The Author(s) *BMC Bioinformatics* 2017, **18**(Suppl 3):76
DOI 10.1186/s12859-017-1463-9

BMC Bioinformatics

RESEARCH Open Access

DTL reconciliation repair

Weiyun Ma[1], Dmitriy Smirnov[2] and Ran Libeskind-Hadas[1*]

From The Fifteenth Asia Pacific Bioinformatics Conference
Shenzhen, China. 16–18 January 2017

Abstract

Background: Maximum parsimony phylogenetic tree reconciliation is an important technique for reconstructing the evolutionary histories of hosts and parasites, genes and species, and other interdependent pairs. Since the problem of finding temporally feasible maximum parsimony reconciliations is NP-complete, current methods use either exact algorithms with exponential worst-case running time or heuristics that do not guarantee optimal solutions.

Results: We offer an efficient new approach that begins with a potentially infeasible maximum parsimony reconciliation and iteratively "repairs" it until it becomes temporally feasible.

Conclusions: In a non-trivial number of cases, this approach finds solutions that are better than those found by the widely-used Jane heuristic.

Keywords: DTL reconciliation, Phylogenetic reconciliation, Undated trees

Background

Phylogenetic tree reconciliation is a fundamental technique for studying the evolution of pairs of entities such as gene families and species, parasites and their hosts, and species and their geographical habitats. The reconciliation problem takes as input two trees and the associations between their leaves and seeks to find a mapping between the trees that accounts for their incongruence. In the Duplication-Transfer-Loss (DTL) model, four types of events are considered: *speciation*, *duplication*, *transfer*, and *loss* [1–7].

Reconciliation in the DTL model is typically performed using a maximum parsimony formulation where each event type has an assigned cost and the objective is to find a reconciliation of minimum total cost. Figure 1a shows a small example of a host and parasite tree and their leaf associations. Figure 1b and c show two different reconciliations of these trees with labels on the events. Speciation is generally considered a "null event" and given cost 0 while the other event types are given positive costs. For example if duplication, transfer, and loss each have cost 1, then the reconciliation in Fig. 1b is optimal and incurs one speciation and one transfer, with total cost of 1. However, if duplication and loss have cost 1 and transfer has cost greater than 4, then the reconciliation in Fig. 1c is optimal, incurring one speciation, one duplication, and three losses, with total cost of 4. Henceforth, we use the terms *optimal* and *maximum parsimony* interchangeably.

A host tree is said to be *dated* if the relative times of its internal nodes are known. For dated host trees, maximum parsimony reconciliations can be found in polynomial time [6, 8, 9]. However, accurately dating host trees is generally difficult [10], and estimated dates may be unreliable. Thus, much of the literature on DTL reconciliation assumes that the host tree is *undated*. Our work addresses the case of undated host trees.

In undated host trees, maximum parsimony reconciliations can be found in polynomial time using dynamic programming [1, 7, 9, 11], but these reconciliations may be *temporally infeasible* in the sense that there exists no ordering of the internal nodes that is consistent with the reconciliation. An example of a temporally infeasible reconciliation is illustrated in Fig. 2. Temporal infeasibility can be detected in polynomial time [11] but the

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

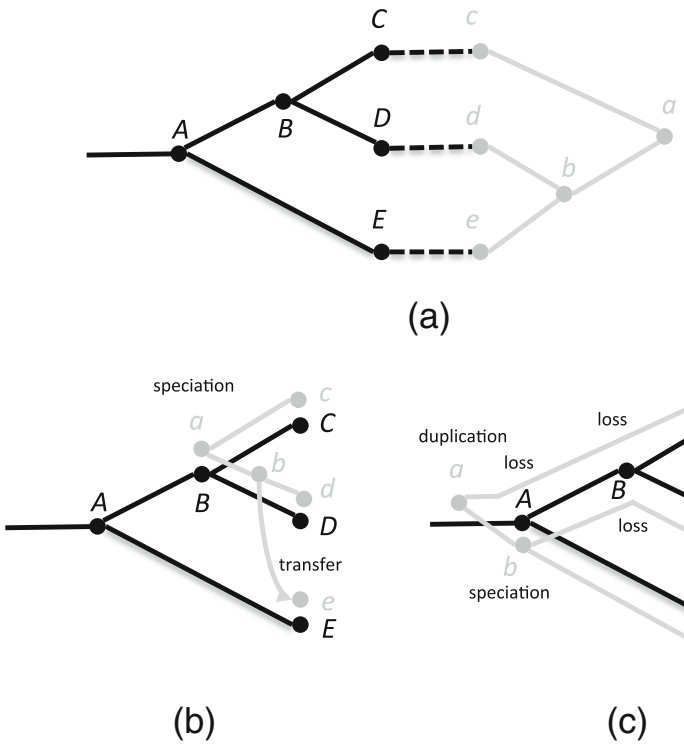

Fig. 1 a A host tree in black and a parasite tree in gray with the leaf mapping shown with dotted lines. Two different reconciliations that are optimal for different event costs are shown in **b** and **c**

problem of finding temporally feasible maximum parsimony reconciliations is NP-complete [7, 12]. Nonetheless, the dynamic programming solution provides a lower bound on the cost of a temporally feasible optimal solution.

