Generalizing the Domain-Gene-Species Reconciliation Framework to Microbial Genes and Domains

Abhijit Mondal and Mukul S. Bansal

Abstract—Protein domains play an important role in the function and evolution of many gene families. Previous studies have shown that domains are frequently lost or gained during gene family evolution. Yet, most computational approaches for studying gene family evolution do not account for domain-level evolution within genes. To address this limitation, a new three-level reconciliation framework, called the Domain-Gene-Species (DGS) reconciliation model, has been recently developed 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. However, the existing model applies only to multi-cellular eukaryotes where horizontal gene transfer is negligible.

In this work, we generalize the existing DGS reconciliation model by allowing for the spread of genes and domains across species boundaries through horizontal transfer. We show that the problem of computing optimal generalized DGS reconciliations, though NP-hard, is approximable to within a constant factor, where the specific approximation ratio depends on the "event costs" used. We provide two different approximation algorithms for the problem and demonstrate the impact of the generalized framework using both simulated and real biological data. Our results show that our new algorithms result in highly accurate reconstructions of domain family evolution for microbes.

Index Terms: Protein domains, microbial gene family evolution, phylogenetic reconciliation, horizontal transfer, approximation algorithms

1 Introduction

Many proteins are known to consist of one or more independently folding structural and/or functional units called protein domains (or just domains for short). A key characteristic of domains is that the same type of domain (i.e., domains from the same domain family) can be found in proteins from more than one protein family. Thus, a domain can be viewed as a recurrent structural or functional building block that can be found in proteins from multiple distinct protein-coding gene families. Many thousands of domain families have been identified in the literature, e.g., [8], and it is well understood that domains play an important role in the function and evolution of many gene families [1], [23], [30], [32]. In fact, it has been estimated that about 70% of eukaryotic and 40% of prokaryotic protein-coding genes consist of one or more domains [14], [16].

Domain families evolve within gene families and domains are frequently lost or gained during gene family evolution, so genes from the same gene family may have different domain compositions. In other words, just as gene families evolve within a species tree, domain families evolve within one or more gene families. Given the importance of understanding domain evolution, several computational methods have been developed for reconstructing the evolutionary histories of domain families in the context of changes in domain composition or architectures (i.e., arrangements of domains within genes) [7], [34], [35]. These methods take as input a collection of domain trees (where a domain tree is a phylogenetic tree depicting evolutionary relationships between domains from the same domain family) along with domain compositions or architectures for the corresponding genes, with the method of Wu et al. [35] also requiring a species tree. The methods seek to reconstruct the compositions/architectures for ancestral genes, along with events such as gene fusions and fissions. However, these methods do not capture the interdependence of domain, gene, and species-level evolution, and do not explicitly consider gene family evolution (i.e. do not use gene trees).

More recently, a new class of phylogenetic reconciliation based approaches has been developed for studying domain level and gene level evolution in a unified framework [18]–[20], [28]. These reconciliation based methods take as input domain, gene, and species trees and seek to explicitly reconstruct the evolution of domain trees within gene trees and of gene trees within species trees. The earliest work in this direction was that of Stolzer et al. [28] where they used the Duplication-Transfer-Loss (DTL) reconciliation model [2], [12], [27], [29], traditionally used for reconciling gene trees with species trees, to additionally reconcile domain trees with gene trees. Their approach takes as input a domain tree, a gene tree, and a species tree and first reconciles the gene tree with the species tree, then reconciles the domain tree with the gene tree, and then interprets the domain tree to gene tree reconciliation in light of the gene tree to species tree reconciliation. Being based on the DTL reconciliation model, the approach of Stolzer at al. allows for gene duplication, horizontal gene transfer, and gene loss when reconciling the gene tree with the species tree and

• *Abhijit Mondal and Mukul S. Bansal are with the Department of Computer Science & Engineering at the University of Connecticut, Storrs, USA.* abhijit.mondal@uconn.edu, mukul.bansal@uconn.edu

for domain duplication, domain horizontal transfer, and domain loss when reconciling the domain tree with the gene tree. Domain transfers are then further interpreted as being intra-species or inter-species based on whether the genes involved in that domain transfer are from the same species or from different species according to the computed gene tree species trees reconciliation. This provides a unified view of how the domain tree evolved inside the gene tree and how the gene tree evolved inside the species tree.

