An Integer Linear Programming Solution for the Domain-Gene-Species Reconciliation Problem

Lei Li
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT, USA
lei.li@uconn.edu

Mukul S. Bansal
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT, USA
mukul.bansal@uconn.edu

ABSTRACT

It is well-understood that most eukaryotic genes contain one or more protein domains and that the domain content of a gene can change over time. This change in domain content, through domain duplications, transfers, or losses, has important evolutionary and functional consequences. Recently, a powerful new reconciliation framework, called Domain-Gene-Species (DGS) reconciliation, was introduced to simultaneously model the evolution of a domain family inside one or more gene families and the evolution of those gene families inside a species tree.

The underlying computational problem in DGS reconciliation is NP-hard and a heuristic algorithm is currently used to estimate optimal DGS reconciliations. However, this heuristic has several undesirable limitations. First, it offers no guarantee of optimality or near-optimality. Second, it can result in biologically unrealistic evolutionary scenarios. And third, it only computes a single DGS reconciliation even though there can be multiple optimal DGS reconciliations. In this work, we introduce the first exact algorithm for computing optimal DGS reconciliations that addresses all three limitations. Our algorithm is based on an integer linear programming formulation of the problem, which we solve iteratively by solving a series of linear programming relaxations. Our experimental results on over 3,400 domain trees and over 7,000 gene trees from 12 fly species shows that our new algorithm is highly scalable and that it leads to significant improvement in DGS reconciliation inference. An implementation of our exact algorithm is available freely from http://compbio.engr.uconn.edu/software/seadog/.

CCS CONCEPTS

• **Theory of computation → Theory and algorithms for application domains**; • **Applied computing → Molecular evolution**;

KEYWORDS

Phylogenetic reconciliation; protein domains; gene family evolution; linear programming

ACM-BCB'18, August 29–September 1, 2018, Washington, DC, USA

ACM ISBN 978-1-4503-5794-4/18/08…$15.00
https://doi.org/10.1145/3233547.3233603

ACM Reference Format:
Lei Li and Mukul S. Bansal. 2018. An Integer Linear Programming Solution for the Domain-Gene-Species Reconciliation Problem. In *ACM-BCB '18: 9th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, August 29–September 1, 2018, Washington, DC, USA.* ACM, New York, NY, USA, 12 pages.
https://doi.org/10.1145/3233547.3233603

1 INTRODUCTION

It is estimated that more than 60% of genes in eukaryotes and 40% of genes in prokaryotes [8, 11] consist of multiple *protein domains* (well-characterized functional units). These protein domains can be independently lost or gained during gene evolution [17], which has important functional and evolutionary consequences for the affected genes [2, 16, 25, 26]. Many phylogenetic reconciliation methods exist for studying the evolution of gene families (or gene trees) inside species trees, e.g. [1, 3, 6, 7, 9, 10, 15, 18, 20, 21, 23, 24], but these methods ignore domain-level events such as domain-gain and loss. The inability of these methods to take domains into consideration not only affects their accuracy but also makes it difficult to study domain evolution itself. Likewise, several methods exist for studying the evolution of domain families (or domain trees), but these methods either do not take gene trees into account at all [5, 27, 30] or do not account for the inter-dependence of domain, gene, and species level evolution [22].

Recently, a powerful new phylogenetic reconciliation framework, called *Domain-Gene-Species (DGS) reconciliation*, was introduced to simultaneously model the evolution of a domain family inside one or more gene families and the evolution of those gene families inside a species tree [14]. DGS reconciliation framework explicitly captures the interdependence of domain, gene, and species level evolution, and simultaneously optimizes the domain-gene and gene-species aspects of the reconciliation. Simultaneous analysis of domain, gene, and species level evolution provides insights that cannot be obtained by analyzing only domain-gene, domain-species, or gene-species evolution separately. In particular, DGS reconciliation makes it possible to trace the evolution of a domain family within and across gene trees, enables a fine-grained view of gene family evolution with domain gains and losses clearly specified, and yields more accurate gene-species reconciliations by computing a *joint* reconciliation between the domain trees, gene trees, and species tree. As shown in [14], DGS reconciliation often results in evolutionary inferences that are markedly different than inferences obtained using only pairwise reconciliation between either domain trees and gene trees, domain trees and species trees, or gene trees and species trees.

No exact algorithms currently exist for DGS reconciliation, and a heuristic algorithm based on dynamic programming is used to estimate optimal DGS reconciliations [14]. However, this heuristic has several important limitations. First, it offers no guarantee of optimality or near-optimality. Second, it can result in biologically unrealistic evolutionary scenarios. And third, it only computes a single DGS reconciliation even though there can be multiple optimal DGS reconciliations. These limitations make it hard to properly interpret any DGS reconciliation computed by the heuristic; in particular, one cannot determine if a significantly more optimal reconciliation exists, or if there are other equally optimal but distinct reconciliations, or if a more biologically plausible reconciliation can be obtained.

Our contributions. In this work, we introduce the first exact algorithm for computing optimal DGS reconciliations that addresses these limitations. Our algorithm is based on an integer linear programming formulation of the problem and works by finding all optimal DGS reconciliations within a specified search space. This search space can either be unrestricted, allowing for consideration of any valid DGS reconciliation, or restricted to a more biologically meaningful subset of the full search space. We show how this search space can be appropriately defined for biological realism and plausibility, which also has the added benefit of allowing large instances of the DGS reconciliation problem (with domain trees and gene trees containing hundreds of leaves) to be solved exactly when confined to that search space. A crucial part of our algorithmic strategy is that we do not need to use an ILP solver. Instead, we show how to solve the ILP formulation efficiently in practice by iteratively solving a series of linear programming relaxations.

We implemented and applied our algorithm to a large dataset of $3,761$ domain trees and $7,165$ gene trees from 12 fly species. We found that our ILP algorithm was highly scalable and could be applied to $3,479$ (92.4%) of the domain trees in the dataset, even when the domain and gene trees had hundreds of leaves. The ILP algorithm was also highly effective and, even when restricted to only biologically plausible scenarios, produced more optimal DGS reconciliations than the heuristic algorithm for 264 (i.e., 7.6%) of the $3,479$ domain families, resulting in an average reduction of 9.4% in the DGS reconciliation cost for these domain trees. We also found that a small but significant fraction of the domain trees had multiple optimal gene-species reconciliations. The ILP algorithm, when restricted to only biologically plausible scenarios, was able to find either more optimal or equally optimal DGS reconciliations for $3,464$ of the $3,479$ domain trees, showing that optimal DGS reconciliations are almost always biologically plausible. The experimental results also show that solutions computed by the heuristic are usually optimal or near-optimal, suggesting that when input instances are too large for the exact ILP algorithm the heuristic offers a good tradeoff between scalability and accuracy.

The remainder of this manuscript is organized as follows: The next section starts with preliminaries and problem definitions. In Section 3 we show how to transform the computational problem first into a flow problem on graphs and then into an integer linear program. In Section 4 we describe our iterated linear programming solution for the ILP formulation. In Section 5 we show how to define the search space of allowed gene-species mappings. Detailed experimental results appear in Section 6, and concluding remarks in Section 7.

2 DEFINITIONS AND PRELIMINARIES

We follow the notation, basic definitions, and problem formulations from [14].

Preliminaries. Throughout this manuscript, the term *tree* refers to binary rooted trees. Given a tree T, we denote its node, edge, and leaf sets by $V(T)$, $E(T)$, and $Le(T)$ respectively. The set of internal nodes of T, denoted $I(T)$, is defined to be $V(T) \setminus Le(T)$. The root node of T is denoted by $rt(T)$, the parent of a node $v \in V(T)$ by $pa_T(v)$, its set of children by $Ch_T(v)$, and the (maximal) subtree of T rooted at v by $T(v)$. The set of *internal nodes* of T, denoted $I(T)$, is defined to be $V(T) \setminus Le(T)$. We define $\leq_T$ to be the partial order on $V(T)$ where $x \leq_T y$ if y is a node on the path between $rt(T)$ and x. The partial order $\geq_T$ is defined analogously, i.e., $x \geq_T y$ if x is a node on the path between $rt(T)$ and y. We say that y is an *ancestor* of x, or that x is a *descendant* of y, if $x \leq_T y$ (note that, under this definition, every node is a descendant as well as ancestor of itself). We say that x and y are *incomparable* if neither $x \leq_T y$ nor $y \leq_T x$. Given a non-empty subset $L \subseteq Le(T)$, we denote by $lca_T(L)$ the least common ancestor (LCA) of all the leaves in L in tree T; i.e., $lca_T(L)$ is the unique smallest upper bound of L under $\leq_T$.

The input for DGS reconciliation is a domain tree D, a collection of gene trees $\mathcal{G}$, and a species tree S. The *species tree* is a tree showing the evolutionary history for a chosen set of species. Each *gene tree* is a tree showing the evolutionary history for a set of genes related by common ancestry (but avoiding domain chaining [12]), called a *gene family*, restricted to the species represented in the species tree. Similarly, a *domain tree* shows the evolutionary history of a set of domains related by common ancestry, called a *domain family*, restricted to the species present in the species tree. For convenience, we extend the notions of leaf set and vertex set of a tree as follows: $Le(\mathcal{G}) = \cup_{G \in \mathcal{G}} Le(G)$ and $V(\mathcal{G}) = \cup_{G \in \mathcal{G}} V(G)$.

Each leaf in a gene tree is labeled by the species from which that leaf (gene) was sampled. Similarly, each leaf in a domain tree is labeled with the gene from which that leaf (domain) was taken. This defines a leaf-to-leaf mapping from the domain trees to the gene trees, and from the gene trees to the species tree. Since a gene may have multiple domains, there may be multiple domains (possibly from different domain trees) mapping to the same gene. Similarly, since domains from the same domain family may be present in multiple gene families, different leaves of a single domain tree may map to genes from different gene families.

