# Simultaneous Multi-Domain-Multi-Gene Reconciliation under the Domain-Gene-Species Reconciliation Model

Lei Li and Mukul S. Bansal

Dept. of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA.
lei.li@uconn.edu, mukul.bansal@uconn.edu

**Abstract.** The recently developed Domain-Gene-Species (DGS) reconciliation framework, which jointly models the evolution of a domain family inside one or more gene families and the evolution of those gene families inside a species tree, represents one of the most powerful computational techniques for reconstructing detailed histories of domain and gene family evolution in eukaryotic species. However, the DGS reconciliation framework allows for the reconciliation of only a single domain tree (representing a single domain family present in one or more gene families from the species under consideration) at a time, i.e., each domain tree is reconciled separately without consideration of any other domain families that might be present in the gene trees under consideration. However, this can lead to conflicting gene-species reconciliations for gene trees containing multiple domain families.

In this work, we address this problem by extending the DGS reconciliation model to simultaneously reconcile a set of domain trees, a set of gene trees, and a species tree. The new model, which we call the *multi-DGS (mDGS) reconciliation model*, produces a consistent joint reconciliation showing the evolution of each domain tree in its corresponding gene trees and the evolution of each gene tree inside the species tree. We formalize the mDGS reconciliation framework and define the associated computational problem, provide a heuristic algorithm for estimating optimal mDGS reconciliations (both the DGS and mDGS reconciliation problems are NP-hard), and apply our algorithm to a large dataset of over $3,800$ domain trees and over $7,100$ gene trees from 12 fly species. Our analysis of this dataset reveals interesting underlying patterns of co-occurrence of domains and genes, demonstrates the importance of mDGS reconciliation, and shows that the proposed heuristic is effective at estimating optimal mDGS reconciliations.

## 1 Introduction

Most eukaryotic genes are known to contain one or more protein domains [2, 4] and it is well understood that the domain content of genes can change over time due to evolutionary events such as domain duplications, transfers, or losses [8]. Changes in the domain content of genes have important functional consequences [11, 12] and it is therefore important to reconstruct the history of these changes in the evolution of gene families. Several methods have been developed for studying the evolution of domain families (or domain trees), but these methods either do not take gene trees into account [1, 13, 15] or do not account for the inter-dependence of domain, gene, and species level evolution [10].

The recently developed Domain-Gene-Species (DGS) reconciliation framework [6, 7], which jointly models the evolution of a domain family inside one or more gene families and the evolution of those gene families inside a species tree, represents one of the most powerful computational techniques for reconstructing detailed histories of domain and gene family evolution in eukaryotic species. However, the DGS reconciliation framework allows for the reconciliation of only a single domain tree (representing a single domain family present in one or more gene families from the species under consideration) at a time, i.e., each domain tree is reconciled separately without consideration of any other domain families that might be present in the gene trees under consideration. This poses a problem since many gene families (or gene trees) have multiple protein domains; specifically, solving the DGS reconciliation problem on different domain trees that are represented in the same gene tree can yield conflicting reconciliations for that gene tree with the species tree.

**Our contributions.** In this work, we address this problem by extending the DGS reconciliation model to simultaneously reconcile a set of domain trees, a set of gene trees, and a species tree. The new model, which we call the *multi-DGS (mDGS) reconciliation model*, produces a consistent joint reconciliation showing the evolution of each domain tree in its corresponding gene trees and the evolution of each gene tree inside the species tree. We formalize the mDGS reconciliation framework and define the associated computational problem, provide a heuristic algorithm for estimating optimal mDGS reconciliations (both the DGS and mDGS reconciliation problems are NP-hard), and apply our algorithm to a large dataset of over $3,800$ domain trees and over $7,100$ gene trees from 12 fly species. Our experimental results demonstrate the importance of mDGS reconciliation and show that the proposed heuristic is effective at estimating optimal mDGS reconciliations. We also develop a technique to further improve the accuracy of mDGS reconciliation by using appropriately chosen subsets of the domain and gene trees under consideration and provide a clustering algorithm to find such subsets. An implementation of our heuristic for mDGS reconciliation is available freely from `https://compbio.engr.uconn.edu/software/seadog/`.

## 2 Definitions and Preliminaries

We follow the notation and basic definitions from [6, 7].

