Supplement: An Integrated Reconciliation Framework for Domain, Gene, and Species Level Evolution

– Lei Li and Mukul S. Bansal

S.1 Computing gene and domain losses

Let T be a tree. Given $x, y \in V(T)$, $x \rightarrow_T y$ denotes the unique path from x to y in T. We denote by $dist_T(x, y)$ the number of edges on the path $x \rightarrow_T y$; note that if $x = y$ then $dist_T(x, y) = 0$.

Definition S1 (Gene-Losses). *Given a DGS-reconciliation $\alpha = \langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta, \Xi, \tau \rangle$ for $D, \mathcal{G}$ and S, let $g \in I(\mathcal{G})$ and $\{g', g''\} = Ch(d)$. The number of* gene-losses *$GeneLoss_\alpha(g)$ at node g, is defined to be:*

- $|dist_S(\mathcal{M}^{\mathcal{G}}(g), \mathcal{M}^{\mathcal{G}}(g')) - 1| + |dist_S(\mathcal{M}^{\mathcal{G}}(g), \mathcal{M}^{\mathcal{G}}(g'')) - 1|$, *if* $g \in \Sigma^{\mathcal{G}}$.
- $dist_S(\mathcal{M}^{\mathcal{G}}(g), \mathcal{M}^{\mathcal{G}}(g')) + dist_S(\mathcal{M}^{\mathcal{G}}(g), \mathcal{M}^{\mathcal{G}}(g''))$, *if* $g \in \Delta^{\mathcal{G}}$.

The total number of gene-losses in the DGS-reconciliation α is defined to be $GeneLoss_\alpha = \sum_{g \in I(\mathcal{G})} GeneLoss_\alpha(g)$.

Definition S2 (Domain-Losses). *Given a DGS-reconciliation $\alpha = \langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta, \Xi, \tau \rangle$ for $D, \mathcal{G}$ and S, let $d \in I(D)$ and $\{d', d''\} = Ch(d)$. The number of* domain-losses *$DomainLoss_\alpha(d)$ at node d, is defined to be:*

- $|dist_{\mathcal{G}}(\mathcal{M}^D(d), \mathcal{M}^D(d')) - 1| + |dist_{\mathcal{G}}(\mathcal{M}^D(d), \mathcal{M}^D(d'')) - 1|$, *if* $d \in \Sigma^D$.
- $dist_{\mathcal{G}}(\mathcal{M}^D(d), \mathcal{M}^D(d')) + dist_{\mathcal{G}}(\mathcal{M}^D(d), \mathcal{M}^D(d''))$, *if* $d \in \Delta^D$.
- $dist_{\mathcal{G}}(\mathcal{M}^D(d), \mathcal{M}^D(d'')) + dist_{\mathcal{G}}(\tau(d), \mathcal{M}^D(d'))$ *if* $(d, d') \in \Xi$.

The total number of domain-losses in the DGS-reconciliation α is defined to be $DomainLoss_\alpha = \sum_{d \in I(D)} DomainLoss_\alpha(d)$.

S.2 Solving the extended-DTL problem

The algorithm for solving the extended-DTL problem is based on the classical dynamic programming algorithm for computing an optimal DTL reconciliation between a gene tree and species tree [1]. In the current setting, we are interested in computing an extended-DTL reconciliation between a domain tree D and one or more gene trees, $\mathcal{G}$, given a fixed mapping $\mathcal{M}^{\mathcal{G}}$ between $\mathcal{G}$ and the species tree S. Following [1], we first introduce some basic notations and definitions.