In many applications, it is important that reconciliations be both temporally feasible and as close to optimal as possible [13]. Conclusions drawn from temporally infeasible or suboptimal solutions are dramatically weakened [14]. For example, in cophylogenetic studies of hosts and parasites, congruence of the host and parasite trees is assessed via randomization tests: The maximum parsimony cost of the original pair of trees is compared to the maximum parsimony costs for a sample of randomized versions of the data (e.g., randomization of the leaf-mapping or the parasite tree). The fraction of random samples whose cost is as good or better than the cost of the original pair of trees provides an empirical p-value for the null hypothesis that the trees are congruent by chance. Conclusions about the evolutionary histories of hosts and parasites depend, therefore, on accurate comparisons of the samples - meaning that they should all be temporally feasible and their costs should be as close to optimal as possible.

A number of algorithms and software tools have been developed to find temporally feasible maximum parsimony reconciliations in the DTL model. For example, TreeMap [15] uses an exact but exponential time algorithm. ILPEACE [14] and CoRe-ILP [16] find optimal solutions using integer linear programming, which also has exponential worst-case running time. In contrast, the widely-used Jane [17] tool uses a faster meta-heuristic that searches a portion of the large space of possible datings of the host tree and, for each one, finds the maximum parsimony reconciliation for that dating, resulting in a temporally feasible but not necessarily optimal solution.

The solutions found by Jane are often optimal, which is verified when the cost of Jane's solution is equal to the lower bound found by the dynamic programming solution. However, in a substantial number of cases, the dynamic programming solution is not temporally feasible and there is a gap between its cost and the least cost solution found by the Jane heuristic. In such cases, it is desirable to find better temporally feasible solutions using another approach.

In this paper, we propose a new approach for finding temporally feasible reconciliations. This approach runs in polynomial time and, in a non-trivial number of cases (11% in our experiments using the Tree of Life dataset [5]), gives more parsimonious solutions than those found by Jane. Even relatively small improvements in the parsimony cost for a fraction of cases can have profound impact on analyses and conclusions based on tree reconciliation.

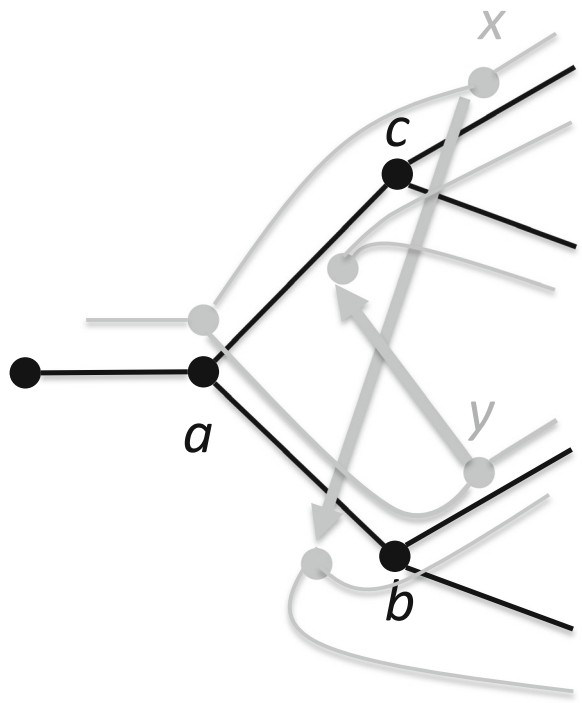

Fig. 2 A fragment of a temporally infeasible reconciliation for a host tree (*black*) and parasite tree (*gray*). Node *x* transfers one child to the species edge from *a* to *b*, implying that *x* must occur before *b* and thus that *c* occurs before *b*. Node *y* transfers one child to the species edge from *a* to *c*, meaning that *y* must occur before *c* and thus that *b* occurs before *c*, contradicting the constraint that *c* occurs before *b*

Our approach uses a combination of both existing and new algorithms. We use the efficient U-MPR dynamic programming algorithm [1] to find a maximum parsimony reconciliation. (Similar algorithms were proposed in [7, 9]. We note that our algorithm is self-contained and fully general and can be applied to reconciliations found by any algorithm). Next, we test that reconciliation for temporal feasibility using an algorithm similar to one proposed by [7]. If the reconciliation is temporally feasible then it is, necessarily, an optimal solution. If, however, the reconciliation is determined to be temporally infeasible, we apply an iterative "repair" process that successively modifies the reconciliation until it becomes feasible. This process terminates, is efficient, and has an upper-bound on the increase in the cost of the solution.

We note that seminal work by Tofigh et al. [7] explores repairing temporally infeasible reconciliations in the Duplication-Transfer model. They give an exact algorithm that runs in time exponential in the cost of the reconciliation. The differences between that work and ours is that our algorithm addresses the Duplication-Transfer-Loss model, our algorithm runs in polynomial time but is not exact (i.e., does not guarantee optimal solutions), and we compare our results to the prevailing heuristic tool and show that it performs better in a non-trivial fraction of cases. Indeed, Tofigh et al. [7] note that the exponential running time of their algorithm is not a concern because in a large analysis of synthetically generated data sets, the classical dynamic programming algorithm for maximum parsimony never constructed a temporally infeasible solution [18]. Subsequent work [19] corroborated this pattern in other synthetically generated data but found that in a large real data set from the Tree of Life [5], over 17% of maximum parsimony reconciliations were temporally infeasible.

