SUPPLEMENTARY MATERIAL

For the paper titled "Efficient Algorithms for the Reconciliation Problem with Gene Duplication, Horizontal Transfer, and Loss", by Mukul S. Bansal, Eric J. Alm, and Manolis Kellis

S.1 Proofs

Proof of Theorem 3.1

PROOF. We will show that Algorithm *U-Reconcile* correctly solves the U-MPR problem on G and S in $O(mn)$ time.

Correctness: It suffices to show that the value $c(g, s)$, for each $g \in V(G)$ and $s \in V(S)$, is computed correctly. Specifically, we will show that the values *in*(g, s), *inAlt*(g, s), *out*(g, s), $c_\Sigma(g, s)$, $c_\Delta(g, s)$, and $c_\Theta(g, s)$ are computed correctly for each g and s.

For each $g \in Le(G)$, consider the values *in*(g, s), *inAlt*(g, s), and $c(g, s)$, for any $s \in V(S)$. Observe that each of these values has been assigned correctly (in accordance with Definition 2.1) after the execution of the 'for' loop of lines 3 through 4. These values form the base case of our inductive argument.

Now let $g \in I(G)$ and $\{g', g''\} = Ch_G(g)$. Let us assume that the values *in*(g', x), *in*(g'', x), *inAlt*(g', x), *inAlt*(g'', x), $c(g', x)$ and $c(g'', x)$ are computed correctly for each $x \in V(S)$. Note that, given the values *inAlt*(g', x) and *inAlt*(g'', x), the values *out*(g', x) and *out*(g'', x) are also correctly computed, for each $x \in V(S)$, in the 'for' loop of steps 23 through 25. We will show that the values *in*(g, s), *inAlt*(g, s), and $c(g, s)$, for any $s \in V(S)$ are then computed correctly as well. Observe that the values $c_\Sigma(g, s)$, $c_\Delta(g, s)$, and $c_\Theta(g, s)$ are computed in accordance with Definitions 2.1 and 2.2 (in steps 9 through 11 if s is a leaf node, and in steps 17 through 19 otherwise), based on the values of *in*$(\cdot, \cdot)$, *out*$(\cdot, \cdot)$, and $c(\cdot, \cdot)$ computed previously. Thus, the value of $c(g, s)$ is computed correctly as well (steps 12 and 20), which, in turn, implies that the values of *in*(g, s) and *inAlt*(g, s) are also correctly assigned (steps 13, 21 and 14, 22 respectively).

Induction completes the proof.

Complexity: We analyze the complexity of Algorithm *U-Reconcile* step-by-step. The 'for' loops from steps 1 through 4 require $O(mn)$ time. Steps 7 through 22 each require $O(1)$ time, and are each executed $O(mn)$ times (through the 'for' loops at lines 5 and 6), yielding a total time complexity of $O(mn)$ for these steps. Similarly, steps 24 though 25 each require $O(1)$ per execution and are executed a total of $O(mn)$ time (through the 'for' loops at lines 5 and 23). Finally, step 26 requires $O(n)$ time. The total time complexity of Algorithm *U-Reconcile* is thus $O(mn)$. □

Proof of Theorem 3.2

PROOF. We will show that Algorithm *D-Reconcile* solves the D-MPR problem in $O(mn \log n)$ time.

Correctness: Since the values *in*$(\cdot, \cdot)$, $c_\Sigma(\cdot, \cdot)$, $c_\Delta(\cdot, \cdot)$, and $c(\cdot, \cdot)$ are assigned in exactly the same way as in Algorithm *U-Reconcile*, it suffices to show that each $c_\Theta(\cdot, \cdot)$ is computed correctly (in part (a) of step 19). Consider the 'for' loop in steps 8 through 16. The purpose of that loop is simply to assign, for each $x \in Ch(g)$ and each time zone i, the edges *Best*(x, i) and *secondBest*(x, i). It is not hard to see that these edges are assigned correctly; the purpose of the slightly round-about approach employed by us to achieve this is only to maintain the efficiency of the algorithm. These computed *Best*(x, i) and *secondBest*(x, i) edges are then added to a dynamic range minimum query data structure Φ_x, indexed by i.