**Preliminaries.** Throughout this manuscript, the term *tree* refers to rooted binary trees. Given a tree $T$, we denote its node, edge, and leaf sets by $V(T)$, $E(T)$, and $\mathit{Le}(T)$ respectively. The root node of $T$ is denoted by $\mathit{rt}(T)$, the parent of a node $v \in V(T)$ by $\mathit{pa}_T(v)$, its set of children by $\mathit{Ch}_T(v)$, 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 $V(T) \setminus \mathit{Le}(T)$. 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 $\mathit{rt}(T)$ 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 $\mathit{rt}(T)$ 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 $L \subseteq \mathit{Le}(T)$, we denote by $\mathit{lca}_T(L)$ 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 mDGS reconciliation is a collection of domain trees $\mathcal{D}$, 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, 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 mDGS reconciliation, we require that the collections $\mathcal{D}$ and $\mathcal{G}$ be "complete", in the sense that all gene families represented in any domain tree from $\mathcal{D}$ should be present as a gene tree in $\mathcal{G}$ and all domain families represented in any gene tree of $\mathcal{G}$ should be present as a domain tree in $\mathcal{D}$. We refer to any such "complete" pair of collections $\mathcal{D}$ and $\mathcal{G}$ as a *DG-group*. Essentially, a DG-group can be viewed as a connected component in a bipartite graph where the node set corresponds to all domain families and all gene families present in the species under consideration and an edge connects a domain family node and a gene family node if a domain from that domain family exists in a gene from that gene family.

As in DGS reconciliation [6,7], 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.

For convenience, we extend the notions of the leaf set, vertex set, and edge set of a tree as follows: $Le(\mathcal{G}) = \cup_{G\in\mathcal{G}} Le(G)$, $V(\mathcal{G}) = \cup_{G\in\mathcal{G}} V(G)$, and $E(\mathcal{G}) = \cup_{G\in\mathcal{G}} E(G)$. And $Le(\mathcal{D}) = \cup_{D\in\mathcal{D}} Le(D)$, $V(\mathcal{D}) = \cup_{D\in\mathcal{D}} V(D)$, and $E(\mathcal{D}) = \cup_{D\in\mathcal{D}} E(D)$.

**mDGS reconciliation.** The *multi-Domain-Gene-Species (mDGS)* reconciliation model defines what constitutes a valid joint reconciliation of the given gene trees with the species tree and of the given domain trees with the gene trees. As with DGS reconciliation, mDGS reconciliation models the primary evolutionary events that shape gene family evolution in multicellular eukaryotes: *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: *co-divergence*, *domain transfer*, *domain duplication*, and *domain loss*. Formally:

**Definition 1 (mDGS-reconciliation).** *Given a collection of domain trees $\mathcal{D}$ and a collection of gene trees $\mathcal{G}$ that form a DG-group, and given a species tree $S$ and leaf-mappings $\mathcal{L}^{\mathcal{D}}\colon Le(\mathcal{D}) \to Le(\mathcal{G})$ and $\mathcal{L}^{\mathcal{G}}\colon Le(\mathcal{G}) \to Le(S)$, an mDGS reconciliation for $\mathcal{D}$, $\mathcal{G}$, and $S$ is a nine-tuple $\langle \mathcal{M}^{\mathcal{D}}, \mathcal{M}^{\mathcal{G}}, \Sigma^{\mathcal{D}}, \Sigma^{\mathcal{G}}, \Delta^{\mathcal{D}}, \Delta^{\mathcal{G}}, \Theta, \Xi, \tau\rangle$, where $\mathcal{M}^{\mathcal{D}}\colon V(\mathcal{D}) \to V(\mathcal{G})$ and $\mathcal{M}^{\mathcal{G}}\colon V(\mathcal{G}) \to V(S)$ map each node of $\mathcal{D}$ to a node from $\mathcal{G}$ and each node from $\mathcal{G}$ to a node of $S$, respectively, the sets $\Sigma^{\mathcal{D}}$, $\Delta^{\mathcal{D}}$, and $\Theta$ partition $I(\mathcal{D})$ 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, respec-*

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

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 [3, 9] 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. Overall, the mDGS reconciliation model is nearly identical to the DGS reconciliation model [6, 7], except that we reconcile multiple domain trees instead of just one. We refer the reader to [7] for a detailed explanation of the underlying model and of each constraint. Figure 1 shows an example of a valid mDGS reconciliation.

We point out that the interdependence between domain-gene and gene-species reconciliations stems from Constraint 6d, which specifies which genes may be designated as the recipient gene for any given domain-transfer event. 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. It is this relationship between gene-species mappings and domain-transfer events that necessitates the computation of a *joint* reconciliation, so that one cannot simply compute optimal DGS or mDGS reconciliations by optimizing domain-gene and gene-species reconciliations independently. It is also important to note that mDGS reconciliation is not a direct generalization of the DGS problem since mDGS reconciliation requires $\mathcal{D}$ and $\mathcal{G}$ to form a DG-group. Valid input instances for DGS reconciliation

