BMC Bioinformatics

RESEARCH Open Access

Maximum parsimony reconciliation in the DTLOR model

Jingyi Liu[1], Ross Mawhorter[3], Nuo Liu[1,2], Santi Santichaivekin[1], Eliot Bush[2] and Ran Libeskind-Hadas[1*]

From The 19th Asia Pacific Bioinformatics Conference (APBC 2021) Tainan, Taiwan. 3-5 February 2021

*Correspondence:
hadas@cs.hmc.edu
[1] Department of Computer Science, Harvey Mudd College, Claremont, CA, USA
Full list of author information is available at the end of the article

Abstract

Background: Analyses of microbial evolution often use reconciliation methods. However, the standard duplication-transfer-loss (DTL) model does not account for the fact that species trees are often not fully sampled and thus, from the perspective of reconciliation, a gene family may enter the species tree from the outside. Moreover, within the genome, genes are often rearranged, causing them to move to new syntenic regions.

Results: We extend the DTL model to account for two events that commonly arise in the evolution of microbes: *origin* of a gene from outside the sampled species tree and *rearrangement* of gene syntenic regions. We describe an efficient algorithm for maximum parsimony reconciliation in this new DTLOR model and then show how it can be extended to account for non-binary gene trees to handle uncertainty in gene tree topologies. Finally, we describe preliminary experimental results from the integration of our algorithm into the existing xenoGI tool for reconstructing the histories of genomic islands in closely related bacteria.

Conclusions: Reconciliation in the DTLOR model can offer new insights into the evolution of microbes that is not currently possible under the DTL model.

Keywords: Microbial evolution, Phylogenetic trees, Maximum parsimony reconciliation

Background

Microbes occupy a vast range of ecological niches [1]. Understanding how particular species have come to occupy their niches requires us to reconstruct how their genomes have evolved over time.

In a clade of closely related microbes with a known gene and species tree, inferring the genetic history can be done through a process called *reconciliation*. This process maps the gene tree to the species tree, and in doing so implies genetic events that explain the discordance between the two trees. The DTL model considers duplication, horizontal gene transfer, and loss events whereas some models consider a subset of these events

© The Author(s), 2021. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/. The Creative Commons Public Domain Dedication waiver (http://creativecommons.org/publicdomain/zero/1.0/) applies to the data made available in this article, unless otherwise stated in a credit line to the data.

(e.g., only duplication and loss) or different types of events (e.g,. incomplete lineage sorting).

While the DTL model is applicable to evolution in microbes, it only allows horizontal transfer between species that are part of the species tree. In the analysis of the evolution of microbes in particular, it is quite common that the species tree is not fully sampled. Thus, from the perspective of performing a reconciliation analysis, a gene family may effectively enter the given species tree via transfer from the outside [2, 3].

In this paper we describe the DTLOR model that addresses this issue by extending the DTL model to allow some or all of the evolution of a gene family to occur outside of the given species tree and for transfers events to occur from the outside. To facilitate the recognition of such entry events, the model also keeps track of the *syntenic region* of each gene as it evolves in the species tree. Two genes are said to be in the same syntenic region if they share a substantial fraction of core genes in a relatively large window around them and, second, they share a certain amount of similarity among all genes in a smaller window around them [2]. Thus, in addition to duplication, transfer, and loss events, the DTLOR model adds *origin* events to indicate that a gene is transferred from outside of the species tree and *rearrangement* events that account for changes in the syntenic regions of genes on the genome.

In the DTL model, reconciliation is generally performed using a maximum parsimony formulation. A positive cost is associated with each type of event and the objective is to find a reconciliation that minimizes the total cost of the incurred events. Efficient algorithms have been developed for maximum parsimony reconciliations (MPRs) in the DTL model [4–6], and several software tools implement these algorithms [7–10].

In earlier related work, Delabre et al. studied a related problem of reconciliation with synteny information in the Duplication-Loss model; horizontal transfer events were not considered in that work [11]. Szöllősi et al. [3] proposed an event called “transfer from the dead” to account for gene evolution that occurs outside of the species tree and Jacox et al. [10] described an extension of an existing DTL maximum parsimony reconciliation algorithm to compute most parsimonious reconciliations with this additional event. Our work differs in two significant ways from that prior work. First, while “transfer from the dead” allows gene lineages to transfer out of and back into the sampled species trees multiple times, the DTLOR model only permits transfer from the outside under the assumption that the species tree comprises closely-related species and, thus, transfers out of the species tree and back in are considered to be relatively rare. Second, the DTLOR model captures rearrangement events, which are not considered in conjunction with DTL events in previous models. Reconstructing rearrangement events is particularly important in identifying genomic islands in bacteria [2].

In summary, in this paper we extend the DTL model to allow origin (O) and rearrangement (R) events. We give an exact polynomial-time algorithm for maximum parsimony reconciliation in the DTLOR model. Since gene trees are often non-binary due to lack of signal in their sequence data, we show how maximum parsimony reconciliations can be found in fixed-parameter polynomial time for non-binary gene trees where the parameter is the maximum branching factor of a node. Finally, we describe preliminary results from the integration of the DTLOR MPR algorithm into the xenoGI tool [2] which may provide new insights into microbial evolution.

Definitions

An instance of the DTLOR reconciliation problem comprises undated rooted species and gene trees, S and G, respectively; a positive integer syntenic region number associated with each leaf vertex (extant gene) in G; and a mapping of the leaves of G to the leaves of S. We assume that both trees are binary, but consider the case that gene trees may be non-binary in Sect. 4. Some leaves of the gene tree may be in the same syntenic region while others may be in unique syntenic regions. The DTLOR model comprises the standard DTL events (duplication, transfer, and loss; described in detail below) [4] and two additional events called *origin* and *rearrangement*. Each of those five event types has an associated positive cost.

A syntenic region number is a positive integer from the set of syntenic region numbers of the leaves of the gene tree (called an *actual syntenic region*) or the special *unknown syntenic region* symbol $*$. When a gene vertex is labeled with $*$, that vertex is assumed to be evolving outside the species tree. When a gene vertex is assigned an actual syntenic region but its parent has unknown syntenic region, this means that the gene entered the species tree through transfer from outside, inducing an origin event. Rearrangement indicates a change in syntenic region that happens during the course of evolution within the species tree.

The rules for syntenic region numbering are as follows:

1. If a vertex u is labeled $*$ and v is a child of u, then v may be labeled with either $*$ or an actual syntenic region number.
2. If a vertex u is labeled with an actual syntenic region number, then its children must be labeled with actual syntenic region numbers. Note that this implies that any vertex labeled with an actual syntenic region number has the property that all of its descendants are also labeled with actual syntenic region numbers.

Constraint 1 ensures that genes can originate outside the species tree while constraint 2 ensures that once a gene is found in the species tree it continues evolving within the tree.

The DTL events in this model are analogous to those in the DTL model. O and R events are induced as follows:

1. If a vertex u is labeled $*$ and a child v is labeled with an actual syntenic region number, then vertex v induces an O event.
2. If a vertex u and its child v have actual syntenic region numbers and those two syntenic region numbers are different, then an R event is induced on the edge between u and v.

The objective of the DTLOR maximum parsimony reconciliation (DTLOR MPR) problem is to map the vertices and edges of the gene tree onto the species tree and to identify a syntenic region number with each internal vertex in the gene tree, minimizing the total cost of the induced events. Note that this model implicitly assumes that duplications are tandem or proximal duplications, and thus a duplication event by itself does not imply a change of syntenic region. A duplication that gives rise to a copy at a different syntenic region is modeled implicitly by a duplication and a rearrangement event. The model can be extended to permit other types of duplication events.

Notation

Let S and G denote a pair of undated species and gene trees, respectively. Throughout this and the next section, we assume that S and G are binary. In Sect. 4 we extend results to non-binary trees.

For a tree T, let root(T) be the root and Le(T) be the set of leaves or *tips*. For a non-root vertex v in the tree, $p(v)$ is the parent of v. For a non-leaf vertex v, v_1 and v_2 denote its two children. We assume that each tree T has an additional *handle* edge, namely an edge $(u, \text{root}(T))$. The handle of S is denoted e^S and the handle of G is denoted e^G. For a vertex v of T, we let $T(v)$ be the subtree of T rooted at v, including its own handle edge e^v from $p(v)$ to v. An edge of a tree T is said to be a *leaf edge* if its terminus is a leaf and is said to be an *internal edge* otherwise.