2.1 DGS reconciliation

In the *domain-gene-species (DGS)* reconciliation model, the goal is to find a reconciliation of the given gene trees with the species tree, and of the given domain tree with the gene trees [14]. The reconciliation of a gene tree with a species tree models the primary evolutionary events that shape gene family evolution within species; in the case of multi-cellular organisms these are *speciation*, *gene duplication*, and *gene loss*. Similarly, the reconciliation of a domain tree with one or more gene trees models the elementary evolutionary events that shape domain family evolution within genes; in this case *co-divergence*, *domain transfer*, *domain duplication*, and

domain loss. Each event is assigned a cost and the computational objective is to find a DGS reconciliation of minimum total cost [14].

DGS-reconciliation is formally defined as follows:

Definition 2.1 (DGS-reconciliation [14]). *Given a domain tree D, collection of gene trees $\mathcal{G}$, a species tree S, and leaf-mappings $\mathcal{L}^D\colon Le(D) \to Le(\mathcal{G})$ and $\mathcal{L}^{\mathcal{G}}\colon Le(\mathcal{G}) \to Le(S)$, a DGS-reconciliation for $D, \mathcal{G}$ and S is a nine-tuple $\langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta, \Xi, \tau\rangle$, where $\mathcal{M}^D\colon V(D) \to V(\mathcal{G})$ and $\mathcal{M}^{\mathcal{G}}\colon V(\mathcal{G}) \to V(S)$ map each node of D to a node from $\mathcal{G}$ and each node from $\mathcal{G}$ to a node of S, respectively, the sets Σ^D, Δ^D, and Θ partition $I(D)$ into co-divergence, domain-duplication, and domain-transfer nodes, respectively, the sets $\Sigma^{\mathcal{G}}$ and $\Delta^{\mathcal{G}}$ partition $I(\mathcal{G})$ into speciation and gene-duplication nodes, respectively, Ξ is a subset of domain tree edges that represent domain-transfer events, and $\tau\colon \Theta \to V(\mathcal{G})$ specifies the recipient gene for each domain-transfer event, subject to:*

Gene-Species constraints:

(1) If $g \in Le(\mathcal{G})$, then $\mathcal{M}^{\mathcal{G}}(g) = \mathcal{L}^{\mathcal{G}}(g)$.
(2) If $g \in I(\mathcal{G})$ and g' and g'' denote the children of g, then,
  (a) $\mathcal{M}^{\mathcal{G}}(g) \geq_S lca(\mathcal{M}^{\mathcal{G}}(g'), \mathcal{M}^{\mathcal{G}}(g''))$,
  (b) $g \in \Sigma^{\mathcal{G}}$ if and only if $\mathcal{M}^{\mathcal{G}}(g) = lca(\mathcal{M}^{\mathcal{G}}(g'), \mathcal{M}^{\mathcal{G}}(g''))$ and $\mathcal{M}^{\mathcal{G}}(g')$ and $\mathcal{M}^{\mathcal{G}}(g'')$ are incomparable,
  (c) $g \in \Delta^{\mathcal{G}}$ only if $\mathcal{M}^{\mathcal{G}}(g) \geq_S lca(\mathcal{M}^{\mathcal{G}}(g'), \mathcal{M}^{\mathcal{G}}(g''))$.

Domain-Gene constraints:

(3) If $d \in Le(D)$, then $\mathcal{M}^D(d) = \mathcal{L}^D(d)$.
(4) If $d \in I(D)$ and d' and d'' denote the children of d, then,
  (a) $\mathcal{M}^D(d) \not\leq_{\mathcal{G}} \mathcal{M}^D(d')$ and $\mathcal{M}^D(d) \not\leq_{\mathcal{G}} \mathcal{M}^D(d'')$,
  (b) At least one of $\mathcal{M}^D(d')$ and $\mathcal{M}^D(d'')$ is a descendant of $\mathcal{M}^D(d)$ (in the same gene tree).
(5) Given any edge $(d, d') \in E(D)$, $(d, d') \in \Xi$ if and only if $\mathcal{M}^D(d)$ and $\mathcal{M}^D(d')$ are in different gene trees or incomparable in the same gene tree.
(6) If $d \in I(D)$ and d' and d'' denote the children of d, then,
  (a) $d \in \Sigma^D$ if and only if $\mathcal{M}^D(d) = lca(\mathcal{M}^D(d'), \mathcal{M}^D(d''))$ (in the same gene tree) and $\mathcal{M}^D(d')$ and $\mathcal{M}^D(d'')$ are incomparable,
  (b) $d \in \Delta^D$ only if $\mathcal{M}^D(d) \geq_{\mathcal{G}} lca(\mathcal{M}^D(d'), \mathcal{M}^D(d''))$ (in the same gene tree),
  (c) $d \in \Theta$ if and only if either $(d, d') \in \Xi$ or $(d, d'') \in \Xi$.
  (d) If $d \in \Theta$ and $(d, d') \in \Xi$, then $\mathcal{M}^D(d)$ and $\tau(d)$ must either be in different gene trees or incomparable in the same gene tree, $\mathcal{M}^{\mathcal{G}}(\mathcal{M}^D(d)) = \mathcal{M}^{\mathcal{G}}(\tau(d))$, and $\mathcal{M}^D(d') \leq_{\mathcal{G}} \tau(d)$.

Constraints 1 and 2 above apply to the reconciliation of the gene trees with the species tree and are based on the classical *Duplication-Loss* model [9, 18] extended to allow suboptimal gene-species reconciliations. Constraints 3, 4, 5, and 6 apply to the reconciliation of the domain tree with the gene trees. Constraint 3 ensures that the mapping $\mathcal{M}^D$ is consistent with the leaf-mapping $\mathcal{L}^{\mathcal{D}}$. Constraint 4a imposes on $\mathcal{M}^D$ the temporal constraints (ancestor-descendant relationships) implied by the gene trees. Constraint 4b implies that any internal node in D may represent at most one domain-transfer event. Constraint 5 determines the edges of D that are domain-transfer edges. Constraints 6a, 6b, and 6c state the conditions under which an internal node of $\mathcal{G}$ may represent a co-divergence, domain-duplication, and domain-transfer

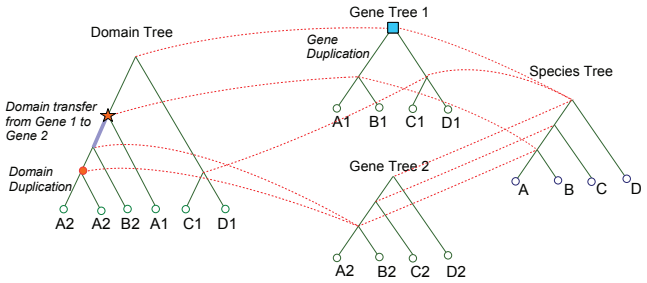

Figure 1: DGS reconciliation. (a) The figure shows a DGS reconciliation for a domain tree whose domains come from two gene trees and a species tree on 4 taxa. The Domain-Gene-Species (DGS) reconciliation model simultaneously optimizes the mapping of the domain tree into the gene trees and of the gene trees into the species tree. These mappings are shown by the dotted red lines in the figure. Domain-gene leaf associations are specified by shared leaf labels, and gene-species leaf associations are specified by shared letters (*A*, *B*, *C*, or *D*). The DGS reconciliation in the figure shows how the two gene trees evolved inside the species tree and how the domain tree evolved inside the two gene trees. In the gene-species reconciliation, a gene-duplication event (marked by the blue square) is invoked at the root of gene tree 1 while all other internal nodes of the gene trees represent speciation events. In the domain-gene reconciliation, a domain duplication event is invoked at the node with the orange circle, and a domain-transfer event is invoked at the node with the orange star. The bolded edge in the domain tree represents the domain-transfer edge, where the domain is copied from gene tree 1 to gene tree 2. As required by the model, the donor gene from gene tree 1 and the recipient gene from gene tree 2 both map to the same species tree node. This DGS reconciliation invokes one domain-loss on the domain tree and three gene-losses on gene tree 1. Figure adapted from [14].

respectively. Constraint 6d specifies which genes may be designated as the recipient gene for any given domain-transfer event. Note that, in the absence of horizontal gene transfer, the transfer of a domain from one gene to another can only happen within the same genome. Thus, Constraint 6d explicitly enforces that the donor gene and recipient gene for any domain transfer event must map to the same species in the species tree. Figure 1 shows an example of a valid DGS-reconciliation.

Each evolutionary event other than speciation and co-divergence is assigned a positive cost. $P^{\mathcal{G}}_{\Delta}$ and $P^{\mathcal{G}}_{loss}$ denote the gene-duplication and gene-loss costs, while P^{D}_{Δ}, P^{D}_{Θ}, and $P^{\mathcal{G}}_{loss}$ denote domain-duplication, domain-transfer, and domain-loss costs. The model allows for the use of two separate costs $P^{D}_{\Theta 1}$ and $P^{D}_{\Theta 2}$ instead of a single P^{D}_{Θ}, so that a distinction can be made between domain transfers that remain within the same gene family from those that cross gene family boundaries.

Notation: Given any DGS reconciliation $\alpha = \langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta, \Xi, \tau \rangle$, we can separate its domain-gene and gene-species reconciliation components. The domain-gene component defines the reconciliation of the domain tree with the gene tree(s) and is denoted by $\alpha_D = \langle \mathcal{M}^D, \Sigma^D, \Delta^D, \Theta, \Xi, \tau \rangle$, and gene -species component defines the reconciliation of the gene tree(s) with the species tree and is denoted by $\alpha_{\mathcal{G}} = \langle \mathcal{M}^{\mathcal{G}}, \Sigma^{\mathcal{G}}, \Delta^{\mathcal{G}} \rangle$. Note that α is completely specified if we are given both α_D and $\alpha_{\mathcal{G}}$.

