Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss

Mukul S. Bansal1,∗, Eric J. Alm2 and Manolis Kellis1,3

1Computer Science and Artificial Intelligence Laboratory, 2Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA and 3Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA

ABSTRACT

Motivation: Gene family evolution is driven by evolutionary events such as speciation, gene duplication, horizontal gene transfer and gene loss, and inferring these events in the evolutionary history of a given gene family is a fundamental problem in comparative and evolutionary genomics with numerous important applications. Solving this problem requires the use of a reconciliation framework, where the input consists of a gene family phylogeny and the corresponding species phylogeny, and the goal is to reconcile the two by postulating speciation, gene duplication, horizontal gene transfer and gene loss events. This reconciliation problem is referred to as duplication-transfer-loss (DTL) reconciliation and has been extensively studied in the literature. Yet, even the fastest existing algorithms for DTL reconciliation are too slow for reconciling large gene families and for use in more sophisticated applications such as gene tree or species tree reconstruction.

Results: We present two new algorithms for the DTL reconciliation problem that are dramatically faster than existing algorithms, both asymptotically and in practice. We also extend the standard DTL reconciliation model by considering distance-dependent transfer costs, which allow for more accurate reconciliation and give an efficient algorithm for DTL reconciliation under this extended model. We implemented our new algorithms and demonstrated up to 100,000-fold speed-up over existing methods, using both simulated and biological datasets. This dramatic improvement makes it possible to use DTL reconciliation for performing rigorous evolutionary analyses of large gene families and enables its use in advanced reconciliation-based gene and species tree reconstruction methods.

Availability: Our programs can be freely downloaded from http://compbio.mit.edu/ranger-dtl.

Contact: mukul@csail.mit.edu; manoli@mit.edu

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

Gene families evolve through complex evolutionary processes such as speciation, gene duplication, horizontal gene transfer and gene loss. Accurate inference of these events is crucial not only to understanding gene and genome evolution but also for reliably reconstructing ancestral gene content and dating gene birth [Chen et al. 2008; David and Alm 2011; Ma et al. 2008], accurate gene tree reconstruction [Rasmussen and Kellis 2011; Vilella et al. 2004], and for whole genome species tree reconstruction [Janardan et al. 2005; Burleigh et al. 2011]. Indeed, the problem of inferring gene family evolution has been extensively studied. In the typical formulation of this problem, the goal is to reconcile an input gene tree (gene family phylogeny) to the corresponding rooted species tree by postulating speciation, duplication, transfer and loss events. Much of the previous work in gene tree–species tree reconciliation has focused on either duplication loss (DL), duplication and gene loss (DL) (Bonizzoni et al. 2005; Chauve et al. 2006; Duran et al. 2003; Halperin and Gurevich 1999; Goodman et al. 1993), gene transfer and loss (TL) [Howe et al. 2010; Halperin and Gurevich 2000; Iliopoulos et al. 2011; Hochstenbach et al. 2004; Jin et al. 2009; Naklibeh et al. 2004; Ronquist 1995], but not on duplication, transfer and loss together. However, duplication and transfer events frequently occur together, particularly in prokaryotes, and the analysis of such events requires reconciliation methods that can simultaneously consider duplication, transfer and loss events. This problem of gene tree–species tree reconciliation by duplication, transfer and loss is referred to as the duplication–transfer–loss (DTL) reconciliation problem.

Previous work. The DTL reconciliation problem has a long history and is well studied in the literature. This is partly due to its close association with the host–parasite cophylogeny problem [Charleston and Hedin, 2004; Hedin and Charleston, 2004] which seeks to understand the evolution of parasites (analogous to genes) within hosts (analogous to species). Almost all known formulations of the DTL reconciliation problem are based on a parsimony framework [Charleston et al. 1999; Conow et al. 2011; David and Alm 2011; Dowen et al. 2011; Gerbunov and Liubasik, 2004; Hickey and Charleston, 2004; Mark and Middendorf, 2004; Mark et al. 2010; Orzad et al. 2011; Ronquist 2003; Rothb et al. 2011] (but see also Roch et al. 2009 for an example of a probabilistic formulation, as well as Csuros and Miklos 2001 for a probabilistic framework based on gene content). Under this framework, each duplication, transfer and loss event is assigned a cost and the goal is to find a reconciliation that has the lowest total reconciliation cost. Optimal DTL reconciliations can sometimes violate temporal constraints; that is, the transfers are such that they induce contradictory constraints on the dates for the internal nodes of the species tree. We refer to such paradoxical reconciliations as time-inconsistent (after Dowen et al. 2011). In general, it is desirable to consider only those DTL reconciliations that are time-consistent (i.e., paradox-free). Henceforth, we refer to the problem of specifically computing optimal time-consistent
DTL reconciliations as the tcDTL Reconciliation problem. Although the DTL reconciliation problem can be solved in polynomial time, solving the tcDTL reconciliation problem is NP-hard [Ovadia et al. 2011; Tofigh et al. 2011]. If divergence time information is available for the nodes of the species tree (or if there is a known relative temporal ordering for each pair of internal nodes), then any proposed DTL reconciliation must also respect the temporal constraints imposed by the available timing information, i.e., transfers must be restricted to occur only between coexisting species. When such divergence timing information is available, even the tcDTL reconciliation problem becomes polynomially solvable [Libeskind-Hadas and Charleston, 2008]. (Note, however, that time-consistency cannot be guaranteed just by ensuring that transfers occur between coexisting species.) In general, the input species tree can be undated, partially dated, or fully dated depending on whether the divergence timing information associated with the nodes of the species tree is absent, partial, or complete, respectively. Thus, in practice, when the species tree is undated or partially dated, one solves the DTL reconciliation problem, and if the species tree is fully dated, one can solve either the DTL reconciliation or the tcDTL reconciliation problem.