The DTLOR MPR problem

An instance of the DTLOR-MPR problem is a 10-tuple $(S, G, L, \phi, \gamma, \mathbf{D}, \mathbf{T}, \mathbf{L}, \mathbf{O}, \mathbf{R})$ where:

- $S = (V_S, E_S)$ and $G = (V_G, E_G)$ are binary species and gene trees, respectively;
- L is a finite set of syntenic regions which are represented by counting numbers;
- $\phi : \text{Le}(G) \rightarrow \text{Le}(S)$ is a mapping that associates each leaf of G with a leaf of S;
- $\gamma : \text{Le}(G) \rightarrow L$ is a surjective mapping that associates each leaf of G with a syntenic region;
- Parameters **D**, **T**, **L**, **O**, **R** are positive costs for duplication, transfer, loss, origin, and rearrangement events, described in detail below.

A *reconciliation* in the DTLOR model comprises a pair of mappings (Φ, Γ) that extend the mappings ϕ and γ. Specifically, $\Phi : V(G) \rightarrow V(S) \cup \{N\}$ maps the vertices of G to the vertices of S or the special N location representing a species that is not in the species tree S. The constraints on Φ are as follows:

1. $\Phi(g) = \phi(g)$ for each leaf g of G;
2. If g is an internal vertex of G and $\Phi(g) \neq N$ then the children of g, denoted g_1 and g_2, have the properties that

   (a) $\Phi(g_1) \neq N$ and $\Phi(g_2) \neq N$;
   (b) Neither $\Phi(g_1)$ nor $\Phi(g_2)$ is an ancestor of $\Phi(g)$; and
   (c) At least one of $\Phi(g_1)$ or $\Phi(g_2)$ is equal to or a descendant of $\Phi(g)$.

Constraint 1 ensures that the mapping Φ is consistent with the leaf mapping ϕ while constraint 2 ensures that for any gene vertex mapped to a species vertex, (a) the children of g are also mapped to species vertices, (b) the children are not mapped to species vertices that are ancestral to their parent, and (c) at most one child can transfer to a different clade.

Note that we assume that the trees are undated and it is therefore possible that a mapping that satisfies these constraints is, nonetheless, time-inconsistent in the sense that there is no ordering of the internal nodes of the species tree that is consistent with the set of duplication, transfer, and loss events. But time-inconsistencies in an MPR can be detected in polynomial-time [12, 13]. Moreover, the problem of finding MPRs that are guaranteed to be time-consistent is NP-hard [14].

Note also that unlike the DTL model, which requires every gene vertex to be mapped to a vertex in the species tree, the DTLOR model allows gene vertices to be mapped to the N location which is outside the sampled species tree.

The mapping Φ induces four types of events. For an internal gene tree vertex g, with children g_1 and g_2, and $\Phi(g) \neq N$, the events induced by Φ are as follows:

Speciation event: Vertex g induces a speciation event if one of $\Phi(g_1)$ and $\Phi(g_2)$ is in the left subtree and the other is in the right subtree of $\Phi(g)$.

Duplication event: Vertex g induces a duplication event if each of $\Phi(g_1)$ and $\Phi(g_2)$ is either equal to or a descendant of $\Phi(g)$ but does not satisfy the requirements for a speciation event.

Transfer event: Vertex g induces a transfer event if exactly one of $\Phi(g_1)$ and $\Phi(g_2)$ is either equal to or a descendant of $\Phi(g)$ and the other is neither an ancestor nor a descendant of $\Phi(g)$.

Loss events: Each non-root vertex g (including leaf vertices) may induce zero or more loss events as follows: If $\Phi(p(g)) \neq N$ is ancestral to $\Phi(g)$, then each species vertex s on the path from $\Phi(p(g))$ to $\Phi(g)$ induces a loss event, except for $\Phi(g)$ and also not $\Phi(p(g))$ if $p(g)$ induces a speciation event. For each loss induced by a vertex s on the path from $\Phi(p(g))$ to $\Phi(g)$, we say that g *passes* through s.

If $\Phi(g) = N$ then g induces none of these four types of events.

The mapping $\Gamma : V(G) \rightarrow L \cup \{*\}$ maps each vertex g in G to an element of L or the special syntenic region represented by $*$ indicating that it is in an unknown syntenic region because it occurs outside of the species tree. The constraints on Γ and its relationship to Φ are as follows:

1. $\Gamma(g) = \gamma(g)$ for each leaf g of G;
2. $\Phi(g) = N$ if and only if $\Gamma(g) = *$;
3. If $\Gamma(g) \neq *$ and g has children g_1 and g_2 then $\Gamma(g_1) \neq *$ and $\Gamma(g_2) \neq *$.

Constraint 1 ensures that the mapping Γ is consistent with the leaf mapping γ, constraint 2 ensures that if a gene vertex is mapped outside of the species tree then its syntenic region is not yet established, and constraint 3 ensures that once the syntenic region for a gene node is established, the syntenic regions of its children are also established. The mapping Γ induces events as follows:

Origin event: A non-root vertex g induces an origin event if $\Gamma(p(g)) = *$ and $\Gamma(g) \neq *$. The root vertex $\text{root}(G)$ induces an origin event if $\Gamma(\text{root}(G)) \neq *$.

Rearrangement event: A non-root vertex g induces a rearrangement event if $\Gamma(g) \neq *$, $\Gamma(p(g)) \neq *$, and $\Gamma(p(g)) \neq \Gamma(g)$.

The cost of a reconciliation is defined to be the sum of the number of duplication, transfer, loss, origin, and rearrangement events scaled by the event costs **D**, **T**, **L**, **O**, and **R**, respectively. Speciation events are assigned an implicit cost of zero because a gene is expected to diverge when the species that carries it diverges.

Methods

When a gene vertex g induces an origin event, all of the genes in the subtree $G(g)$ rooted at g must have actual syntenic regions (by rule 3 in the definition of Γ), and the genes in that subtree are mapped to species in S (by rule 2 in the definition of Γ), that is, $\Phi(g') \in V_S$ and $\Gamma(g') \in L$ for all $g' \in G(g)$. The mappings Φ and Γ are only related by the constraint that $\Phi(g) = N$ iff $\Gamma(g) = *$. Thus, if g induces an origin event, then the pair of mappings Φ and Γ restricted to the domain $G(g)$ are independent. Therefore, for an *origin subtree*, a subtree of G whose root induces an origin event, the process of finding an optimal species mapping Φ can be decoupled from the process of finding an optimal syntenic region mapping Γ. Further, by definition, vertices that induce origin events cannot be ancestrally related. Thus, in a reconciliation (Φ, Γ) where g', g'' induce origin events, the species and syntenic region mappings restricted to the origin subtree $G(g')$ are independent of the mappings restricted to the origin subtree $G(g'')$.

For binary gene trees, we use a dynamic programming algorithm to compute the optimal cost of a species mapping of each subtree of the gene tree. Then, we use a second dynamic programming algorithm to compute the optimal cost for the syntenic region mapping for each subtree. Finally, a third algorithm combines these results to find an optimal solution to the DTLOR MPR problem. For non-binary gene trees, this decoupling is no longer possible and a different (and less efficient) algorithm is presented in Sect. 9.

Computing the species map

Next, we give an efficient algorithm for computing an optimal species mapping for each origin subtree $G(g)$. The algorithm is similar to other DTL reconciliation algorithms [4], but the variant used here is useful in the extensions and generalizations in later sections.

For a species mapping Φ, or its restriction to an origin subtree of the gene tree, we say that a gene tree edge e_g is *placed* on species tree edge e_s if either $\Phi(g) = s$ or if the path from $\Phi(p(g))$ to $\Phi(g)$ includes vertex s, unless $p(g)$ is involved in a speciation event at s. As a special case, if g is the root of an origin subtree, then $\Phi(p(g)) = N$. In this case, there is no path from $\Phi(p(g))$ to $\Phi(g)$, so e_g is placed on e_s if and only if $\Phi(g) = s$. If

$\Phi(g) = s$ we say that e_g *terminates* on edge e_s and if $\Phi(g)$ is a descendant of s then a loss event is induced and we say that e_g *continues* on the corresponding child edge of e_s.

Let $C(g)$ denote the optimal cost for a species mapping restricted to the domain of $G(g)$ and let $C(e_g, e_s)$ denote the optimal cost for a species mapping of $G(g)$ such that e_g is placed on e_s. Then $C(g) = \min_{e_s \in E_S} C(e_g, e_s)$.