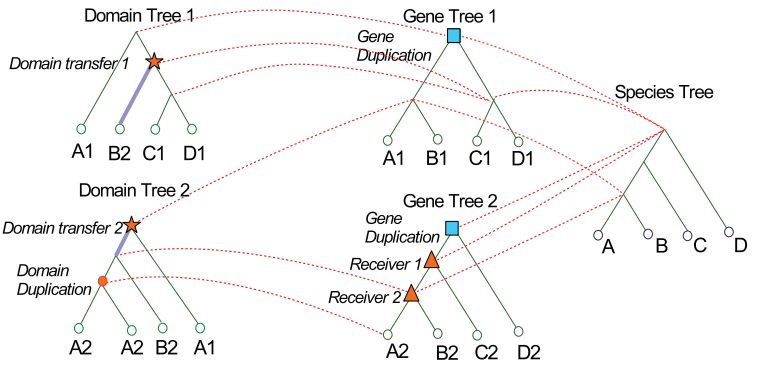


Fig. 1: The figure shows an mDGS reconciliation for two domain trees, two gene trees, and a species tree on 4 taxa. The mappings of the domain trees into the gene trees and of the gene trees into the species tree are shown by the dotted red lines. Domain-gene leaf associations are specified by shared leaf labels, and gene-species leaf associations are specified by shared letters ($A$, $B$, $C$, or $D$). 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, two domain transfer events are invoked at the nodes with the orange star, one in domain tree 1 and one in domain tree 2, and duplication event is invoked at the node with the orange circle in domain tree 2. The bolded edges in the domain trees represent the domain-transfer edges; in both domain trees the domains are copied from gene tree 1 to gene tree 2, and the recipient genes for domain transfer 1 and domain transfer 2 are marked as "receiver 1" and "receiver 2", respectively. As required by the model, for both transfer events, the donor gene and recipient gene both map to the same species tree node.

may therefore not be valid input instances for mDGS reconciliation. In the remainder of this paper we assume that $\mathcal{D}$ and $\mathcal{G}$ form a DG-group.

We define a parsimony based problem formulation for finding an optimal mDGS reconciliation. Thus, each evolutionary event other than speciation and co-divergence is assigned a positive cost, and the computational objective is to find an mDGS reconciliation of minimum total cost. $P^{\mathcal{G}}_{\Delta}$ and $P^{\mathcal{G}}_{loss}$ denote the gene-duplication and gene-loss costs, while $P^{\mathcal{D}}_{\Delta}$, $P^{\mathcal{D}}_{\Theta}$, and $P^{\mathcal{G}}_{loss}$ denote domain-duplication, domain-transfer, and domain-loss costs. The model allows for the use of two separate costs $P^{\mathcal{D}}_{\Theta 1}$ and $P^{\mathcal{D}}_{\Theta 2}$ instead of a single $P^{\mathcal{D}}_{\Theta}$, so that a distinction can be made between domain transfers that remain within the same gene family from those that cross gene family boundaries.

**Definition 2 (Reconciliation cost).** *Given an mDGS reconciliation $\alpha$, the* reconciliation cost *for $\alpha$ is the total cost of all events invoked by $\alpha$.*

Note that, while domain-duplication, domain-transfer, and gene-duplication events are directly specified in the mDGS reconciliation, domain-losses and gene-losses are not. However, given an mDGS reconciliation, one can directly count the minimum number of gene-losses and domain-losses implied by the reconciliation as shown in [7].

**Definition 3 (Optimal mDGS Reconciliation Problem).** *Given $\mathcal{D}$, $\mathcal{G}$ and $S$, along with $P^{\mathcal{G}}_{\Delta}$, $P^{\mathcal{G}}_{loss}$, $P^{\mathcal{D}}_{\Delta}$, $P^{\mathcal{D}}_{\Theta 1}$, $P^{\mathcal{D}}_{\Theta 2}$, and $P^{\mathcal{D}}_{loss}$, the* Optimal mDGS Reconciliation *problem is to find an mDGS reconciliation for $\mathcal{D}$, $\mathcal{G}$ and $S$ with minimum reconciliation cost.*

The NP-hardness of the optimal mDGS reconciliation problem follows from the NP-hardness proof for optimal DGS reconciliation [7]. Specifically, even though mDGS reconciliation is not a direct generalization of DGS reconciliation, the gadget used in [7] yields a valid input instance (i.e., the domain tree and gene trees form a DG-group) for the optimal mDGS reconciliation problem as well.

## 3 A heuristic for optimal mDGS reconciliation problem

Algorithms for DGS reconciliation cannot be used for computing mDGS reconciliations due to differences in the problem formulation and final objective. However, optimal DGS reconciliations may still serve as a good starting point for computing optimal mDGS reconciliations (we demonstrate this later in our experiments). Our proposed heuristic is based on this idea and we show how to modify an existing algorithm for DGS reconciliation to estimate optimal mDGS reconciliations.