Let m denote the number of leaves in the input gene tree and n the number of leaves in the species tree. Both the DTL reconciliation problem and the tcDTL reconciliation problem, along with some of their variants, have been extensively studied in the literature [Charleston, 1999; Conow et al. 2014; David and Alm, 2011; Doyon et al. 2011; Gorbunov and Liubetskii, 2009; Libeskind-Hadas and Charleston, 2008; Merkle and Middendorf, 2006; Merkle et al. 2010; Novak and Charleston, 2009; Tofigh et al. 2011]. The most recent systematic work on these problems include [Doyon et al. 2011; Novak and Charleston, 2009; Tofigh et al. 2011].

The paper by Tofigh et al. 2011 studies a restricted version of the reconciliation model that ignores losses (equivalent to assigning a cost of zero for loss events under the DTL reconciliation problem) and that the species tree is undated. This restricted model, the DTL reconciliation problem on undated trees can be solved in $O(mn^2)$ time. They also gave a fixed parameter tractable algorithm for enumerating all most parsimonious reconciliations. The time complexity of the $O(mn^2)$-time algorithm was further improved to $O(mn)$ in Tofigh et al. 2011 (under the same restricted reconciliation model). However, with the increasing availability of whole-genome datasets, such a restriction on the reconciliation model can be problematic as losses are a rich source of information that can be critical for accurate reconciliation. Indeed, losses play a fundamental role in the ability to distinguish between duplications and transfers as well as in mapping the nodes of the gene tree into the nodes of the species tree, and thus should be explicitly considered during reconciliation. The paper by Doyon et al. 2011 showed that, for fully dated species trees, the tcDTL reconciliation problem could be solved in $O(mn^2)$ time. Recently, an $O(mn^{3/2})$-time algorithm for the tcDTL reconciliation problem on fully dated trees has also been independently developed for Version 2 of the program Jane [Conow et al. 2014]. Finally, the recent paper by David and Alm 2011 gave an $O(mn^{3/2})$-time algorithm for the DTL reconciliation problem on undated trees.

In summary, in spite of tremendous methodological and algorithmic advances, even the fastest existing algorithms for DTL reconciliation [David and Alm 2011; Merkle et al. 2010] as well as for tcDTL reconciliation on fully dated trees [Doyon et al. 2011] still have a time complexity of $O(mn^2)$. This makes them too slow to reconcile trees with more than a few hundred taxa, and completely unsuitable for all but the smallest trees when used in sophisticated applications such as reconciliation-based gene tree or species tree reconstruction that require the reconciliation of a multitude of trees while searching through tree space [Bansal et al. 2011; Burleigh et al. 2011; Eriksen and Kellett 2011; Ovadia et al. 2009].

Our contributions. Recall that the DTL reconciliation problem, even on fully dated species trees, does not guarantee that the optimal reconciliation is time-consistent, whereas the tcDTL reconciliation problem does. However, the tcDTL reconciliation problem suffers from two major drawbacks that limit its applicability in practice. First, the tcDTL reconciliation problem can only be solved efficiently when the species tree is fully dated. This limits its application to only those species tree that contain a relatively small number of taxa (say <100). This is because, it can be extremely difficult to accurately date large species trees [Hausmann 2009] and the accuracy of tcDTL reconciliation relies implicitly on having a correctly dated species tree. Second, the time complexity of the fastest known algorithm for the tcDTL reconciliation problem is $O(mn^2)$, which makes it too slow to be used with large datasets (as we also demonstrate later). This also makes it too slow for reconciliation-based gene tree reconstruction of even relatively small gene trees, as it involves repeatedly reconciling a multitude of candidate gene trees against the species tree. Furthermore, the tcDTL reconciliation problem cannot be used for reconciliation-based whole-genome species tree construction (also called gene tree parsimony), as the topology of the species tree is repeatedly modified and so at each step, the species tree is undated.

Thus, in this work, we focus on the DTL reconciliation problem. In particular, we improve upon the current state of the art for the DTL reconciliation problem in the following ways:

1. We provide an $O(mn)$-time algorithm for the DTL reconciliation problem on undated species trees. This improves on the fastest known algorithm for this problem by a factor of n. The DTL reconciliation problem on undated trees is the most common version of the DTL reconciliation problem and arises whenever the species tree cannot be accurately dated, as is usually the case with large gene families, and in applications such as reconciliation-based species tree reconstruction.