We now describe an algorithm for computing $C(e_g, e_s)$. The algorithm computes the C table by considering edges in the gene tree bottom-up (postorder): An edge e_g is considered if either g is a leaf or the children edges e_{g_1} and e_{g_2} have already been considered. For each edge e_g under consideration, we now consider each edge e_s in the species tree in postorder.

To compute $C(e_g, e_s)$, we enumerate the four possible cases:

- In the base case, if g and s are leaves, then:

$$C(e_g, e_s) = \begin{cases} 0 & \text{if } \phi(g) = s \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

- If neither g nor s is a leaf, then either g is mapped to s or not. If g is not mapped to s, then it induces a loss at s by being mapped to one of its children. Otherwise, g is mapped to s, which induces either a speciation, duplication, or transfer event, which incurs a corresponding cost. Thus,

$$\begin{aligned} C(e_g, e_s) = &\min\{\text{Spec}(e_g, e_s), \text{Loss}(e_g, e_s) \\ &\text{Dup}(e_g, e_s), \text{Transfer}(e_g, e_s)\} \end{aligned} \tag{2}$$

  where the computation of Spec, Loss, Dup, and Transfer are described below.
- If g is not a leaf but s is a leaf, then speciation and loss at g are not possible, so:

$$C(e_g, e_s) = \min\{\text{Dup}(e_g, e_s), \text{Transfer}(e_g, e_s)\} \tag{3}$$

- If g is a leaf but s is not a leaf, then speciation, duplication and transfer at g are not possible, so:

$$C(e_g, e_s) = \text{Loss}(e_g, e_s) \tag{4}$$

The functions Spec(e_g, e_s), Loss(e_g, e_s), Dup(e_g, e_s), and Transfer(e_g, e_s) are computed as follows:

$$\begin{aligned} \text{Spec}(e_g, e_s) = &\min\{C(e_{g_1}, e_{s_1}) + C(e_{g_2}, e_{s_2}), \\ &C(e_{g_1}, e_{s_2}) + C(e_{g_2}, e_{s_1})\} \end{aligned} \tag{5}$$

$$\text{Loss}(e_g, e_s) = \mathbf{L} + \min\{C(e_g, e_{s_1}), C(e_g, e_{s_2})\} \tag{6}$$

$$\text{Dup}(e_g, e_s) = \mathbf{D} + C(e_{g_1}, e_s) + C(e_{g_2}, e_s) \tag{7}$$

$$\text{Transfer}(e_g, e_s) = \mathbf{T}+ \min \begin{cases} C(e_{g_1}, e_s) + \text{Best-Transfer}(e_{g_2}, e_s), \\ C(e_{g_2}, e_s) + \text{Best-Transfer}(e_{g_1}, e_s) \end{cases} \tag{8}$$

The speciation term (5) considers both ways in which the children edges of e_g can be placed onto the children edges of e_s in a speciation event. The loss term (6) considers both ways in which edge e_g can continue, either on one child of e_s or the other. The duplication term (7) places both children edges of e_g on e_s. In the transfer term (8), we consider both ways of selecting the transferred child edge. The non-transferred child edge of e_g remains on e_s, but the transferred child edge is placed on a species edge determined by Best-Transfer; Best-Transfer(e_j, e_s) denotes the minimum cost of a mapping of the subtree $G(g_j)$ assuming that e_j is placed on a species edge that is neither ancestral nor descendant to e_s. In order to compute these values, the algorithm maintains another table called Best-Entry(e_g, e_s) which stores the minimum of $C(e_g, e_i)$ over all e_i in the subtree rooted at e_s. The algorithm is given in Algorithm 1.

```
for each gene edge $e_g$ in postorder do
    for each species edge $e_s$ in postorder do
        if $g$ is not a leaf then
            if $s$ is not a leaf then
                $C(e_g, e_s) = \min\{\text{SPEC}(e_g, e_s), \text{LOSS}(e_g, e_s)$
                  $\text{DUP}(e_g, e_s), \text{TRANSFER}(e_g, e_s)\}$;
            else
                $C(e_g, e_s) = \min\{\text{DUP}(e_g, e_s), \text{TRANSFER}(e_g, e_s)\}$;
            end
        else
            if $s$ is not a leaf then
                $C(e_g, e_s) = \text{LOSS}(e_g, e_s)$
            else
                if $\phi(g) = s$ then $C(e_g, e_s) = 0$ else
                  $C(e_g, e_s) = \infty$;
            end
        end
        if $s$ is a leaf then
            $\text{BEST-ENTRY}(e_g, e_s) = C(e_g, e_s)$
        else
            $\text{BEST-ENTRY}(e_g, e_s) =$
              $\min\{C(e_g, e_s), \text{BEST-ENTRY}(e_g, e_{s_1}),$
              $\text{BEST-ENTRY}(e_g, e_{s_2})\}$;
        end
    end
    $\text{BEST-TRANSFER}(e_g, e^S) = \infty$ ;
    for each non-handle species edge $e_s$ in preorder do
        $\text{BEST-TRANSFER}(e_g, e_{s_1}) =$
          $\min\{\text{BEST-TRANSFER}(e_g, e_s), \text{BEST-ENTRY}(e_g, e_{s_2})\}$;

        $\text{BEST-TRANSFER}(e_g, e_{s_2}) =$
          $\min\{\text{BEST-TRANSFER}(e_g, e_s), \text{BEST-ENTRY}(e_g, e_{s_1})\}$;

    end
    $C(g) = \min_{e_s \in E_S} C(e_g, e_s)$
end
```

Computing the synteny map

We use another dynamic programming algorithm to find the optimal cost for a syntenic region mapping for each subtree $G(g)$. Let syn(g) denote the optimal cost for a syntenic region mapping of $G(g)$. Let $\text{syn}(g, \ell)$ denote the optimal cost for a syntenic region mapping of $G(g)$ such that g has the syntenic region ℓ. Then $\text{syn}(g) = \min_{\ell \in L} \text{syn}(g, \ell)$.

If g is a leaf, then $\Gamma(g) = \gamma(g)$ (by rule 1 in the definition of Γ). Thus,

$$\text{syn}(g, \ell) = \begin{cases} 0 & \text{if } \gamma(g) = \ell \\ \infty & \text{otherwise} \end{cases} \tag{9}$$

If g is not a leaf, then (recalling that g_1 and g_2 denote the children of g):

$$\begin{aligned} \text{syn}(g, \ell) = &\min\{\text{syn}(g_1, \ell), \mathbf{R} + \text{syn}(g_1)\} + \\ &\min\{\text{syn}(g_2, \ell), \mathbf{R} + \text{syn}(g_2)\} \end{aligned} \tag{10}$$

This accounts for each child g_1 and g_2 either remaining in the same syntenic region as g or potentially changing to a new region and incurring a cost of $\mathbf{R}$. The algorithm for computing $\text{syn}(g, \ell)$ is summarized in Algorithm 2.

```
1  for each gene vertex $g$ in postorder do
2      for each actual syntenic region $\ell \in L$ do
3          if $g$ is a leaf then
4              $syn(g, \ell) = 0$ if $\ell = \gamma(g)$
5              $syn(g, \ell) = \infty$ otherwise
6          else
7              $\text{SYN}(g, \ell) =$
                 $\min\{\text{SYN}(g_1, \ell), \mathbf{R} + \text{SYN}(g_1)\} +$
                 $\min\{\text{SYN}(g_2, \ell), \mathbf{R} + \text{SYN}(g_2)\}$
8          end
9      end
10     $\text{SYN}(g) = \min_{\ell \in L} \text{SYN}(g, \ell)$
11 end
```

Solving the DTLOR MPR problem

Let $\text{Origin}(g)$ denote the cost of reconciling an origin subtree $G(g)$ which, as noted earlier, can be computed as $\text{Origin}(g) = \mathbf{O} + C(g) + \text{syn}(g)$. To find a maximum parsimony reconciliation, we must therefore determine the optimal locations for origin events.