The reconciliation cost of a given DGS-reconciliation is defined as follows.

Definition 2.2 (Reconciliation cost). *Given a DGS-reconciliation α, the* reconciliation cost *for α is the total cost of all events invoked by α. Equivalently, the* reconciliation cost *for α is the total cost of all events invoked by α_D and $\alpha_{\mathcal{G}}$*

Note that, while domain-duplication, domain-transfer, and gene-duplication events are directly specified in the DGS-reconciliation, domain-losses and gene-losses are not. However, given a DGS-reconciliation, one can directly count the minimum number of gene- and domain-losses required by that reconciliation [14].

The computational objective is to find an optimal, or most parsimonious, reconciliation, i.e., a DGS-reconciliation that has minimum reconciliation cost. More formally:

Definition 2.3 (ODGS Problem). *Given D, $\mathcal{G}$ and S, along with $P^{\mathcal{G}}_{\Delta}$, $P^{\mathcal{G}}_{loss}$, P^{D}_{Δ}, $P^{D}_{\Theta 1}$, $P^{D}_{\Theta 2}$, and P^{D}_{loss}, the* Optimal DGS-Reconciliation (ODGS) *problem is to find a DGS-reconciliation for D, $\mathcal{G}$ and S with minimum reconciliation cost.*

2.2 Restricted ODGS problem

The ODGS problem imposes no restriction on the gene-species mapping $\mathcal{M}^{\mathcal{G}}$, which can lead to inferences that are biologically implausible. The space of possible DGS reconciliations can be reduced, and biological realism simultaneously improved, by imposing reasonable constraints on the gene-species mapping $\mathcal{M}^{\mathcal{G}}$. In particular, let Γ denote a set of candidate gene-species mappings $\mathcal{M}^{\mathcal{G}}$. We define the Γ-restricted ODGS problem to be the ODGS problem under the constraint that the gene-species mapping must be present in Γ. More formally:

Definition 2.4 (Γ-ODGS Problem). *Given D, $\mathcal{G}$ and S, along with $P^{\mathcal{G}}_{\Delta}$, $P^{\mathcal{G}}_{loss}$, P^{D}_{Δ}, $P^{D}_{\Theta 1}$, $P^{D}_{\Theta 2}$, and P^{D}_{loss}, and a set Γ of candidate mappings from $\mathcal{G}$ to S, the* Γ-restricted ODGS (Γ-ODGS) *problem is to find a DGS-reconciliation $\langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta, \Xi, \tau \rangle$ with minimum reconciliation cost such that $\mathcal{M}^{\mathcal{G}} \in \Gamma$.*

Observe that if Γ includes all possible gene-species mappings then the Γ-ODGS problem simply becomes the ODGS problem. In the next section, we formulate the Γ-ODGS problem as an integer linear program and show how to define Γ in a biologically meaningful way.

3 AN ILP FORMULATION FOR Γ-ODGS

Given any fixed gene-species mapping $\mathcal{M}^{\mathcal{G}}$, the ODGS problem restricted to only $\mathcal{M}^{\mathcal{G}}$ can be solved in polynomial time using the *extended-DTL* Algorithm described in [14]. Thus, if the set of candidate gene-species mappings Γ is small, then the Γ-ODGS problem can be solved efficiently by simply considering all possible $\mathcal{M}^{\mathcal{G}} \in \Gamma$. However, the space of biologically plausible gene-species mappings can be very large, as we will see later, and the Γ-ODGS problem can be solved far more efficiently by using an integer linear programming formulation instead.

In the following, we first show how to transform the Γ-ODGS problem into a constrained network flow problem on a directed network derived from D, $\mathcal{G}$, and S. The network, denoted H, consists of two components that we call H_1 and H_2. Component H_1 corresponds to possible reconciliations between the domain tree and gene trees, while component H_2 corresponds to the set of candidate gene-species mappings Γ. Later, we will convert the constrained network flow problem into an integer linear programming (ILP) formulation and prove its correctness.

3.1 Construction of network component H_1

Component H_1 represents the reconciliation between D and $\mathcal{G}$, and the vertices in H_1 correspond either to domain nodes from D or gene nodes from $\mathcal{G}$. We refer to the nodes of H_1 that correspond to domains as *domain vertices* and those that correspond to genes as *gene vertices*. Each domain vertex in H_1 represents either the triple consisting of a domain node $i \in V(D)$ and specific mappings for its left and right children, if i is an internal node, or just i itself if i is a leaf node. (We point out that while the domain tree is unordered, for notational convenience we impose an ordering to distinguish between the "left" and "right" child of an internal node.) Thus, each domain node $i \in Le(D)$ has a single representative in H_1, while each $i \in I(D)$ has multiple representatives in H_1 corresponding to all possible mapping assignments for its two children. Specifically, a domain vertex in H_1 is labeled as $d_i^{l,r}$ if it corresponds to domain node $i \in V(D)$ with the left child of i mapping to node $l \in V(\mathcal{G})$ and the right child of i mapping to node $r \in V(\mathcal{G})$. If i is a leaf node then the corresponding vertex in H_1 is labeled $d_i^{0,0}$. Each domain vertex in H_1 is associated with one or more gene vertices in H_1, which captures the possible gene node mappings for the corresponding domain node. Thus, each gene node in $V(\mathcal{G})$ corresponds to zero or more of these gene vertices, and each gene vertex is associated with a specific domain vertex in H_1. Specifically, consider any domain vertex $d_i^{l,r}$. If the domain node i can map to a gene node $j \in V(\mathcal{G})$ under the constraint that the left child of i maps to l and the right child to r, then we create a gene vertex labeled $g^i_{j,R}$ associated with $d_i^{l,r}$. If domain node i represents a domain-transfer event when it maps to gene node j and its left and right children map to l and r, respectively, then $R = \tau(i)$. Otherwise, $R = 0$. Note that when the domain vertex is of the form $d_i^{0,0}$ (i.e., when $i \in Le(D)$), there is only one gene node that i can map to (specified by the given leaf mapping), and so there is only one gene vertex associated with $d_i^{0,0}$.

We now describe how to construct this network H_1.

Algorithm: *Construct-H_1*$(D, \mathcal{G}, S, \mathcal{L}^D, \mathcal{L}^{\mathcal{G}})$

1: **for** each domain node i in a post order traversal of D **do**
2: If $i \in Le(D)$ then add domain vertex $d_i^{0,0}$ to H_1. Otherwise, add all domain vertices $d_i^{l,r}$ corresponding to i into H_1.
3: Add gene vertices $g^i_{j,R}$, for each $j, R \in V(\mathcal{G})$.
4: **if** $i \in I(D)$ **then**

Let i' and i'' denote the left and right child of i, respectively.

For each newly inserted domain vertex of the form $d_i^{l,r}$, find all gene vertices of the form either $g_{l,R}^{i'}$ or $g_{r,R}^{i''}$ and add an edge from these gene vertices to that domain vertex.

for each domain vertex in H_1 of the form $d_i^{l,r}$, where $l, r \in V(\mathcal{G})$ **do**

Add an edge from $d_i^{l,r}$ to gene vertex $g_{j,R}^{i}$, if it is possible (under the definition of DGS-reconciliation) to map domain node i to gene node j under the constraint that the left child of i maps to l and the right child to r. In addition, R must be as follows: If i is a domain-transfer event under this mapping then R must be such that there exists a valid gene-species mapping under which j and R map to the same species node; if i is not a domain-transfer event then $R = 0$.

else if $i \in Le(D)$ **then**

Add an edge from $d_i^{0,0}$ to $g_{j,0}^{i}$, where $j = \mathcal{L}^D(i)$.

Add a *source* vertex to H_1 and add an edge from this vertex to each domain vertex that corresponds to a leaf node of D.

As a *sink* vertex and add an edge from all gene vertices of the form $g_{j,R}^{rt(D)}$, where $j, R \in V(\mathcal{G})$, to this sink.

Delete all gene vertices that do not have any incoming edges.

Figure 2 illustrates how H_1 is constructed.

3.2 Reducing the size of the network

We can reduce the size of H_1 without sacrificing optimality by using a branch and bound approach. The existing dynamic programming heuristic for DGS-reconciliation, described in [14], can be used to compute an upper bound on the domain-gene reconciliation cost of subtree $D(i)$, for any $i \in V(D)$, under the constraint that i maps to gene node j. We denote this upper bound by $U(i, j)$.

We can also use the extended DTL reconciliation model and its exact algorithm [14], which optimally reconciles the domain tree with the gene trees without considering the species tree or the species constraint on domain-transfers, to compute a lower bound on the reconciliation cost for subtree $D(i)$, for any $i \in V(D)$, under the constraint that i maps to gene node j. We denote this lower bound by $L(i, j)$.

Consider any domain vertex $d_i^{l,r}$, where i is an internal node of D, and i', i'' denote the two children of i. Suppose there is an edge from $d_i^{l,r}$ to gene vertex $g_{j,R}^{i}$. Consider the domain-gene reconciliation scenario where node i maps to j, i' to l, and i'' to r, and R denotes the recipient gene node in case i is a domain-transfer. Let $\mathcal{F}$ denote the sum of the cost for the event (co-divergence, domain-duplication, or domain-transfer) at i and the loss costs along the two edges (i, i') and (i, i''). Then, we can safely delete the gene vertex $g_i^{j,R}$ from H_1 (along with the relevant edges) if $U(i, j) \leq \mathcal{F} + L(i', l) + L(i'', r)$.