The approach of Stolzer et al. [28], while groundbreaking, has two important limitations: First, it does not adequately model the interdependence of domain, gene, and species evolution. Specifically, it uses a simpler problem formulation that assumes a fixed gene to species mapping and does not seek a *joint* reconciliation of the domain, gene, and species trees. And second, it only allows for the reconciliation of a domain tree with a single gene tree. This can be quite limiting since domain families generally evolve within multiple gene families, with domains frequently transferring between genes from different gene families. Thus multiple gene trees must be simultaneously considered to infer the complete evolutionary history of most domain families. Later work by Li and Bansal [18], [19] addressed these limitations by introducing an integrated reconciliation framework, called the Domain-Gene-Species (DGS) reconciliation model, for joint reconciliation of domain, gene, and species trees. This DGS reconciliation model takes as input a domain tree, a collection of gene trees (i.e., all gene trees that contain at least one domain from that domain family), and a species tree and seeks a jointly optimal reconciliation of all trees. Thus, DGS reconciliation explicitly models the interdependence of domain-, gene-, and species-level evolution. Li and Bansal also proved that the associated optimization problem (computing an optimal DGS reconciliation) is NP-hard and provided an efficient heuristic algorithm and an exact integer linear programming solution for the problem [18], [19]. A key limitation of the DGS reconciliation model is that it is designed for analyzing domains and genes not affected by horizontal transfer. Specifically, each gene tree is assumed to evolve in the species tree under a duplication-loss model (i.e., horizontal gene transfer is not allowed), and domains can be transferred from one gene to another (within or across gene families) only if both those genes are present in the same species/genome. The absence of inter-species transfer of genes or domains makes DGS reconciliation well-suited to most multi-cellular eukaryotes but makes it inapplicable to microbial species.

Probabilistic models combing domain, gene, and species level evolution have also been developed [22], [24]. Muhammad et al. [24] introduced DomainDLRS, a generative probabilistic model and inference framework that takes as input a dated species tree and a multiple sequence alignment for each domain family and estimates a posterior distribution over the reconciled gene tree and domain trees. However, DomainDLRS operates under a duplication-loss framework and does not allow for the horizontal transfer of either genes or domains. It also requires the species tree to be dated and can only consider the evolution of domains in a single gene family. More recently, Menet et al. [22] introduced a probabilistic framework for host-symbiont-gene phylogenetic reconciliation. This framework, though developed in a different phylogenetic context, can also be applied to domain-gene-species reconciliation and has similar goals as our current work. We discuss this probabilistic approach of Menet et al. [22] in detail in Section 5.

Our contributions. In this work, we address a key current limitation of DGS reconciliation by introducing a *Generalized Domain-Gene-Species (Gen-DGS) reconciliation model* that explicitly models inter-species gene and domain horizontal transfer. We formalize the Gen-DGS reconciliation model and define the associated computational problem, which takes as input a domain tree, a collection of gene trees, a species tree, and event costs and seeks an optimal *joint reconciliation* of all trees. Gen-DGS reconciliation includes all of the original events types from the DGS reconciliation model and additionally allows for the horizontal transfer of genes and domains across species boundaries. While the problem of computing optimal Gen-DGS reconciliations is NP-hard (follows from the NP-hardness of computing optimal DGS reconciliations), we show that it is approximable to within a constant factor, where the specific approximation ratio depends on the event costs used for the reconciliation. We provide two different approximation algorithms with this approximation ratio and demonstrate the impact of Gen-DGS reconciliation using both simulated and real biological data. On simulated data, we find that our new algorithms result in highly accurate reconstructions of domain family evolution, showing much higher accuracy than SEADOG, which implements a heuristic for the DGS reconciliation problem [19], and far greater applicability than Notung-DM, which implements the approach of Stolzer et al. [28]. On the real dataset, which consists of 11 cyanobacterial species, 2587 gene trees, and 2347 domain trees, we found that Notung-DM could be correctly applied to fewer than 25% of the domain families. An analysis of the multi-gene domain families in this dataset using our algorithms also sheds light on the relative prevalence of different types of domain-transfer events in real microbial domain families. We found, for example, that about 75% of all domain-transfer events were *within-gene-family* transfers, with over half of all domain-transfer events being within-gene-family and inter-species. An implementation of our algorithms for Gen-DGS reconciliation is available freely from https://compbio.engr.uconn.edu/software/seadog-gen/.