Currently, two algorithms exist for DGS reconciliation problem: An efficient dynamic programming based heuristic algorithm from [7], and an exact integer linear programming (ILP) based algorithm from [6]. Since, problem instances for mDGS recnciliation are generally much larger (more domain trees and gene trees) than those for DGS reconciliation, the exact ILP based algorithm is not well-suited. We therefore focused on extending the efficient dynamic programming based heuristic algorithm from [7] which has also been previously shown to compute optimal DGS reconciliations (i.e., same as those computed using the exact ILP approach) in the vast majority of test cases [6]. We will refer to this dynamic programming heuristic as the *DGS-algorithm*. We refer the reader to [7] for a complete description of the DGS-algorithm; however, for the current discussion it suffices to view it as a black box that estimates optimal DGS reconciliations. The DGS-algorithm takes as input a single domain tree $D$, set of associated gene trees $\mathcal{G}$, and a species tree $S$ for the species under consideration. The output of the algorithm is a domain-gene reconciliation of $D$ with $\mathcal{G}$ and gene-species reconciliations for each $G \in \mathcal{G}$ with $S$ (with the domain-gene reconciliation satisfying the constraints imposed on it by the gene-species reconciliations, and vice versa).

Observe that algorithms for DGS reconciliation cannot be used for computing mDGS reconciliations since reconciling each domain tree of $\mathcal{D}$ individually may lead to conflicting gene-species reconciliations for one or more gene trees. This is illustrated by the DGS reconciliations shown in Figure A1 in the Appendix, which shows the two separate DGS reconciliations for the two domain trees from Figure 1. As shown in Figure A1, DGS reconciliations for the two domain trees assign different mappings for the parent of node C2 in gene tree 2. Our heuristic for mDGS reconciliation, which we will refer to as the *mDGS-algorithm*, identifies such conflicts and resolves them. In particular, it preserves the domain-gene reconciliations inferred through DGS reconciliation, but adjusts any conflicting gene-species mappings to create a single gene-species mapping for each gene tree. Before we describe the algorithm in detail, we need the following

notation: Given any gene tree $G \in \mathcal{G}$, let $\mathcal{D}_G$ be the set containing those domain trees from $\mathcal{D}$ that are represented in $G$. Analogously, given any domain tree $D \in \mathcal{D}$, let $\mathcal{G}_D$ denote the set containing exactly those gene trees from $\mathcal{G}$ that are represented in $D$.

*mDGS-algorithm* $(\mathcal{D}, \mathcal{G}, S, \mathcal{L}^{\mathcal{D}}, \mathcal{L}^{\mathcal{G}})$

1. For each domain tree $D \in \mathcal{D}$
   (a) Run *DGS-algorithm*$(D, \mathcal{G}_D, S, \mathcal{L}^D, \mathcal{L}^{\mathcal{G}_D})$. This yields a gene-species mapping for each $G \in \mathcal{G}_D$.
2. For each gene tree $G \in \mathcal{G}$
   (a) Consider the (up to) $|\mathcal{D}_G|$ different gene-species mapping for $G$ generated above. Let these mapping be denoted by $\mathcal{M}^G_1, \ldots, \mathcal{M}^G_{|\mathcal{D}_G|}$.
   (b) For each $g \in I(G)$ in post order, let $\mathcal{M}^G(g) = \mathit{lca}(\mathcal{M}^G(g'), \mathcal{M}^G(g''), \mathcal{M}^G_1(g), \ldots, \mathcal{M}^G_{|\mathcal{D}_G|}(g))$, where $g'$ and $g''$ denote the two children of $g \in V(G)$.
3. For each domain tree $D \in \mathcal{D}$
   (a) For each transfer event $d$ in a post-order traversal of $D$
      i. Let $g$ and $g'$ denote the donor and recipient gene nodes for the transfer event at $d$, and let $G$ and $G'$ denote the gene trees containing $g$ and $g'$, respectively.
      ii. If $\mathcal{M}^G(g) \neq \mathcal{M}^{G'}(g')$ then $\mathcal{M}^G(g) = \mathcal{M}^{G'}(g') = \mathit{lca}(\mathcal{M}^G(g), \mathcal{M}^{G'}(g'))$.
4. Repeat Steps 2 and 3 above until no further changes are made to $\mathcal{M}^{\mathcal{G}}$.
5. Return the domain-gene reconciliation for each $D \in \mathcal{D}$ as computed in Step 1, and the gene-species reconciliation $\mathcal{M}^G$ for each $G \in \mathcal{G}$ as computed above.