3.3 Construction of network component H_2

Component H_2 corresponds to the set of candidate gene-species mappings Γ. This component consists simply of a set of vertices, which we call "solution vertexes", corresponding to the set Γ. Thus each vertex in H_2 represents a specific mapping of the gene tree into the species tree. These "solution vertices" are used to restrict the edges of H_1 that can be used to define a valid DGS-reconciliation. Consider a vertex p of the form $d_i^{l,r}$ and vertex q of the form $g_{j,R}^{i}$ from H_1. If (p, q) is an edge in H_1 then we denote by $H_2(p, q)$ the subset of vertices of H_2 (i.e., subset of gene-species reconciliations from Γ) that are compatible with the (partial) domain-gene reconciliation imposed by the edge (p, q). Specifically, if domain node i maps to gene node j, the left child of i maps to l, the right child of i maps to r, i is a domain-transfer event, and R is the recipient of that domain-transfer, then $H_2(p, q)$ contains exactly those gene-species mappings from Γ in which j (the donor) and R (the recepient) map to the same species node.

We discuss how the set Γ, and therefore H_2, can be appropriately defined in practice in Section 5.

3.4 Constrained Network Flow and its ILP formulation

We will first describe the specific constrained network flow problem that must be solved on H to compute optimal solutions for the Γ-ODGS problem, and then show how to cast the resulting optimization problem as an ILP. By understanding the intuition behind the network flow formulation, it becomes much easier to understand the ILP formulation.

3.4.1 Constrained minimum cost network flow. Observe that each domain-to-gene edge, say $(d_i^{l,r}, g_{j,R}^{i})$, in H_1 corresponds to a specific "local" domain-gene reconciliation for domain node i and its two children, defining the mapping for those three nodes as well as the event type for i. We will assign a weight to each domain-to-gene edge of H_1 based on the cost of this "local" reconciliation. More precisely: For an edge (p, q) in H_1, if $p = d_i^{l,r}$, for $i \in V(D)$ and $l, r \in V(\mathcal{G})$, is a domain vertex and $q = g_i^{j,R}$, for $j \in V(\mathcal{G})$, is a gene vertex, then we set the weight of (p, q), denoted $\mathcal{W}(p, q)$, equals to the evolutionary cost of the event indicated by (p, q), plus the loss costs along the lineages of the domain node d_i and its two children on the domain tree. All other edges are assigned weight 0.

Now suppose, for simplicity, that Γ consists of only a single candidate gene-species mapping, i.e., network component H_2 has exactly one vertex, and wish to find an optimal domain-gene reconciliation that is compatible with that specific gene-species mapping. This problem can be solved using a minimum cost constrained network flow formulation as follows: We first identify all those domain-to-gene edges whose "local" reconciliations are incompatible with the given gene-species mapping and assign them a capacity of 0. All other edges are assigned a capacity of 1. Given any edge (p, q) in H_1, we define $\mathcal{F}(p, q)$ to be the flow value for that edge. The objective now is to find an integer valued flow in H_1 that minimizes the sum $\sum_{(p,q)}(\mathcal{F}(p, q) \times \mathcal{W}(p, q))$, subject to the following three constraints:

(1) The total flow out of the source should equal $|Le(D)|$, i.e., each edge going out of the source should be saturated with flow.

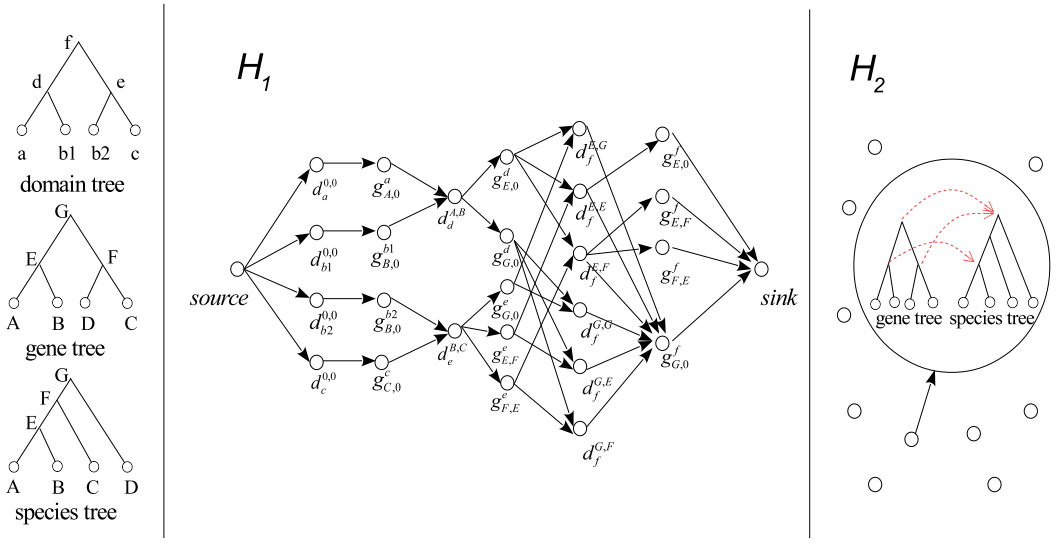

Figure 2: Example showing how the network H is constructed. Given the domain tree D, gene tree G, and species tree S on the left, the resulting network component H_1 is shown in the middle. The domain-gene leaf mapping is given by shared first letters in the leaf labels (lowercase letters in D and uppercase letters in G), and the gene-species leaf mapping is defined by shared leaf labels. The set of "solution vertices" that comprise network component H_2, is shown on the right. As depicted, each of the vertices in H_2 corresponds to a specific gene-species mapping from set of allowed gene-species mappings Γ. Components H_1 and H_2 together define H.

(2) The total flow out of a domain vertex of the form $d_i^{l,r}$, where $i \in I(D)$, should equal half the incoming flow.
(3) For any $i \in I(D)$, with i' and i'' denoting its two children, let q' be a gene vertex of the form $g_{j,R}^{i'}$ and q'' be a gene vertex of the form $g_{j,R}^{i''}$. Then, for each domain vertex p of the form $d_i^{l,r}$, we must have $\sum_{q'} \mathcal{F}(q', p) = \sum_{q''} \mathcal{F}(q'', p)$.

Here, the objective function minimizes the total domain-gene reconciliation cost, the way the edge capacities are set ensures that only domain-gene reconciliations compatible with the specific gene-species mapping are considered, the first constraint ensures that the entire domain tree is reconciled, the second constraint ensures that each internal domain node has both of its children reconciled without violating the capacity constraint at any edge (it may help to envision flow moving up from the leaves the domain tree towards the root, each internal node would have a incoming flow of 2 but an outgoing flow of 1), and the third constraint ensures that the edges with non-zero flow are consistent with a single domain-gene reconciliation (i.e., that all the chosen "local" reconciliations are correctly formed).

This formulation can be easily extended to the case when Γ contains multiple candidate gene-species mappings. This can be done by defining a binary (0 or 1) "usage" value for each vertex in H_2, constraining the total usage value for all vertices of H_2 to be 1, and setting the capacities of each domain-to-gene edge in H_1 to be equal to the sum of the usage values for all vertices (i.e., gene-species mappings) in H_2 that are compatible with that edge.

The constrained minimum cost network flow problem described above can be directly formulated as an integer linear program (ILP) as described next. Later, in Section 3.5, we prove that this ILP formulation solves the Γ-ODGS problem optimally, thereby also proving the correctness of the constrained minimum cost network flow formulation.

3.4.2 Integer linear programming formulation. We define the following variables. We denote the flow value along any edge $(p, q) \in E(H_1)$ by $f_{(p,q)}$. Given any vertex $k \in V(H_2)$, we denote its usage value by u_k, where $u_k \in \{0, 1\}$. Each edge $(p, q) \in E(H_1)$ is assigned a fixed weight $\mathcal{W}(p, q)$ as defined previously. We denote the fixed gene-species reconciliation cost of any vertex $k \in V(H_2)$ by $C(k)$.
Objective function. The objective function captures the total DGS-reconciliation cost of the solution and can be written as

$$\text{Minimize} \sum_{(p,q)\in E(H_1)} (\mathcal{W}(p, q) \times f_{(p,q)}) + \sum_{k\in V(H_2)} (C(k) \times u_k) \tag{1}$$

Here, the term $\Sigma_{(p,q)\in E(H_1)}(\mathcal{W}(p, q)\times f_{(p,q)})$ captures the domain-gene reconciliation cost. Recall that we only allow one of the vertices $k \in V(H_2)$ to have usage value equal to 1 and the term $\Sigma_{k\in V(H_2)}(C(k) \times u_k))$ therefore captures the gene-species reconciliation cost of the chosen gene-species mapping.
Constraints.

Our first set of constraints capture flow conservation. For each gene vertex q of the form $g_{j,R}^{i}$, where $i \in V(D)$ and $j, R \in V(\mathcal{G})$ we define the usual flow conservation constraint as follows.

$$\sum_{p|(p,q)\in E(H_1)} f_{(p,q)} = \sum_{x|(q,x)\in E(H_1)} f_{(q,x)}. \quad (2)$$

For each domain vertex p of the form $d_i^{l,r}$, where $i \in I(D)$, we define a scaled flow conservation constraint (corresponding to the second constraint in the network flow formulation above) as follows.

$$2 \cdot \sum_{q|(p,q)\in E(H_1)} f_{(p,q)} = \sum_{x|(x,p)\in E(H_1)} f_{(x,p)}. \quad (3)$$

The next set of constraints enforces that the edges out of the source are saturated with flow (corresponding to the first constraint in the network flow formulation above). For each domain vertex p of the form $d_i^{0,0}$, where $i \in Le(D)$ we require.

$$\sum_{q|(p,q)\in E(H_1)} f_{(p,q)} = 1. \quad (4)$$

Next, we enforce that every domain vertex p of the form $d_i^{l,r}$, where $i \in I(D)$ must receive the same amount of flow from vertices corresponding to i's left and right children. (This corresponds to the third constraint in the network flow formulation above.) Formally, if i' and i'' denote the two children of domain node i in $I(D)$, then for each domain vertex p of form $d_i^{l,r}$ we require.

$$\sum_{j\in V(\mathcal{G})} \sum_{R\in V(\mathcal{G})\cup\{0\}} f_{(g_{j,R}^{i'},p)} = \sum_{j\in V(\mathcal{G})} \sum_{R\in V(\mathcal{G})\cup\{0\}} f_{(g_{j,R}^{i''},p)}. \quad (5)$$

Finally, we have the usage and capacity constraints that together enforce that only those domain-gene edges are assigned a flow that are compatible with a single "solution vertex" from H_2. Thus, we have

$$\sum_{k\in V(H_2)} (u_k) = 1, \quad (6)$$

and, for each domain vertex p of the form $d_i^{l,r}$, where $i \in I(D)$, and edge $(p,q) \in E(H_1)$, we require

$$f_{(p,q)} \leq \sum_{k\in H_2(p,q)} u_k. \quad (7)$$

Note that all variables in this ILP are required to be integral. Also note that the total number of variables and constraints is polynomial in the sizes of the domain tree, gene trees, and $|\Gamma|$.