Now, to show that part (a) of step 19 computes the value of $c_\Theta(g, s)$ correctly, it suffices to show that the edges $(pa(u), u)$ and $(pa(v), v)$, assigned in step 18, are such that (i) u is that node from $V(S)$ for which *in*(g', u) has minimum value, subject to the constraints that $u \neq s$ and $(pa(u), u)$ has at least one time zone in common with $(pa(s), s)$, and (ii) v is that node from $V(S)$ for which *in*(g'', v) has minimum value, subject to the constraints that $v \neq s$ and $(pa(v), v)$ has at least one time zone in common with $(pa(s), s)$. Observe that one of the edges *Best*(g', i) and *secondBest*(g', i), for $t(pa(s)) \leq i \leq t(s) - 1$, must be such an edge $(pa(u), u)$, and one of the edges *Best*(g'', i) and *secondBest*(g'', i), for $t(pa(s)) \leq i \leq t(s) - 1$, must be such an edge $(pa(v), v)$. In other words, such $(pa(u), u)$ and $pa(v), v)$ can be found among the edges stored in the data structures $\Phi_{g'}$ and $\Phi_{g''}$ between (and including) the indexes $t(pa(s))$ and $t(s) - 1$. In light of this observation, it is easy to verify that a call to Procedure *bestReceiver* with parameters (g, g', s) returns exactly such an edge $(pa(u), u)$, and when the parameters are (g, g'', s) exactly such an edge $(pa(v), v)$. Thus, the value of each $c_\Theta(\cdot, \cdot)$ is computed correctly by Algorithm *D-reconcile*.

Complexity: The structure of Algorithm *D-Reconcile* is the same as the structure of Algorithm *U-Reconcile* except for the preprocessing step, the addition of the 'for' loop in steps 8 through 16, and the use of the procedure *bestReceiver* in step 18. For the preprocessing step, computing the *end*(i) and *begin*(i) sets, for each time zone i, can be accomplished in $O(n)$ time by a single post order traversal of S.

We now analyze the complexity of the 'for' loop from steps 8 through 16.

We now analyze the complexity of the 'for' loop from steps 8 through 16. Creating an empty binomial heap in Step 9 requires $O(1)$ time. Steps 10 through 11 require $O(n \log n)$ time, since each of these steps requires at most $O(n)$ insertions or queries on the binomial heap. In the 'for' loop of steps 12 through 14, we update the binomial heap by deleting and inserting at most a total of $O(n)$ edges (since each edge of S is inserted or deleted at most once), and query the binomial heap a total of $O(k)$ times (which is $O(n)$) to set the *Best* values and perform $O(k)$ deletions, reinsertions, and queries to set the *secondBest* values. Thus, the total time complexity of this 'for' loop is $O(n \log n)$. Step 15 involves adding $O(k)$ values to a dynamic range minimum query data structure, which requires a total of $O(k \log k)$, i.e., $O(n \log n)$ time. Since the 'for' loop of steps 8 through 16 only executes steps 10 through 15 two times, the total time complexity of that 'for' loop is $O(n \log n)$. Now, since this 'for' loop itself is executed a total of $O(m)$ times, the total contribution of steps 8 through 16 to the complexity of Algorithm *D-Reconcile* is $O(mn \log n)$

Finally, we analyze the time complexity of step 16. Observe that the total contribution of step 18 to the time complexity of Algorithm *D-Reconcile* is just the total time spent by the algorithm inside procedure *bestReceiver*. Observe also that Procedure *bestReceiver* is called exactly once each for each triplet (g, x, s), for any $g \in I(G)$, $x \in Ch(g)$, and $s \in V(S) \setminus rt(S)$. For any fixed g, x, and s, the time complexity of Procedure *bestReceiver* depends on the number of times that the dynamic range minimum query data structure Φ_x is queried: And Φ_x is only queried again if the edge returned by the current query is the same as the edge $(pa(s), s)$. Note, however,

that for each index in $\Phi(x)$, there is only one edge from $V(S)$ that could cause Φ_x to be queried again. Thus, for any fixed g and x, as s varies, there are at most $O(k)$ repeated queries overall (since there are $O(k)$ indexes in Φ_x. This implies that, for any fixed g and x, and over all $s \in V(S) \setminus rt(S)$, the total time spent in Procedure *bestReceiver* is simply the time required for $O(n+k)$ queries and $O(n+k)$ insertions and deletions on Φ_x, which is $O(n \log n)$. Since there are $O(m)$ ways to choose g, and for any given g only two ways to choose x, the total time spent spent inside Procedure *bestReceiver* is $O(mn \log n)$.

Thus, the total time complexity of Algorithm *D-Reconcile* is $O(mn \log n)$. □

Proof of Theorem 3.3

PROOF. We will show that Algorithm *Reconcile* correctly solves the MPR problem with distance-dependent transfer costs in $O(mn^2)$ time.

Correctness: Observe that, by updating Steps 11 and 19 of Algorithm *U-Reconcile*, Algorithm *Reconcile* correctly assigns the value $c_\Theta(g,s)$, for any $g \in I(G)$ and $s \in V(S)$, with respect to the given time constraints and distance-dependent transfer cost function. The remainder of the argument is analogous to that of the correctness of Algorithm *U-Reconcile* in the proof of Theorem 3.1.