Given any $d \in I(D)$ and $g \in V(G)$, let $c_\Sigma(d, g, \mathcal{M}^{\mathcal{G}})$ denote the cost of an optimal extended-DTL reconciliation of $D(d)$ with $\mathcal{G}$ such that d maps to g, $d \in \Sigma^D$ and all invoked domain-transfer events respect the species constraint on domain transfers imposed the mapping $\mathcal{M}^{\mathcal{G}}$. The terms $c_\Delta(d, g, \mathcal{M}^{\mathcal{G}})$, $c_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}})$ and $c_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}})$ are defined similarly for $d \in \Delta^D$, and $d \in \Theta 1^D$, and $d \in \Theta 2^D$, respectively. Recall that, $\Theta 1^D$ and $\Theta 2^D$ represent domain-transfer events that remain within the same gene family and those that cross gene family boundaries, respectively. Given any $d \in V(D)$ and $g \in V(\mathcal{G})$, define $c(d, g, \mathcal{M}^{\mathcal{G}})$ to be the cost of an optimal reconciliation of $D(d)$ with $\mathcal{G}$ such that d maps to g and all invoked domain-transfer events respect the species constraint on domain transfers imposed by the mapping $\mathcal{M}^{\mathcal{G}}$. Note that, for $d \in I(D)$, $c(d, g, \mathcal{M}^{\mathcal{G}}) = \min\{c_\Sigma(d, g, \mathcal{M}^{\mathcal{G}}), c_\Delta(d, g, \mathcal{M}^{\mathcal{G}}), c_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}}), c_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}})\}$. The extended-DTL algorithm performs a nested post-order traversal of the domain tree and the gene trees, computing the value $c(d, g, \mathcal{M}^{\mathcal{G}})$ for each $d \in I(D)$ and $g \in V(\mathcal{G})$ based on previously computed values of $c(d', \cdot, \mathcal{M}^{\mathcal{G}})$ and $c(d'', \cdot, \mathcal{M}^{\mathcal{G}})$, where d' and d'' denote the two children of d. The dynamic programming table is initialized

based on the given leaf mappings from D to $\mathcal{G}$. Once all the $c(\cdot,\cdot,\mathcal{M}^{\mathcal{G}})$ values are computed, the minimum extended-DTL reconciliation cost of D and $\mathcal{G}$, given $\mathcal{M}^{\mathcal{G}}$, is simply $\min_{g\in V(\mathcal{G})} c(\mathit{rt}(D), g, \mathcal{M}^{\mathcal{G}})$.

To help compute $c_{\Sigma}(d,g,\mathcal{M}^{\mathcal{G}})$, $c_{\Delta}(d,g,\mathcal{M}^{\mathcal{G}})$, $c_{\Theta 1}(d,g,\mathcal{M}^{\mathcal{G}})$ and $c_{\Theta 2}(d,g,\mathcal{M}^{\mathcal{G}})$, we also define, for each $d \in V(D)$ and $g \in V(\mathcal{G})$,

$$\begin{aligned}
\mathit{in}(d,g,\mathcal{M}^{\mathcal{G}}) &= \min_{x\in V(G(g))} \{P^D_{loss}\cdot \mathit{dist}_G(g,x) + c(d,x,\mathcal{M}^{\mathcal{G}})\},\\
\mathit{out\text{-}1}(d,g,\mathcal{M}^{\mathcal{G}}) &= \min_{x\in V(G) \text{ where } x \text{ is incomparable to } g \text{ and } \mathcal{M}^{\mathcal{G}}(g)=\mathcal{M}^{\mathcal{G}}(x)} c(d,x,\mathcal{M}^{\mathcal{G}}), \text{ and}\\
\mathit{out\text{-}2}(d,g,\mathcal{M}^{\mathcal{G}}) &= \min_{x\in V(\mathcal{G})\setminus V(G) \text{ where } \mathcal{M}^{\mathcal{G}}(g)=\mathcal{M}^{\mathcal{G}}(x)} c(d,x,\mathcal{M}^{\mathcal{G}}),
\end{aligned}$$

where G represents the specific gene tree in which node g appears, i.e., $g \in V(G)$.

The complete algorithm for solving the extended-DTL problem can now be formally described as follows:

Algorithm *extended-DTL*$(D,\mathcal{G},S,\mathcal{L}^D,\mathcal{M}^{\mathcal{G}})$

for each $d \in V(D)$ and $g \in V(G)$ **do**
  Initialize $c(d,g,\mathcal{M}^{\mathcal{G}})$, $c_{\Sigma}(d,g,\mathcal{M}^{\mathcal{G}})$, $c_{\Delta}(d,g,\mathcal{M}^{\mathcal{G}})$, $c_{\Theta 1}(d,g,\mathcal{M}^{\mathcal{G}})$ and $c_{\Theta 2}(d,g,\mathcal{M}^{\mathcal{G}})$ to ∞.
end for
for each $d \in \mathit{Le}(D)$ **do**
  Initialize $c(d,\mathcal{L}^D(d),\mathcal{M}^{\mathcal{G}})$ to 0.
end for
for each $d \in I(D)$ in post-order **do**
  for each $G \in \mathcal{G}$ **do**
    for each $g \in V(G)$ in post-order **do**
      Let $\{d',d''\} = \mathit{Ch}_D(d)$.
      if $g \in \mathit{Le}(G)$ **then**
        $c_{\Sigma}(d,g,\mathcal{M}^{\mathcal{G}}) = \infty$.
        $c_{\Delta}(d,g,\mathcal{M}^{\mathcal{G}}) = P^D_{\Delta} + c(d',g,\mathcal{M}^{\mathcal{G}}) + c(d'',g,\mathcal{M}^{\mathcal{G}})$.
        If $g \neq \mathit{rt}(G)$, then $c_{\Theta 1}(d,g,\mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 1} + \min\{\mathit{in}(d',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d'',g,\mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d',g,\mathcal{M}^{\mathcal{G}})\}$.
        If $|\mathcal{G}| > 1$, then $c_{\Theta 2}(d,g,\mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 2} + \min\{\mathit{in}(d',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d'',g,\mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d',g,\mathcal{M}^{\mathcal{G}})\}$.
        $c(d,g,\mathcal{M}^{\mathcal{G}}) = \min\{c_{\Sigma}(d,g,\mathcal{M}^{\mathcal{G}}), c_{\Delta}(d,g,\mathcal{M}^{\mathcal{G}}), c_{\Theta 1}(d,g,\mathcal{M}^{\mathcal{G}}), c_{\Theta 2}(d,g,\mathcal{M}^{\mathcal{G}})\}$.
      else
        Let $\{g',g''\} = \mathit{Ch}_G(g)$.
        $c_{\Sigma}(d,g,\mathcal{M}^{\mathcal{G}}) = \min\{\mathit{in}(d',g',\mathcal{M}^{\mathcal{G}}) + \mathit{in}(d'',g'',\mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'',g',\mathcal{M}^{\mathcal{G}}) + \mathit{in}(d',g'',\mathcal{M}^{\mathcal{G}})\}$.
        $c_{\Delta}(g,s) = P^D_{\Delta} + \min\{\mathit{in}(d',g,\mathcal{M}^{\mathcal{G}}) + \mathit{in}(d'',g,\mathcal{M}^{\mathcal{G}})\}$.
        If $s \neq \mathit{rt}(S)$, then $c_{\Theta 1}(d,g,\mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 1} + \min\{\mathit{in}(d',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d'',g,\mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d',g,\mathcal{M}^{\mathcal{G}})\}$.
        If $|\mathcal{G}| > 1$, then $c_{\Theta 2}(d,g,\mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 2} + \min\{\mathit{in}(d',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d'',g,\mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'',g,\mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d',g,\mathcal{M}^{\mathcal{G}})\}$.
        $c(d,g,\mathcal{M}^{\mathcal{G}}) = \min\{c_{\Sigma}(d,g,\mathcal{M}^{\mathcal{G}}), c_{\Delta}(d,g,\mathcal{M}^{\mathcal{G}}), c_{\Theta 1}(d,g,\mathcal{M}^{\mathcal{G}}), c_{\Theta 2}(d,g,\mathcal{M}^{\mathcal{G}})\}$.
      end if
    end for
  end for

27: **end for**
28: Return $\min_{g \in V(\mathcal{G})} c(\mathit{rt}(D), g, \mathcal{M}^{\mathcal{G}})$.

Note that the values $\mathit{in}(d, g, \mathcal{M}^{\mathcal{G}})$, $\mathit{out\text{-}1}(d, g, \mathcal{M}^{\mathcal{G}})$ and $\mathit{out\text{-}2}(d, g, \mathcal{M}^{\mathcal{G}})$, for each $d \in D$ and $g \in V(\mathcal{G})$, can be easily computed as needed in the above algorithm. The correctness of Algorithm *extended-DTL* follows easily from the correctness of the algorithm for computing optimal DTL reconciliations [1]. A formal proof of correctness is therefore excluded and we refer the reader to [1].