Let $\text{Null}(g)$ be the optimal cost of reconciling $G(g)$ such that g has the unknown syntenic region $*$. Since the given mapping γ of leaves to syntenic regions must be respected, g may not be assigned syntenic region $*$ if g is a leaf. Thus, $\text{Null}(g)$ is calculated as:

$$\text{Null}(g) = \begin{cases} \infty & \text{if } g \text{ is a leaf} \\ \min\{\text{Null}(g_1), \text{Origin}(g_1)\} + & \\ \min\{\text{Null}(g_2), \text{Origin}(g_2)\} & \text{otherwise} \end{cases} \tag{11}$$

The optimal cost for reconciling the entire gene tree G is given by:

$$\text{Opt} = \min\{\text{Null}(\text{root}(G)), \text{Origin}(\text{root}(G))\} \tag{12}$$

The algorithm for computing $\text{Null}(g)$ is summarized in Algorithm 3.

```
1 for each gene vertex $g$ in postorder do
2     $\textsc{Origin}(g) = \mathbf{O} + \textsc{syn}(g) + C(g)$
3     if $g$ is a leaf then
4         $\textsc{Null}(g) = \infty$
5     else
6         $\textsc{Null}(g) =$
            $\min\{\textsc{Null}(g_1), \textsc{Origin}(g_1)\} +$
            $\min\{\textsc{Null}(g_2), \textsc{Origin}(g_2)\}$
7     end
8 end
9 return $\min\{\textsc{Null}(\mathsf{root}(G)), \textsc{Origin}(\mathsf{root}(G))\}$
```

Note that if we wish to reconstruct an optimal solution, the DP tables C, syn, Null can be annotated in the standard way, allowing the solutions to be reconstructed by tracing through the table. We first trace through the Null table to find a set of origin events that produce an optimal solution. For any gene vertex not in any of the origin subtrees induced by these origin events, they are labeled with the unknown syntenic region $*$. For each origin subtree, we trace through the syn table to get an optimal syntenic region mapping, and then we trace through the C table to get an optimal species mapping. Because of loss events, there can be multiple $C(e_g, e_s)$ entries that involve the same gene vertex in an optimal solution. The mapping of that gene vertex corresponds to the lowest such e_s.

The proofs of the following are given in the Appendix:

Lemma 1 *Algorithm 1 correctly computes C(g) for every gene vertex* $g \in V(G)$.

Lemma 2 *Algorithm 2 correctly computes* syn(g) *for every gene vertex* $g \in V(G)$.

Theorem 1 *Algorithm 3 correctly computes the optimal solution to the DTLOR-MPR problem.*

Time complexity

Computing each entry of the C table takes constant time, so the running time for computing the C table is $O(|G||S|)$. Computing the syn table takes $O(|G||L|)$ time, and computing the Origin and Null entries takes $O(|G|)$ time. In total, the asymptotic running time of this algorithm is $O(|G||S| + |G||L|)$.

Non-binary gene trees

While it is generally possible to construct accurate species trees using a variety of methods, gene trees are susceptible to ambiguity due to the relatively little information available in their sequence data. Consequently, phylogenetic trees for genes often have non-binary vertices, also known as *multifurcations* or *soft polytomies*, which correspond to an unknown ordering of the underlying sequence of divergences [15]. In this case, we wish to expand each multifurcation into a sequence of binary divergences, leading to a binary gene tree. Such an expansion is called a *resolution* or *binarization* of the non-binary tree. The DTLOR MPR problem for non-binary trees seeks to find the optimal reconciliation of G and S over all possible resolutions of G.

Unfortunately, the number of resolutions of a non-binary tree can be exponential in the number of vertices in the tree. It is, therefore, impractical to explicitly consider every resolution. However, Kordi and Bansal [15] and Jacox et al. [16] demonstrated the existence of fixed-parameter polynomial-time algorithms for finding maximum parsimony reconciliations for non-binary trees in the DTL model. These algorithms operate in polynomial time assuming the maximum number of children of any non-binary vertex is bounded by some constant k. More precisely, a fixed-parameter algorithm in this context runs in time $O(f(k)p(m, n))$ where m and n denote the sizes of the gene and species trees, k is the maximum branching factor of any gene vertex, $p(m, n)$ is a polynomial in m and n, and $f(k)$ is some function of k which may even be exponential in k. In particular, $f(k)$ in this context is the number of distinct binary resolutions of a tree comprising a root and k children times the size of such a binary resolution, which can be shown to be $f(k) = O(2^k(k-1)!)$. For any fixed k, this value is a fixed constant. Jacox et al. [16] offer an approach that results in an $f(k)$ which is smaller but still potentially exponential in k. Importantly, a fixed-parameter polynomial-time algorithm is much more efficient and practical than an exponential time algorithm such as the naive approach of enumerating all possible resolutions of a non-binary tree which would have running time $O((2^k(k-1)!)^n)$.

In this section, we describe a fixed-parameter polynomial time algorithm for the DTLOR MPR problem. Following the approach of Kordi and Bansal [15], our algorithm expands each individual non-binary vertex into every possible binary resolution but avoids enumerating all possible binary resolutions of the entire tree, thus resulting in a fixed-parameter polynomial-time algorithm rather than an exponential-time algorithm. While our algorithm leverages the important ideas for resolving non-binary vertices first proposed by Kordi and Bansal [15], it requires a new algorithm due to the advent of O and R events.

Each binary resolution of a non-binary gene tree implies a different topology for the gene tree, which induces potentially different costs for both the species mapping and the syntenic region mapping. Note that one resolution may be most favorable for minimizing the cost of the species mapping while a different one may admit the least expensive syntenic region mapping (Fig. 1). Thus, while in binary gene trees the species mapping and syntenic region mappings could be efficiently solved independently and then merged into an optimal solution for the DTLOR MPR problem, the situation is more complicated in the presence of non-binary gene trees. The algorithm presented here considers the species mapping and syntenic region mapping simultaneously as non-binary vertices are resolved one-by-one. Importantly, once the best

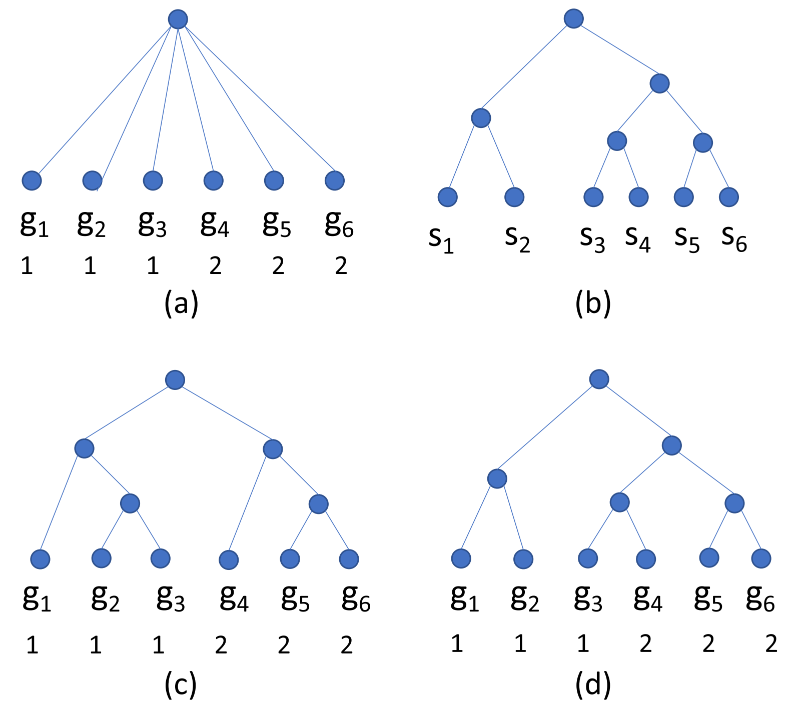
g_1 g_2 g_3 g_4 g_5 g_6
1 1 1 2 2 2
(a)
s_1 s_2 s_3 s_4 s_5 s_6
(b)
g_1 g_2 g_3 g_4 g_5 g_6
1 1 1 2 2 2
(c)
g_1 g_2 g_3 g_4 g_5 g_6
1 1 1 2 2 2
(d)

Fig. 1 An example showing that one resolution of a multifurcation can be optimal for species mapping while another may be optimal for syntenic region mapping. **a** A gene tree with six leaves labeled with their syntenic regions. **b** A species tree. The tip association is $\phi(g_i) = s_i, 1 \leq i \leq 6$. **c** A binary resolution of the gene tree in which the optimal species mapping cost is necessarily greater than the origin cost **O** since this tree is not isomorphic to the species tree; the optimal number of rearrangements in this case is 1. **d** Another binary resolution of the gene tree in which the optimal species mapping cost is just the origin cost **O** since this tree is isomorphic to the species tree; the optimal number of rearrangements in this case is 2

resolution is found for the subtree rooted at a given gene vertex, that value can be saved and used as the dynamic program processes the ancestors of g. For this reason, it is not necessary to consider complete resolutions of the gene tree but, instead, the non-binary vertices can be resolved one-at-a-time. This results in an algorithm that is additive, rather than multiplicative, in the number of resolutions of individual non-binary vertices.