2. For the DTL reconciliation problem on fully dated species trees, we provide an $O(mn \log n)$-time algorithm, which improves on the fastest known algorithm for this problem by a factor of $n \log n$. Even though the fully dated version of DTL reconciliation does not guarantee time-consistency, as we show later using thorough experimental studies, optimal DTL reconciliations closely approximate optimal tcDTL reconciliations. This algorithm is thus meant as a faster alternative to the $O(mn^2)$-time algorithm for tcDTL reconciliation.

3. We give a simple $O(mn^2)$-time algorithm for DTL reconciliation that can handle distance-dependent transfer costs and can work with undated, partially dated, or fully dated species trees. This is a factor of n faster than the fastest known algorithm that can handle distance-dependent transfer costs [Conow et al. 2014]. Distance-dependent transfer costs capture the biology of transfers more accurately than having...
Reconciliation using Duplication, Transfer, and Loss

2 DEFINITIONS AND PRELIMINARIES

Given a tree $T$, we denote its node, edge and leaf sets by $V(T)$, $E(T)$ and $L(T)$, respectively. If $T$ is rooted, the root node of $T$ is denoted by $r(T)$, the parent of a node $v \in V(T)$ by $p_T(v)$, its set of children by $C_T(v)$, and the (maximal) subtree of $T$ rooted at $v$ by $T_v$. If two nodes in $T$ have the same parent, they are called siblings. The set of internal nodes of $T$, denoted $I(T)$, is defined to be $V(T) \setminus L(T)$. We define $\leq_T$ to be the partial order on $V(T)$, where $x \leq_T y$ if $x$ is a node on the path between $r(T)$ and $y$. The partial order $\leq_T$ is defined analogously, i.e., $x \leq_T y$ if $x$ is a node on the path between $r(T)$ and $x$. We say that $y$ is an ancestor of $u$, or that $u$ is a descendant of $v$, if $u \leq_T y$ (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 $u \leq_T y$ nor $y \leq_T u$. Given a non-empty subset $L \subseteq L(T)$, we denote by $lca_T(L)$, the least 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 $x \rightarrow_T y$. Throughout this work, unless otherwise stated, the term tree refers to a rooted binary 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. We assume that each leaf of the gene trees is labeled with the species from which that gene was sampled. This labeling defines a leaf-mapping $G : \mathcal{L}(G) \rightarrow \mathcal{L}(S)$ that maps a leaf node $g \in \mathcal{L}(G)$ to that unique leaf node $r \in \mathcal{L}(S)$ which has the same label as $g$. Note that gene trees may have more than one gene sampled from the same species. Throughout this work, we denote the gene tree and species tree under consideration by $G$ and $S$, respectively, and will implicitly assume that $\mathcal{L}(G) \subseteq \mathcal{L}(S)$ is well defined.

2.1 Reconciliation and DTL scenarios

Reconciling a gene tree with a species tree involves mapping the gene tree into the species tree. Such a mapping allows us to infer the evolutionary events that gave rise to that particular gene tree. In this case, the evolutionary events of interest are speciation, gene duplication, horizontal gene transfer and gene loss. Next, we define what constitutes a valid reconciliation; specifically, we define a DTL scenario for $G$ and $S$ that characterizes the mappings of $G$ 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 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 $g'$ and $g''$ denote the children of $g$, represents the transfer edge on $S$, and identify the recipient species of the corresponding transfer. Incorporating available divergence time information.

Incorporating available divergence time information, when accurate divergence time information is available, for some or all of the nodes of the species tree, DTL scenarios must respect the temporal constraints imposed by the available timing information. Specifically, those transfer events that are inconsistent with the available timing information are disallowed (as transfer events could only have happened between two coexisting species). If there is no divergence time information available, then transfers are allowed to occur between any pair of incomparable species on the species tree.

The definition of a DTL scenario below is a generalization of the definition from [3]. The generalization is necessary to correctly handle optimal reconciliations in cases when the species tree is dated. Specifically, we enforce that, if the species tree is dated, then transfers can only occur between coexisting species and introduce an additional variable to explicitly specify the recipient species for any transfer event.

DEFINITION 2.1 (DTL scenario). A DTL scenario for $G$ and $S$ is a seven-tuple $(\mathcal{L}, M, \Sigma, \Delta, \Theta, \Xi, \tau)$, where \mathcal{L}. Let $G(\rightarrow) \rightarrow \mathcal{L}(S)$ represents the leaf mapping from $G$ to $S$, $M : \mathcal{L}(G) \rightarrow \mathcal{L}(S)$ maps each node of $G$ to a node of $S$, the sets $\Sigma, \Delta$ and $\Theta$ partition $I(G)$ into speciation, duplication and transfer nodes, respectively, $\Xi$ is a subset of gene tree edges that represent transfer edges, and $\tau : \Theta \rightarrow \mathcal{L}(S)$ specifies the recipient species for each transfer event, subject to the following constraints:

1. If $g \in \mathcal{L}(G)$, then $M(g) = \mathcal{L}(g)$.
2. If $g \in \mathcal{L}(G)$, and $g'$ and $g''$ denote the children of $g$, then,
(a) $M(g) \not\leq_M M(g')$ and $M(g) \not\leq_M M(g'')$.
(b) At least one of $M(g')$ and $M(g'')$ is a descendant of $M(g)$.
(3) Given any edge $(g, g') \in E(G)$, $(g, g') \in \Xi$ if and only if $M(g)$ and $M(g')$ are incomparable.
(4) If $g \in I(G)$ and $g'$ and $g''$ denote the children of $g$, then,
   (a) $g \in \Xi$ only if $M(g) = \text{lca}(M(g'), M(g''))$ and $M(g')$ and $M(g'')$ are incomparable,
   (b) $g \in \Delta$ only if $M(g) \geq_{\text{lca}}(M(g'), M(g''))$,
   (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 $t(g)$ must be incomparable, the species represented by them must be potentially coexisting with respect to the available divergence time estimates, and $M(g')$ must be a descendant of $t(g)$, i.e., $M(g') \subseteq_T t(g)$.