3.5 Correctness of the ILP formulation

It is not very difficult to show that solving the Γ-ODGS problem is equivalent to finding an optimal integral solution for the ILP formulation described above. Specifically, the optimal value of the objective function must equal the DGS-reconciliation cost of an optimal solution to the Γ-ODGS problem, and an optimal DGS-reconciliation is defined by the domain-to-gene edges of H_1 that are assigned a flow value of 1 and by the vertex from H_2 that is assigned a usage value of 1. We therefore have the following claim.

Claim 1. *Solving the Γ-ODGS problem is equivalent to finding an integral solution that minimizes the objective function value in the ILP formulation.*

Proof. **Forward Direction**

Given an optimal 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 the Γ-ODGS problem, we will show how to assign each variable of the ILP formulation so that all constraints are satisfied and the objective function value equals the DGS-reconciliation cost for α.

Given an internal domain node $i \in I(D)$, we denote its two children by i' and i''. We will assign the variables in the ILP formulation as follows:

(1) For each $k \in V(H_2)$, assign $u_k = 1$ if k corresponds to $\mathcal{M}^{\mathcal{G}}$, and $u_k = 0$ otherwise.
(2) For each $i \in Le(D)$, let domain vertex $p = d_i^{0,0}$ and gene vertex $q = g^i_{\mathcal{M}^D(i),0}$. Assign $f_{(p,q)} = 1$.
(3) For each $i \in I(D)$, if $i \in \Theta$ then, for domain vertex $p = d_i^{\mathcal{M}^D(i'),\mathcal{M}^D(i'')}$ and gene vertex $q = g^i_{\mathcal{M}^D(i),\tau(i)}$, assign $f_{(p,q)} = 1$. If $i \neq rt(D)$, and $\hat{i}$ denotes the sibling of i, then also assign $f_{(q,r)} = 1$, where $r = d_{pa(i)}^{\mathcal{M}^D(i),\mathcal{M}^D(\hat{i})}$. If $i = rt(D)$, assign $f_{(q,sink)} = 1$.
(4) For each $i \in I(D)$, if $i \notin \Theta$ then, for domain vertex $p = d_i^{\mathcal{M}^D(i'),\mathcal{M}^D(i'')}$ and gene vertex $q = g^i_{\mathcal{M}^D(i),0}$, set $f_{(p,q)} = 1$. If $i \neq rt(D)$, and $\hat{i}$ denotes the sibling of i, then also assign $f_{(q,r)} = 1$, where $r = d_{pa(i)}^{\mathcal{M}^D(i),\mathcal{M}^D(\hat{i})}$. If $i = rt(D)$, assign $f_{(q,sink)} = 1$.
(5) Assign $f_{(p,q)} = 0$ for all other $(p,q) \in E(H_1)$.

Observe that the left term in the objective function, $\sum_{(p,q)\in E(H_1)}(\mathcal{W}(p,q) \times f_{(p,q)})$, evaluates to the domain-gene reconciliation cost for α_D and the right term, $\sum_{k\in V(H_2)}(C(k) \times u_k)$, is simply the gene-species reconciliation cost for $\alpha_{\mathcal{G}}$. Thus, the total objective function value equals the DGS-reconciliation cost for α. The following two observations follow directly from the assignment of flow values above.

Observation 1. *For each domain node $i \in V(D)$, there exists exactly one domain vertex p of form $d_i^{l,r}$ or $d_i^{0,0}$ satisfying $f(p,q) = 1$ for any $(p,q) \in E(H_1)$.*

Observation 2. *For each domain node $i \in V(D)$, there exists exactly one gene vertex q of form $g^i_{j,R}$ satisfying $f(q,x) = 1$ for any $(q,x) \in E(H_1)$.*

Next, we show that our assignment of flow and usage values satisfies all constraints in the ILP formulation.

Integrality constraints: Since $f(p,q) \in \{0,1\}$ for each $(p,q) \in E(H_1)$, and $u_k \in \{0,1\}$ for each $k \in V(H_2)$, the integrality constraints are satisfied.

Constraints from Equation 2: For each gene vertex q of the form $g^i_{j,R}$, Observation 1 implies that $\sum_{p|(p,q)\in E(H_1)} f_{(p,q)}$ must equal 1. Likewise, Observation 2 implies that $\sum_{x|(q,x)\in E(H_1)}$ also equals 1. Thus, constraints from Equation 2 are satisfied.

Constraints from Equation 3: Let $i \in I(D)$ and i', i'' denote the children of i. Observation 1 implies that $2 \cdot \sum_{q|(p,q)\in E(H_1)} f_{(p,q)}$ must equal 2. And applying Observation 2 to the domain nodes i' and i'' we can infer that $\sum_{x|(x,p)\in E(H_1)} f_{(x,p)}$ must also equal 2. Thus, constraints from Equation 3 are satisfied.

Constraints from Equation 4: This follows directly from Observation 1.

Constraints from Equation 5: Let $i \in I(D), i', i''$ denote the children of i, and p be a domain vertex of the form $d_i^{l,r}$. Recall that in our assignment of flow values we assigned $f_{(q',x)} = 1$ and $f_{(q'',x)} = 1$, where $q' = g^{i'}_{\mathcal{M}^D(i'),y}$ for $y \in \{\tau(i'), 0\}$, $q'' = g^{i''}_{\mathcal{M}^D(i''),z}$ for $z \in \{\tau(i'), 0\}$ and $x = d^{\mathcal{M}^D(i'),\mathcal{M}^D(i'')}_{pa(i)}$. From Observation 1, it follows that $\sum_{j\in V(\mathcal{G})} \sum_{R\in V(\mathcal{G})\cup\{0\}} f_{(g^{i'}_{j,R},p)}$ is either 0 or 1, depending on p. Suppose $\sum_{j\in V(\mathcal{G})} \sum_{R\in V(\mathcal{G})\cup\{0\}} f_{(g^{i'}_{j,R},p)} = 1$ for the given domain vertex p, then there must exist $q' = g^{i'}_{\mathcal{M}^D(i'),y}$, for $y \in \{\tau(i'), 0\}$, such that $f(q', p) = 1$. Per our assignment of flow values there must also exist $q'' = g^{i''}_{\mathcal{M}^D(i''),z}$, for $z \in \{\tau(i''), 0\}$, such that $f(q'', p) = 1$. Thus, $\sum_{j\in V(\mathcal{G})} \sum_{R\in V(\mathcal{G})\cup\{0\}} f_{(g^{i'}_{j,R},p)}$ must also be 1, satisfying the constraint. A similar argument applies to the case when $\sum_{j\in V(\mathcal{G})} \sum_{R\in V(\mathcal{G})\cup\{0\}} f_{(g^{i'}_{j,R},p)} = 0$.

Constraint from Equation 6: This is clearly satisfied since there is exactly one vertex k in $V(H_2)$ for which $u_k = 1$.

Constraints from Equation 7: Consider any edge $(p, q) \in E(H_1)$ where p is a domain vertex of the form $d_i^{l,r}$ and $i \in I(D)$. Per our assignment of flow values, $f_{(p,q)} \in \{0, 1\}$, and $f_{(p,q)} = 1$ only if the local reconciliation defined by that edge is compatible with the vertex $k \in V(H_2)$ that corresponds to $\mathcal{M}^{\mathcal{G}}$. Since we assigned $u_k = 1$, the constraints from Equation 7 must be satisfied.

Backward Direction

We now prove that any integral solution to the ILP formulation corresponds to a valid DGS-reconciliation α, and that the reconciliation cost of α equals the objective function value for that integral solution. Given any integral solution to the ILP, each flow value and usage value must be either 0 or 1. Moreover, there must be exactly one vertex $k \in V(H_2)$ for which $u_k = 1$. Thus, the gene-species mapping in α is uniquely defined, and the right term of the objective function, $\sum_{k\in V(H_2)}(C(k) \times u_k)$, must equal the reconciliation cost for $\alpha_{\mathcal{G}}$. It suffices to show that the flow values in H_1 correspond to a valid domain-gene reconciliation α_D (that is compatible with the gene-species mapping represented by k) and that the left term of the objective function, $\sum_{(p,q)\in E(H_1)}(\mathcal{W}(p, q) \times f_{(p,q)})$, equals the domain-gene reconciliation cost for α_D.

Consider any domain vertex p of the form $d_i^{l,r}$ and gene vertex q of the form $g^i_{j,R}$. If $f(p, q) = 1$, then that defines a "local" reconciliation for domain node i and its two children. In particular, $\mathcal{M}^D(i) = j$, the left and right children of i must map to gene nodes l and r, respectively, and if $R \neq 0$ then $\tau(i) = R$. Observe that, by the construction of network H_1 and by constraint 4, it follows that for each $i \in Le(D)$, $\mathcal{M}^D(i)$ is assigned correctly. Thus, to show that the flow values in H_1 correspond to a valid domain-gene reconciliation, it is sufficient to show that, for each $i \in I(D)$, (i) the mapping for i is uniquely defined, (ii) the mapping of i, along with the mappings of its two children, represents a valid evolutionary event, and (iii) if i represents a domain-transfer event then the donor gene node and recipient gene node must map to the same species node.