The following definitions and equations assume that the terminus vertex g of edge e_g is either a leaf or has exactly two children. Later, we show how to apply these to non-binary trees. Let $C(e_g, e_s, \ell)$ denote the optimal cost of reconciling the subtree $G(g)$ with S such that e_g is placed on e_s and g has the syntenic region ℓ. Note that in contrast to $C(e_g, e_s)$ used in the previous section, $C(e_g, e_s, \ell)$ also encodes the constraint that g has syntenic region ℓ and the total cost includes the cost of rearrangement events in the subtree $G(g)$. We define Best-Entry(e_g, e_s, ℓ) and Best-Transfer(e_g, e_s, ℓ) analogously to Best-Entry(e_g, e_s) and

Best-Transfer(e_g, e_s), respectively, in the previous section. We define $C(e_g, e_s, L) = \min_{\ell \in L} C(e_g, e_s, \ell)$ and

$$\text{Best-Transfer}(e_g, e_s, L) = \min_{\ell \in L} \text{Best-Transfer}(e_g, e_s, \ell)$$

We compute $C(e_g, e_s, \ell)$ in postorder. There are four cases:

- In the base case, if g and s are leaves, then:

$$C(e_g, e_s, \ell) = \begin{cases} 0 & \text{if } \phi(g) = s \text{ and } \gamma(g) = \ell \\ \infty & \text{otherwise} \end{cases} \tag{13}$$

- If neither g nor s is a leaf, then:

$$\begin{aligned} C(e_g, e_s, \ell) = \min\{&\text{Spec}(e_g, e_s, \ell), \text{Loss}(e_g, e_s, \ell), \\ &\text{Dup}(e_g, e_s, \ell), \text{Transfer}(e_g, e_s, \ell)\} \end{aligned} \tag{14}$$

  where the computation of Spec, Loss, Dup, and Transfer are given below.
- If g is not a leaf but s is a leaf, then:

$$C(e_g, e_s, \ell) = \min\{\text{Dup}(e_g, e_s, \ell), \text{Transfer}(e_g, e_s, \ell)\} \tag{15}$$

- If g is a leaf but s is not a leaf, then:

$$C(e_g, e_s, \ell) = \text{Loss}(e_g, e_s, \ell) \tag{16}$$

The functions $\text{Spec}(e_g, e_s, \ell)$, $\text{Loss}(e_g, e_s, \ell)$, $\text{Dup}(e_g, e_s, \ell)$, and $\text{Transfer}(e_g, e_s, \ell)$ are computed as follows:

$$\begin{aligned} &\text{Spec}(e_g, e_s, \ell) = \min\{ \\ &\quad \min\{C(e_{g_1}, e_{s_1}, \ell), \mathbf{R} + C(e_{g_1}, e_{s_1}, L)\} + \\ &\quad\ \min\{C(e_{g_2}, e_{s_2}, \ell), \mathbf{R} + C(e_{g_2}, e_{s_2}, L)\}, \\ &\quad \min\{C(e_{g_1}, e_{s_2}, \ell), \mathbf{R} + C(e_{g_1}, e_{s_2}, L)\} + \\ &\quad\ \min\{C(e_{g_2}, e_{s_1}, \ell), \mathbf{R} + C(e_{g_2}, e_{s_1}, L)\}\} \end{aligned} \tag{17}$$

$$\text{Loss}(e_g, e_s, \ell) = \mathbf{L} + \min\{C(e_g, e_{s_1}, \ell), C(e_g, e_{s_2}, \ell)\} \tag{18}$$

$$\begin{aligned} &\text{Dup}(e_g, e_s, \ell) = \\ &\quad \mathbf{D} + \min\{C(e_{g_1}, e_s, \ell), \mathbf{R} + C(e_{g_1}, e_s, L)\} \\ &\qquad + \min\{C(e_{g_2}, e_s, \ell), \mathbf{R} + C(e_{g_2}, e_s, L)\} \end{aligned} \tag{19}$$

$$\begin{aligned} &\text{Transfer}(e_g, e_s, \ell) = \mathbf{T} + \min\{ \\ &\quad \min\{C(e_{g_1}, e_s, \ell), \mathbf{R} + C(e_{g_1}, e_s, L)\} + \\ &\quad\ \min\{\text{Best-Transfer}(e_{g_2}, e_s, \ell), \\ &\quad\ \mathbf{R} + \text{Best-Transfer}(e_{g_2}, e_s, L)\}, \\ &\quad\ \min\{C(e_{g_2}, e_s, \ell), \mathbf{R} + C(e_{g_2}, e_s, L)\} + \\ &\quad\ \min\{\text{Best-Transfer}(e_{g_1}, e_s, \ell), \\ &\quad \mathbf{R} + \text{Best-Transfer}(e_{g_1}, e_s, L)\}\} \end{aligned} \tag{20}$$

In order to compute Best-Transfer, we compute Best-Entry(e_g, e_s, ℓ) as follows. If s is a leaf, then

$$\text{Best-Entry}(e_g, e_s, \ell) = C(e_g, e_s, \ell). \tag{21}$$

Otherwise,

$$\begin{aligned}\text{Best-Entry}(e_g, e_s, \ell) = \min\{&C(e_g, e_s, \ell),\\ &\text{Best-Entry}(e_g, e_{s_1}, \ell), \text{Best-Entry}(e_g, e_{s_2}, \ell)\}\end{aligned} \tag{22}$$

Best-Transfer(e_g, e_s, ℓ) is then computed in preorder: First, for the handle edge e^S

$$\text{Best-Transfer}(e_g, e^S, \ell) = \infty \tag{23}$$

For all other edges, e_s with child edges e_{s_1} and e_{s_2}
Best-Transfer$(e_g, e_{s_1}, \ell) = \min\{$

$$\text{Best-Transfer}(e_g, e_s, \ell), \text{Best-Entry}(e_g, e_{s_2}, \ell)\} \tag{24}$$

Best-Transfer$(e_g, e_{s_2}, \ell) = \min\{$

$$\text{Best-Transfer}(e_g, e_s, \ell), \text{Best-Entry}(e_g, e_{s_1}, \ell)\} \tag{25}$$

Now, we consider the case that each internal vertex g has an arbitrary number of children denoted $g_1, \ldots g_k$, $k \geq 2$. A *binary resolution for g* is defined to be a binary tree whose root is g and whose leaves are $g_1, g_2, \ldots, g_k$. Let $BR(g)$ denote the set of all binary resolutions for g. Note that if g has two children, then it has just one binary resolution. Note also that a binary resolution for a vertex g is different from a binary resolution for the entire gene tree; the former only resolves g into a binary subtree whereas the latter resolves all non-binary vertices in G.

Let Null(g) be the optimal cost of reconciling $G(g)$ with S such that g has the unknown syntenic region $*$. Let Origin(g) be the optimal cost of reconciling $G(g)$ with S such that g induces an origin event. Let H be a binary resolution for g and let $G^H(g)$ denote the subtree $G(g)$, along with its handle, such that g and its children have been replaced by H. Note that if g has exactly two children, then $G^H(g) = G(g)$. Let C^H, Best-Entry^H, Best-Transfer^H, Origin^H, Null^H correspond to C, Best-Entry, Best-Transfer, Origin, and Null (from the previous section) for $G^H(g)$.