2 Definitions and Preliminaries

2.1 Preliminaries

We follow basic notation, definitions and preliminaries from [19]. Given a rooted tree T, we denote its node, edge, and leaf sets by $V(T)$, $E(T)$, and $Le(T)$ respectively. 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) sub-tree 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 $y \leq_T x$. A tree is *binary* if all of its internal nodes have exactly two children. 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; that is, $lca_T(L)$ is the unique smallest upper bound of L under $\leq_T$. Throughout this work, the term *tree* refers to rooted binary trees.

As with DGS reconciliation, the Gen-DGS reconciliation model requires a species tree S, a collection of gene trees $\mathcal{G}$, and a domain tree D. These three types of trees are defined as follows. A *species tree* is a tree that depicts the evolutionary relationships for a set of species of interest. A *gene tree* is a tree that depicts the evolutionary relationships among the gene sequences belonging to a specific gene family, restricted to the species represented in the species tree. Finally, a *domain tree* depicts the evolutionary relationships among the domain sequences belonging to a specific domain family, restricted to the species represented in the species tree. Note that domains from the same domain family may be found in multiple different gene families. Thus, the domain sequences present in the domain tree may be associated with several different gene families. Accordingly, we implicitly assume that all gene families represented in the domain tree have a corresponding gene tree in the collection $\mathcal{G}$. For convenience, we use $Le(\mathcal{G})$ to denote $\cup_{G \in \mathcal{G}} Le(G)$, and analogously use $V(\mathcal{G})$, $I(\mathcal{G})$, and $E(\mathcal{G})$.

Each leaf of each gene tree is labeled with the species from which that gene sequence was obtained. This defines a leaf-to-leaf mapping from each gene tree of $\mathcal{G}$ to the species tree, denoted $\mathcal{L}^{\mathcal{G}}\colon Le(\mathcal{G}) \rightarrow Le(S)$ that maps each leaf node $g \in Le(\mathcal{G})$ to that unique leaf node $s \in Le(S)$ which has the same species label as g. Note that a gene tree may have more than one gene from the same species. Likewise, each leaf in a domain tree is labeled with the gene in which that domain sequence was found. This defines a leaf-to-leaf mapping from the domain tree to the gene trees, denoted $\mathcal{L}^{\mathcal{D}}\colon Le(D) \rightarrow Le(\mathcal{G})$ that maps each leaf node $d \in Le(D)$ to that unique leaf node $g \in Le(\mathcal{G})$ which has the same gene label as d. Note that, since domains from the same domain family may be present in multiple gene families, different leaves of the domain tree may map to genes from different gene trees.

2.2 Gen-DGS Reconciliation

Like the DGS reconciliation model [19], the Gen-DGS reconciliation model defines what constitutes a biologically valid joint reconciliation of the domain tree D, collection of gene trees $\mathcal{G}$, and species tree S. DGS reconciliation [19] models key evolutionary events that shape gene family and domain family evolution in multicellular eukaryotes: *speciation*, *gene duplication*, and *gene loss* for gene family evolution, and *co-divergence*, *intra-species domain transfer* (i.e., acquisition of a domain by a gene from another gene in the same species/genome), *domain duplication*, and *domain loss* for domain family evolution. Gen-DGS reconciliation extends upon DGS reconciliation in two ways: (i) it allows for *horizontal gene transfer*, in addition to speciation, gene duplication, and gene loss, for gene family evolution, and (ii) it allows for *inter-species horizontal domain transfer*, in addition to co-divergence, intra-species domain transfer, domain duplication, and domain loss, for domain family evolution. Thus, Gen-DGS reconciliation is a generalization of DGS reconciliation.