Constraint 1 above ensures that the mapping $A$ is consistent with the leaf mapping $C$. Constraint 2(a) imposes on $A$ the temporal constraints implied by $S$. Constraint 2(b) implies that any internal node in $G$ may represent at most one transfer event. Constraint 3 determines the edges of $G$ that are transfer edges. Constraints 4(a–c) state the conditions under which an internal node of $G$ may represent a speciation, duplication and transfer, respectively. Constraint 4(d) 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 $G$ and $S$ implied by any DTL scenario. Figure 3 shows two simple DTL scenarios. Given a DTL scenario, one can directly count the minimum number of gene losses in the corresponding reconciliation as follows:

Definition 2.2 (Losses). Given a DTL scenario $a = (L, M, \Xi, \Delta, \Theta, \Xi, \Theta, \Xi, \Theta, \Xi, \Theta)$ for $G$ and $S$, let $g \in V(G)$ and $(g', g'') \in E(G)$. The number of losses $Loss_G(g)$ at node $g$ is defined to be

$$|d_G(M(g), M(g')) - 1| + |d_G(M(g), M(g'')) - 1|, \quad \text{if } g \in \Xi,$$
$$d_G(M(g), M(g')) - 1, \quad \text{if } g \in \Delta 	ext{ and } M(g) = M(g'),$$
$$d_G(M(g), M(g')) + d_G(M(g), M(g'')) + d_G(M(g), M(g'')) - 1, \quad \text{if } g \in \Delta, M(g) \neq M(g') \neq M(g''),$$
$$d_G(M(g), M(g')) + d_G(M(g), M(g'')) - 1, \quad \text{if } (g', g'') \in \Xi.$$

Fig. 1. Simple DTL scenarios. (a) and (b) depict two possible reconciliations of $G$ and $S$; the dotted arcs show the mapping $A$ (with the leaf mapping being specified by the leaf labels on the gene tree), and the label at each internal node of $G$ specifies the type of event represented by that node. The reconciliation in (a) requires two transfers and one loss and the one in (b) requires one duplication and two losses.

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

Let $P_\Delta$, $P_\Theta$ and $P_{\text{loss}}$ denote the costs associated with duplication, transfer and loss events respectively. The cost of reconciling $G$ and $S$ according to a DTL scenario $\alpha$ is defined as follows:

Definition 2.3 (Reconciliation cost of a DTL scenario). Given a DTL scenario $\alpha = (L, M, \Xi, \Delta, \Theta, \Xi, \Theta, \Xi, \Theta, \Xi, \Theta)$ for $G$ and $S$, the reconciliation cost associated with $\alpha$ is given by $R_\alpha = P_\Delta |\Delta| + P_\Theta |\Theta| + P_{\text{loss}} |\Xi|$.

Given $G$ and $S$, our goal is to find a most parsimonious reconciliation of $G$ and $S$. More formally,

Problem 1. [Most parsimonious reconciliation (MPR)] Given $G$ and $S$, the MPR problem is to find a DTL scenario for $G$ and $S$ with minimum reconciliation cost.

Based on whether the species tree is undated or fully dated, we distinguish two versions of the MPR problem: (i) The undated MPR (U-MPR) problem where the species tree is undated and (ii) the fully dated MPR (D-MPR) problem where every node of the species tree has an associated divergence time estimate (or there is a known total order on the internal nodes of the species tree). We will exploit the local structure unique to each version to develop faster algorithms for them.

3 COMPUTING THE MOST PARSIMONIOUS RECONCILIATION

In this section, we first develop our fast algorithms for the U-MPR and D-MPR problems and then give a simple $O(n\mu \tau)$-time algorithm for the (general) MPR problem that can efficiently handle distance-dependent transfer costs. Before we proceed, we need a few definitions and additional notation.