Let e_h be an edge in H. If e_h is a leaf edge in H (and thus h is one of the children of g in G), then $C^H(e_h, e_s, \ell) = C(e_h, e_s, \ell)$ for all e_s, ℓ, Origin$^H(h) =$ Origin(h), and Null$^H(h) =$ Null(h). Thus, as the algorithm considers each gene edge e_g in postorder, it then considers each binary resolution H of g and the induced subtree $G^H(g)$. Within $G^H(g)$ it considers the edges of H in postorder to compute the optimal reconciliation for $G^H(g)$. Finally, the optimal reconciliation over all binary resolutions H of g yields the optimal reconciliation for $G(g)$. The algorithm is summarized in Algorithm 4. (Recall that h_1 and h_2 denote the two children of vertex h.)

```
Initialize all entries in tables used below to $\infty$;
for each gene edge $e_g$ in postorder do
    if $e_g$ is a leaf edge then
        $\textsc{Origin}(g) = \mathbf{O}$ ;
        $C(e_g, e_{\phi(g)}, \gamma(g)) = 0$;
    end
    for each $H \in BR(g)$ do
        for each leaf edge $e_h$ in $H$ do
            Initialize $\textsc{Origin}^H(h) = \textsc{Origin}(h)$;
            Initialize $\textsc{Null}^H(h) = \textsc{Null}(h)$;
            for each syntenic region $\ell$ do
                for each species edge $e_s$ in postorder do
                    Initialize $C^H(e_h, e_s, \ell) = C(e_h, e_s, \ell)$;
                end
            end
        end
        for each internal edge $e_h$ in $H$ in postorder do
            for each syntenic region $\ell$ do
                for each species edge $e_s$ in postorder do
                    Compute $C^H(e_h, e_s, \ell)$ ;
                    Compute $\textsc{Best-Entry}^H(e_h, e_s, \ell)$;
                end
                for each species edge $e_s$ in preorder do
                    Compute
                      $\textsc{Best-Transfer}^H(e_h, e_{s_1}, \ell)$;
                    Compute
                      $\textsc{Best-Transfer}^H(e_h, e_{s_2}, \ell)$;
                end
            end
            $\textsc{Origin}^H(h) = \min_{e_s,\ell} C^H(e_h, e_s, \ell) + \mathbf{O}$;
            $\textsc{Null}^H(h) =$
              $\min\{\textsc{Null}^H(h_1), \textsc{Origin}^H(h_1)\} +$
              $\min\{\textsc{Null}^H(h_2), \textsc{Origin}^H(h_2)\}$;
        end
        for each syntenic region $\ell$ do
            for each species edge $e_s$ in postorder do
                $C(e_g, e_s, \ell) =$
                  $\min\{C^H(e_g, e_s, \ell), C(e_g, e_s, \ell)\}$;
                $C(e_g, e_s, L) =$
                  $\min\{C(e_g, e_s, \ell), C(e_g, e_s, L)\}$;
            end
        end
        $\textsc{Origin}(g) = \min\{\textsc{Origin}^H(g), \textsc{Origin}(g)\}$
          $\textsc{Null}(g) = \min\{\textsc{Null}^H(g), \textsc{Null}(g)\}$
    end
end
return $\min\{\textsc{Origin}(\text{root}(G)), \textsc{Null}(\text{root}(G))\}$
```

Theorem 2 *Algorithm 4 correctly computes the optimal solution to the DTLOR-MPR problem for non-binary gene trees.*

The proof is summarized in the Appendix.

Time complexity

The algorithm first initializes Origin, Null, C entries for all the leaf edges of the gene tree, which takes $O(|G|)$ time. Then, for each internal gene edge e_g, the algorithm loops through all binary resolutions H at the gene vertex and computes C^H, Best-Entry^H, Best-Transfer^H, Origin^H, and Null^H, which takes $O(|H||S||L|) = O(k|S||L|)$ time. (Here $|H|$ is the size of any binary resolution at a gene vertex, which is bounded by $O(k)$ where k is the maximum degree of any vertex in the gene tree.) For each H, the algorithm updates all entries of $C(e_g, e_s, \ell)$ and $C(e_g, e_s, L)$, which takes $O(|L||S|)$ time. In total, the running time for computing all the DP entries for all gene edges and binary resolutions is $O(f(k)k|G||S||L|)$ time, where $f(k)$ upper bounds the number of binary resolutions at any gene vertex.

Results

The implementation of the DTLOR MPR Algorithm (Algorithm 3) was integrated into the xenoGI software package [2] which seeks to reconstruct the history of genome evolution in clades of microbes. xenoGI takes as input a set of sequenced genomes, identifies gene families within this set, and groups those families by common origin. The previous version of xenoGI created gene families in a species-tree aware way, but did not make use of reconciliations. It was able to map gene families onto the species tree and identify their point of origin. However, it was not able reconstruct events in the subsequent evolution of the gene family (e.g. losses or rearrangements). The integration of the DTLOR MPR algorithm allows xenoGI to reconstruct these subsequent events, providing potentially important new insights into microbial evolution.

Within the new DTLOR version of xenoGI, we construct a gene tree for every family, then reconcile it with the species tree. The resulting reconciliation can be used to refine the family (e.g. split it into multiple parts based on the placement of origin events) and to provide detailed information about the family's subsequent evolution.

Table 1 shows the running time for the DTLOR MPR Algorithm within xenoGI on all gene trees given inputs ranging from 4 to 15 bacterial genomes (species). Trees were constructed using FastTree [17] and MUSCLE [18] In each case, DTLOR was run on every binary gene tree with more than two leaves, with gene trees being rooted in all possible ways. These calculations were performed on a commodity server (50 AMD Opteron 6276 2.3 GHz processors, 503 GB RAM). The DTLOR costs were set at 1, 1, 1, 2, 2, respectively.

In one of our enteric bacterial test data sets (Dataset B in Table 1), we examined DTLOR output for a known genomic island, the Acid Fitness Island (AFI) [19]. The corresponding species tree is shown in Fig. 2. This island is thought to have originated with an insertion of 19 genes on branch s2 (the branch leading to the internal vertex s2) via horizontal transfer from outside the clade. It then evolved in the clade and was inherited in the four descendant strains, with the notable loss of nine genes along the branch leading to E. coli K12 [2]. For nearly all the gene families in this island, DTLOR produced reconciliations that place the origin of the family on the branch leading to s2, and it correctly recognized loss events on the K12 branch where those occurred. In a few cases,

Table 1 Runtimes for four different species trees

Species Tree	# Genomes	# Gene Trees	Median GT size	Total Time (s)
A	4	3288	4	4.3
B	6	4032	6	10.6
C	11	4750	9	128.8
D	15	5510	9	563.8

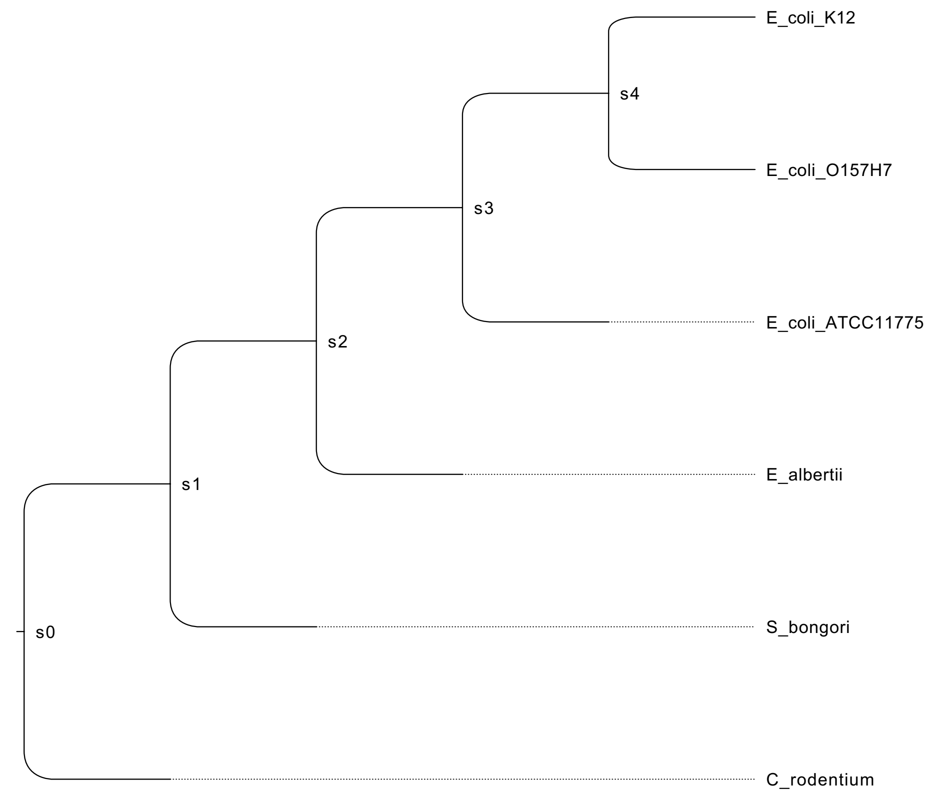

Fig. 2 The species tree for Dataset B

there were multiple most parsimonious reconciliations (MPRs), one of which agreed with the above scenario and was deemed correct, and the others did not agree. Finally there is one family (glutamate decarboxylase) with an evidently complicated post-insertion evolution that is not fully understood. In this case, none of the MPRs using the selected event costs appear to be correct. (The evolution of this family likely involved gene conversion, but the MPR lacks transfer events using the event costs that we used in this experiment.)

Discussion

Several important problems remain to be studied. First, the impact of event costs is not well-understood. Just as in the case of the DTL model, different event costs can give rise to different reconciliations which, in turn, can lead to different conclusions. We believe that the costscape algorithm developed for the DTL model [20] may be extendible to the DTLOR model, which would provide insights into the impact of event costs to the solution space.