Under the Gen-DGS reconciliation model, gene families evolve under the well-established *Duplication-Transfer-Loss (DTL)* reconciliation framework [2], [3], [11], [12], [25], [27], [29]. Specifically, the constraints imposed by the Gen-DGS reconciliation model on a valid reconciliation of any gene tree $G \in \mathcal{G}$ with species tree S are identical to the constraints imposed by the DTL reconciliation framework [2], [3]. Accordingly, we provide a definition of DTL reconciliation below [2], [3], suitably extended to a collection of gene trees $\mathcal{G}$.

Definition 2.1 (DTL reconciliation of $\mathcal{G}$ and S [2], [3])**.** *Given a collection of gene trees $\mathcal{G}$ and species tree S, along with the corresponding leaf-mapping $\mathcal{L}^{\mathcal{G}}\colon Le(\mathcal{G}) \rightarrow Le(S)$, a* DTL reconciliation *for $\mathcal{G}$ and S is a six-tuple $\langle \mathcal{M}^{\mathcal{G}}, \Sigma^{\mathcal{G}}, \Delta^{\mathcal{G}}, \Theta^{\mathcal{G}}, \Xi^{\mathcal{G}}, \tau^{\mathcal{G}} \rangle$, where $\mathcal{M}^{\mathcal{G}}\colon V(\mathcal{G}) \rightarrow V(S)$ maps each node of $\mathcal{G}$ to a node of S, the sets $\Sigma^{\mathcal{G}}$, $\Delta^{\mathcal{G}}$ and $\Theta^{\mathcal{G}}$ partition $I(\mathcal{G})$ into speciation, gene-duplication, and (horizontal) gene-transfer nodes, respectively, $\Xi^{\mathcal{G}}$ is a subset of gene tree edges that represent gene-transfer events and $\tau^{\mathcal{G}} : \Theta^{\mathcal{G}} \rightarrow V(S)$ specifies the recipient species for each gene-transfer event, subject to the following 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) *$\mathcal{M}^{\mathcal{G}}(g) \not<_S \mathcal{M}^{\mathcal{G}}(g')$ and $\mathcal{M}^{\mathcal{G}}(g) \not<_S \mathcal{M}^{\mathcal{G}}(g'')$,*
   b) *At least one of $\mathcal{M}^{\mathcal{G}}(g')$ and $\mathcal{M}^{\mathcal{G}}(g'')$ is a descendant of $\mathcal{M}^{\mathcal{G}}(g)$.*
3) *Given any edge $(g, g') \in E(\mathcal{G})$, $(g, g') \in \Xi^{\mathcal{G}}$ if and only if $\mathcal{M}^{\mathcal{G}}(g)$ and $\mathcal{M}^{\mathcal{G}}(g')$ are incomparable.*
4) *If $g \in I(\mathcal{G})$ and g' and g'' denote the children of g, then,*
   a) *$g \in \Sigma^{\mathcal{G}}$ 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,*
   b) *$g \in \Delta^{\mathcal{G}}$ only if $\mathcal{M}^{\mathcal{G}}(g) \geq_S lca(\mathcal{M}^{\mathcal{G}}(g'), \mathcal{M}^{\mathcal{G}}(g''))$,*
   c) *$g \in \Theta^{\mathcal{G}}$ if and only if either $(g, g') \in \Xi^{\mathcal{G}}$ or $(g, g'') \in \Xi^{\mathcal{G}}$.*
   d) *If $g \in \Theta^{\mathcal{G}}$ and $(g, g') \in \Xi^{\mathcal{G}}$, then $\mathcal{M}^{\mathcal{G}}(g)$ and $\tau^{\mathcal{G}}(g)$ must be incomparable, and $\mathcal{M}^{\mathcal{G}}(g')$ must be a descendant of $\tau^{\mathcal{G}}(g)$, i.e., $\mathcal{M}^{\mathcal{G}}(g') \leq_S \tau^{\mathcal{G}}(g)$.*

Constraints 1, 2, and 3 in the definition above constrain the mapping of the gene tree(s) into the species tree and are necessary (though not necessarily sufficient) for biological feasibility. The remaining constraints specify the conditions under which the events speciation (Constraint 4a), gene-duplication (Constraint 4b), and gene-transfer (Constraints 4c and 4d) can be invoked. We note that, under the above definition, a DTL reconciliation is allowed to potentially violate temporal constraints imposed by the species tree; specifically, the invoked transfer events could, in some cases, induce contradictory constraints on possible datings (orderings) of the internal nodes of the species tree [27], [29]. We refer the reader to [2], [3], [29] for further details on this DTL reconciliation model.