Definitions: Given any $g \in I(G)$ and $s \in V(S)$, let $c_G(g, s)$ denote the cost of an optimal reconciliation of $G(g)$ with $S$ such that $g$ maps to $s$ and $g \in \Xi$. The terms $c_G(g, s)$ and $c_G(g, s)$ are defined similarly for $g \in \Delta$ and $g \in \Theta$, respectively. Given any $g \in V(G)$ and $s \in V(S)$, we define $c_G(g, s)$ to be the cost of an optimal reconciliation of $G(g)$ with $S$ such that $g$ maps to $s$. Thus,

$$c_G(g, s) = \begin{cases} 0, & \text{if } g \in L(G) \text{ and } s = M(g), \\ \infty, & \text{if } g \in L(G) \text{ and } s \neq M(g), \\ \min[c_G(g, s), c_G(g, s), c_G(g, s)] & \text{otherwise.} \\ \end{cases}$$

Furthermore, let $\text{in}_G(g, s) = \min_{s \in V(S)} \{P_{\text{loss}} d_G(s, x) + c_G(g, s)\}$, $\text{out}_G(g, s) = \min_{s \in V(S)} \{P_{\text{loss}} d_G(s, x) + c_G(g, s)\}$, and $\text{inAlt}(g, s) = \min_{s \in V(S)} \{P_{\text{loss}} d_G(s, x) + c_G(g, s)\}$. In other words, $\text{inAlt}(g, s)$ is the cost of an optimal reconciliation of $G(g)$ with $S$ such that $g$ may map to any node in $V(S)$; $\text{out}_G(g, s)$ is the cost of an optimal reconciliation of $G(g)$ with $S$ such that $g$ may map to any node from $V(S)$ that is incomparable to $s$; and $\text{in}(g, s)$ is the cost of an optimal reconciliation of $G(g)$ with $S$ such that $g$ may map to any node, say $x$, in $V(S)$ but with an additional reconciliation cost of one loss event for each edge on the path from $s$ to $x$.

Note that the optimal reconciliation cost of $G$ and $S$ is simply $\min_{s \in V(S)} \{\text{in}_G(G, S)\}$. The equation for $c_G(g, s)$ above, used in a dynamic programming framework and coupled with methods for computing the values of $c_G(g, s)$, $c_G(g, s)$ and $c_G(g, s)$, forms the basis of all our algorithms.
3.1 An $O(mn)$-time algorithm for U-MPR

The following algorithm solves the U-MPR problem in $O(mn)$ time. Our algorithm builds on the $O(mn)$-time dynamic programming algorithm from [19] that computes optimal reconciliation scenarios under a simpler reconciliation cost that ignores losses. We compute the values $c_{\Delta}(g, s)$, $c_{\Delta}(g, s)$ and $c_{\Delta}(g, s)$ for each $g \in V(G)$ and $s \in V(S)$ by performing a nested post-order traversal of $G$ and $S$. For efficiency, we save and reuse as much of the computation from previous steps as possible, and the values $int(\cdot, \cdot)$, inAlt(\cdot, \cdot), and out(\cdot, \cdot, \cdot) help us in efficiently computing the values $c_{\Delta}(g, s)$, $c_{\Delta}(g, s)$ and $c_{\Delta}(g, s)$ at each dynamic programming step. For instance, for any $g \in V(G)$, the value of $c_{\Delta}(g, s)$ is simply:

$$
c_{\Delta}(g, s) = \infty$$

where $[g', \ell'] = \text{Ch}_2(g)$ and $[g', \ell'] = \text{Ch}_3(s)$, if $s \in I(S)$. The values of $c_{\Delta}(g, s)$ and $c_{\Omega}(g, s)$ can be similarly computed; see Steps 16 and 18 of Algorithm U-Reconcile for $c_{\Delta}(g, s)$ and Steps[21] and[22] for $c_{\Omega}(g, s)$. The nested post-order traversal ensures that when computing the values $c_{\Delta}(g, s)$, $c_{\Delta}(g, s)$ and $c_{\Omega}(g, s)$ at nodes $g \in G$ and $s \in S$, all the required $int(\cdot, \cdot)$, inAlt(\cdot, \cdot), and out(\cdot, \cdot, \cdot)$ values have already been computed.

**Algorithm U-Reconcile$(G, S, C)$**

1. For each $g \in V(G)$ and $s \in V(S)$ do
   1. Initialize $c_{\Delta}(g, s)$, $c_{\Delta}(g, s)$, $c_{\Omega}(g, s)$, $\text{int}(g, s, t)$, $\text{inAlt}(g, s, t)$, and $\text{out}(g, s, t, t, t')$ to $\infty$.
2. For each $g \in V(G)$ do
   1. Initialize $c_{\Delta}(g, s)$ to 0. and, for each $g \in V(S)$, initialize $\text{int}(g, s) = P_{\text{loss}}$, $\text{inAlt}(g, s)$, and $\text{out}(g, s, t)$ to 0.
3. For each $g \in V(G)$ in post-order do
   1. Let $[g', \ell'] = \text{Ch}_2(g)$.
   2. For all $s \in V(S)$ then
      1. $c_{\Delta}(g, s) = \infty$.
   3. $c_{\Delta}(g, s) = P_{\text{loss}} + c_{\Delta}(g', s) + c_{\Delta}(g', s)$.
   4. If $g' \neq \text{tt}(S)$, then $c_{\Delta}(g, s) = P_{\text{loss}} + \min\left[\text{int}(g', s), \text{out}(g', s), \text{int}(g', s), \text{out}(g', s), \right]$
   5. $\text{int}(g, s) = c_{\Delta}(g, s)$.
   6. $\text{inAlt}(g, s) = c_{\Delta}(g, s)$.
   7. else
      1. Let $[s', \ell'] = \text{Ch}_2(g)$.
   2. $c_{\Delta}(g, s) = \min\left[\text{int}(g', s'), \text{int}(g', s'), \text{int}(g', s'), \text{int}(g', s'), \right]$
   3. $c_{\Delta}(g, s) = P_{\text{loss}} + \min\left[\text{int}(g', s'), \text{int}(g', s'), \text{int}(g', s'), \text{int}(g', s'), \right]$
   4. $\text{int}(g, s) = c_{\Delta}(g, s)$.
   5. $\text{inAlt}(g, s) = c_{\Delta}(g, s)$.
   6. if $g' \neq \text{tt}(S)$, then $c_{\Delta}(g, s) = P_{\text{loss}} + \min\left[\text{int}(g', s), \text{out}(g', s), \text{out}(g', s), \right]$
   7. $\text{out}(g, s) = \min\left[\text{out}(g, s), \text{out}(g, s), \text{out}(g, s), \text{out}(g, s), \right]$
   8. if $g' \neq \text{tt}(S)$, then $c_{\Delta}(g, s) = P_{\text{loss}} + \min\left[\text{int}(g', s), \text{out}(g', s), \text{out}(g', s), \right]$
   9. $\text{inAlt}(g, s) = \min\left[\text{inAlt}(g, s), \text{inAlt}(g, s), \text{inAlt}(g, s), \right]$