S.3 Detailed description of the dynamic programming heuristic

Our heuristic for the ODGS problem builds upon the *extended-DTL* algorithm described above. As explained in the main text, the main idea is to allow for the modification of the mapping $\mathcal{M}^{\mathcal{G}}$ to enable domain transfer events as needed, and accounting for the additional gene-species reconciliation cost incurred due to the modification of $\mathcal{M}^{\mathcal{G}}$.

The initial gene-species mapping $\mathcal{M}^{\mathcal{G}}$ is defined to be the LCA mapping (which is the unique mapping that minimizes the duplication-loss reconciliation cost between $\mathcal{G}$ and S). Specifically, for any $G \in \mathcal{G}$ and $g \in I(G)$, we define $\mathcal{M}^{\mathcal{G}}(g) = \mathit{lca}_S(\mathcal{L}^{\mathcal{G}}(\mathit{Le}(G(g))))$. Let γ denote the reconciliation cost of $\mathcal{G}$ and S given by this mapping. Consider a domain-transfer event whose donor and recipient are nodes g_1 and g_2 from $V(\mathcal{G})$ (such that g_1 and g_2 do not have an ancestor-descendant relationship, as required for domain-transfer events). To enable this domain-transfer event, the mapping $\mathcal{M}^{\mathcal{G}}$ may need to be modified so that g_1 and g_2 both map to the same species node on the species tree. Let $\mathcal{M}^{\mathcal{G}}_{g_1,g_2})$ denote this modified mapping, such that $\mathcal{M}^{\mathcal{G}}_{g_1,g_2})$ has the least reconciliation cost for $\mathcal{G}$ and S among all such mappings. Let γ_{g_1,g_2} denote the reconciliation cost of $\mathcal{G}$ and S under the modified mapping $\mathcal{M}^{\mathcal{G}}_{g_1,g_2})$. The additional gene-species cost incurred in allowing the domain-transfer event between g_1 and g_2 is defined to be

$$\mathit{Add}(g_1, g_2) = \gamma_{g_1,g_2} - \gamma.$$

Based on the given set of gene trees $\mathcal{G}$ and species tree S, the value of $\mathit{Add}(g_1, g_2)$ can be easily precomputed for each $g_1, g_2 \in V(\mathcal{G})$ in accordance with the standard gene duplication/loss reconciliation model. Our description of the heuristic algorithm below assumes that all $\mathit{Add}(\cdot, \cdot)$ values have been precomputed. Note that, for some choices of g_1 and g_2, the mapping $\mathcal{M}^{\mathcal{G}}_{g_1,g_2})$ may not exist (e.g., when g_1 and g_2 are leaf nodes mapping to different species). In such cases, the value $\mathit{Add}(g_1, g_2)$ is ∞.

Observe that the heuristic algorithm does not compute the values $c(d, g, \mathcal{M}^{\mathcal{G}})$, $c_{\Sigma}(d, g, \mathcal{M}^{\mathcal{G}})$, $c_{\Delta}(d, g, \mathcal{M}^{\mathcal{G}})$, $c_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}})$, and $c_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}})$, as defined in the previous section. Rather, the heuristic computes upper-bounds for these values, and we denote these upper bounds by $c'(d, g, \mathcal{M}^{\mathcal{G}})$, $c'_{\Sigma}(d, g, \mathcal{M}^{\mathcal{G}})$, $c'_{\Delta}(d, g, \mathcal{M}^{\mathcal{G}})$, $c'_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}})$, and $c'_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}})$.