The Gen-DGS reconciliation model can now be formally defined as follows:

Definition 2.2 (Gen-DGS reconciliation)**.** *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* Gen-DGS reconciliation *for $D, \mathcal{G}$, and S is a thirteen-tuple $\langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta_1^D, \Theta_2^D, \Theta^{\mathcal{G}}, \Xi^D, \Xi^{\mathcal{G}}, \tau^D, \tau^{\mathcal{G}}\rangle$, 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, $\Sigma^{\mathcal{G}}$, Θ_1^D, and Θ_2^D partition $I(D)$ into co-divergence, domain-duplication, intra-species domain-transfer, and inter-species domain-transfer nodes, respectively, the sets $\Sigma^{\mathcal{G}}$, $\Delta^{\mathcal{G}}$, and $\Theta^{\mathcal{G}}$ partition $I(\mathcal{G})$ into speciation, gene-duplication, and gene-transfer nodes, respectively, Ξ^D is a subset of domain tree edges that represent domain-transfer events, and $\Xi^{\mathcal{G}}$ is a subset of gene tree edges that represent gene-transfer events and $\tau^D : \Theta^D \to V(\mathcal{G})$ specifies the recipient gene for each domain-transfer event, and $\tau^{\mathcal{G}} : \Theta^{\mathcal{G}} \to V(S)$ specifies the recipient species for each gene-transfer event, subject to:*

Gene-species constraints:

1) *The six-tuple $\langle \mathcal{M}^{\mathcal{G}}, \Sigma^{\mathcal{G}}, \Delta^{\mathcal{G}}, \Theta^{\mathcal{G}}, \Xi^{\mathcal{G}}, \tau^{\mathcal{G}}\rangle$ must be a valid DTL reconciliation of $\mathcal{G}$ and S (per Definition 2.1).*

Domain-Gene constraints:

2) *If $d \in Le(D)$, then $\mathcal{M}^D(d) = \mathcal{L}^D(d)$.*
3) *If $d \in I(D)$ and d' and d'' denote the two 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).*
4) *Given any edge $(d, d') \in E(D)$, $(d, d') \in \Xi^D$ 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.*
5) *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) *$d \in \Theta^D$ if and only if either $(d, d') \in \Xi^D$ or $(d, d'') \in \Xi^D$.*
   d) *If $d \in \Theta_1^D$, where $(d, d') \in \Xi^D$, then $\mathcal{M}^D(d)$ and $\tau^D(d)$ must either be in different gene trees or be incomparable in the same gene tree, $\mathcal{M}^{\mathcal{G}}(\mathcal{M}^D(d)) = \mathcal{M}^{\mathcal{G}}(\tau^D(d))$, and $\mathcal{M}^D(d') \leq_{\mathcal{G}} \tau^D(d)$.*
   e) *If $d \in \Theta_2^D$, where $(d, d') \in \Xi^D$, then $\mathcal{M}^D(d)$ and $\tau^D(d)$ must either be in different gene trees or be incomparable in the same gene tree, $\mathcal{M}^{\mathcal{G}}(\mathcal{M}^D(d)) = \mathcal{M}^{\mathcal{G}}(\tau^D(d))$, and $\mathcal{M}^D(d') \leq_{\mathcal{G}} \tau^D(d)$.*

