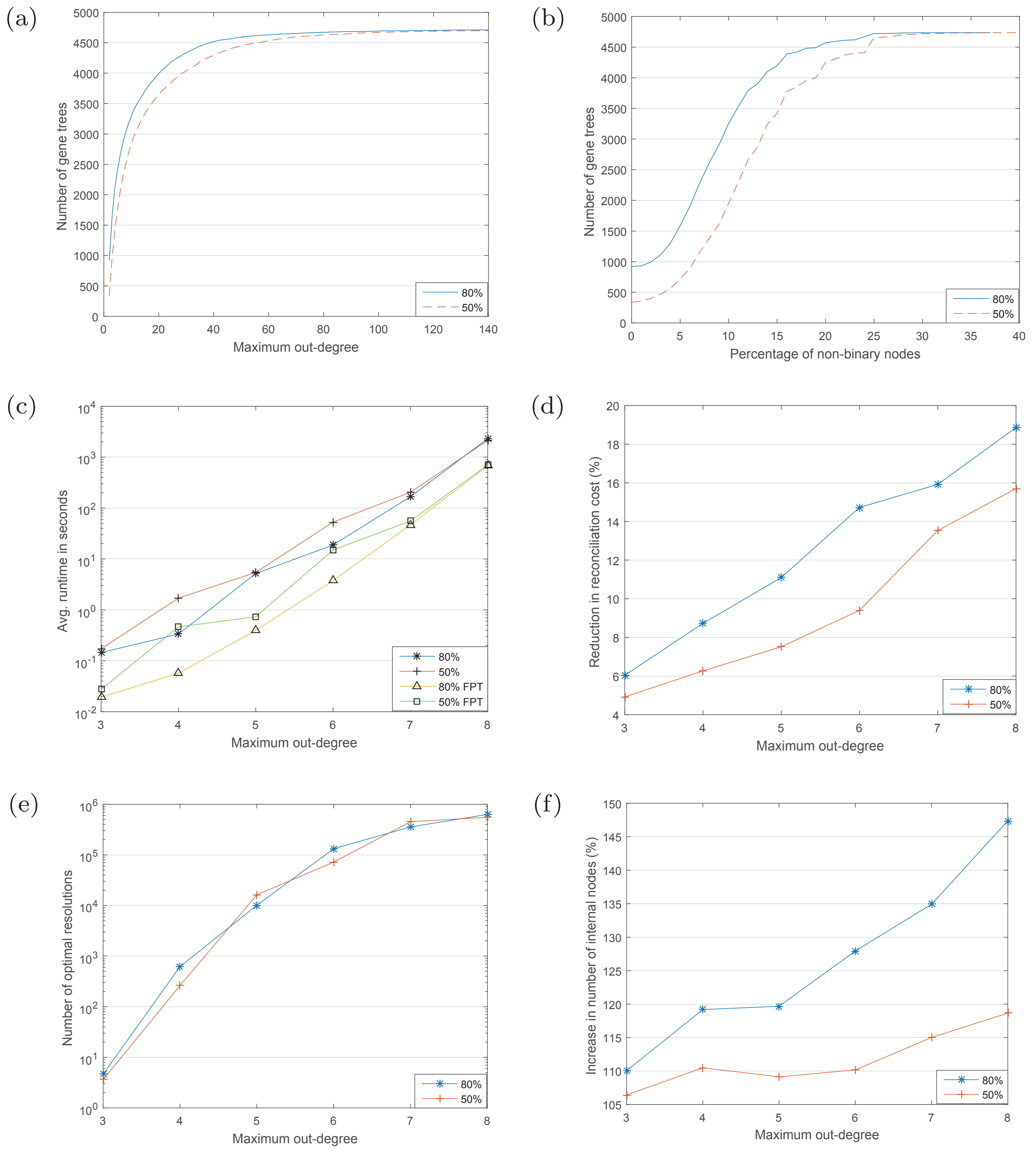


Fig. 2. **Experimental results.** (a) Number of gene trees (cumulative) plotted against their maximum out-degrees for the 80% and 50% cutoffs. (b) Number of gene trees (cumulative) plotted against the percentage of their internal nodes that are non-binary, for the 80% and 50% cutoffs. (c) Average running time (in seconds, on a $\log$ scale) of the FPT and enumeration algorithms on gene trees with maximum out-degrees 3 through 8, for both 50% and 80% bootstrap cutoffs. (d) Average reduction in reconciliation cost for the gene trees with maximum out-degrees 3 through 8, for 50% and 80% bootstrap cutoffs. (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. Results are shown for gene trees with maximum out-degrees 3 through 8, for both 50% and 80% bootstrap cutoffs.