23. For each $s \in I(S)$ in pre-order do
24. Let $\{s', s''\} = \text{Ch}_3(s)$.
25. $\text{out}(g, s') = \min\left[\text{out}(g, s), \text{inAlt}(g, s', s''), \text{out}(g, s')\right]$, and $\text{out}(g, s'') = \min\left[\text{out}(g, s), \text{inAlt}(g, s', s'')\right]$
26. Return $\min_{s \in I(S)} (t, t'(S), g).

Remarks: (i) Note that, while the above algorithm only outputs the optimal reconciliation cost, it can be easily adapted, without affecting its time complexity, to output the DTL scenario itself. (ii) The algorithm above implicitly assumes that if $g \in I(G)$ is a transfer node such that $(g, g') \in \mathbb{Z}$, then $t = \text{Alt}(g')$. The reason for this is easy to see: any reconciliation in which $t = \text{Alt}(g')$ (and losses have a strictly positive cost), cannot be most parsimonious. This, however, only holds true for the U-MPR problem, and we will be unable to make this assumption when working with partially or fully dated species trees.

We have the following theorem. (all proofs are available in the Supplementary Material).

**Theorem 3.1.** The U-MPR problem on $G$ and $S$ can be solved in $O(mn)$ time.

3.2 An $O(m\log n)$-time algorithm for D-MPR

In the D-MPR problem, there exists a total ordering of the internal nodes of the species tree based on their divergence times. Thus, in this setting, for any given pair of species tree edges, it is known whether the two species represented by those edges overlapped in their time of existence, and transfers are only allowed between two species if they are coexisting.

We assign consecutive positive integers, starting with one, to the internal nodes of the species tree according to the total order. These numbers are referred to as time stamps and they represent the temporal order in which the species represented by these nodes diverged. Given a node $s \in V(S)$, we denote its time stamp by $(s)$. If the largest time stamp assigned to the internal nodes is $k$, then we assume time stamp 1 to 1 each leaf of $S$. Any two consecutive time stamps $s, s' \in V(S)$ have a strictly positive cost, cannot be most parsimonious. Thus, however, only holds true for the U-MPR problem, and we will be unable to make this assumption when working with partially or fully dated species trees.

Our algorithm for the D-MPR problem, called Algorithm D-Recoume, makes use of the same overall dynamic programming structure as Algorithm U-Reconcile, and the procedure for computing the values $c_{\Delta}(\cdot, \cdot)$ and $c_{\Omega}(\cdot, \cdot, \cdot)$ remains identical. The difference is in the way $c_{\Delta}(\cdot, \cdot, \cdot)$ is computed, as we can no longer rely on the $\text{out}(\cdot, \cdot, \cdot)$ values. Instead, we need a more elaborate procedure that can efficiently yield the "best receiver" for a transfer originating at the species tree node currently under consideration, from among the relevant time zones. More concretely, suppose we want to compute the value $c_{\Delta}(s, s)$ assuming that $(g, g') \in \mathbb{Z}$, where $g' \in \text{Ch}(g)$, for each $s \in V(S)$. Our algorithm first efficiently computes the locally best and locally second-best receivers of gene $g$ in each time zone based on the values $c_{\Delta}(\cdot, \cdot)$. Then, for each candidate node $s$ under consideration, we efficiently compute the best receiver, for a transfer originating at $s$, by choosing the globally
optimal value from among the previously computed locally best and locally second-best receivers for the relevant time zones. For efficiency, our algorithm makes use of (i) a binomial heap data structure and (ii) a dynamic range minimum query data structure. The binomial heap data structure maintains a set of P-values while supporting find-min, insert and delete operations in O(1) time, and O(log n) space, respectively (Kemeny et al. 2009; Vuillemin 1978). The dynamic range minimum query data structure maintains an ordered list of elements and can answer queries that seek the smallest element in a given query range in O(log n) time and also supports update operations that change the value of an element in the list in O(log n) time (Brodal et al. 2011).