In the above definition, Constraint 1 specifies what constitutes a valid reconciliation of the gene tree(s) with the species tree, while the remaining constraints apply to the reconciliation of the domain tree with the gene tree(s). Observe that these domain-gene constraints are similar to the gene-species constraints imposed by DTL reconciliation; i.e., notice the correspondence between Constraints 1 through 4 of Definition 2.1 and Constraints 2 through 5 of Definition 2.2. Essentially, under Gen-DGS reconciliation, the reconciliation of the domain tree with the gene tree(s) employs a DTL-like reconciliation model, with the following two key differences: (i) the domain tree can evolve within more than one gene tree, and (ii) the type of domain-transfer event (i.e., intra-species or inter-species) additionally depends on the reconciliation between the gene tree(s) and the species tree. This interdependence between domain-gene and gene-species reconciliations stems from Constraint 5d, which enforces that the donor gene and recipient gene for any intra-species domain-transfer event must map to the same species in the species tree. This relationship between gene-species mappings and intra-species domain-transfer events necessitates the computation of a *joint* reconciliation at the domain-gene and gene-species levels when seeking optimal Gen-DGS reconciliations. Figure 1 provides a simple illustrative example of Gen-DGS reconciliation.

Parsimony based problem formulation. Given D, $\mathcal{G}$, and S, our goal is to find a most parsimonious Gen-DGS reconciliation for them. Under this framework, we assign positive costs for all gene-level and domain-level evolutionary events except for speciation and co-divergence (since those are null events), and seek a Gen-DGS reconciliation with smallest total cost of invoked events. Formally,

Definition 2.3 (Reconciliation cost)**.** *Given a Gen-DGS reconciliation α for D, $\mathcal{G}$, and S, the* reconciliation cost *for α is the total cost of all evolutionary events invoked by α.*

We point out that while speciation, gene-duplication, gene-transfer, co-divergence, domain-duplication, intra-species domain-transfer, and inter-species domain-transfer events are explicitly specified by any Gen-DGS reconciliation, gene-losses and domain-losses are not. However, the minimum number of gene-losses and domain-losses required by a given Gen-DGS reconciliation can be easily computed as in the DTL reconciliation model [2].

Event costs. We use $P_{\Delta}^{\mathcal{G}}$, $P_{\Theta}^{\mathcal{G}}$, and $P_{loss}^{\mathcal{G}}$ to denote gene-duplication, gene-transfer, and gene-loss costs, respectively, and P_{Δ}^{D}, $P_{\Theta_1}^{D}$, $P_{\Theta_2}^{D}$, and P_{loss}^{D} to denote domain-duplication, intra-species domain-transfer, inter-species domain transfer, and domain-loss costs, respectively. We further distinguish between domain-transfers for which the donor and recipient genes are within the same gene family and those in which the donor and recipient genes are from different gene families. Thus, instead of requiring a single cost for all intra-species domain-transfer events, we allow for a *within-gene-family intra-species domain-transfer* cost, denoted $P_{\Theta_1 w}^{D}$, and an *across-gene-family intra-species domain-transfer* cost, denoted $P_{\Theta_1 a}^{D}$. Likewise, instead of a single cost for all inter-species domain-transfer events, we allow for a *within-gene-family inter-species domain-transfer* cost, denoted $P_{\Theta_2 w}^{D}$, and an *across-gene-family inter-species domain-transfer* cost, denoted $P_{\Theta_2 a}^{D}$.

The problem of computing an optimal Gen-DGS reconciliation can now be stated as follows:

Definition 2.4 (Optimal Gen-DGS reconciliation (O–

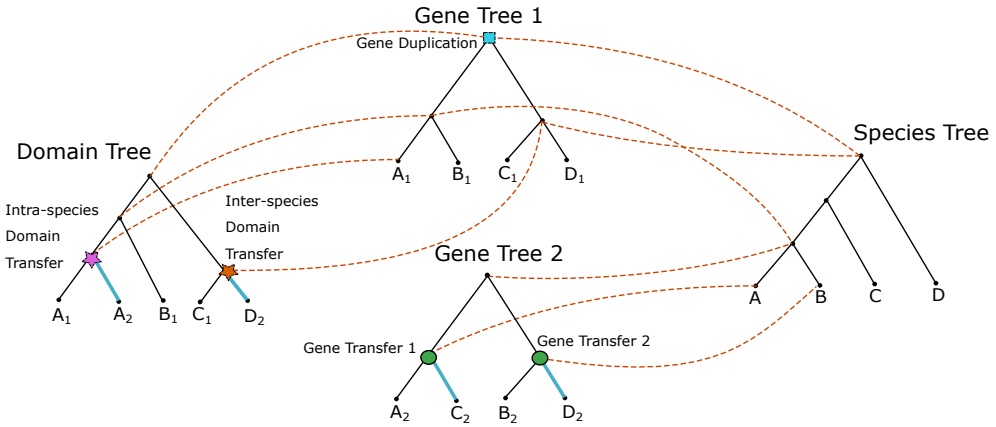

Fig. 1. **Gen-DGS Reconciliation.** This figure shows a Gen-DGS-reconciliation between a domain tree, two gene trees, and a species tree on four taxa. The mapping of the domain tree into the gene trees and of the gene trees into the species tree are shown by the dotted red lines. Domain-gene leaf associations are shown by shared leaf labels, and gene-species leaf associations are shown 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 that gene tree represent speciation events. In Gene Tree 2, we observe a speciation event at the root and gene-transfer events at the other two internal nodes (marked by green circles). In the domain-gene reconciliation, two domain transfer events are invoked at the nodes marked with the purple (left) and orange (right) stars, and all other internal nodes of the domain tree represent co-divergence events. Bolded (blue) edges in the domain tree represent domain-transfer edges. The purple star represents an *intra-species* domain-transfer event since its donor and recipient genes are mapped to the same species A in the species tree. So, we denote this event as an intra-species domain transfer event (Θ_1). In contrast, the orange star represents an *inter-species* domain-transfer event since its donor and recipient genes map to different species, C and D, in the species tree. Observe that both these domain-transfer events are *across-gene-family* transfers.

Gen-DGS) problem). . *Given $D, \mathcal{G}$, and S, along with event costs $P^{\mathcal{G}}_{\Delta}$, $P^{\mathcal{G}}_{\Theta}$, $P^{\mathcal{G}}_{loss}$, P^{D}_{Δ}, $P^{D}_{\Theta_{1w}}$, $P^{D}_{\Theta_{1a}}$, $P^{D}_{\Theta_{2w}}$, $P^{D}_{\Theta_{2a}}$, and P^{D}_{loss}, the* O-Gen-DGS problem *is to find a Gen-DGS reconciliation for $D, \mathcal{G}$ and S with minimum reconciliation cost.*

2.3 Relationship to the approach of Stolzer et al.

It is worth noting the similarities between Gen-DGS reconciliation and the domain event inference approach of Stolzer et al. [28]. Like Gen-DGS reconciliation, Stolzer et al. also use the DTL reconciliation model (albeit a slightly different one) for reconciling the gene tree with the species tree and the domain tree with the gene tree. They also distinguish between intra-species domain transfer events, called domain insertion events in [28], and inter-species domain transfer events based on the species mapping of the donor and recipient genes, just like in Gen-DGS reconciliation. The two key differences between Gen-DGS reconciliation and the approach of Stolzer et al. [28] are that (i) Gen-DGS allows for the domain tree to evolve within multiple gene trees while the approach of Stolzer et al. allows for only a single gene tree, and (ii) the O-Gen-DGS problem seeks jointly optimal domain-gene and gene-species reconciliations while the approach of Stolzer et al. computes the domain-gene and gene-species reconciliations separately and independently.

2.4 NP-hardness of the O-Gen-DGS problem

The optimal DGS reconciliation problem is known to be NP-hard [19], and that result also implies the NP-hardness of the O-Gen-DGS problem. This is based on the observation that if gene-transfer events and inter-species domain-transfer events are disallowed then an optimal Gen-DGS reconciliation must, in fact, be an optimal DGS reconciliation. Thus, the O-Gen-DGS problem can be used to compute optimal DGS reconciliations if the event costs $P^{\mathcal{G}}_{\Theta}$, $P^{D}_{\Theta_{2w}}$, and $P^{D}_{\Theta_{2a}}$ are set large enough to prevent the invocation of those events.

Theorem 2.1. *The O-Gen-DGS problem is NP-hard.*

Since a reduction from the decision version of the optimal DGS reconciliation problem [19] to the decision version of the O-Gen-DGS problem is straightforward, we omit a formal proof.

