Efficient Algorithms for the Reconciliation Problem with Gene Duplication, Horizontal Transfer and Loss

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1093/bioinformatics/bts225">http://dx.doi.org/10.1093/bioinformatics/bts225</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Oxford University Press (OUP)</td>
</tr>
<tr>
<td>Version</td>
<td>Final published version</td>
</tr>
<tr>
<td>Accessed</td>
<td>Fri Jun 23 10:41:03 EDT 2017</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/72150">http://hdl.handle.net/1721.1/72150</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-NonCommercial 3.0 Unported License</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc/3.0/">http://creativecommons.org/licenses/by-nc/3.0/</a></td>
</tr>
</tbody>
</table>
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

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 inferring orthology, paralogy and xenology (Koonin, 2005; Mikkelsen and Aravind, 2008). 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) (Bonizzoni et al. 2005; Chauve et al. 2008; Durand et al. 2003; Haluška and Vingron, 1994; Goodman et al. 1979; Golecki and Tiuryn, 2006; Mirkin, 2007), or transfer loss (TL) (Bocchini et al. 2010; Brühlmann and Lagergren, 2003; Doyon et al. 2010; Huchon et al. 2004; Pin et al. 2009; Nakhleh et al. 2004; Ronquist et al. 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 families 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 simultaneously is referred to as the duplication TL (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 Hedges, 1998) 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, 1998; Conow et al. 2011; David and Alm, 2011; Gerbunov and Libanov, 2008; Huchon et al. 2004; Vernot et al. 2010; Vernot et al. 2011; Ronquist et al. 2003; Ronquist et al. 2011) (but see also Ovadia et al. 2008 for an example of a probabilistic formulation, as well as Csürös and Miklós, 2004 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 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 Down 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 \textit{tcDTL Reconciliation} problem. Although the DTL reconciliation problem can be solved in polynomial time, solving the t
\textit{tcDTL reconciliation} problem becomes non-polynomially solvable \cite{Bansal}. 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 \textit{tcDTL reconciliation} problem becomes non-polynomially solvable \cite{Libeskind-Hadas}. (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 \cite{Burny, Libeskind-Hadas, Doyon, Conow, Libeskind-Hadas, Charleston, Merkle, Merkle, Morandi, Tofigh, Tofigh}. The most recent algorithmic work on these problems include \cite{Doyon, Tofigh, Tofigh, David, Alt}. The paper by Tofigh et al. \cite{Tofigh} 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 shows that, under 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 \cite{Tofigh} (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 duplcations 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. \cite{Doyon} showed that, for fully dated species trees, the t\textit{c} DT\textit{L} reconciliation problem could be solved in $O(mn^2)$ time. Recently, an $O(mn^2)$-time algorithm for the tc\textit{c} DT\textit{L} reconciliation problem on fully dated trees has also been independently developed for Version 2 of the program \textit{Jane} \cite{Conow}. Finally, the recent paper by David and Alt \cite{David} gave an $O(mn^2)$-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 \cite{David, Merkle} as well as for tc\textit{c} DT\textit{L} reconciliation on fully dated trees \cite{Doyon} 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. \cite{Bansal}, Burleigh et al. \cite{Burleigh}, Forssen and Kellett \cite{Forssen}, and Merkle et al. \cite{Merkle} discuss this challenge in detail.

\textbf{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 \cite{Libeskind-Hadas} and the accuracy of tc\textit{c} DT\textit{L} 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 tc\textit{c} DT\textit{L} 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 tc\textit{c} DT\textit{L} reconciliation.

(3) We give a simple $O(m^2)$-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 \cite{Conow}. 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 \( C_f(v) \), 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 \( x \preceq_T y \) to be the partial order on \( V(T) \), where \( x \preceq_T y \) if \( x \) is a node on the path between \( n(T) \) and \( y \). The partial order \( \preceq_T \) is defined analogously, i.e., \( x \preceq_T y \) if \( x \) is a node on the path between \( n(T) \) and \( x \). We say that \( y \) is an ancestor of \( u \), or that \( u \) is a descendant of \( v \), if \( u \preceq v \) (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 \( u \preceq_T v \) nor \( x \preceq_T y \), where \( u \) is an empty set. \( L(T) \)