Definitions. Let k denote the number of time zones on the species tree. Given a time zone i (1 ≤ i ≤ k), let Z(i) denote the set of edges from E(S) that are associated with time zone i. Let Best(y,i) and secondBest(y,i) denote, respectively, the two edges from Z(i) with the smallest value of in(y.g).

Preprocessing. Before running Algorithm D-Reconcile, we assume that we have precomputed, for each time zone i (1 ≤ i ≤ k), the following: (i) the set of edges (pa(y), x) ∈ E(S) for which rt(x) = i (i.e. (pa(y), x) is associated with Z(i)), but not with Z(i+1), referred to as end(i) and (ii) the set of edges (pa(y), x) ∈ E(S) for which rt(x) = i (i.e. (pa(y), x) is associated with Z(i)), but not with Z(i−1), referred to as begin(i).

The algorithm below makes use of the procedure bestReceiver which takes as input a node g ∈ I(G), a child x of g, and an edge e from S and returns, from among all those edges that share at least one common node x with an edge (pa(y), y) for which the value in(y.g) is smallest. Essentially, the returned edge (pa(y), y) implies that, in a scenario where g maps to x and g is a transfer node with (g, x) ∈ Σ, the best possible mapping for x (i.e. one for which c(g,x) is minimized) is 2i−1, and secondBest(x,i) to index 2i, and querying the data structure accordingly. We denote this data structure by Φx.

16. Delete the heap H.
17. For each s ∈ V(S) in post-order do
18. If s ≠ rt(S), then let (pa(s), s) = bestReceiver(g, g', s), and (pa(s), s) = bestReceiver(g, g', s).

19. This part of the algorithm is identical to Steps 3 through 22 of Algorithm U-Reconcile, except:
   (a) Steps 13 and 20 are replaced by the following:
      If s ≠ rt(S), then c(g, s) = P(s) + min{w(g', y) + c(g', y), w(g', y) + c(g', y)}, and,
   (b) Steps 13 and 20 are removed.
20. Delete the data structures Φx and Φy.

Procedure bestReceiver is implemented as follows:

Procedure bestReceiver(g, s)
1. Query the data structure Φx with the query range (rt(pa(s)), rt(s)−1). Let e denote the returned edge.
2. If e happens to be the edge (pa(s), s), then remove e from Φx, and repeat the above step.
3. Reinsert any removed edges back into Φx.
4. Return e.

Theorem 3.2. The D-MPR problem on G and S can be solved in O(mlog n) time.

3.3 Considering distance-dependent transfer costs

Under the current reconciliation model, all transfers have the same cost irrespective of the span of the transfer. However, it has been observed that transfers are more likely to occur between closely related species than between distantly related ones (Andam and Gogarten, 2008). This suggests that, ideally, the cost of a transfer should depend on the phylogenetic distance between the donating and receiving species. Such a cost scheme could be implemented in several different ways: one straightforward way to implement this is to define the transfer cost between species u and b to be P(a, b) = d(a, b) · δy, where δy ≥ 0. If branch lengths are available on the species tree, d(a, b) could also be replaced by a term that counts the total branch length between u and b. A simpler alternative is to have different constant transfer costs for different ranges of transfer spans.

Next, we give a simple O(mn²)-time algorithm for the (general) MPR problem that can work with undated, partially dated, or fully dated species trees and can handle distance-dependent transfer costs. This makes it a factor of n faster than the fastest known algorithm that can handle distance-dependent transfer costs. Our algorithm, which we will refer to as algorithm D-MPR, is essentially the same as algorithm U-Reconcile, except that we remove our dependence on the out array and assign a cost of ∞ to those transfers that violate any given time constraints. Specifically, we (i) remove Lines 18 and 21 through 22 and (ii) replace Steps 13 and 20 with the following five:

Let X = {x ∈ V(S): x is incomparable to and potentially coexisting with x}.

If X ≠ ∅ then...
for each \( x \in X \)

\[
\text{Temp}(x) = P_d(x) + \min \left[ \text{Temp}(g'_x), \text{Temp}(g''_x) \right]
\]

\[
\text{cost}(g_x) = \min_{x \in X} \text{Temp}(x)
\]

Given any \( a \) and \( b \), the value of \( P_d(a, b) \) under distance-dependent transfer costs can be computed in constant time as long as the value \( d_2(a, b) \) (or its equivalent in terms of branch lengths) can be computed in constant time. This can be achieved after an \( O(n) \) preprocessing of the species tree, which (i) allows constant time LCA querying \( \text{Bender et al.} \) \( 2000 \) and (ii) labels each species tree node with its distance (or total branch length) from the root. This yields the following theorem.

**Theorem 3.3.** The MPR problem on \( G \) and \( S \) with distance-dependent transfer costs can be solved in \( O(mn^2) \) time.