For the heuristic, we modify the definitions of $out - 1$ and $out - 2$ to account for the additional costs of modifying $\mathcal{M}^{\mathcal{G}}$ to enable the considered transfer events. The new definitions for $out - 1$ and $out - 2$ are as follows:

$$\mathit{out\text{-}1}(d, g, \mathcal{M}^{\mathcal{G}}) = \min_{x \in V(G) \text{ where } x \text{ is incomparable to } g} \left(c(d, x, \mathcal{M}^{\mathcal{G}}) + \mathit{Add}(g, x)\right), \text{ and}$$

$$\mathit{out\text{-}2}(d, g, \mathcal{M}^{\mathcal{G}}) = \min_{x \in V(\mathcal{G}) \setminus V(G)} \left(c(d, x, \mathcal{M}^{\mathcal{G}}) + \mathit{Add}(g, x)\right)$$

where G represents the specific gene tree in which node g appears, i.e., $g \in V(G)$. The definition of $\mathit{in}(d, g, \mathcal{M}^{\mathcal{G}})$ remains unchanged.

The heuristic algorithm can now be written as follows:

Algorithm *Dynamic-Programming-Heuristic*$(D, \mathcal{G}, S, \mathcal{L}^D, \mathcal{L}^{\mathcal{G}})$

Initialize $\mathcal{M}^{\mathcal{G}}$ to be the LCA mapping from $\mathcal{G}$ to S.
for each $d \in V(D)$ and $g \in V(G)$ **do**
  Initialize $c'(d, g, \mathcal{M}^{\mathcal{G}})$, $c'_{\Sigma}(d, g, \mathcal{M}^{\mathcal{G}})$, $c'_{\Delta}(d, g, \mathcal{M}^{\mathcal{G}})$, $c'_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}})$ and $c'_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}})$ to ∞.
end for
for each $d \in \mathit{Le}(D)$ **do**
  Initialize $c'(d, \mathcal{L}^D(d), \mathcal{M}^{\mathcal{G}})$ to 0.
end for
for each $d \in I(D)$ in post-order **do**
  for each $G \in \mathcal{G}$ **do**
    for each $g \in V(G)$ in post-order **do**
      Let $\{d', d''\} = \mathit{Ch}_D(d)$.
      if $g \in \mathit{Le}(G)$ **then**
        $c'_{\Sigma}(d, g, \mathcal{M}^{\mathcal{G}}) = \infty$.
        $c'_{\Delta}(d, g, \mathcal{M}^{\mathcal{G}}) = P^D_{\Delta} + c'(d', g, \mathcal{M}^{\mathcal{G}}) + c'(d'', g, \mathcal{M}^{\mathcal{G}})$.
        If $g \neq \mathit{rt}(G)$, then $c'_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 1} + \min\{\mathit{in}(d', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d'', g, \mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d', g, \mathcal{M}^{\mathcal{G}})\}$.
        If $|\mathcal{G}| > 1$, then $c'_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 2} + \min\{\mathit{in}(d', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d'', g, \mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d', g, \mathcal{M}^{\mathcal{G}})\}$.
        $c'(d, g, \mathcal{M}^{\mathcal{G}}) = \min\{c'_{\Sigma}(d, g, \mathcal{M}^{\mathcal{G}}), c'_{\Delta}(d, g, \mathcal{M}^{\mathcal{G}}), c'_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}}), c'_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}})\}$.
      else
        Let $\{g', g''\} = \mathit{Ch}_G(g)$.
        $c'_{\Sigma}(d, g, \mathcal{M}^{\mathcal{G}}) = \min\{\mathit{in}(d', g', \mathcal{M}^{\mathcal{G}}) + \mathit{in}(d'', g'', \mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'', g', \mathcal{M}^{\mathcal{G}}) + \mathit{in}(d', g'', \mathcal{M}^{\mathcal{G}})\}$.
        $c'_{\Delta}(g, s) = P^D_{\Delta} + \min\{\mathit{in}(d', g, \mathcal{M}^{\mathcal{G}}) + \mathit{in}(d'', g, \mathcal{M}^{\mathcal{G}})\}$.
        If $s \neq \mathit{rt}(S)$, then $c'_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 1} + \min\{\mathit{in}(d', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d'', g, \mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}1}(d', g, \mathcal{M}^{\mathcal{G}})\}$.
        If $|\mathcal{G}| > 1$, then $c'_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}}) = P^D_{\Theta 2} + \min\{\mathit{in}(d', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d'', g, \mathcal{M}^{\mathcal{G}}),\ \mathit{in}(d'', g, \mathcal{M}^{\mathcal{G}}) + \mathit{out\text{-}2}(d', g, \mathcal{M}^{\mathcal{G}})\}$.
        $c'(d, g, \mathcal{M}^{\mathcal{G}}) = \min\{c'_{\Sigma}(d, g, \mathcal{M}^{\mathcal{G}}), c'_{\Delta}(d, g, \mathcal{M}^{\mathcal{G}}), c'_{\Theta 1}(d, g, \mathcal{M}^{\mathcal{G}}), c'_{\Theta 2}(d, g, \mathcal{M}^{\mathcal{G}})\}$.
      end if
    end for
  end for
end for
Return $\min_{g \in V(\mathcal{G})} c'(\mathit{rt}(D), g, \mathcal{M}^{\mathcal{G}})$.