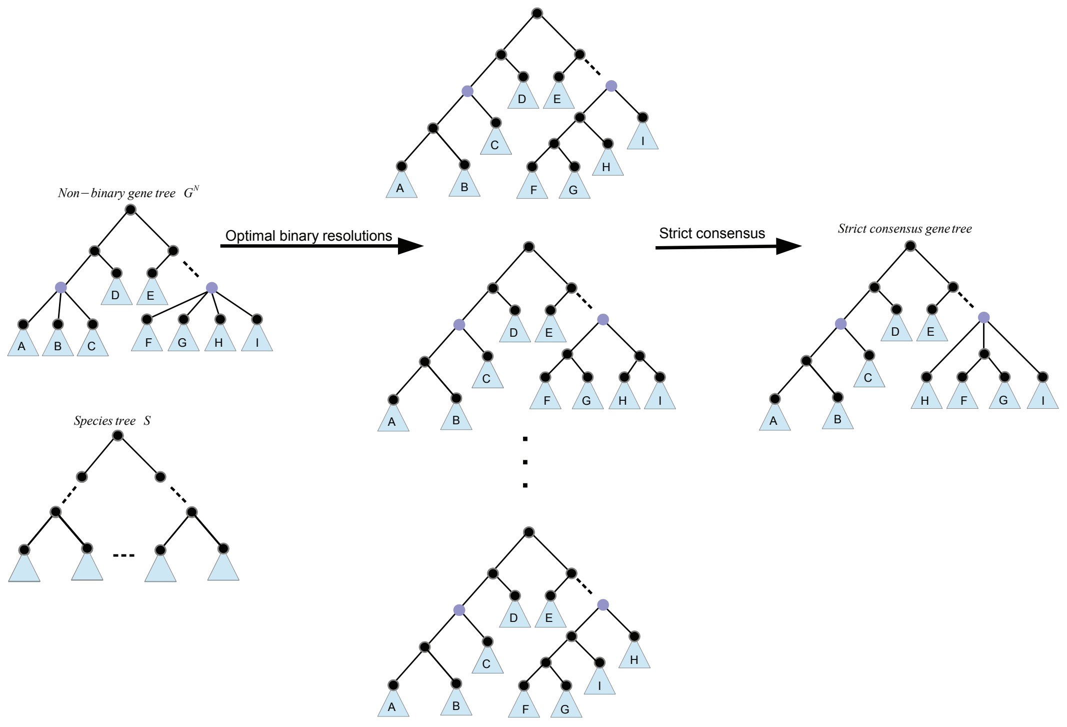

Non−binary gene tree G^N
Optimal binary resolutions
Strict consensus
Strict consensus gene tree
Species tree S
A
B
C
D
E
F
G
H
I

Fig. 3. **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.

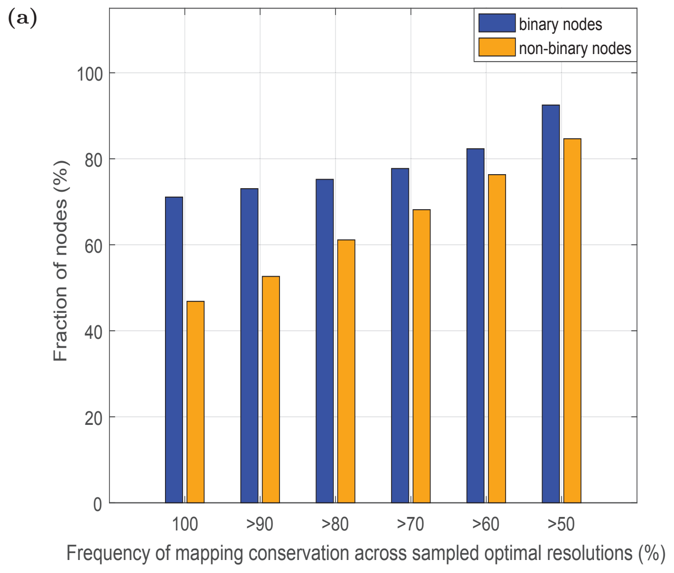
6.4 Impact on inference of gene family evolution

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) [21]. 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 3. 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 2(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 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.

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 [2], we computed 100 optimal DTL-reconciliations, sampled uniformly at random from the space of all optimal reconciliations, for each opti-

2. For gene trees that had more than 20,000 optimal resolutions, we chose 20,000 samples uniformly at random for computing the strict consensus.