It is easy to see that this heuristic is guaranteed to yield a valid mDGS reconciliation. It is also not difficult to show that, after the initial runs of DGS-algorithm in Step 1, the heuristic above requires no more than $O((m \times n \times |\mathit{Le}(S)|)$ time, where $m$ is the total number of leaves in all domain trees of $\mathcal{D}$ and $n$ is the total number of leaves in all gene trees of $\mathcal{G}$. We found the heuristic to be very efficient in practice, requiring less than an hour to run on our entire dataset of $3,847$ domain trees and $7,165$ gene trees from 12 species (described in the next section) using a single core on a desktop computer.

**Empirical justification.** Observe that the mDGS-algorithm resolves conflicts by simply taking their least common ancestor in case of conflicting mappings for the same gene node. Despite its simplicity, this algorithm is expected to work well if (i) the number of gene nodes that are assigned conflicting mappings under different domain trees is small, and/or (ii) for the gene nodes that do have conflicting mappings, those conflicting mappings are close together on the species tree. This is exactly what we find in our empirical data analysis. Specifically, we find that different domain trees are remarkably consistent in their gene-species mappings under DGS reconciliation and only a very small fraction of gene nodes had conflicting mappings that had to be resolved by the mDGS-algorithm. These results appear in the next section.

## 4 Experiments and Results

**Dataset.** To experimentally study the impact of using mDGS reconciliation instead of DGS reconciliation, we used a biological dataset of $3,847$ rooted domain trees and

$7,165$ rooted gene trees from $12$ fly species. This dataset was first created and used in [7] to evaluate the performance of the heuristic algorithm for DGS reconciliation and was subsequently also used in [6]. The domain trees and gene trees in this dataset were constructed and error-corrected using state-of-the-art methods [7, 14], and each gene tree contains at least one domain present in the domain trees. On average, each gene in the dataset contains $1.4$ domains, each gene family contains $1.68$ domain families, and each domain family is associated with $2.93$ gene families.

**Structure of DG-groups.** We first computed all DG-groups on our dataset and studied their structural properties. We found that the $3,847$ domain trees and $7,165$ gene trees could be partitioned into $2,010$ DG-groups. Among these, $1,241$ DG-groups consist of a single domain tree and single gene tree, and $386$ DG-groups has a single domain tree but at least two gene trees. Note that, for these two types of DG-groups, using mDGS reconciliation is the same as using DGS reconciliation. The remaining $383$ DG-groups each had multiple domain trees and we refer to these as *complex DG-groups*. Among the $383$ complex DG-groups, $149$ had a single gene tree and $234$ had multiple gene trees. One of the complex DG-groups is extremely large and contains $1,205$ domain trees and $2,394$ gene trees, constituting almost one-third of the entire dataset. For the remaining $382$ complex DG-groups the average number of domain and gene trees is $2.74$ and $2.85$, respectively, with the largest DG-group having $15$ domain trees and $23$ gene trees.

Among the $2,220$ domain trees in the $383$ complex DG-groups, $1,032$ evolve inside only one gene tree and the others in multiple gene trees, including $239$ that evolve inside more than five. Likewise, among the $3418$ gene trees in these DG-groups, $1,823$ are associated with only one domain tree, $1,061$ with two, and only $61$ gene trees are associated with more than $5$ domain trees.

**Impact of mDGS reconciliation.** We applied our mDGS-algorithm on the $383$ complex DG-groups and compared the resulting gene-species reconciliations with those inferred through DGS reconciliation. We observed that gene-species mappings inferred through DGS reconciliation are highly consistent in general, but that there are several gene nodes for which different domain trees imply conflicting gene-species mappings. Overall, we found there were $12,201$ internal gene tree nodes that were assigned gene-species mappings by at least two domain trees, and among these gene nodes $148$ were assigned conflicting mappings. Thus, only a small fraction of the total of $66,854$ internal gene tree nodes present in the $383$ complex DG-groups was assigned conflicting mappings. This shows that, in the vast majority of cases, optimal mDGS reconciliations are composed of optimal DGS reconciliations.

We also found that the mDGS-algorithm rectified these conflicts without significantly increasing the total gene-species reconciliation cost or significantly affecting other conflict-free gene-species mappings. Specifically, in the largest DG-group the total gene-species reconciliation cost for the $2,394$ gene trees increased by only $4.6\%$ compared to DGS reconciliation, and total number of gene nodes that deviate from their LCA (least common ancestor) mapping increased by only $294$ (increased from $501$ to $795$) among a total of $46,693$ total internal gene nodes. These are very small numbers considering that there are $6,577$ domain transfer events in the largest DG-group. Similarly, in the remaining $382$ DG-groups, the total gene-species reconciliation