In summary, in some cases existing heuristics do not find the least cost temporally feasible reconciliations and, in those cases, it is highly desirable to find lower cost reconciliations if possible. In this paper:

1. We show how a combination of existing and new algorithms can be used to efficiently find temporally feasible DTL reconciliations.
2. We provide experimental results that demonstrate that this approach finds better solutions than Jane in over 10% of cases in a large real dataset.
3. We provide a software package called *Cheeta* (www.cs.hmc.edu/~hadas/cheeta) that implements this algorithm and compares the results to those found by Jane.

Preliminaries

We adopt definitions and notation from Bansal [1]. Let T be a rooted tree and denote the sets of nodes, edges, leaves, and internal nodes of T by $V(T)$, $E(T)$, $Le(T)$, and $I(T)$ respectively. Let $rt(T)$ denote the root node of T, $pa_T(v)$ the parent of node v, $Ch_T(v)$ the set of children of v, and $T(v)$ the maximal subtree of T rooted at v. Let $d_T(x, y)$ be the number of edges on the path from x to y. Let $x \preceq_T y$ if y is a node on the path between $rt(T)$ and x (inclusive) and $x \succeq_T y$ if x is a node on the path between $rt(T)$ and y (inclusive). Nodes x and y are said to be *incomparable* if neither $x \preceq_T y$ nor $y \preceq_T x$. Let $lca_T(x, y)$ be the least common ancestor (LCA) of x and y in tree T; that is, $lca_T(x, y)$ is the node z furthest (with respect to d_T) from the root such that $z \succeq_T x$ and $z \succeq_T y$.

DTL-scenarios and reconciliations

Next we give definitions from [1] leading to the definition of the maximum parsimony reconciliation problem.

Definition 1 (DTL-scenario [1]) *A DTL-scenario for trees G and S is a seven-tuple* $(\mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau)$*, where* $\mathcal{L}: Le(G) \rightarrow Le(S)$ *represents a leaf-mapping from G to S,* $\mathcal{M}: V(G) \rightarrow V(S)$ *maps each node of G to a node of S, the sets* Σ*,* Δ*, and* Θ *partition* $I(G)$ *into speciation, duplication, and transfer nodes respectively,* Ξ *is a subset of edges of G that represent transfer edges, and* $\tau: \Theta \rightarrow V(S)$

specifies the recipient (or "landing site") for each transfer event, subject to the following constraints:

Constraint 1: *If $g \in Le(G)$, then $\mathcal{M}(g) = \mathcal{L}(g)$.*

Constraint 2: *If $g \in I(G)$ and g_ℓ and g_r denote the children of g, then,*

1. *$\mathcal{M}(g) \not<_S \mathcal{M}(g_\ell)$ and $\mathcal{M}(g) \not<_S \mathcal{M}(g_r)$,*
2. *At least one of $\mathcal{M}(g_\ell)$ and $\mathcal{M}(g_r)$ is a descendant of $\mathcal{M}(g)$.*

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

Constraint 4: *If $g \in I(G)$ and g_ℓ and g_r denote the children of g, then,*

1. *$g \in \Sigma$ only if $\mathcal{M}(g) = lca_S(\mathcal{M}(g_\ell), \mathcal{M}(g_r))$ and $\mathcal{M}(g_\ell)$ and $\mathcal{M}(g_r)$ are incomparable,*
2. *$g \in \Delta$ only if $\mathcal{M}(g) \geq_S lca_S(\mathcal{M}(g_\ell), \mathcal{M}(g_r))$,*
3. *$g \in \Theta$ if and only if either $(g, g_\ell) \in \Xi$ or $(g, g_r) \in \Xi$.*
4. *If $g \in \Theta$ and $(g,g') \in \Xi$, then $\mathcal{M}(g)$ and $\tau(g)$ must be incomparable, and $\mathcal{M}(g')$ must be a descendant of $\tau(g)$, i.e., $\mathcal{M}(g') \leq_S \tau(g)$.*

These four constraints ensure that (1) $\mathcal{M}$ extends $\mathcal{L}$, (2) $\mathcal{M}$ satisfies the temporal constraints from S and that each internal node in G is associated with at most one transfer event, (3) a transfer can only be to a non-ancestrally related node and (4) an internal node of G is designated with one of the four event types.

Note that while DTL-scenarios represent reconciliations, these reconciliation are not guaranteed to be temporally feasible. Next, losses are inferred from DTL scenarios according to the following definition from [1].

Definition 2 (Losses [1]) *Given a DTL-scenario $\alpha = (\mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau)$ for G and S, let $g \in V(G)$ and $\{g_\ell, g_r\} = Ch_G(g)$. The number of losses $Loss_\alpha(g)$ at node g, is defined to be:*

- *$(d_S(\mathcal{M}(g), \mathcal{M}(g_\ell)) - 1) + (d_S(\mathcal{M}(g), \mathcal{M}(g_r)) - 1)$, if $g \in \Sigma$,*
- *$d_S(\mathcal{M}(g), \mathcal{M}(g_\ell)) + d_S(\mathcal{M}(g), \mathcal{M}(g_r))$, if $g \in \Delta$, and*
- *$d_S(\mathcal{M}(g), \mathcal{M}(g_r)) + d_S(\tau(g), \mathcal{M}(g_\ell))$ if $(g, g_\ell) \in \Xi$.*

The total number of losses in the reconciliation corresponding to the DTL-scenario α is defined to be $Loss_\alpha = \sum_{g \in I(G)} Loss_\alpha(g)$.

Speciations are assumed to have zero cost. Duplications, transfers, and losses have positive costs denoted C_Δ, C_Θ, and C_Λ, respectively.