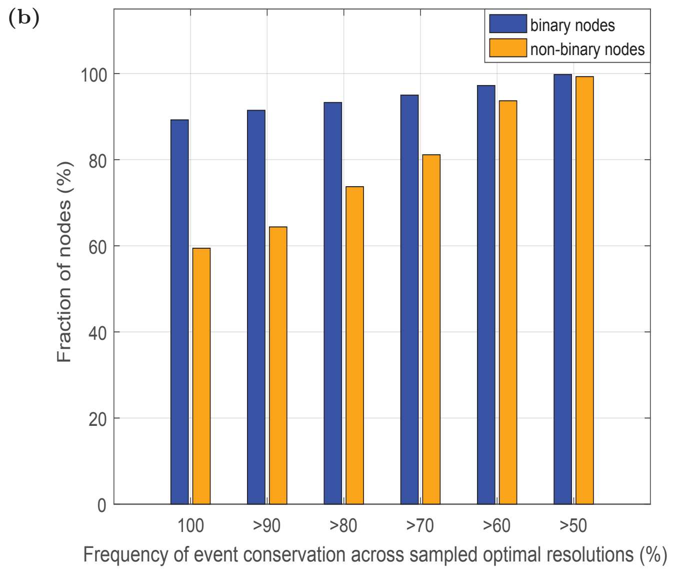


Fig. 4. **Stability of mapping and event assignments across optimal resolutions.** The plot in part (a) shows the fraction of binary and non-binary nodes from the input non-binary gene trees that are assigned the same mapping to the species tree at least a certain fraction of times across a randomly 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.

mal resolution (using the algorithm described in [2]). 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 4. 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 [1], available at http://compbio.engr.uconn.edu/software/RANGER-DTL.

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 paper 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.

Funding: This work was supported in part by NSF CAREER award IIS 1553421, NSF award MCB 1616514, and by startup funds from the University of Connecticut to MSB.

References

[1] M. S. Bansal, E. J. Alm, and M. Kellis. Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. *Bioinformatics*, 28(12):i283–i291, 2012.

[2] M. S. Bansal, E. J. Alm, and M. Kellis. Reconciliation revisited: Handling multiple optima when reconciling with duplication, transfer, and loss. *J. Comput. Biol.*, 20(10):738–754, 2013.

[3] M. S. Bansal, Y.-C. Wu, E. J. Alm, and M. Kellis. Improved gene tree error correction in the presence of horizontal gene transfer. *Bioinformatics*, 31(8):1211–1218, 2015.

[4] J. G. Burleigh, M. S. Bansal, O. Eulenstein, S. Hartmann, A. Wehe, and T. J. Vision. Genome-scale phylogenetics: Inferring the plant tree of life from 18,896 gene trees. *Syst. Biol.*, 60(2):117–125, 2011.

[5] W. Chang and O. Eulenstein. Reconciling gene trees with apparent polytomies. In D. Z. Chen and D. T. Lee, editors, *Computing and Combinatorics, 12th Annual International Conference, COCOON 2006, Taipei, Taiwan, August 15-18, 2006, Proceedings*, volume 4112 of *Lecture Notes in Computer Science*, pages 235–244. Springer, 2006.

[6] K. Chen, D. Durand, and M. Farach-Colton. Notung: dating gene duplications using gene family trees. In *RECOMB*, pages 96–106, 2000.

[7] F. Chevenet, J.-P. Doyon, C. Scornavacca, E. Jacox, E. Jousselin, and V. Berry. SylvX: a viewer for phylogenetic tree reconciliations. *Bioinformatics*, 2015.

[8] L. A. David and E. J. Alm. Rapid evolutionary innovation during an archaean genetic expansion. *Nature*, 469:93–96, 2011.

[9] B. Donati, C. Baudet, B. Sinaimeri, P. Crescenzi, and M.-F. Sagot. EUCALYPT: efficient tree reconciliation enumerator. *Algorithms for Molecular Biology*, 10(1):1–11, 2015.

[10] J.-P. Doyon, C. Scornavacca, K. Y. Gorbunov, G. J. Szöllosi, V. Ranwez, and V. Berry. An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. In *RECOMB-CG*, pages 93–108, 2010.

[11] D. Durand, B. V. Halldórsson, and B. Vernot. A hybrid micro-macroevolutionary approach to gene tree reconstruction. *J. Comput. Biol.*, 13(2):320–335, 2006.

[12] J. Felsenstein. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, 39:783–791, 1985.