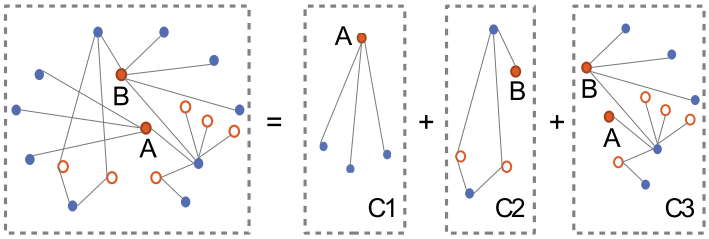


Fig. 2: This figure shows how the DG-group on the left can be split into three smaller DG-communities. Blue dots represent domain trees. Solid orange circles (labeled $A$ and $B$) represent connecting gene trees, and hollow orange circles represent other gene trees. As shown, each DG-community is connected to at least one other DG-community through connecting gene trees.

cost for the $1,024$ gene trees increased by only $3.47\%$ compared to DGS reconciliation, and total number of gene nodes that deviate from their LCA mapping increased by only $51$ (increased from $106$ to $157$) among a total of $20,161$ total internal gene nodes. The total number of domain transfers in these DG-groups was $1,786$.

**Splitting large DG-groups into smaller communities.** As seen in our dataset, DG-groups can become extremely large, comprising of thousands of domain trees and gene trees. Upon closer inspection of the largest complex DG-group in our dataset (with $1,205$ domain trees and $2,394$ gene trees), we found that it is composed of many small well-connected communities of domain and gene trees, with different communities connected to each other through small numbers of shared gene trees. We refer to these communities within a larger DG-group as *DG-communities*, and gene trees that "connect" different DG-communities as *connecting gene trees*. Figure 2 illustrates how a larger DG-group can be split into smaller DG-communities connected through connecting gene trees.

To systematically identify DG-communities within large DG-groups and study their relevance, we devised a simple algorithm for identifying clusters in bipartite graphs. While many clustering algorithms exist for bipartite graphs, we found that these could not be directly applied for identifying DG-communities since most clustering algorithms seek to partition the set of nodes into distinct clusters (effectively by deleting edges). This would not work in the current setting since we wish to retain all domain-gene edges and some gene trees must therefore appear in multiple DG-communities.

Our new clustering algorithm is specifically designed for identifying DG-communities. It partitions all domain trees into different DG-communities, but allows some gene trees to appear in multiple DG-communities. The algorithm makes use of a similarity measure between domain trees to do the clustering and we define this similarity in a manner that is meaningful for detecting DG-communities. Specifically, given domain trees $D_1$ and $D_2$, we define the similarity between them, denoted $sim(D_1, D_2)$, as follows:

$$sim(D_1, D_2) = \frac{|\mathcal{G}_{D_1} \cap \mathcal{G}_{D_2}|}{|\mathcal{G}_{D_1}|} + \frac{|\mathcal{G}_{D_1} \cap \mathcal{G}_{D_2}|}{|\mathcal{G}_{D_2}|}. \quad (1)$$

A high-level description of the proposed clustering algorithm follows. In addition to $\mathcal{D}$ and $\mathcal{G}$, the algorithm takes as input a clustering parameter $\rho$.

*Find-Communities* $(\mathcal{D}, \mathcal{G}, \mathcal{L}^{\mathcal{D}}, \rho)$

1. Compute $sim(D_1, D_2)$ for each pair of domain trees $D_1, D_2 \in \mathcal{D}$.
2. Initialize the set *pool* to include all domain trees in $\mathcal{D}$.
3. While $|pool| \geq 2$ and $\max_{D_1, D_2 \in pool} sim(D_1, D_2) \geq \rho$.
   (a) Choose a pair of domain trees from *pool* with greatest similarity and create a new community with that pair. Add all gene trees associated with the two domain trees to this community.
   (b) Repeatedly choose one domain tree from *pool* that has maximal average similarity to the domain trees in the current community and add this domain tree to the current community. Add all gene trees associated with this new domain tree to the community. Repeat this step until the maximal average similarity falls below $\rho$.
4. Add all remaining domain trees in *pool* to their own single-domain communities, along with their associated gene trees.

There are several crucial reasons for decomposing large DG-groups into smaller DG-communities. First, DG-communities are expected to represent clusters of domains and genes that are closely related and biologically meaningful, whereas the domains and genes in a large DG-group are likely to be only weakly associated. Second, DG-communities reveal the underlying structure of DG-groups and help identify connecting gene families. And third, each DG-community can be viewed as a smaller DG-group for the purposes of mDGS reconciliation and it may be more appropriate to use these smaller DG-communities than larger weakly connected DG-groups.