### 3.4 Algorithmic extensions

**Unrooted gene trees.** If the input gene trees are unrooted, each possible rooted version of the unrooted gene tree is reconciled against the species tree and the goal is to find a reconciliation that has minimum cost among all rootings. Each of our three algorithms described earlier can be easily extended to work with unrooted gene trees without any increase in their respective time complexities. This is done by relying on the oft-used observation \( \text{Chen et al.} \) \( 2009 \) that, w.r.t. any internal node \( g \), all rootings of the tree can be partitioned into three sets, depending on which of the tree edges incident on the node is closest to the root node. We have implemented this feature into our software RANGER-DTL.

**Multiple optimal solutions.** It should be noted that, for any given values of the event costs \( P_A, P_B \), and \( P_{\text{loss}} \), there may be more than one optimal solution for the MPR problem. The \( O(mn^2) \) algorithm above can be easily adapted to output all possible optimal reconciliations for any given problem instance.

**Further enhancements.** It is also possible to extend each of our three algorithms, without any increase in their time complexities, to consider more complex biological scenarios, such as transfers from potentially extinct or unsampled lineages, or transfer from a species tree of 100 (predominantly prokaryotic) species sampled across the tree of life. This biological dataset was analyzed using the same cost parameters \((P_A = 2, P_B = 3, P_{\text{loss}} = 1)\) used in \( \text{David and Alm, 2011} \).

**Running time.** To compare the running time of our algorithms, we used an implementation of our algorithm for DTL reconciliation on undated species trees, referred to as the RANGER-DTL-U program, and compared it against AnGST \( \text{David and Alm, 2011} \) and Mowgli \( \text{Doyon et al., 2011} \), which are two of the most advanced programs implementing the fastest known algorithms for DTL reconciliation on undated species trees and tcDLT reconciliation on fully dated species trees, respectively. When running RANGER-DTL-U and AnGST on these datasets, all divergence-time information (branch lengths) on the nodes of the species tree was ignored. Moreover while both RANGER-DTL and AnGST can efficiently handle unrooted gene trees, Mowgli cannot; thus, we first randomly rooted each of the 4733 gene trees of the biological dataset. Table 1 depicts the results. We find a dramatic improvement in runtime and scalability over both AnGST and Mowgli. For instance, on the 100 simulated 100-taxon datasets, RANGER-DTL-U is an impressive 300 and 4500 times faster than AnGST and Mowgli, respectively. Similar speedups are observed on the biological dataset as well, with RANGER-DTL-U requiring just over a minute to analyze the entire dataset of 4733 gene trees. (Even when run directly on the original unrooted gene trees, it requires only about 2 min to analyze the entire dataset.) Moreover, the speedups are, as anticipated, even greater for larger datasets. AnGST required between 8 and 10 h on each of the 10 randomly chosen 500-taxon datasets that we tried, suggesting a running time of at least 800 h on all 100 datasets, and it crashed immediately on the 1000-taxon datasets. Similarly, Mowgli crashed after \( \sim 4 \) h of running time on each of the 10 randomly chosen 500-taxon datasets that we tried, and did not terminate in 60 h (after which we stopped the program) on any one of the 10 1000-taxon datasets we ran it on. This suggests a total running time of at least 400 and 6000 h on all 100 of the 500- and 1000-taxon datasets, respectively, for Mowgli. In contrast, RANGER-DTL-U required \( \leq 2 \) s on each 1000-taxon dataset, which is, remarkably, over 100 000 times faster than Mowgli. While neither AnGST nor Mowgli can be run on the 10 000-taxon dataset, RANGER-DTL-U required only \( \sim 4 \) h to analyze it.

**Solution quality.** Note that it is ineffective to compare the actual reconciliations themselves as the presence of multiple
Table 1. Runtime comparison

<table>
<thead>
<tr>
<th>Dataset type</th>
<th>Dataset size</th>
<th>RANGER-DTL-U</th>
<th>AnGST</th>
<th>Mowgli</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulated</td>
<td>50 taxa (100 datasets)</td>
<td>2 s</td>
<td>3 m:26 s</td>
<td>28 h:30 s</td>
</tr>
<tr>
<td></td>
<td>100 taxa (100 datasets)</td>
<td>3 s</td>
<td>15 m:4 s</td>
<td>3 h:52 m</td>
</tr>
<tr>
<td></td>
<td>200 taxa (100 datasets)</td>
<td>9 s</td>
<td>1 h:2 m</td>
<td>29 h:43 m</td>
</tr>
<tr>
<td></td>
<td>500 taxa (100 datasets)</td>
<td>35 s</td>
<td>&gt;800 h</td>
<td>&gt;400 h</td>
</tr>
<tr>
<td></td>
<td>1000 taxa (100 datasets)</td>
<td>2 m:57 s</td>
<td>—</td>
<td>&gt;600 h</td>
</tr>
<tr>
<td></td>
<td>10 000 taxa (1 dataset)</td>
<td>4 h:7 m</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Biological</td>
<td>4733 gene trees, 100 taxa species tree</td>
<td>1 m:03 s</td>
<td>3 h:45 m</td>
<td>41 h:36 m</td>
</tr>
</tbody>
</table>

This table shows the runtimes of RANGER-DTL-U, AnGST and Mowgli on simulated and biological datasets. Times are shown in hours (h), minutes (m) and seconds (s). Experiments were performed on a desktop computer with a 3.2 GHz Intel Core i3 processor and 4 GB of RAM.
REFERENCES