Definition 3 (Reconciliation cost of a DTL-scenario [1]) *Given a DTL-scenario $\alpha = (\mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau)$ for G and S, the reconciliation cost associated with α is given by $C_\Delta \cdot |\Delta| + C_\Theta \cdot |\Theta| + C_\Lambda \cdot Loss_\alpha$.*

An instance of the maximum parsimony reconciliation problem comprises a gene tree G, a species tree S, a leaf mapping $\mathcal{L} : Le(G) \rightarrow Le(S)$, and positive costs C_Δ, C_Θ, and C_Λ for duplication, transfer, and loss events, respectively. A maximum parsimony reconciliation, henceforth denoted *MPR*, is a DTL-reconciliation of minimum total cost with respect to the given set of event costs.

A number of closely-related dynamic programming algorithms have been given for finding MPRs in undated trees [1, 7, 9]. Here, we use the U-MPR Algorithm from Bansal et al. [1] which has running time of $O(|G||S|)$.

The reconciliation repair algorithm

In this section, we describe a process for computing temporally feasible MPRs. The process begins by using the U-MPR algorithm [1] to find a most parsimonious DTL-scenario. Next, this scenario is tested for temporal feasibility using an algorithm similar to one described by Tofigh [11]. If the scenario is not temporally feasible, a particular gene node is selected and re-mapped to a species node higher up in the species tree, resulting in a new DTL-scenario. The process of choosing a gene node, remapping it, and testing the resulting scenario for feasibility is repeated until the scenario becomes temporally feasible. In this section, we describe the repair process, analyze it, and prove its correctness.

We determine if a DTL-scenario is temporally feasible using a *temporal feasibility graph* $F = (V, E)$ constructed as follows:

1. $V = I(S) \cup I(G) \cup \{\ell\}$ where $I(S)$ and $I(G)$ represent the internal nodes of S and G, respectively, and ℓ is a single node representing all of the leaves of S and G.
2. For each pair of nodes $u, v \in V$ such that u is the parent of v in either S or G, there is a directed edge $(u, v) \in E$.
3. For each $v \in V$ such that its corresponding node in S or G is the parent of a leaf, there is an edge $(v, \ell) \in E$.
4. For each gene node g associated with species node s in the DTL-scenario:
   (a) If the association is via a speciation event, g and s are identified (i.e., g is removed from the graph and all edges entering g are redirected to enter s and all nodes leaving g now leave s).
   (b) If the association is via a duplication event, we add the directed edges $(pa_S(s), g)$ (unless $s = rt(S)$) and (g, s).
   (c) If the association is via a transfer event with landing site s', we add the directed edges $(pa_S(s), g)$, (g, s), $(pa_S(s'), g)$, (g, s').

A directed edge (u, v) represents the constraint that node u must have a date that comes before the date of v. Thus, the edges in 2 and 3 above enforce that ancestor nodes must have dates that come before their descendants. The edges in 4 enforce the relative dates of genes and the species with which they are associated. In particular, 4(c) ensures that the dates of takeoff and landing sites for transfers are contemporaneous. It is easily verified that there exists a dating of the tree in which all events are temporally consistent if and only if the temporal feasibility graph is acyclic. Moreover, if the graph is acyclic, a topological ordering of that graph gives a feasible dating for the species tree in the given DTL-scenario. (One difference between this test and the one in [11] is that we test that the reconciliation is temporally consistent given the takeoff and landing sites for each transfer event. The test in [11] does not specify the landing sites of transfer events and thus may determine that the scenario is feasible by moving landing sites to locations that are not consistent with any DTL-scenario).

If the DTL-scenario is found to be temporally infeasible due to a cycle in the temporal feasibility graph F, then the "repair" process identifies a gene node g, currently mapped to a species node $\mathcal{M}(g)$, such that g is on a cycle in F and such that no other gene node on a cycle in F is mapped to a descendant of $\mathcal{M}(g)$. In general, node g is not unique and the algorithm breaks ties arbitrarily.

Next, the mapping $\mathcal{M}$ is altered so that g is re-mapped (or "pulled up") in the species tree by either moving to the parent of its current species node (if g is a duplication or transfer node) or to the edge above it (if g is a speciation node). The temporal feasibility graph is recomputed and this process is repeated until the temporal feasibility graph has no cycles, resulting in a temporally feasible DTL-scenario. Figure 3 illustrates a small example, and the process is described formally in Algorithms 1, 2, and 3 where Algorithm 1 is the main algorithm which invokes Algorithm 2 to select a gene node on a cycle and Algorithm 3 to move that gene node upward in the species tree.

Algorithm 1: DTL Repair Algorithm

Input: A gene tree G, a species tree S, and event costs for duplication, transfer and loss

Output: A temporally consistent DTL-scenario

```
1 α = result of U-MPR algorithm [1] on the given input
2 g = FindFirstCycle(G, S, α)
3 while g ≠ null do
4 |   α = PullUpGeneNode(G, S, α, g)
5 |   g = FindFirstCycle(G, S, α)
6 return α
```

Algorithm 2: FindFirstCycle

Input: A gene tree G, a species tree S, and a DTL-scenario $\alpha = \langle \mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau \rangle$ for G and S

Output: A node $g \in I(G)$ contained in some cycle in the temporal feasibility graph corresponding to α such that $\mathcal{M}(g)$ comes before $\mathcal{M}(g')$ in post-order for all such g', or *null* if α is temporally consistent

```
1 Compute the temporal feasibility graph F = (V, E) for G, S, and α
2 for g ∈ I(G) sorted by M(g) in post-order do
3 |   if there exists some cycle in F containing g then
4 |   |   return g
5 return null
```