**Analyzing DG-communities.** We applied our clustering algorithm to the largest complex group in our dataset with clustering parameter $\rho = 1.0$. This resulted in the identification of $532$ DG-communities, of which $304$ DG-communities contain only one domain tree and the remaining $228$ DG-communities together contain $901$ domain trees. Among the $2,394$ gene trees in the largest complex DG-group, $647$ (or $1.22$ per DG-community on average) were identified as connecting gene trees. We found that these connecting gene trees were often larger in size and contained more domains, on average, than the other gene trees. More precisely, the $647$ connecting gene trees contained $2.8$ domains each, on average, compared to $1.81$ domains over all gene trees within the DG-group. Similarly, connecting gene trees each contained $29.2$ leaf nodes on average, compared to $20.0$ leaf nodes for all gene trees in the DG-group. This is not entirely surprising since any connecting gene tree must necessarily contain domains from at least two different domain trees while no such constraint applies to other gene trees.

Next, we applied our mDGS reconciliation heuristic to each DG-community separately and compared the resulting gene-species reconciliations against those obtained by applying the heuristic to the entire DG-group. Recall that, when mDGS reconciliation was applied to the entire DG-group, the total number of gene nodes that deviate from their LCA mapping increased to $795$ from the $501$ observed for the base DGS-algorithm. In contract, when the mDGS-algorithm is applied separately to each DG-community in this DG-group, the total number of gene nodes that deviate from

their LCA mapping increases to only $567$. In other words, to make the underlying DGS reconciliations consistent in their gene-species mappings, mDGS reconciliation on the entire DG-group required $294$ additional gene tree nodes to deviate from their LCA mappings, while this number falls dramatically to only $66$ gene nodes when mDGS reconciliation is applied to all DG-communities in that DG-group. Thus, the vast majority of gene nodes that deviate from their LCA mappings appear on connecting gene trees. One possible explanation for this surprising result is that nodes in connecting gene trees are more likely to be assigned conflicting mappings by their associated domain trees; however, we observed that this was not the case. In fact, we found that conflicting gene trees had only $31$ gene tree nodes with conflicting mapping assignments compared to $115$ for all $2,394$ gene trees in the DG-group. This implies that the abundance of gene nodes on conflicting gene trees that deviate from their LCA mappings is caused by greater disagreement between the conflicting mappings (i.e., the conflicting mappings may be far apart on the species tree), causing the mapping of such nodes to be moved higher up towards the root than for other nodes with conflicting mappings.

We performed further analysis to assess if the sizes or other features of connecting gene trees may explain this overabundance of gene nodes deviating from their LCA mappings. We found that even though connecting gene trees are larger, on average, than other gene trees, they together contained less than $40\%$ of the total number of gene tree nodes in this DG-group. We also evaluated if the larger number of domain families, on average, represented in connecting gene trees may explain the overabundance, but found that connecting gene trees constitute only $61\%$ of all gene trees with at least two domain families and that these gene trees contain the same average number of domain families as connecting gene trees. Thus, the overabundance of gene nodes deviating from their LCA mappings on connecting gene trees is adequately explained neither by their size nor by their domain content.

One possible explanation for this surprising result is a higher error rate for connecting gene trees. Such error in gene trees could be caused by domain chaining, discordance in domain evolutionary histories, and other reasons. Thus, the identification of DG-communities within DG-groups may not only lead to more accurate mDGS reconciliations but also help identify erroneous multi-domain gene trees.

## 5 Conclusion

In this work, we extended the existing DGS reconciliation framework to address the problem of inconsistent gene-species mappings. We introduced the mDGS reconciliation framework and provided an efficient heuristic for estimating optimal mDGS reconciliations. Using an extensive experimental study on real biological data, we demonstrated the importance of mDGS reconciliation and showed that the proposed heuristic is effective at estimating optimal mDGS reconciliations. We also developed a technique to further improve the accuracy of mDGS reconciliation by introducing the notion of a DG-community, which is a subset of the domain and gene trees under consideration, and providing a clustering algorithm to find such DG-communities.

Several important research questions remain to be explored. First, our heuristic for mDGS reconciliation is rather simplistic, changing only the gene-species mappings to

achieve consistency and preserving the domain-gene mappings computed using DGS reconciliation. Simultaneous correction of both the domain-gene and gene-species mappings may lead to more optimal reconciliations. Second, a thorough simulation study is needed to systematically assess the impact of using mDGS reconciliation instead of DGS reconciliation and to properly assess the effectiveness of the proposed heuristic. The recent development of a probabilistic simulation framework for gene and subgene evolution [5] will facilitate such studies. And third, it would be interesting to further study the connecting gene families identified by our algorithm for finding DG-communities. It is possible that many connecting gene families represent cases of domain chaining.