[13] K. Y. Gorbunov and V. A. Liubetskii. Reconstructing genes evolution along a species tree. *Molekuliarnaia Biologiia*, 43(5):946–958, Oct. 2009.

[14] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: Assessing the performance of phyml 3.0. *Systematic Biology*, 59(3):307–321, 2010.

[15] E. V. Koonin. Orthologs, paralogs, and evolutionary genomics. *Annual Review of Genetics*, 39(1):309–338, 2005.

[16] M. Kordi and M. S. Bansal. Exact algorithms for duplication-transfer-loss reconciliation with non-binary gene trees. In *Proceedings of the 7th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB '16*, page in press. ACM, 2016.

[17] M. Kordi and M. S. Bansal. On the complexity of Duplication-Transfer-Loss reconciliation with non-binary gene trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, in press, 2016.

[18] M. Lafond, K. Swenson, and N. El-Mabrouk. An optimal reconciliation algorithm for gene trees with polytomies. In B. Raphael and J. Tang, editors, *Algorithms in Bioinformatics*, volume 7534 of *Lecture Notes in Computer Science*, pages 106–122. Springer Berlin Heidelberg, 2012.

[19] R. Libeskind-Hadas and M. Charleston. On the computational complexity of the reticulate cophylogeny reconstruction problem. *J. Comput. Biol.*, 16:105–117, 2009.

[20] R. Libeskind-Hadas, Y.-C. Wu, M. S. Bansal, and M. Kellis. Pareto-optimal phylogenetic tree reconciliation. *Bioinformatics*, 30(12):i87–i95, 2014.

[21] F. R. McMorris, D. B. Meronk, and D. A. Neumann. A view of some consensus methods for trees. In J. Felsenstein, editor, *Numerical Taxonomy*, pages 122–126. Springer Berlin Heidelberg, 1983.

[22] D. Merkle, M. Middendorf, and N. Wieseke. A parameter-adaptive dynamic programming approach for inferring cophylogenies. *BMC Bioinformatics*, 11(Suppl 1):S60, 2010.

[23] Y. Ovadia, D. Fielder, C. Conow, and R. Libeskind-Hadas. The cophylogeny reconstruction problem is NP-complete. *J. Comput. Biol.*, 18(1):59–65, 2011.

[24] C. Scornavacca, E. Jacox, and G. J. Szollosi. Joint amalgamation of most parsimonious reconciled gene trees. *Bioinformatics*, 31(6):841–848, 2015.

[25] C. Scornavacca, W. Paprotny, V. Berry, and V. Ranwez. Representing a set of reconciliations in a compact way. *J. Bioinform. Comput. Biol.*, 11(02):1250025, 2013.

[26] J. B. Slowinski. Molecular polytomies. *Molecular Phylogenetics and Evolution*, 19(1):114 – 120, 2001.

[27] M. Stolzer, H. Lai, M. Xu, D. Sathaye, B. Vernot, and D. Durand. Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. *Bioinformatics*, 28(18):409–415, 2012.

[28] G. J. Szollosi, E. Tannier, N. Lartillot, and V. Daubin. Lateral gene transfer from the dead. *Systematic Biology*, 62(3):386, 2013.

[29] A. Tofigh, M. T. Hallett, and J. Lagergren. Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 8(2):517–535, 2011.

[30] A. J. Vilella, J. Severin, A. Ureta-Vidal, L. Heng, R. Durbin, and E. Birney. EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research*, 19(2):327–335, 2009.

[31] Y. Zheng and L. Zhang. Reconciliation with non-binary gene trees revisited. In R. Sharan, editor, *Research in Computational Molecular Biology*, volume 8394 of *LNCS*, pages 418–432. 2014.

Misagh Kordi received the BS degree in computer science and engineering from Kharazmi University, Iran, in July 2010, and the MS degree in computer science and engineering from the University of Tehran, Iran, in July 2013. He is currently working towards the PhD degree in the Department of Computer Science and Engineering at the University of Connecticut, USA. His research interests include computational biology and phylogenetics, graph theory, complexity theory, approximation algorithms, and algorithms in general.

Mukul S. Bansal is currently an assistant professor with the Department of Computer Science and Engineering at the University of Connecticut, USA. His research interests are in computational biology and bioinformatics, with an emphasis on computational molecular evolution. He is especially interested in computational problems related to understanding the evolution of genes, genomes, and species. He received the PhD degree in computer science from Iowa State University in 2009. He was an Edmond J. Safra postdoctoral fellow at the School of Computer Science at Tel Aviv University in Israel until December 2010, and a postdoctoral associate at the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology until August 2013.