Algorithm 3: PullUpGeneNode

Input: A gene tree G, a species tree S, a DTL-scenario $\alpha = \langle \mathcal{L}, \mathcal{M}, \Sigma, \Delta, \Theta, \Xi, \tau \rangle$ for G and S and a node $g \in I(G)$ such that $\mathcal{M}(g) \neq rt_S(S)$ to be pulled up

Output: A modified DTL-scenario α'

```
1  if g ∈ Σ then
2  |   Σ = Σ − g
3  |   Δ = Δ + g
4  else if g ∈ Δ then
5  |   M(g) = pa_S(M(g))
6  else if g ∈ Θ then
7  |   M(g) = pa_S(M(g))
8  |   if M(g) >_S τ(g) then
9  |   |   Θ = Θ − g
10 |   |   Δ = Δ + g
11 |   |   {g_ℓ, g_r} = Ch_G(g)
12 |   |   if (g, g_ℓ) ∈ Ξ then
13 |   |   |   Ξ = Ξ − (g, g_ℓ)
14 |   |   else
15 |   |   |   Ξ = Ξ − (g, g_r)
16 |   |   τ(g) = null
17 return α
```

Lemma 1 *Algorithm* 1 *terminates.*

Proof We first prove that Algorithm 2 never returns a node $g \in I(G)$ such that $\mathcal{M}(g) = rt(S)$. By way of contradiction, assume that Algorithm 2 returns such a node g.

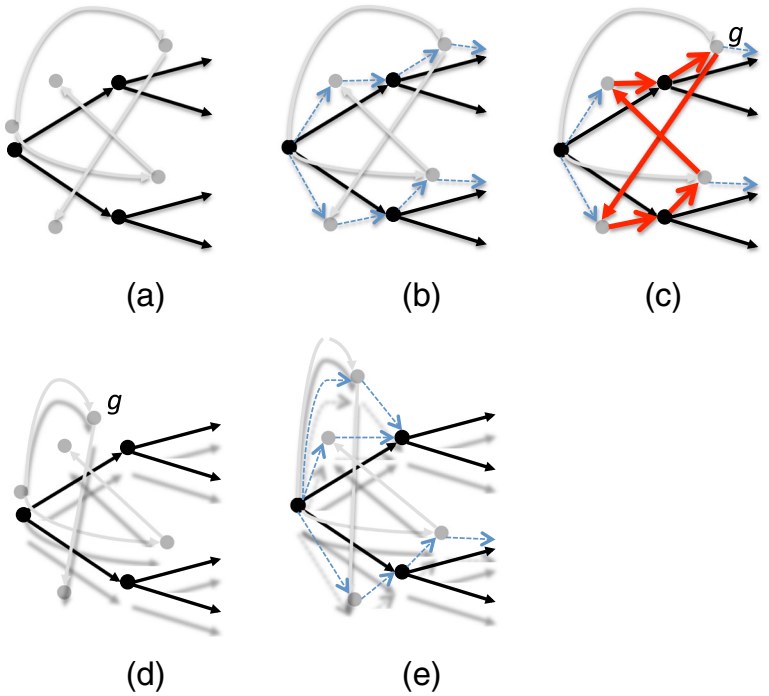

Fig. 3 a The reconciliation from Fig. 2 with unimportant edges removed and edges directed from parent to children nodes. **b** The temporal feasibility graph for this reconciliation. **c** A cycle in the temporal feasibility graph with a minimal vertex g in the gene tree. **d** The modified reconciliation after vertex g is pulled up one level in the species tree. **e** The resulting temporal feasibility graph is acyclic and thus the resulting reconciliation is temporally feasible

Then g is contained in some cycle C in the temporal feasibility graph $F = (V, E)$. Because the for loop on line 2 of Algorithm 2 iterates over the internal nodes of G sorted by $\mathcal{M}(g)$ in post-order, any other node $g' \in I(G)$ (or the node that g' is identified with) contained in cycle C, if it exists, also has $\mathcal{M}(g') = rt(S)$. By construction of F, $(s, g') \notin E$ for any pair of nodes $s \in I(S) \cup \{\ell\}$ and $g' \in I(G)$ such that $\mathcal{M}(g') = rt(S)$. Therefore cycle C does not contain any node $s \in I(S) \cup \{\ell\}$ and consists only of nodes in the set $\{g' \in I(G) : \mathcal{M}(g') = rt(S)\}$. However, by construction of F, the subgraph induced by $\{g' \in I(G) : \mathcal{M}(g') = rt(S)\}$ is acyclic. Thus, we have a contradiction.

Let h_S denote the height of tree S. We now prove that throughout Algorithm 1, Algorithm 2 will return any given node $g \in I(G)$ at most h_S times. If initially $g \notin \Sigma$, then each time after g is returned by Algorithm 2, $\mathcal{M}(g)$ gets remapped to $pa_S(\mathcal{M}(g))$. If initially $g \in \Sigma$, the first time after g is returned by Algorithm 2, $\mathcal{M}(g)$ is not altered, but after every subsequent iteration the above claim applies. Since we have proven that Algorithm 2 will never return a node $g \in I(G)$ such that $\mathcal{M}(g) = rt(S)$, Algorithm 2 returns any $g \in I(G)$ at most h_S times.