**Funding:** This work was supported in part by National Science Foundation award IIS 1553421.

## References

1. B. Behzadi and M. Vingron. Reconstructing domain compositions of ancestral multi-domain proteins. In G. Bourque and N. El-Mabrouk, editors, *Comparative Genomics*, volume 4205 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2006.
2. D. Ekman, Åsa K. Björklund, J. Frey-Skött, and A. Elofsson. Multi-domain proteins in the three kingdoms of life: Orphan domains and other unassigned regions. *Journal of Molecular Biology*, 348(1):231 – 243, 2005.
3. M. Goodman, J. Czelusniak, G. W. Moore, A. E. Romero-Herrera, and G. Matsuda. Fitting the gene lineage into its species lineage. a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Zoology*, 28:132–163, 1979.
4. J.-H. Han, S. Batey, A. A. Nickson, S. A. Teichmann, and J. Clarke. The folding and evolution of multidomain proteins. *Nature Reviews Molecular Cell Biology*, 8:319–330, 2007.
5. S. Kundu and M. S. Bansal. SaGePhy: An improved phylogenetic simulation framework for gene and subgene evolution. *Bioinformatics (in press)*, 2019.
6. L. Li and M. S. Bansal. An integer linear programming solution for the domain-gene-species reconciliation problem. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, pages 386–397, New York, NY, USA, 2018. ACM.
7. L. Li and M. S. Bansal. An integrated reconciliation framework for domain, gene, and species level evolution. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(1):63–76, 2019.
8. A. D. Moore, A. K. Bjorklund, D. Ekman, E. Bornberg-Bauer, and A. Elofsson. Arrangements in the modular evolution of proteins. *Trends in Biochem. Sci.*, 33:444–451, 2008.
9. R. D. M. Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst. Biol.*, 43(1):58–77, 1994.
10. M. Stolzer, K. Siewert, H. Lai, M. Xu, and D. Durand. Event inference in multidomain families with phylogenetic reconciliation. *BMC Bioinformatics*, 16(14):S8, 2015.
11. H. Tordai, A. Nagy, K. Farkas, L. Banyai, and L. Patthy. Modules, multidomain proteins and organismic complexity. *FEBS Journal*, 272(19):5064–5078, 2005.
12. C. Vogel, M. Bashton, N. D. Kerrison, C. Chothia, and S. A. Teichmann. Structure, function and evolution of multidomain proteins. *Current Opinion in Structural Biology*, 14(2):208 – 216, 2004.
13. J. Wiedenhoeft, R. Krause, and O. Eulenstein. The plexus model for the inference of ancestral multidomain proteins. *IEEE/ACM Trans. Comp. Biol. Bioinf.*, 8(4):890–901, 2011.

14. Y.-C. Wu, M. S. Bansal, M. D. Rasmussen, J. Herrero, and M. Kellis. Phylogenetic identification and functional characterization of orthologs and paralogs across human, mouse, fly, and worm. *bioRxiv*, 2014.
15. Y.-C. Wu, M. D. Rasmussen, and M. Kellis. Evolution at the subgene level: Domain rearrangements in the drosophila phylogeny. *Mol. Biol. Evol.*, 29(2):689–705, 2012.

# Appendix

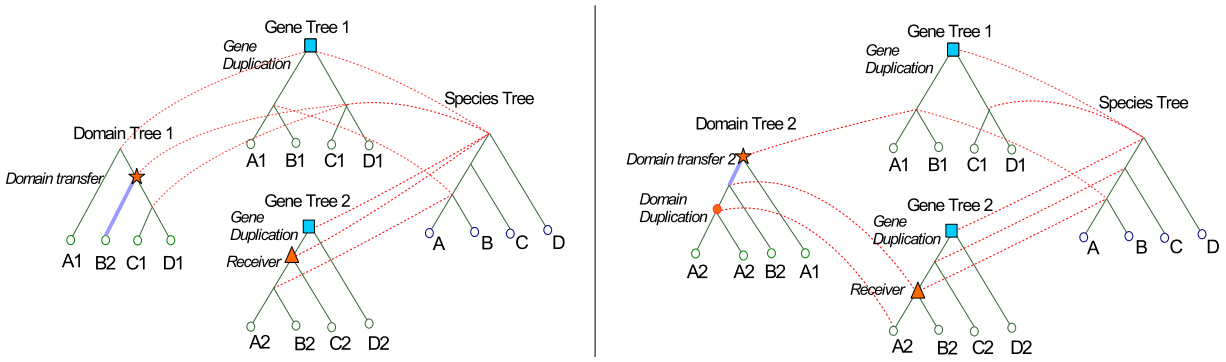


**Fig. A1.** DGS reconciliations for the two domain trees of Figure 1 with the same two gene trees and species tree. Observe that the two DGS reconciliations assign conflicting mappings to the parent of node C2 in gene tree 2. All other gene tree nodes are assigned the same mapping under both DGS reconciliations. Conflicts such as these can not arise under mDGS reconciliation.