Given a non-empty subset \( S \subseteq L(T) \), we denote by \( L\left( T \right) \setminus \left( L\left( S \right) \right) \) the unique smallest upper bound of \( L \) under \( \preceq_T \). Given \( x, y \in V(T) \) and \( x \preceq_T y \), the unique path from \( x \) to \( y \) in \( T \). We denote by \( d_p(x, y) \) 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 \( \chi_G : L(G) \rightarrow L(S) \) that maps a leaf node \( g \in L(G) \) to that unique leaf node \( r(Le(S)) \) with 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 \( \chi_G(g) \) 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 \( T \) 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, \( g \), 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. 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 Tofigh et al. (2011). 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 \( (L, M, \Sigma, \Lambda, \Theta_0, \Theta, \tau) \), where \( L \cdot Le(G) \rightarrow Le(S) \) represents the leaf mapping from \( G \) to \( S \), \( M : V(G) \rightarrow V(S) \) maps each node of \( G \) to a node of \( S \), the sets \( \Sigma, \Lambda, \Theta_0 \) and \( \Theta \) partition \( I(G) \) into speciation, duplication and transfer nodes, respectively; \( \Sigma \subseteq V(S) \) is a subset of gene tree edges that represent transfer edges, and \( \tau : \Theta_0 \rightarrow V(S) \) specifies the recipient species for each transfer event, subject to the following constraints:

1. If \( g \in L(G) \), then \( M(g) = \chi_G(g) \).
2. If \( g \in L(G) \), and \( g' \) and \( g'' \) denote the children of \( g \), then,
Given a DTL scenario, one can directly count the minimum loss associated with each transfer event. Constraint 3(a) imposes on $G$ the leaf mapping $L$ defined to be $L = \{(s_0, g_0), (s_1, g_1), \ldots, (s_n, g_n)\}$ such that $L$ may map to any node, say $x$.

The number of losses $\text{losses}(x)$ at node $x$ is defined to be

$$\text{losses}(x) = 0$$

if $x \notin G$, and $\text{losses}(x)$ is the cost of an optimal reconciliation of $G(x)$ with $S$ such that $g$ maps to $x$ and $g \in G(x)$.

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

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

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

(d) If $g \in \Theta$ and $(g, g') \in \Xi$, then $M(g)$ and $t$ 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 $t(x)$, i.e., $M(g'') \leq M(x)$.

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:
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 [10] that computes optimal reconciliation scenarios under a simpler reconciliation cost that ignores losses. We compute the values $c_{g}(g,s)$, $c_{A}(g,s)$, and $c_{O}(g,s)$ 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 in, out, $\in Alt(-)$, and out(-) help us in efficiently computing the values $c_{g}(g,s)$, $c_{A}(g,s)$, and $c_{O}(g,s)$ at each dynamic programming step.

For instance, for any $g \in G$, the value of $c_{g}(g,s)$ is simply: \[ c_{g}(g,s) = \infty \] if $s \in L(S)$, and \[ c_{g}(g,s) = \min\{c_{g}^{'}(g,s^{'}), c_{g}^{'}(g,s^{''})\} + \mu\] where $\{g^{'}, s^{'}\} = Ch(g)$ and $\{g^{''}, s^{''}\} = Ch(s)$, if $s \notin L(S)$. The values of $c_{g}(g,s)$ and $c_{A}(g,s)$ can be similarly computed; see Steps 4, 10, 11, 12, 13, 14, and 15. The described post-ordered traversal ensures that when computing the values $c_{g}(g,s)$, $c_{A}(g,s)$, and $c_{O}(g,s)$ at nodes $g \in G$ and $s \in S$, all the required in(-), out(-), and (c-) values have already been computed.