Post-processing. The pseudocode above shows how to compute the final DGS-reconciliation cost. An actual DGS reconciliation can easily be computed by back-tracking through the dynamic programming table, as in any dynamic programming algorithm. For our algorithm, backtracking gives a final domain-gene mapping $\mathcal{M}^D$, along with the partition of $I(D)$ into Σ^D, Δ^D, and Θ, but it does not directly give the final (modified) gene-species mapping $\mathcal{M}^{\mathcal{G}}$. The final gene-species mapping can be efficiently computed by considering the donors and recipients for each domain-transfer event in Θ, and modifying the initial LCA mapping between $\mathcal{G}$ and S to satisfy all the species mapping constraints required for all donor-receiver pairs for the domain-transfers in Θ. Specifically, consider any node $g \in V(\mathcal{G})$ that appears as donor or recipient for at least one of the domain-transfers. Let $\{g_1, g_2, \ldots, g_z\}$ be the set of nodes from $V(\mathcal{G})$ that are paired as a donor or receiver with g. Then, the final mapping for g is assigned to be $\mathit{lca}_S(\mathcal{L}^{\mathcal{G}}(\mathit{Le}(\mathcal{G}(g))) \cup \mathcal{L}^{\mathcal{G}}(\mathit{Le}(\mathcal{G}(g_1))) \cup \ldots \cup \mathcal{L}^{\mathcal{G}}(\mathit{Le}(\mathcal{G}(g_z))))$.

Time complexity of the heuristic. For any fixed $g_1, g_2 \in V(\mathcal{G})$, the value of $Add(g_1, g_2)$ can be computed in $O(|\,Le(\mathcal{G})| + |\,Le(S)|)$ time (using linear-time LCA-mapping computation [2] and a constant number of traversals of $\mathcal{G}$ and S. Thus, all $Add(g_1, g_2)$ values can be precomputed in $O(|\,Le(\mathcal{G})|^3 + |\,Le(S)| \cdot |\,Le(\mathcal{G})|^2)$ time. Inside Algorithm *Dynamic-Programming-Heuristic*, the initial mapping $\mathcal{M}^{\mathcal{G}}$ can be computed in $O(|\,Le(\mathcal{G})| + |\,Le(S)|)$ time [2]. Computation of each $in(\cdot,\cdot,\cdot)$, $out\text{-}1(\cdot,\cdot,\cdot)$ or $out\text{-}2(\cdot,\cdot,\cdot)$ value requires $O(|\,Le(\mathcal{G})|)$ time by brute force. Steps 11 through 24 therefore require $O(|\,Le(\mathcal{G})|)$ time overall. Steps 11 through 24 are executed a total of $O(|\,Le(D)| \cdot |\,Le(\mathcal{G})|)$ times through the "for" loops in Steps 8, 9 and 10. Thus, the total time spent in the "for" loop of Step 8 is $O(|\,Le(D)| \cdot |\,Le(\mathcal{G})|^2)$. Step 29 requires only $O(|\,Le(\mathcal{G})|)$ time. The post-processing step to finalize the gene-species mapping requires an additional $O(|\,Le(\mathcal{G})|^2)$ time. The total time complexity of the dynamic programming heuristic is thus $O(|\,Le(\mathcal{G})|^2 \cdot (|\,Le(\mathcal{G})| + |\,Le(S)| + |\,Le(D)|))$.

Figure S1 shows the running time of our implementation of this heuristic in practice on the trees in our dataset.