Second, it is often the case that there are many distinct MPRs. In fact, even in the DTL model, the number of MPRs can be exponential in the size of the two trees [21]. A subset of these MPRs may be of particular interest because they contain certain evolutionary events that are strongly believed to have occurred (e.g., a horizontal transfer on a particular branch of the species tree). It is desirable, therefore, to efficiently filter the set of MPRs to only include those that contain a specified set of events, count the number of

MPRs in the filtered set, compute support values on the constituent events in that space of MPRs, and select representative reconciliations from this set.

Finally, further systematic studies are needed to determine the full impact of the DTLOR MPR algorithm on the analyses that can now be performed with the enhanced xenoGI tool, including the case of non-binary gene trees.

Conclusions

In this paper, we have described the DTLOR model which extends the well-known DTL model to include origin and rearrangement events. This model is particularly applicable to the evolution of microbes where the species tree is, in many cases, not fully sampled. Therefore, reconciliations must be able to account for transfer events from outside the sampled tree. In addition, the DTLOR model allows for syntenic rearrangement, which is also prevalent in microbial gene families.

We have described efficient algorithms for maximum parsimony reconciliation in the DTLOR model. In binary gene trees, our algorithm solves the DTL reconciliation problem and the sytnenic region problems independently and then combines the results of those two algorithms, resulting in a particularly efficient solution. When the gene tree is non-binary, the two subproblems can no longer be decoupled in this way, and our algorithm for this case considers all of the events simultaneously.

Appendix

This appendix contains proofs of results from the main paper.

Proof of Lemma 1

Proof

We prove that the algorithm correctly computes $C(e_g, \cdot)$ and Best-Transfer$(e_g, \cdot)$ *by structural induction on G. For the base case for $C(e_g, \cdot)$, consider a leaf edge e_g. We perform structural induction on S. In the base case where e_s is a leaf edge, g must map to $\phi(g)$, so $C(e_g, e_s)$ is computed correctly by equation 1. In the inductive step, consider a non-leaf edge e_s. In this case, g must induce a loss event at s since g is a leaf, so $C(e_g, e_s) = \text{Loss}(e_g, e_s)$ (by Eq. 4). By the inductive hypothesis, for each descendant branch $e_{s'}$ of e_s, $C(e_g, e_{s'})$ is computed correctly, so* Loss(e_g, e_s) *is computed correctly (by Eq. 6). This concludes the base case for $C(e_g, \cdot)$.*

In the base case for Best-Transfer$(e_g, \cdot)$, we consider a leaf edge e_g. Since its correctness relies on Best-Entry$(e_g, \cdot)$, we first use structural induction on S to prove that Best-Entry$(e_g, \cdot)$ is correctly computed. In the base case, e_s is a leaf edge, so the only choice for e_g to enter the subtree rooted at e_s is at e_s. Since $C(e_g, e_s)$ is correctly computed, Best-Entry(e_g, e_s) is also correctly computed on line 17. In the inductive step, consider a non-leaf edge e_s. By the inductive hypothesis, Best-Entry$(e_g, e_{s'})$ is computed correctly for each descendant edge $e_{s'}$ of e_s. The ways for e_g to enter the subtree rooted at e_s are at e_s, the left subtree of e_s, or the right subtree of e_s, thus Best-Entry(e_g, e_s) is computed correctly on line 19.

Now we prove the base case for Best-Transfer$(e_g, \cdot)$ using structural induction on S from the handle edge e^S. In the base case where $e_s = e^S$, all edges in S is a descendent of e^S, so there is no valid species edge for the child edge of e_g to transfer to. Thus Best-Transfer(e_g, e^S) is computed correctly on line 22. In the inductive step, we consider a non-root edge e_{s_1}, which has a sibling edge e_{s_2}. By the inductive hypothesis Best-Transfer$(e_g, p(e_{s_1}))$ is computed correctly and by the inductive proof on Best-Entry$(e_g, \cdot)$, Best-Entry(e_g, e_{s_2}) is computed correctly. Since the species edges that e_g are allowed to transfer to from e_{s_1} not only include the same edges if e_g were to transfer from $p(e_{s_1})$, but also edges in the subtree rooted at e_{s_2}, and the optimal cost of placing e_g inside the subtree rooted at e_{s_2} is given by Best-Entry(e_g, e_{s_2}), Best-Transfer(e_g, e_s) is computed correctly. This concludes the base case for Best-Transfer$(e_g, \cdot)$.

To conclude the proof of correctness of $C(e_g, \cdot)$, we consider a non-leaf edge e_g. We use structural induction on S. In the base case, e_s is a leaf edge; the only two possibilities are e_g duplicates on e_s or transfers on e_s. The correctness of Dup(e_g, e_s) is guaranteed by the correctness of $C(e_{g'}, \cdot)$, while the correctness of Transfer(e_g, e_s) is guaranteed by the correctness of both $C(e_{g'}, \cdot)$ and Best-Transfer$(e_{g'}, \cdot)$ for each descendant edge $e_{g'}$ of e_g . In the inductive step, e_s is not a leaf edge, then $C(e_g, e_s)$ is, by definition, the minimum of Spec(e_g, e_s), Dup(e_g, e_s), Transfer(e_g, e_s), and Loss(e_g, e_s) (see Eq. 2). Again, Spec(e_g, e_s), Dup(e_g, e_s) and

Transfer(e_g, e_s) are computed correctly by the correctness of $C(e_{g'}, \cdot)$ and Best-Transfer$(e_{g'}, \cdot)$. The correctness of Loss(e_g, e_s) is guaranteed by the inductive hypothesis on the correctness of $C(e_g, e_{s'})$ for every descendant edge $e_{s'}$ of e_s. This concludes the inductive step for $C(e_g, e_s)$. The inductive step for Best-Transfer$(e_g, \cdot)$ is analogous to the proof for the base case.

Finally, $C(g)$ is the optimal cost for a species mapping of $G(g)$ such that g can be mapped to any species in S. By definition, $C(g) = \min_s \mathcal{C}(g, s)$ where $\mathcal{C}(g, s)$ denotes the optimal cost for a species mapping of $G(g)$ in which g is mapped to s.

Consider $\min_{e_s} C(e_g, e_s)$. Since $C(e_g, e_s)$ is the optimal cost of a species mapping where e_g is placed on e_s, this implies a mapping of g to s or one of its descendants. If g is mapped to s, then $C(e_g, e_s) = \mathcal{C}(g, s)$. If $C(e_g, e_s)$ involves a mapping of g to s' which is a descendant of s, then it induces loss events. Because loss events have a non-negative cost, $C(e_g, e_s') \leq C(e_g, e_s)$, and thus $\min_{e_s} C(e_g, e_s)$ only includes those entries of $C(e_g, e_s)$ where g is mapped to s. Thus, $C(g) = \min_s \mathcal{C}(g, s) = \min_{e_s} C(e_g, e_s)$.

□

Proof of Lemma 2

Proof
We prove the correctness of $\mathrm{syn}(g,\ell)$ *and* $\mathrm{syn}(g)$ *by induction on g. In the base case, g is a leaf, then the syntenic region of g must be* $\gamma(g)$*, so both* $\mathrm{syn}(g,\ell)$ *and* $\mathrm{syn}(g)$ *are computed correctly on line 5. In the inductive step, consider an internal vertex g. By the inductive hypothesis,* $\mathrm{syn}(g_i,\ell')$ *and* $\mathrm{syn}(g_i)$ *is computed correctly for* $g_i \in \{g_1,g_2\}$ *and* $\ell' \in L$*. The syntenic regions for the left and the right child are chosen independently, and the total cost for* $\mathrm{syn}(g,\ell)$ *is the sum of the costs for choosing syntenic regions for the left subtree and the right subtree of g. If* g_i *is a child of g with syntenic region* ℓ*, then no R event is induced and the cost is* $\mathrm{syn}(g_i,\ell)$*. If an R event is induced by a change in syntenic region, then it is optimal to choose a syntenic region that minimizes the total cost of choosing syntenic regions for* $G(g_i)$*. Thus, the cost for a syntenic region mapping of* $G(g_i)$ *with an R event is* $\mathrm{syn}(g_i) + \mathbf{R}$*. The cost for a syntenic region mapping of* $G(g_i)$ *as a whole is the minimum of these two possibilities, and this is the same for both children of g. Thus,* $\mathrm{syn}(g,\ell)$ *is computed correctly on line 7. Then, by definition,* $\mathrm{syn}(g)$ *is also computed correctly by taking the minimum of* $\mathrm{syn}(g,\ell)$ *over all* $\ell \in L$*.* □