We will first show that for any $i \in I(D)$, the mapping for i is uniquely defined. As the flow moves from domain vertices of the form $d_i^{0,0}$, towards the sink, the integrality constraint on flow values, together with the flow conservation constraints from Equations 2 and 3, ensure that, for each domain or gene vertex with incoming flow, the flow out of that vertex is exactly 1 along exactly one outgoing edge. Combined with the fact that only one outgoing edge from any domain vertex of the form $d_i^{0,0}$ carries a flow of 1, and that, by the constraints of Equation 5, each domain vertex must receive the same quantity of flow from vertices corresponding to its left and right children, this implies that for each internal node $i \in I(D)$ there is a single vertex of the form $d_i^{l,r}$ with an outgoing flow of 1. This translates into a unique mapping for i.

Next, we will show that for any $i \in I(D)$, the uniquely assigned mappings of i and its two children must represent a valid evolutionary event. Let i' and i'' denote the left and right child of i in D, respectively. Recall that each domain vertex $p = d_i^{l,r}$ is a representative of domain node $i \in I(D)$ under the constraint that i' maps to gene node l and i'' maps to gene node r. We add an edge from p to a gene vertex $q = g^i_{j,R}$ only if it is possible (under the definition of DGS-reconciliation) to map domain node i to gene node j under the constraint that i' maps to l and i'' to r, and where R represents either the recipient gene node, if i is a domain-transfer event, or $R = 0$ otherwise. Thus, if $f_{(p,q)} = 1$, then i must map to j and would represent a valid evolutionary event as long as i' and i'' map to l and r, respectively. We will show that if $f_{(p,q)} = 1$ then i' and i'' map indeed map to l and r. Suppose $p' = d_{i'}^{l',r'}, q' = g^{i'}_{j',R'}$ and $f(p', q') = 1$, and $p'' = d_{i''}^{l'',r''}, q'' = g^{i''}_{j'',R''}$ and $f(p'', q'') = 1$. Thus, i' maps to j' and i'' maps to j''. It suffices to show that $j' = l$ and $j'' = r$. By the constraints of Equation 3, we know that $\sum_{x|(x,p)\in E(H_1)} f_{(x,p)} = 2$. However, if $j' \neq l$, then there cannot be an edge from q' to p, and so the flow along edge (p', q') would not reach p. This would necessarily imply that $\sum_{x|(x,p)\in E(H_1)} f_{(x,p)} < 2$, a contradiction. An analogous argument applies to j'', establishing that $j' = l$ and $j'' = r$.

If $i \in I(D)$ represents a domain-transfer event, then let $(p, q) \in E(H_1)$ denote the specific edge with flow 1 that corresponds to the local reconciliation at i (as shown above this edge is unique). By the constraint of Equation 7, we must have $\sum_{k\in H_2(p,q)} u_k = 1$, implying that the vertex of H_2 that corresponds to the gene-species reconciliation $\alpha_{\mathcal{G}}$ must be the only element of $H_2(p, q)$. Thus, the domain-transfer event at i must be compatible with this single gene-species mapping from H_2, which means that the donor and recipient must map to the same species tree node per $\alpha_{\mathcal{G}}$.

Finally, observe that the cost of this domain-gene reconciliation is equal to the total cost of all local reconciliations, which equals the total weight of all domain-to-gene edges with a flow value of 1, which is simply $\sum_{(p,q)\in E(H_1)}(\mathcal{W}(p, q) \times f_{(p,q)})$. □

4 ITERATIVE LINEAR PROGRAMMING SOLUTION

The problem of computing an optimal integral solution for an integer linear program is NP-hard [13] and rather than use an exact

ILP solver, which would not be scalable, we chose to use an iterative linear programming approach based on solving a series of linear programming relaxations of the ILP formulation. This approach is guaranteed to result in an optimal solution, but with no guarantee on the number of iterations required to converge to an optimal solution. However, as we discuss in detail in Section 6, we found that the number of iterations required to converge was very small in practice, making our approach highly scalable. It is worth noting that generic branch and bound techniques based on linear programming relaxations are also used by ILP solvers to compute optimal integral solutions, but our iterative linear programming approach is especially tailored for the Γ-ODGS problem and makes use of a customized branching rule and customized heuristic for computing upper bounds. Our iterative linear programming approach can be described easily as follows:

(1) Relax the ILP by removing the integrality constraint.
(2) Let *sol* denote the objective function value obtained by solving the relaxed linear program, and let $U = \{k \mid k \in V(H_2) \text{ and } u_k \neq 0\}$.
(3) If $|U| = 1$, then return the solution of the relaxed linear program. Otherwise, if $|U| > 1$, set *sol* as the lower bound.
(4) For each $k \in U$, construct a DGS reconciliation that uses the fixed gene-species mapping represented by k and compute its reconciliation cost. (This is doable efficiently in polynomial time as shown in [14]). Keep track of the lowest reconciliation cost seen and use that to update the upper bound.
(5) Remove each $k \in U$ from the solution pool. This can be done by fixing $u_k = 0$ for each $k \in U$.
(6) Iteratively run the algorithm until the lower bound is no less than the upper bound.
(7) Output the DGS reconciliation that uses the fixed gene-species mapping represented by the vertex from H_2 that yielded this upper bound.

This algorithm can also be easily extended to output all optimal gene-species mappings, rather than just one. The correctness of this algorithm follows from the claim below.

Claim 2. *The iterative linear programming algorithm described above outputs an optimal solution for the* Γ*-ODGS problem.*

Proof. It suffices to prove that the gene-species mapping represented by the vertex from H_2 that yields the final upper bound must, in fact, be an optimal gene-species mapping for the Γ-ODGS problem. We represent this vertex of H_2 by k. Given any vertex $x \in V(H_2)$, let $\mathcal{R}(x)$ denote the minimum DGS-reconciliation cost for the specific gene-species mapping represented by x. We wish to show that $k \in \arg\min_{x \in V(H_2)} \mathcal{R}(x)$.

Suppose, for contradiction, that there exists a different vertex $k' \in V(H_2)$ for which $\mathcal{R}(k') < \mathcal{R}(k)$. Now, it is not possible that k' was a element of U in any of the iterations of the algorithm, since then the final upper bound would not have been defined by vertex k. Thus, k' must have been part of the solution pool in the final iteration of the algorithm. However, then the lower bound computed in the final iteration using the LP relaxation would have been less than or equal to $\mathcal{R}(k')$. During that final iteration, the upper bound was defined by vertex k and so its value was $\mathcal{R}(k)$. Thus, in the final iteration, the lower bound would have been strictly smaller than the upper bound. This is a contradiction, since the algorithm only terminates when the lower bound becomes equal to or greater than the upper bound. □

5 DEFINING THE SEARCH SPACE Γ

A critical component of our algorithmic strategy is to define the set of candidate gene-species mappings, Γ, appropriately. This restriction on the space of possible DGS-reconciliations serves two important purposes: To ensure that computed DGS-reconciliations are biologically meaningful, and to limit the search space so that optimal solutions can be computed even for large input instances.

The main insight behind effective restriction of the search space is as follows. Under the duplication-loss reconciliation model, the most parsimonious mapping of a gene tree into a species tree is the unique Least Common Ancestor (LCA) mapping (also known as the Most Recent Common Ancestor (MRCA) mapping) [10]. A large deviation from the LCA mapping is highly implausible biologically and, in most cases, we expect the gene-species mapping in a DGS-reconciliation to be the LCA mapping itself. Even when the gene-species mapping deviates from the LCA mapping, we expect most gene nodes to follow the LCA mapping. Indeed, in DGS-reconciliation analysis of biological data [14], it was observed that 75% of the domain families resulted in a DGS-reconciliation in which the gene-species mapping was the LCA mapping, and in the remaining 25% of domain families only an average of 1.8 nodes deviated from their LCA mapping.

This insight forms the basis for our procedure to define the restricted set of gene-species mappings Γ. Specifically, for each problem instance, we identify up to 10 gene nodes that *could* deviate from their LCA mappings, fix the remaining gene nodes to their LCA mappings, and allow any of the (up to) 10 chosen nodes to map to any node along the path from the root of the species tree to LCA mapping for that node. This results in a large number of possible mappings.

The gene nodes to be chosen can be identified in various ways. For our experiments, described in the next section, we ran the dynamic programming heuristic developed in [14] five times for each domain tree and identified all gene nodes that deviated from their LCA mapping in either of the five resulting DGS-reconciliations. If the number of identified gene nodes was less than or equal to 10 then we chose all those gene nodes; otherwise, we sorted the five DGS-reconciliations by their reconciliation costs and greedily chose only 10 gene nodes. For domain trees where the gene trees had a total of less than 11 internal nodes, we simply chose all internal nodes, effectively solving the (unrestricted) ODGS problem for those domain trees. To assess if this strategy for identifying candidate gene nodes was effective, we tested our method on a subset of our data set (described in the next section) that only contained domain trees of which the corresponding gene trees had no more than 17 total leaf nodes. For the 1,769 domain trees in this subset, we ran two versions of our ILP algorithm, one in which Γ was restricted as described above, and another in which Γ was unrestricted, i.e., the (unrestricted) ODGS problem was solved exactly. We found that there were only 3 domain families for which the unrestricted DGS-reconciliation was more optimal.

An additional constraint. To further restrict the search space and impose biological realism, we limited the number of domain-losses invoked at any domain-transfer event. Specifically, given a domain node $d \in \Theta$ and its two children d' and d'' such that $(d, d') \in \Xi$, we restrict the distance between $\tau(d)$ and $\mathcal{M}^D(d')$ on the gene tree to be no more than 5 edges.