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

1. for each $g \in V(G)$ and $s \in V(S)$ do
2. Initialize $c_{g}(g,s) = \infty$, $c_{A}(g,s) = c_{O}(g,s) = \max$, $\in Alt(g,s)$, and $\max \in Alt(g,s)$ to $\infty$.
3. for each $g \in L(G)$ do
4. Initialize $c_{g}(g,L(g))$ to 0. and, for each $r \geq L(L(g))$, initialize $\min\{c_{g}^{'}(g,s^{'}), c_{g}^{'}(g,s^{''})\}$ to 0.
5. for each $g \in \ell(G)$ in post-order do
6. for each $s \in V(S)$ in post-order do
7. Let $\{g^{'}, s^{'}\} = Ch(g)$.
8. if $s \not\in L(S)$ then
9. $c_{g}(g,s) = \infty$.
10. $c_{A}(g,s) = P_{\Lambda} + c_{g}(g,s) + c_{g}(s)$.
11. if $x \not\in \ell(S)$, then $c_{g}(g,s) = P_{\Lambda} + \min\{c_{g}^{'}(g,s^{'}), c_{g}^{'}(g,s^{''})\} + \max\{c_{g}^{'}(g,s^{'}), c_{g}^{'}(g,s^{''})\}$.
12. $\in Alt(g,s) = \{c_{g}^{'}(g,s^{'}), c_{g}(s)\}$.
13. $\max \in Alt(g,s) = c_{g}(g,s)$.
14. $\in Alt(g,s) = c_{g}(g,s)$.
15. else
16. Let $\{g^{''}, s^{''}\} = Ch(g)$.
17. $c_{g}(g,s) = \min\{c_{g}^{'}(g,s^{'}), c_{g}(s)\}$.
18. $\in Alt(g,s) = \{c_{g}(s), \max\{c_{g}^{'}(g,s^{'}), c_{g}(s)\}\}$.

For each $x \in L(S)$ in pre-order do
21. $\min\{c_{g}(s), \max\{c_{g}^{'}(g,s^{'}), c_{g}(s)\}\}$.
22. $\min\{c_{g}(s), \max\{c_{g}^{'}(g,s^{'}), c_{g}(s)\}\}$.

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

3.2 An $O(mn \log n)$-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 $\tau(s)$. If the largest time stamp assigned to the internal nodes is $k$, then we assign time stamp $k + 1$ to each leaf of $S$. Any two consecutive time stamps $s, x + 1$ define the time zone labeled $x$ on $S$.

Given a node $x \in V(S)$, the species represented by that node exists along the edge $(pa(s), s)$ and is consequently associated with the time stamp interval $[\tau(pa(s)), \tau(s)]$ and the time zones $\tau(pa(s)) \ldots \tau(s) - 1$. Observe that any edge from $E(S)$ is associated with at least one time zone. Given any pair of nodes $s, x' \in V(S) \setminus$ rt($S$), a transfer is allowed between the species represented by those nodes if and only if the two edges $(pa(s), s)$ and $(pa(x'), x')$ overlap in at least one time zone.

Our algorithm for the D-MPR problem, called Algorithm D-Reconcile, makes use of the same overall dynamic programming structure as Algorithm U-Reconcile, and the procedure for computing the values $c_{g}(x)$ and $c_{g}(x)$ remains identical. The difference is in the way $c_{g}(x, s)$ is computed, as we can no longer rely on the out(-) 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 $c_{g}(x, s)$ assuming that $(g, g') \in \mathbb{Z}$, where $g' \in Ch(g)$, 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 $c_{g}(x')$. Then, for each candidate node $s$ under consideration, we efficiently compute the best receiver, for a transfer originating at $s$, by choosing the globally

[287]
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 $O(1)$, $O(\log n)$ and $O(\log \log n)$ time, respectively [98,99]. The dynamic range minimum query data structure maintains an ordered list of numbers and can answer queries that seek the smallest element in a given query range in $O(\log g)$ time and also supports update operations that change the value of an element in the list in $O(\log g)$ time [100,101].

Definitions. Let $k$ denote the number of time zones on the species tree. Given a time zone $t$ ($1 \leq t \leq k$), let $Z(t)$ denote the set of edges from $E(S)$ that are associated with zone $t$. Let $Best(g, i)$ and $secondBest(g, i)$ denote, respectively, the two edges from $Z(i)$ with the smallest value of $in(g)$. Before running Algorithm D-Reconcile, we assume that we have precomputed, for each time zone $t$ ($1 \leq t \leq k$), the following: (i) the set of edges $(pa(g)x, s)\in E(S)$ for which $t(x) \leq t-1$ (i.e. $(pa(s), x)$ is associated with $Z(t-1)$, referred to as end($i$)), and (ii) the set of edges $(pa(s), x)\in E(S)$ for which $t(pa(s)) = t$ (i.e. $(pa(s), x)$ is associated with $Z(i)$, but not with $Z(t-1)$), referred to as begin($i$).