Proof of Theorem 1

Proof
First we prove the correctness of $\mathrm{Origin}(g)$*. Since a species mapping for an origin subtree G(g) is independent of a syntenic region mapping of the same subtree, the optimal cost of a reconciliation of G(g) is the optimal cost C(g) for a species mapping and the optimal cost* $\mathrm{syn}(g)$ *for a syntenic region mapping and the cost of the origin event at g. Thus* $\mathrm{Origin}(g)$ *is computed correctly on line 2.*

Now we prove $\mathrm{Null}(g)$ is computed correctly by an induction on g. In the base case, g is a leaf, so g must have a known syntenic region $\gamma(g)$. Thus $\mathrm{Null}(g) = \infty$ as computed on line 4. In the inductive case, consider an internal vertex g. By the inductive hypothesis, $\mathrm{Null}(g_i)$ is computed correctly for any $g_i \in \{g_1,g_2\}$. There are two cases we need to consider for each child since it can either induce an origin event or remain unassigned to any real syntenic region. Since we already know $\mathrm{Origin}(g_i)$ and $\mathrm{Null}(g_i)$ are computed correctly, taking the minimum over the two cases yield the optimal cost for assigning each child and the sum of the costs for both children is the optimal cost for assigning g to the unknown syntenic region $*$. Thus $\mathrm{Null}(g)$ is computed correctly on line 6.

Since the root of the gene tree can either be mapped to the unknown syntenic region or some actual syntenic region, we minimize over the two cases and get the optimal cost for reconciling the entire gene tree. Using standard DP traceback techniques, we can also obtain the mappings and events involved in an optimal solution. □

Proof of Theorem 2

The correctness of Algorithm 4 is a direct extension of the proof of correctness of Algorithm 1, now using triple induction to account for the third argument, the syntenic region, in the DP table. The correctness for non-binary vertices then extends analogously to the proof in [15] for the DTL model.

Abbreviations
DTLOR: Duplication–transfer–loss–origin–rearrangement; MPR: Maximum parsimony reconciliation.

Acknowledgements
The authors thank Yi-Chieh Wu for valuable discussions and the anonymous reviewers for their comments and suggestions.

About this supplement
This article has been published as part of BMC Bioinformatics Volume 22 Supplement 10 2021: Selected articles from the 19th Asia Pacific Bioinformatics Conference (APBC 2021): bioinformatics. The full contents of the supplement are available at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-22-supplement-10.

Authors' contributions
RLH and EB conceived this research. SS contributed to the model. RLH and NL wrote the first implementation of the algorithm. JL and RM made contributions to the development and analysis of the algorithms, analysis, and proofs of correctness. JL, RM, EB, and RLH wrote the paper. All authors read and approved the final manuscript

Funding
This material is based upon work supported by the National Science Foundation under IIS-1905885 to RLH. Publication costs are funded by support from Harvey Mudd College. The funding agency did not play any role in the design of the methodology, creation of the algorithms, analysis, and interpretation of data, or in writing the manuscript

Availability of data and materials
The DTLOR algorithm for binary trees has been implemented in Python 3 and is available for download at https://github.com/aremath/DTLOR.

Declarations

Competing Interests
The authors declare that they have no competing interests.

Author details
[1]Department of Computer Science, Harvey Mudd College, Claremont, CA, USA. [2]Department of Biology, Harvey Mudd College, Claremont, CA, USA. [3]Department of Computer Science and Engineering, University of California Santa Cruz, Santa Cruz, CA, USA.

Received: 13 June 2021 Accepted: 6 July 2021
Published online: 04 August 2021

References
1. Ochman H, Lawrence JG, Groisman EA. Lateral gene transfer and the nature of bacterial innovation. Nature. 2000;405(6784):299–304.
2. Bush EC, Clark AE, DeRanek CA, Eng A, Forman J, Heath K, Lee AB, Stoebel DM, Wang Z, Wilber M, Wu H. Xenogi: reconstructing the history of genomic island insertions in clades of closely related bacteria. BMC Bioinform. 2018;19(1):32–13211. https://doi.org/10.1186/s12859-018-2038-0.
3. Szöllősi GJ, Tannier E, Lartillot N, Daubin V. Lateral gene transfer from the dead. Syst Biol. 2013;62(3):386–97.
4. Bansal MS, Alm EJ, Kellis M. Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. Bioinformatics. 2012;28(12):283–91.
5. Doyon J-P, Scornavacca C, Gorbunov KY, Szöllosi J G, Ranwez V, Berry V. An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. Comp Genomics. 2011;6398:93–108.
6. Tofigh A. Using trees to capture reticulate evolution: lateral gene transfers and cancer progression. PhD thesis, KTH Royal Institute of Technology (2009).
7. Bansal MS, Kellis M, Kordi M, Kundu S. RANGER-DTL 2.0: rigorous reconstruction of gene-family evolution by duplication, transfer and loss. Bioinformatics 2018;34(18):3214–3216 . https://doi.org/10.1093/bioinformatics/bty314.
8. Chen K, Durand D, Farach-Colton M. NOTUNG: A program for dating gene duplications and optimizing gene family trees. J Comput Biol. 2000;7(3–4):429–47. https://doi.org/10.1089/106652700750050871.
9. Conow C, Fielder D, Ovadia Y, Libeskind-Hadas R. Jane: A new tool for cophylogeny reconstruction problem. Algorithms Mol. Biol. 2010;5(16).
10. Jacox E, Chauve C, Szöllősi GJ, Ponty Y, Scornavacca C. ecceTERA: comprehensive gene tree-species tree reconciliation using parsimony. Bioinformatics. 2016;32(13):2056–8. https://doi.org/10.1093/bioinformatics/btw105.

11. Delabre M, El-Mabrouk N, Huber K, Lafond M, Moulton V, Noutahi E, Castellanos M. Evolution through segmental duplications and losses: a super-reconciliation approach. Algorithms Mol Biol AMB. 2020;15.
12. Ma W, Smirnov D, Libeskind-Hadas R. DTL reconciliation repair. BMC Bioinform. 2017;18(3):76. https://doi.org/10.1186/s12859-017-1463-9.
13. Stolzer M, Lai H, Xu M, Sathaye D, Vernot B, Durand D. Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. Bioinformatics. 2012;28(18):409–15.
14. Ovadia Y, Fielder D, Conow C, Libeskind-Hadas R. The cophylogeny reconstruction problem is NP-complete. J Comput Biol. 2011;18(1):59–65.
15. Kordi M, Bansal MS. Exact algorithms for duplication-transfer-loss reconciliation with non-binary gene trees. IEEE Trans Comput Biol Bioinform. 2017. https://doi.org/10.1109/TCBB.2017.2710342.
16. Jacox E, Weller M, Tannier E, Scornavacca C. Resolution and reconciliation of non-binary gene trees with transfers, duplications and losses. Bioinformatics. 2017;33(7):980–7. https://doi.org/10.1093/bioinformatics/btw778.
17. Price MN, Dehal PS, Arkin AP. Fasttree 2 -approximately maximum-likelihood trees for large alignments. PLoS ONE. 2010;5(3):1–10. https://doi.org/10.1371/journal.pone.0009490.
18. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucl Acids Res. 2004;32(5):1792–7. https://doi.org/10.1093/nar/gkh340.
19. Hommais F, Evelyne K, Coppée J-Y, Lacroix C, Yeramian E, Danchin A, Bertin P. Gade (yhie): a novel activator involved in the response to acid environment in *Escherichia coli*. Microbiology (Reading, England). 2004;150:61–72. https://doi.org/10.1099/mic.0.26659-0.
20. Libeskind-Hadas R, Wu Y-C, Bansal MS, Kellis M. Pareto-optimal phylogenetic tree reconciliation. Bioinformatics. 2014;30(12):87–95.
21. Bansal MS, Alm EJ, Kellis M. Reconciliation revisited: handling multiple optima when reconciling with duplication, transfer, and loss. J Comput Biol. 2013;20(10):738–54. https://doi.org/10.1089/cmb.2013.0073.

Ready to submit your research? Choose BMC and benefit from:

- fast, convenient online submission
- thorough peer review by experienced researchers in your field
- rapid publication on acceptance
- support for research data, including large and complex data types
- gold Open Access which fosters wider collaboration and increased citations
- maximum visibility for your research: over 100M website views per year

At BMC, research is always in progress.

Learn more biomedcentral.com/submissions

BMC