Complexity: Consider the two modifications to Algorithm *U-Reconcile* and how they affect its complexity: Modification (i) has no impact on the time complexity. And, since each $P_\Theta(s,x)$ value can be computed in constant time, modification (ii) increases the time complexity of computing each $c_\Theta(\cdot,\cdot)$ only to $O(n)$. This increases the overall time complexity of the algorithm to $O(mn^2)$. □

S.2 Handling more complex transfer scenarios

Transfers where the host copy is lost. As we illustrate in Figure 2, the existing reconciliation model is unable to correctly handle cases where there is a transfer from a species that then loses its copy of that gene. This is simply because there is no node in the gene tree that can represent that transfer. To the best of our knowledge, the only existing approach that can detect such transfers is the one from Doyon *et al.* (2010), which solves the tcDTL-reconciliation problem on fully dated trees in $O(mn^2)$ time. As we show now, each of our Algorithms *U-Reconcile*, *D-Reconcile*, and *Reconcile* can be extended to detect such transfers without any increase in their time complexities. The idea is as follows: We explicitly create a fifth type of event, denoted by $\mathbb{TL}$, (besides speciations, transfers, duplications, and losses) that represents the case where there is a transfer with corresponding host-copy loss. The cost of a $\mathbb{TL}$ event is denoted by $P_{\mathbb{TL}}$. We add a dummy node on each edge in the gene tree. These dummy nodes have one parent and one child and represent potential $\mathbb{TL}$ events. Note that each internal node of G has two dummy nodes as its children. Given any dummy node g and any $s \in V(S)$, we define $c_{\mathbb{TL}}(g,s)$ to be the cost of an optimal reconciliation of $G(g)$ with S such that g is mapped to s and $g \in \mathbb{TL}$. Consider Algorithm *U-Reconcile*; the algorithm proceeds as usual at the regular (non-dummy) nodes of the gene tree. When at a dummy node g, the algorithm does the following: The dummy node g inherits the values of $c_\Sigma(g,\cdot)$, $c_\Delta(g,\cdot)$, $c_\Theta(g,\cdot)$ from its unique child (which must be a regular node), and in addition computes the value $c_{\mathbb{TL}}(g,s)$ to be $P_{\mathbb{TL}} + out(g',s)$, where g' denotes the unique child of g. At each dummy node g, the value of $c_(g,s)$ is then defined to be $\min\{c_\Sigma(g,s), c_\Delta(g,s), c_\Theta(g,s), c_{\mathbb{TL}}(g,s)\}$. Note that the cost of a $\mathbb{TL}$ event should be set to be at least $P_\Theta + P_{loss}$ to prevent overuse of $\mathbb{TL}$ events. These simple changes enable the algorithm to detect and handle $\mathbb{TL}$ events. It is not hard to show that by employing the dummy-node approach described above, each of the Algorithms *U-Reconcile*, *D-Reconcile*, and *Reconcile* can be modified to detect and account for $\mathbb{TL}$ events without any increase in their time complexities. Further technical details are omitted for brevity.

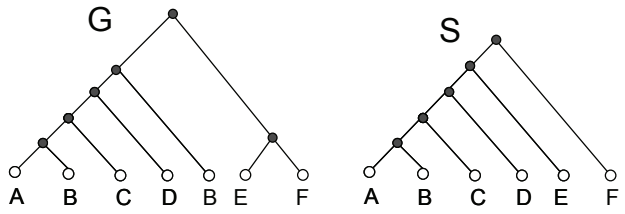

Fig. 2. Transfers where host copy is lost. Consider the gene tree G showing the evolutionary history of some gene in the species in S. G can be obtained from S by two transfers and one loss: Namely, a transfer from species E to species B, and then a transfer from species F to species E that replaces the copy of that gene in E. This evolutionary history is, however, undetectable under the current reconciliation model, and the current MPR requires three transfers and a loss (assuming all events have the same cost)

Transfers from unsampled or extinct species lineages. Incomplete taxon sampling can confound the accuracy of DTL-reconciliation due to transfers from the unsampled lineages to the sampled ones. Even with complete taxon sampling, the species, both extant and ancestral, that can be seen on a species tree represent only those species that have surviving descendant species. This is only a fraction of the total number of species that have ever existed. It is likely that now-extinct species lineages would have participated in horizontal gene transfer events with species that do have surviving descendant species. In conventional HGT detection methods, transfers from unsampled or extinct lineages, sometimes called phantom transfers, can manifest themselves as transfers going forward in time or as transfers going from an ancestor to a descendant. Such phantom transfers can be detected and handled by augmenting the species tree with additional edges (representing unsampled or extinct lineages) and then employing a strategy similar to the one used above for handling transfers where the host copy is lost.