It follows directly that the while loop in Algorithm 1 goes through at most $|I(G)|h_S$ iterations. Therefore, Algorithm 1 terminates. □

Lemma 2 *Algorithm* 3 *returns a valid DTL-scenario.*

Proof Let $\mathcal{M}'(\cdot)$ and $\tau'(\cdot)$ denote the updated mappings $\mathcal{M}(\cdot)$ and $\tau(\cdot)$, respectively, when Algorithm 3 returns α at line 17. Let $\{g_\ell, g_r\} = Ch_G(g)$.

We consider each constraint in Definition 1. Clearly, Constraint 1 holds throughout, since we never change $\mathcal{M}(g)$ for a leaf node $g \in Le(g)$.

If g is initially a speciation node ($g \in \Sigma$), then by Constraint 4.1, $\mathcal{M}(g) = lca_S(\mathcal{M}(g_\ell), \mathcal{M}(g_r))$. When Algorithm 3 terminates, we have $\mathcal{M}'(g) = \mathcal{M}(g)$ and g is now a duplication node. We need only confirm that Constraints 2 and 4.2 hold. Indeed, $\mathcal{M}'(g) = \mathcal{M}(g) = lca_S(\mathcal{M}(g_\ell), \mathcal{M}(g_r)) = lca_S(\mathcal{M}'(g_\ell), \mathcal{M}'(g_r))$, since $\mathcal{M}(g_\ell) = \mathcal{M}'(g_\ell)$ and $\mathcal{M}(g_r) = \mathcal{M}'(g_r)$. It follows trivially that Constraint 2 holds as well.

If g is initially a transfer node ($g \in \Theta$), then since Constraints 2 and 4.3 hold at the beginning of the algorithm, exactly one of (g, g_ℓ) and (g, g_r) is in Ξ. Without loss of generality, suppose $(g, g_\ell) \in \Xi$. Then $\mathcal{M}(g) \geq_S \mathcal{M}(g_r)$.

- If $\mathcal{M}'(g) >_S \tau(g)$, then when Algorithm 3 terminates, g is a duplication node and neither (g, g_ℓ) nor (g, g_r) is in Ξ. We need only confirm that Constraints 2, 3, 4.2 and 4.3 hold. Constraints 2 and 4.2 hold since

$\mathcal{M}'(g) >_S \tau(g) \geq_S \mathcal{M}(g_\ell) = \mathcal{M}'(g_\ell)$ and $\mathcal{M}'(g) = pa_S(\mathcal{M}(g)) >_S \mathcal{M}(g) \geq_S \mathcal{M}(g_r) = \mathcal{M}'(g_r)$. It follows that Constraints 3 and 4.3 hold as well.

- Otherwise, since $\mathcal{M}'(g) = pa_S(\mathcal{M}(g)) >_S \mathcal{M}(g)$ and $\mathcal{M}(g)$ and $\tau(g)$ are incomparable, it must be the case that $\mathcal{M}'(g)$ and $\tau(g)$ are incomparable. When Algorithm 3 terminates, g remains a transfer node. We need only confirm that Constraints 2, 3, 4.3 and 4.4 still hold. Indeed, Constraint 4.3 holds since Ξ remains unchanged during Algorithm 3. Constraint 4.4 holds since $\mathcal{M}'(g)$ and $\tau'(g) = \tau(g)$ are incomparable and $\mathcal{M}'(g_\ell) = \mathcal{M}(g_\ell) \leq_S \tau(g) = \tau'(g)$. It follows that $\mathcal{M}'(g)$ and $\mathcal{M}'(g_\ell)$ are also incomparable, so Constraint 3 also holds. Moreover, $\mathcal{M}'(g_r) = \mathcal{M}(g_r) \leq_S \mathcal{M}(g) <_S pa_S(\mathcal{M}(g)) = \mathcal{M}'(g)$, so Constraint 2 holds.

If g is initially a duplication node ($g \in \Delta$), then by Constraint 4.2, $\mathcal{M}(g) \geq_S lca_S(\mathcal{M}(g_\ell), \mathcal{M}(g_r))$. When Algorithm 3 terminates, g remains a duplication node. We need only confirm that Constraints 2 and 4.2 still hold. Indeed, $\mathcal{M}'(g) = pa_S(\mathcal{M}(g)) >_S \mathcal{M}(g) \geq_S lca_S(\mathcal{M}(g_\ell), \mathcal{M}(g_r)) = lca_S(\mathcal{M}'(g_\ell), \mathcal{M}'(g_r))$. It follows trivially that Constraint 2 holds as well. Therefore, Algorithm 3 returns a valid DTL-scenario. □

Lemma 3 *Algorithm* 1 *returns a valid DTL-scenario.*

Proof The algorithm takes as input a valid DTL-scenario, and by Lemma 2, at every iteration of the while loop, a valid DTL-scenario is returned. Therefore, by induction, Algorithm 1 returns a valid DTL-scenario. □

Theorem 1 *Algorithm* 1 *returns a temporally consistent DTL-scenario for G and S.*

Proof By Lemma 1 and Lemma 2, we know that at some point Algorithm 2 returns *null* with input G, S and a valid DTL-scenario α, which means that there does not exist any cycle containing any $g \in I(G)$ in the temporal feasibility graph F corresponding to α. Moreover, by construction of F, the subgraph induced by $I(S) \cup \{\ell\}$ is acyclic. Therefore F is acyclic. It follows that α is temporally consistent. □

Let h_S and h_G denote the heights of trees S and G, respectively. We now bound the running time of Algorithm 1 and the increase in the number of events introduced.