6 EXPERIMENTAL EVALUATION

For our analysis, we used a dataset of 3,761 error-corrected and rooted domain trees and 7,165 rooted gene trees from 12 fly species. This dataset was first created and used in [14] to evaluate the performance of the heuristic algorithm. The domain trees and gene trees in this dataset were constructed and error-corrected using state-of-the-art methods [14, 28, 29], and each gene tree contains at least one domain present in the domain trees.

We applied our exact algorithm to all those domain trees that did not result in more than 300,000 vertices in the H_1 graph representation and for which the set of candidate gene-species mappings, i.e., $|\Gamma|$, was no greater than 500,000. This resulted in 3,479 of the domain trees (i.e., 92.4% of the full dataset) being analyzed using our exact algorithm. In this subset, the largest domain family had 299 leaf nodes and the largest gene tree had 869 leaf nodes. Following [14], we used event cost 1 for $P^{\mathcal{G}}_{loss}$ and P^{D}_{loss}, 2 for $P^{\mathcal{G}}_{\Delta}$ and P^{D}_{Δ}, 4 for $P^{D}_{\Theta 1}$, and 6 for $P^{D}_{\Theta 2}$. The set of candidate gene-species mappings, Γ, for each domain tree, was computed using the methodology described in the previous section. We found that the size of Γ, averaged across all domain trees, was 1,038, with a low of 2 and a high of 82,944.

6.1 Results

Comparison with heuristic algorithm. To assess the impact of using our new exact algorithm for DGS reconciliation, we compared its results against those obtained by applying the current heuristic algorithm [14] on the same dataset. The heuristic algorithm for DGS reconciliation uses dynamic programming to consider candidate domain-gene reconciliations and adjusts the gene-species mapping as necessary to accommodate the domain-transfer events required by the domain-gene reconciliation. This heuristic has been shown to work well in practice, but it is unable to guarantee any kind of optimality and only generates a single DGS-reconciliations (even though there may be multiple equally optimal ones). It is also susceptible to generating reconciliations that correspond to biologically unrealistic scenarios.

In our comparative study, we focused on comparing the total DGS-reconciliation costs and on the final gene-species mapping in the computed DGS-reconciliations. We focus on only gene-species mappings simply because they represent the most stable part of any DGS-reconciliation; domain-gene mappings are much more variable (even for the same reconciliation cost) due to the presence of domain-transfer events. Furthermore, gene-species mappings are of great biological utility and interest by themselves. In the following, we refer to our new exact algorithm as the *ILP algorithm* and the heuristic algorithm as the *dynamic programming (DP) heuristic*.

We observed that the ILP algorithm outperformed the dynamic algorithm for 264 (i.e., 7.6%) of the domain families in our dataset,

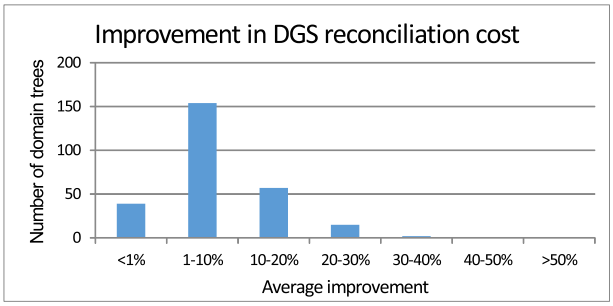

Figure 3: Distribution of average improvement in DGS-reconciliation cost for the 264 domain trees where the ILP algorithm outperformed the DP heuristic.

resulting in an average reduction of 9.4% in the DGS-reconciliation cost for these domain trees. Figure 3 shows the average DGS-reconciliation costs obtained with the ILP algorithm and DP heuristic for different domain tree sizes. For these 264 domain trees, the ILP algorithm and DP heuristic had an average of 1.82 and 1.81 nodes, respectively, that deviated from their LCA mappings. However, the two algorithms almost always chose different nodes to deviate from their LCA mappings. Thus, using the ILP algorithm has a significant impact on inferring DGS reconciliations.

In contrast, the DP heuristic, which places no restriction on the gene-species mapping, produced more optimal reconciliations for only 9 of the domain trees. In addition, we found that the ILP algorithm was unable to find a valid DGS-reconciliation for 6 domain trees; this can happen when none of the gene-species mappings in the search space Γ for a domain tree is consistent with one or more "required" domain-transfer events on that domain tree. Thus, the ILP algorithm finds either better or equally optimal DGS-reconciliations for 3,464 out of the 3,479 domain trees. This strongly suggests that even with the restriction on allowed gene-species mappings, Γ, our ILP algorithm for the Γ-OGTR problem likely recovers optimal solutions for the (unrestricted) OGTR problem. Stated differently, optimal solutions for the OGTR problem appear to be biologically plausible in most cases, further justifying our formulation and exact solution of the Γ-OGTR problem. Interestingly, we found that in all 9 instances when the DP heuristic performed better, our "additional constraint" on the number of domain-losses at any domain-transfer event (described in the previous section) was violated. This indicates that the DP heuristic can sometimes compute DGS reconciliations that are not biologically plausible even when more biologically plausible solutions exist.

Overall, we found that application of the ILP algorithm resulted in 786 (i.e., 22.6%) of the domain trees deviating from the LCA gene-species mapping.

Prevalence of multiple optima. The ILP algorithm is able to compute all optimal gene-species mappings. Since multiple optima represent equally optimal alternative evolutionary scenarios it is important to take multiple optima into account when interpreting the results of any reconciliation method, e.g. [4, 19]. On our

dataset, we found that 79 domain trees had multiple optima, with an average of 2.1 optimal gene-species reconciliations across these domain trees. Surprisingly, we noticed that the prevalence of multiple optima did not depend on the size of the domain tree or the size of Γ. Since the domain trees and gene trees in our dataset have a wide range of sizes, for different domain trees, the size of Γ varies from a low of 2 to a high of 82,944. We observed that the average number of optimal gene-species reconciliation for the 3,479 domain trees remained fairly constant at approximately 1.02 across most different sizes of the set Γ.

Overall, these results suggest that a sizeable fraction of domain trees give rise to multiple optimal gene-species reconciliations and that explicitly considering these multiple optima may be important for proper biological interpretation of the results. On the other hand, the results also demonstrate that the DGS-reconciliation model is surprisingly resistant to the presence of multiple optimal gene-species mappings.

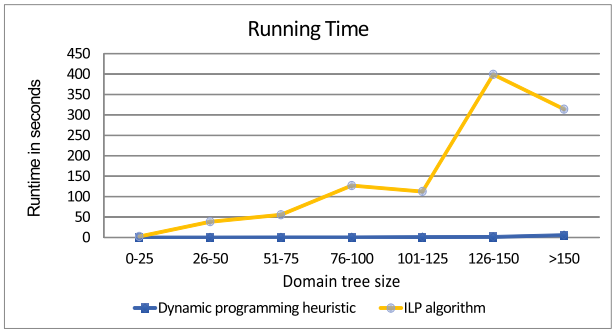

Figure 4: Average running times of the ILP algorithm and DP heuristic across different domain tree sizes. The size of a domain tree is the number of leaves in that domain tree.

6.2 Running time and scalability

We ran our algorithm using a single core on a Linux server with a 2.1 GHz Intel Xeon processor and 64 GB of main memory. To solve the linear programming problems we used the well-known LP solver CPLEX (free for academic use). As Figure 4 shows, the running time of our algorithm increases roughly linearly with domain size. Overall, our exact algorithm is remarkably scalable and, as mentioned previously, we were able to analyze 92.4% of the full dataset (3,479 out of 3,761 domain trees) using this algorithm when the set of candidate gene-species mappings, Γ, was restricted as defined earlier. Even on domain trees with over 150 leaves, the algorithm required less than 6 minutes of running time on average. The largest domain family we analyzed had 299 leaf nodes and the largest gene tree had 869 leaf nodes. However, there were also several much smaller domain trees that we were unable to analyze. For example, of the 2,718 domain trees with at most 30 leaf nodes, we were unable to analyze 8. The time and space complexity of our approach in practice depends not only on the sizes of the domain tree and gene trees but also on the level of discordance between them. In particular, the sizes of the graphs H_1 and H_2 can increase rapidly as the topology of the domain tree becomes more and more "inconsistent" with that of its gene trees, leading to a large number of constraints in the ILP formulation. Still, as these results on real biological data show, the ILP algorithm runs very efficiently in most cases.

We also observed that the iterated linear programming algorithm almost always converged to the optimal solution in a very small number of iterations. For the 3,479 domain trees in our dataset, the average number of iterations was only 1.22, with the optimal solution identified in only a single iteration for a majority of the domain trees. Furthermore, the number of iterations was ten or greater for only 45 of the domain families. This shows that our iterated linear programming technique for solving the ILP formulation is extremely effective in practice.

For the unrestricted version of the problem, where Γ contains all possible gene-species mappings, we were still able to compute exact solutions (but with the additional constraint on domain-losses) for the 1,764 domain trees whose gene trees had up to 17 total leaf nodes. Analyzing the trees in this subset took an average of only 1.98 seconds per tree, with a maximum of 6 minutes for one of the trees.

7 DISCUSSION AND CONCLUSION

In this work, we have introduced the first exact algorithm for the NP-hard DGS-reconciliation problem. Our exact algorithm uses an integer linear programming formulation of the problem, which we then solve using an iterative algorithm based on solving a series of linear programming relaxations of the original ILP. Our exact algorithm has several important advantages over the current heuristic algorithm. First, it can compute optimal DGS-reconciliations. Second, it makes it easy to impose restrictions on allowable gene-species mappings, which makes it possible to restrict the space of candidate DGS-reconciliations to those that are biologically meaningful. And third, it outputs all optimal DGS-reconciliations in the restricted or unrestricted search space. Instead of using an ILP solver to directly solve the ILP formulation, which would work for only small problem instances, we developed an iterative algorithm based on solving a series of linear programming relaxations of the original ILP. This iterative algorithm is guaranteed to solve the ILP formulation exactly, and is often scalable even to domain trees with hundreds of leaves.