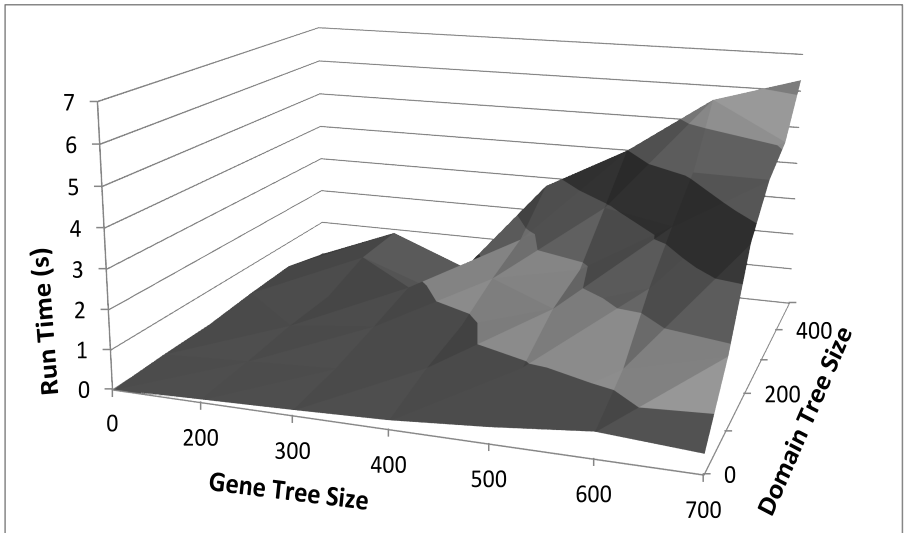

Fig. S1. This figure shows the running time (in seconds) of our heuristic algorithm on our data set of 3761 domain trees and associated gene families from 12 fly species, plotted against the number of nodes (leaves plus internal nodes) in each domain tree and its associated gene trees.

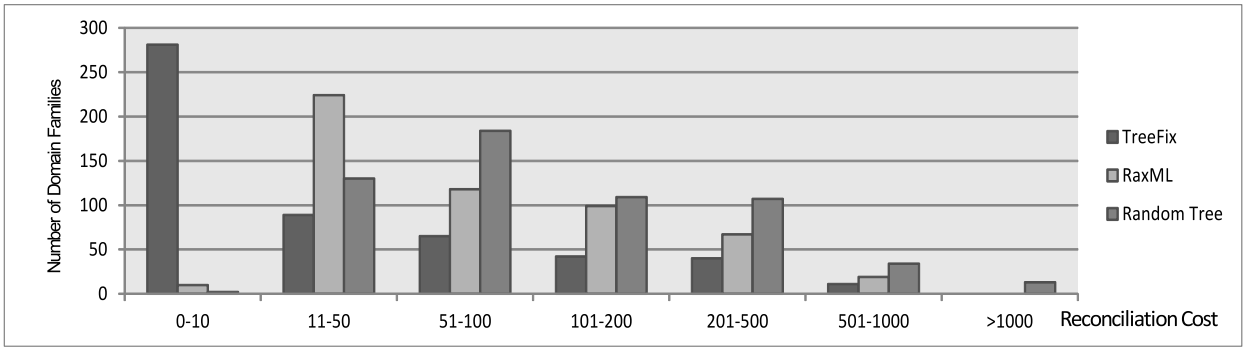

Fig. S2. Comparison of DGS reconciliation costs: This plot shows the distribution of DGS reconciliation costs obtained by our heuristic on the RAxML domain trees, TreeFix domain trees, and randomized domain trees for 650 randomly chosen domain families. The randomized domain trees were generated using the corresponding RAxML domain tree topologies and randomly shuffling their leaf labels. The reconciliation costs for the randomized domain trees are much higher (1.92 times higher on average) than for the RAxML trees, showing that even the RAxML domain trees are at least moderately accurate.

References

1. M. S. Bansal, E. J. Alm, and M. Kellis. Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. *Bioinformatics*, 28(12):283–291, 2012.
2. M. A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *J. Algorithms*, 57(2):75–94, 2005.