Lemma 4 *The worst-case time complexity of Algorithm* 3 *and Algorithm* 2 *is* $O(1)$ *and* $O(|G|^2 + |G||S|)$*, respectively.*

Proof The time complexity of Algorithm 3 is trivially $O(1)$. For Algorithm 2, note that the size of the temporal feasibility graph F corresponding to any α is in $O(|G| + |S|)$. Therefore it takes $O(|F|) = O(|G| + |S|)$ time to construct F and to check if there exists a cycle in F containing any given node g using depth-first search. Also, since the for loop goes through $O(|G|)$ iterations, the total time complexity of Algorithm 2 is $O(|G|^2 + |G||S|)$. □

Theorem 2 *The worst-case time complexity of Algorithm* 1 *is* $O(|G|^3 h_S + |G|^2|S|h_S)$.

Proof There is a single invocation of the U-MPR algorithm whose worst-case running time is $O(|G||S|)$. Since Algorithm 1 goes through at most $|I(G)|h_S \in O(|G|h_S)$ iterations and it calls Algorithm 3 and Algorithm 2 once respectively at each iteration, from Lemma 4 it follows that the total time complexity of Algorithm 1 is $O(|G|^3 h_S + |G|^2|S|h_S)$. □

Lemma 5 *Let* α *denote the initial DTL-scenario and let* Θ *represent the transfer nodes in* α*. If Algorithm* 1 *invokes Algorithm* 2*, which in turn returns* $g \in V(G)$*, then there must exist some* $g' \in \Theta$ *such that* $g \geq_G g'$.

Proof By the definition of Algorithm 2, g is on a cycle in the graph F. Denote the arcs of F, except for those defined by step 4(c) in the construction, as white arcs and let the arcs defined by step 4(c) be black arcs. Note that, by construction, there can be no cycles that involve exclusively white arcs and thus the cycle C detected in line 3 of Algorithm 4 necessarily involves at least one black arc. Consider the subpath of C from g to the first black arc e on C. By construction of F, arc e was introduced by a transfer event $g' \in \Theta$ and that transfer event is reachable by white arcs from g and thus $g \geq_G g'$. □

Theorem 3 *Algorithm* 1 *introduces at most* kh_G *duplication events and* $kh_G h_S$ *loss events, where k is the number of transfer events in the initial MPR* α.

Proof We first prove that Algorithm 1 introduces at most kh_G duplication events. A new duplication event can be introduced at most once for each node $g \in I(G)$. By Lemma 5, we know that only a transfer node and its ancestors may be modified by our algorithm. Assume that, in the worst case, every transfer node and every ancestor of a transfer node is modified and that the sets of ancestors are disjoint for each transfer node. The number of ancestors for a transfer node is strictly less than the height of the tree h_G, and we have a total of k transfer nodes. Therefore, no more than kh_G duplication events are introduced.

We now prove that Algorithm 1 introduces at most $kh_G h_S$ loss events. We may introduce a series of loss events

each time we make g a new duplication node. Based on the definition of the number of losses, $Loss_\alpha(g) \leq h_S$. Therefore, no more than kh_Gh_S loss events are introduced. □

Results

To demonstrate the utility of our approach, we ran our algorithm on a dataset comprising 4848 parasite trees for a host tree comprising 100 (predominantly prokaryotic) species from the Tree of Life [5]. We used DTL values of 2, 3, and 1, respectively. In this dataset 17.4% of MPRs found by the U-MPR algorithm were temporally infeasible. In order to compare the solutions found by our algorithm to those found by Jane tool, we selected the first 100 of the 4848 parasite trees for comparison. (We do not compare performance to TreeMap or the ILP approaches since their exponential worst-case running times make them viable only for very small datasets).

In general, there may be multiple MPRs for a given DTL instance. Our implementation of the U-MPR algorithm can find all MPRs and we chose the first 10 and repaired all of them, if necessary, and reported the best score. Jane uses a genetic algorithm that maintains a population of T candidate datings for the host tree and runs for P iterations. We used $T = 30$ and $P = 30$ in these experiments.

In 27% of the cases, our algorithm found reconciliations with lower costs than those found by Jane: 16% that were temporally feasible and required no repair and 11% that were not temporally feasible and required repair. On average, in these cases, the repaired reconciliations had costs that were 5.5% lower than Jane's costs. In the remaining 73% of cases, Jane performed at least as well as our algorithm. However, given that Jane is a de facto standard in many cophylogenetic studies, it is notable that better solutions can be obtained by our relatively simple and fast algorithm for a non-trivial fraction of cases.

Our code is available in the Cheeta package (www.cs.hmc.edu/~hadas/cheeta) which runs both Jane and our repair algorithm and reports the best solution found (from Jane, from U-MPR with no repair necessary, or from U-MPR with our repair algorithm).

Conclusions

In this work we have described a new approach for finding temporally feasible reconciliations in the DTL model. This algorithm is efficient and, in a significant number of cases, finds solutions that are better than those found by the widely used Jane heuristic. In those cases, the results from our heuristic should be used instead of the results from Jane in order to draw more robust conclusions.