Our exact algorithm can be used in several different ways: (i) it can be used to compute exact solutions under a biologically meaningful restriction of the search space, (ii) it can be used to improve the results of the heuristic algorithm by including the heuristic solution in the set of candidate gene-species mappings Γ, and (iii) it can be used without any restrictions on the search space, by including all possible gene-species mappings in Γ, to compute exact solutions for the ODGS problem when the input instances are small. A comparison of results from our exact algorithm and from the heuristic algorithm shows that the exact algorithm is able to frequently outperform the heuristic even when the search space (defined by Γ) is restricted, and that a significant number of domain trees have non-unique optimal gene-species mappings. It also

shows that when input instances are too large for the exact algorithm, the heuristic offers a good tradeoff between scalability and accuracy.

A limitation of the existing DGS reconciliation framework and our ILP algorithm is that they reconcile each domain tree independently. Since a single gene family often contains multiple domains, it would be useful to extended the DGS reconciliation framework and ILP approach to simultaneously reconcile multiple domain trees with multiple gene trees and species tree. It would also be very informative to systematically assess the accuracy, strengths, and weaknesses of the DGS reconciliation framework using realistic simulated data. A suitable framework for simulating domain trees inside gene trees and species trees does not yet exist, but is currently under development.

ACKNOWLEDGMENTS

Funding: This work was supported in part by National Science Foundation CAREER award 1553421 to MSB.

REFERENCES

[1] Örjan Äkerborg, Bengt Sennblad, Lars Arvestad, and Jens Lagergren. 2009. Simultaneous Bayesian gene tree reconstruction and reconciliation analysis. *P. Natl. Acad. Sci. USA* 106, 14 (2009), 5714–5719. https://doi.org/10.1073/pnas.0806251106

[2] S. Arena, S. Benvenuti, and A. Bardelli. 2005. Genetic analysis of the kinome and phosphatome in cancer. *Cellular and Molecular Life Sciences* 62, 18 (2005), 2092–2099. https://doi.org/10.1007/s00018-005-5205-1

[3] Mukul S. Bansal, Eric J. Alm, and Manolis Kellis. 2012. Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. *Bioinformatics* 28, 12 (2012), 283–291.

[4] Mukul S. Bansal, Eric J. Alm, and Manolis Kellis. 2013. Reconciliation Revisited: Handling Multiple Optima when Reconciling with Duplication, Transfer, and Loss. *Journal of Computational Biology* 20, 10 (2013), 738–754.

[5] Behshad Behzadi and Martin Vingron. 2006. Reconstructing Domain Compositions of Ancestral Multi-domain Proteins. In *Comparative Genomics*, Guillaume Bourque and Nadia El-Mabrouk (Eds.). Lecture Notes in Computer Science, Vol. 4205. Springer Berlin Heidelberg, 1–10. https://doi.org/10.1007/11864127_1

[6] P. Bonizzoni, G. D. Vedova, and R. Dondi. 2005. Reconciling a gene tree to a species tree under the duplication cost model. *Theor. Comput. Sci.* 347, 1-2 (2005), 36–53. https://doi.org/10.1016/j.tcs.2005.05.016

[7] Jean-Philippe Doyon, Celine Scornavacca, K. Yu. Gorbunov, Gergely J. Szöllosi, Vincent Ranwez, and Vincent Berry. 2010. An Efficient Algorithm for Gene/Species Trees Parsimonious Reconciliation with Losses, Duplications and Transfers. In *RECOMB-CG (Lecture Notes in Computer Science)*, Eric Tannier (Ed.), Vol. 6398. Springer, 93–108.

[8] Diana Ekman, Asa K. Bjorklund, Johannes Frey-Skott, and Arne Elofsson. 2005. Multi-domain Proteins in the Three Kingdoms of Life: Orphan Domains and Other Unassigned Regions. *Journal of Molecular Biology* 348, 1 (2005), 231 – 243. https://doi.org/10.1016/j.jmb.2005.02.007

[9] M. Goodman, J. Czelusniak, G. W. Moore, A. E. Romero-Herrera, and G. Matsuda. 1979. Fitting the gene lineage into its species lineage. A parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Zoology* 28 (1979), 132–163.

[10] Pawel Górecki and Jerzy Tiuryn. 2006. DLS-trees: A model of evolutionary scenarios. *Theor. Comput. Sci.* 359 (2006), 378–399.

[11] Jung-Hoon Han, Sarah Batey, Adrian A. Nickson, Sarah A. Teichmann, and Jane Clarke. 2007. The folding and evolution of multidomain proteins. *Nature Reviews Molecular Cell Biology* 8 (2007), 319–330.

[12] Jacob M. Joseph and Dannie Durand. 2009. Family classification without domain chaining. *Bioinformatics* 25, 12 (2009), i45–i53. https://doi.org/10.1093/bioinformatics/btp207

[13] Richard M. Karp. 1972. *Reducibility among Combinatorial Problems*. Springer US, Boston, MA, 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9

[14] Lei Li and Mukul S. Bansal. In Press. An Integrated Reconciliation Framework for Domain, Gene, and Species Level Evolution. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (In Press).

[15] B. Ma, M. Li, and L. Zhang. 2000. From Gene Trees to Species Trees. *SIAM J. Comput.* 30, 3 (2000), 729–752. https://doi.org/10.1137/S0097539798343362

[16] Takashi Miyata and Hiroshi Suga. 2001. Divergence pattern of animal gene families and relationship with the Cambrian explosion. *BioEssays* 23, 11 (2001), 1018–1027.

[17] Andrew D. Moore, Asa K. Bjorklund, Diana Ekman, Erich Bornberg-Bauer, and Arne Elofsson. 2008. Arrangements in the modular evolution of proteins. *Trends in Biochemical Sciences* 33 (2008), 444–451.

[18] R. D. M. Page. 1994. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst. Biol.* 43, 1 (1994), 58–77.

[19] Celine Scornavacca, Wojciech Paprotny, Vincent Berry, and Vincent Ranwez. 2013. Representing a set of reconciliations in a compact way. *Journal of Bioinformatics and Computational Biology* 11, 02 (2013), 1250025. https://doi.org/10.1142/S0219720012500254

[20] Joel Sjostrand, Ali Tofigh, Vincent Daubin, Lars Arvestad, Bengt Sennblad, and Jens Lagergren. 2014. A Bayesian Method for Analyzing Lateral Gene Transfer. *Systematic Biology* 63, 3 (2014), 409–420. https://doi.org/10.1093/sysbio/syu007 arXiv:http://sysbio.oxfordjournals.org/content/63/3/409.full.pdf+html

[21] Maureen Stolzer, Han Lai, Minli Xu, Deepa Sathaye, Benjamin Vernot, and Dannie Durand. 2012. Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. *Bioinformatics* 28, 18 (2012), 409–415.

[22] Maureen Stolzer, Katherine Siewert, Han Lai, Minli Xu, and Dannie Durand. 2015. Event inference in multidomain families with phylogenetic reconciliation. *BMC Bioinformatics* 16, 14 (2015), S8. https://doi.org/10.1186/1471-2105-16-S14-S8

[23] Gergely J. Szollosi, Bastien Boussau, Sophie S. Abby, Eric Tannier, and Vincent Daubin. 2012. Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. *Proceedings of the National Academy of Sciences* 109, 43 (2012), 17513–17518. https://doi.org/10.1073/pnas.1202997109 arXiv:http://www.pnas.org/content/109/43/17513.full.pdf

[24] Ali Tofigh, Michael T. Hallett, and Jens Lagergren. 2011. Simultaneous Identification of Duplications and Lateral Gene Transfers. *IEEE/ACM Trans. Comput. Biology Bioinform.* 8, 2 (2011), 517–535.

[25] Hedvig Tordai, Alinda Nagy, Krisztina Farkas, Laszlo Banyai, and Laszla Patthy. 2005. Modules, multidomain proteins and organismic complexity. *FEBS Journal* 272, 19 (2005), 5064–5078. https://doi.org/10.1111/j.1742-4658.2005.04917.x

[26] Christine Vogel, Matthew Bashton, Nicola D Kerrison, Cyrus Chothia, and Sarah A Teichmann. 2004. Structure, function and evolution of multidomain proteins. *Current Opinion in Structural Biology* 14, 2 (2004), 208 – 216. http://www.sciencedirect.com/science/article/pii/S0959440X04000454

[27] John Wiedenhoeft, Roland Krause, and Oliver Eulenstein. 2011. The Plexus Model for the Inference of Ancestral Multidomain Proteins. *IEEE/ACM Trans. Comp. Biol. Bioinf.* 8, 4 (2011), 890–901. https://doi.org/10.1109/TCBB.2011.22

[28] Yi-Chieh Wu, Mukul S Bansal, Matthew D Rasmussen, Javier Herrero, and Manolis Kellis. 2014. Phylogenetic Identification and Functional Characterization of Orthologs and Paralogs across Human, Mouse, Fly, and Worm. *bioRxiv* (2014). https://doi.org/10.1101/005736 arXiv:http://biorxiv.org/content/early/2014/05/31/005736.full.pdf

[29] Yi-Chieh Wu, Matthew D. Rasmussen, Mukul S. Bansal, and Manolis Kellis. 2013. TreeFix: statistically informed gene tree error correction using species trees. *Systematic Biology* 62, 1 (2013), 110–120.

[30] Yi-Chieh Wu, Matthew D. Rasmussen, and Manolis Kellis. 2012. Evolution at the Subgene Level: Domain Rearrangements in the Drosophila Phylogeny. *Molecular Biology and Evolution* 29, 2 (2012), 689–705. https://doi.org/10.1093/molbev/msr222