The algorithm below makes use of the procedure bestReceiver which takes as input a node $g\in G$, a child $x$ of $g$, and an edge $s$ from $S$ and returns, from among all those edges that share at least one time zone with $x$, an edge $(pa(y), y)$ for which the value $in(g, x)$ 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 $(y, x)\in S$, the best possible mapping for $x$ (i.e. one for which $c_0(g, x)$ is minimized) is $2i-1$, and $secondBest(s, i)$ to index $2i$, and querying the data structure accordingly. We denote this data structure by $\Phi_x$.

16. Delete the heap $H$.
17. For each $s\in V(S)$ in post-order do
18. If $s\neq r(S)$, then let $\text{bestReceiver}(g, s) = \text{bestReceiver}(g, s', s)$, and  $(pa(v), v) = \text{bestReceiver}(g, g, s)$.
19. This part of the algorithm is identical to Steps 4 through 20 of Algorithm U-Reconcile, except:
   (a) Steps 3 and 4 are replaced by the following:
   If $s\neq r(S)$, then $c_0(g, x) = \min\{c_0(g', x) + c(g', x), \text{int}(g', x) + c_0(g, x)\}$, and
   (b) Steps 5 and 6 are removed.
20. Delete the data structures $\Phi_x$ and $\Phi_y$.
21. Return $\text{min}_{u\in G}c_0(r(G), u)$.

Procedure bestReceiver($g, s$)
1. Query the data structure $\Phi_x$ with the query range $\{(pa(s), s), t(s) - 1\}$. Let $e$ denote the returned edge.
2. If $e$ happens to be the edge $(pa(s), x)$, then remove $e$ from $\Phi_x$, and repeat the above step.
3. Reinsert any removed edges back into $\Phi_x$.
4. Return $e$.

Theorem 3.2. The D-MPR problem on $G$ and $S$ can be solved in $O(mn\log n)$ 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 [102]. 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 $a$ and $b$ to be $P_0(a, b) = \delta_1 + \delta_2(a, b) - \delta_2$, where $\delta_1, \delta_2 \geq 0$. If branch lengths are available on the species tree, $P_0(a, b)$ could also be replaced by a term that counts the total branch length between $a$ and $b$. A simpler alternative is to have different constant transfer costs for different ranges of transfer spans.

Next, we give a simple $O(mn^2 \log n)$-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-transfer, is essentially as algorithm U-Reconcile, except that we remove our dependence on the $out$ array and assign a cost of $\infty$ to those transfers that violate any given time constraints. Specifically, we (i) remove Lines 18 and 20 through 33, and (ii) replace Steps 33 and 34 with the following five:

Let $X = \{x\in V(S); x$ is incomparable to $y$ and potentially coexisting with $x\}$.
If $X \neq \emptyset$ then

We implemented our fast algorithms into a software package called RANGER-DTL (Rapid ANalysis of Gene family Evolution using Duplication, Transfer, and Loss) (David and Alm, 2011) which are two of the most advanced programs implementing the fastest known algorithms for DTL reconciliation on undated species trees and tcDTL 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 trees 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 4 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.

When running RANGER-DTL-U and AnGST on these datasets, all divergence-time information (branch lengths) on the nodes of the species trees 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 4 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 takes just over a minute to analyze the entire dataset of 4733 gene trees.)

<table>
<thead>
<tr>
<th>Example Evaluation</th>
</tr>
</thead>
</table>
| We implemented our fast algorithms into a software package called RANGER-DTL (Rapid ANalysis of Gene family Evolution using Duplication, Transfer, and Loss) (David and Alm, 2011) which are two of the most advanced programs implementing the fastest known algorithms for DTL reconciliation on undated species trees and tcDTL 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 trees 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 4 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 takes just over a minute to analyze the entire dataset of 4733 gene trees.)

### 4 EXPERIMENTAL EVALUATION

We implemented our fast algorithms into a software package called RANGER-DTL (Rapid ANalysis of Gene family Evolution using Duplication, Transfer, and Loss) (David and Alm, 2011) which are two of the most advanced programs implementing the fastest known algorithms for DTL reconciliation on undated species trees and tcDTL 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 trees 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 4 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 takes just over a minute 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 ~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 taxa and 1000-taxon datasets, respectively, for Mowgli. In contrast, RANGER-DTL-U required <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 ~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>5 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>85 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 taxon species tree</td>
<td>1 m 03 s</td>
<td>3 h 45 m</td>
<td>4 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