The "repair approach" described here has the desirable property that it begins with a reconciliation whose cost is a lower bound on that of a temporally feasible optimal solution. While we derive a bound on the increase in cost due to successive repair steps, this bound is quite large. Future work is needed to determine if this bound is tight or can be improved. Additionally, there may be other ways to repair temporally infeasible reconciliations that perform even better than the one described here. Finally, it is possible that this approach may lead to approximation algorithms or schemes for the DTL MPR problem.

Acknowledgements
The authors thank Yi-Chieh Wu and Mukul Bansal for many helpful discussions, advice, and assistance with the data used in this paper; Matthew Dohlen, Chen Pekker, and Gabriel Quiroz for assistance in preparing the *Cheeta* package; and Daniel Bork, Ricson Cheng, Jean Sung, and Jincheng Wang for valuable conversations and feedback.

Funding
This work and publication costs were funded by the U.S. National Science Foundation under Grant Number IIS-1419739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work was also funded by HHMI award #52007544.

Availability of data and materials
The datasets generated during and/or analysed during the current study are available in the Cheeta repository, www.cs.hmc.edu/~hadas/cheeta.

Authors' contributions
RLH conceived the repair framework. WM and DS developed, implemented, and analyzed Algorithms 1, 2, and 3. All authors read and approved the final manuscript.

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

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

About this supplement
This article has been published as part of *BMC Bioinformatics* Volume 18 Supplement 3, 2017. Selected articles from the 15th Asia Pacific Bioinformatics Conference (APBC 2017): bioinformatics. The full contents of the supplement are available online https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-18-supplement-3.

Author details
[1]Department of Computer Science, Harvey Mudd College, Claremont, California, USA. [2]Pomona College, Claremont, California, USA.

Published: 14 March 2017

References

1. 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.
2. Bansal MS, Alm EJ, Kellis M. Reconciliation revisited: handling multiple optima when reconciling with duplication, transfer, and loss In: Deng M, Jiang R, Sun F, Zhang X, editors. Research in Computational Molecular Biology 17th Annual International Conference, RECOMB 2013. Beijing: Springer; 2013. p. 1–13.
3. Charleston MA, Perkins SL. Traversing the tangle: algorithms and applications for cophylogenetic studies. J Biomed Inform. 2006;39(1):62–71.
4. Conow C, Fielder D, Ovadia Y, Libeskind-Hadas R. Jane: A new tool for cophylogeny reconstruction problem. Algoritm Mol Biol. 2010;5(16).

5. David LA, Alm EJ. Rapid evolutionary innovation during an archaean genetic expansion. Nature. 2011;469:93–6.
6. Doyon JP, Scornavacca C, Gorbunov KY, Szöllősi GJ, Ranwez V, Berry V. In: Tannier E, editor. An Efficient Algorithm for Gene/Species Trees Parsimonious Reconciliation with Losses, Duplications and Transfers. Berlin: Springer; 2010, pp. 93–108. doi:10.1007/978-3-642-16181-0_9.
7. Tofigh A, Hallett MT, Lagergren J. Simultaneous identification of duplications and lateral gene transfers. ACM Trans Comput Biology Bioinform. 2011;8(2):517–35.
8. Libeskind-Hadas R, Charleston M. On the computational complexity of the reticulate cophylogeny reconstruction problem. J Comput Biol. 2009;16(1):105–17.
9. Yodpinyanee A, Cousins B, Peebles J, Schramm T, Libeskind-Hadas R. Faster dynamic programming algorithms for the cophylogeny reconstruction problem. Technical Report CS-2011-1, Harvey Mudd College, Department of Computer Science; 2011.
10. Rutschmann F. Molecular dating of phylogenetic trees: a brief review of current methods that estimate divergence times. Divers Distrib. 2006;12(1):35–48. doi:10.1111/j.1366-9516.2006.00210.x.
11. Tofigh A. Using trees to capture reticulate evolution: Lateral gene transfers and cancer progression. PhD thesis, KTH Royal Institute of Technology; 2009. p. 152–172.
12. Ovadia Y, Fielder D, Conow C, Libeskind-Hadas R. The cophylogeny reconstruction problem is NP-complete. J Comput Biol. 2011;18(1):59–65.
13. Charleston M, Libeskind-Hadas R. Event-based cophylogenetic comparative analysis In: Garamszegi LZ, editor. Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology: Concepts and Practice. Berlin, Heidelberg: Springer; 2014. p. 465–80. doi:10.1007/978-3-662-43550-2_20.
14. van Iersel L, Scornavacca C, Kelk S. Exact reconciliations of undated trees. 2014. https://arxiv.org/abs/1410.7004.
15. Charleston M, Page RDM. TreeMap. https://sites.google.com/site/cophylogeny/treemap. Accessed 11 Nov 2016.
16. Wieseke N, Hartmann T, Bernt M, Middendorf M. Cophylogenetic reconciliation with ILP. IEEE/ACM Trans Comput Biol Bioinforma. 2015;12(6):1227–35. doi:10.1109/TCBB.2015.2430336.
17. Conow C, Fielder D, Ovadia Y, Ran LH. Jane: a new tool for the cophylogeny reconstruction problem. Algoritm Mol Biol. 2010;5(1):1.
18. Addario-Berry L, Hallett MT, Lagergren J. Towards identifying lateral gene transfer events. Pac Symp Biocomput. 2003;8:279–90.
19. Ma W, Smirnov D, Forman J, Schweickart A, Slocum C, Srinivasan S, Libeskind-Hadas R. DTL-RnB: Algorithms and tools for summarizing the space of DTL reconciliations. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2016;PP(99).