3 Algorithms for O-Gen-DGS

Observe that the introduction of inter-species domain transfer in the Gen-DGS model weakens (but does not completely break) the connection between domain-gene and gene-species reconciliations since a domain may be transferred irrespective of whether the donor and recipient genes map to the same species (or node) in the species tree or not. Thus, while algorithms for DGS reconciliation had to necessarily estimate jointly optimal domain-gene and gene-species reconciliations [18], [19], separately computing optimal domain-gene and gene-species reconciliations may yield near-optimal Gen-DGS reconciliations. In fact, as we show later (Corollary 3.1), if all four types of domain-transfer events are assigned the same cost, then an optimal solution for the O-Gen-DGS problem can be computed in polynomial time by separately computing optimal domain-gene and gene-species reconciliations. We leverage this insight to design two different approximation algorithms for the O-Gen-DGS problem. Both approximation algorithms have the same approximation ratio, but the second approximation algorithm (Section 3.2) has better empirical performance than the first (Section 3.1).

3.1 A Simple Approximation Algorithm

We first propose a very simple approximation algorithm for the O-Gen-DGS problem based on computing separate

optimal reconciliations at the domain-gene and gene-species levels. The overall idea is to simply compute an optimal DTL reconciliation, using the algorithm of Bansal et al. [3], for each gene tree in $\mathcal{G}$ with the species tree S, and then compute an optimal DTL reconciliation, using a trivial extension of the DTL reconciliation algorithm of Bansal et al. [3], for the domain tree and gene trees. Note that this trivial extension of the DTL reconciliation algorithm is required to enable the domain tree to be reconciled with multiple gene trees instead of just a single gene tree. The extension is straightforward and was also described by Li and Bansal [19]. Thus, we use DTL reconciliation at the gene-species level since each gene tree is being reconciled to a single species tree, but use the extended version of DTL reconciliation at the domain-gene level since the domain tree needs to be simultaneously reconciled to all gene trees in $\mathcal{G}$.

A key step in this overall approach is to use the largest of the four domain-transfer costs (i.e., maximum among $P^D_{\Theta_{1w}}$, $P^D_{\Theta_{1a}}$, $P^D_{\Theta_{2w}}$, and $P^D_{\Theta_{2a}}$) as the cost of a transfer event when computing domain-gene DTL reconciliations. Once the domain-gene reconciliation is computed, the inferred domain-transfer events can be easily relabeled/classified as intra-species or inter-species based on the previously computed gene-species reconciliation. If P_{max} denotes the maximum among the four domain-transfer costs and P_{min} denotes the minimum, then doing so ensures that the resulting Gen-DGS reconciliation has total reconciliation cost within a factor of P_{max}/P_{min} of the optimum. A formal description of this approximation algorithm follows:

Algorithm *SimpleApprox*$(D, \mathcal{G}, S, \mathcal{L}^D, \mathcal{L}^{\mathcal{G}})$

1: Let $P^{\mathcal{G}}_{\Delta}, P^{\mathcal{G}}_{\Theta}, P^{\mathcal{G}}_{loss}, P^D_{\Delta}, P^D_{\Theta_{1w}}, P^D_{\Theta_{1a}}, P^D_{\Theta_{2w}}, P^D_{\Theta_{2a}}, P^D_{loss}$ denote the assigned event costs.
2: **for** each gene tree $G \in \mathcal{G}$ **do**
3: Compute an optimal DTL reconciliation of G and S using duplication, transfer, and loss event costs $P^{\mathcal{G}}_{\Delta}, P^{\mathcal{G}}_{\Theta}$, and $P^{\mathcal{G}}_{loss}$, respectively, to obtain $\mathcal{M}^{\mathcal{G}}, \Sigma^{\mathcal{G}}, \Delta^{\mathcal{G}}, \Theta^{\mathcal{G}}, \Xi^{\mathcal{G}}$, and $\tau^{\mathcal{G}}$
4: Let $P_{max} = \max\{P^D_{\Theta_{1w}}, P^D_{\Theta_{1a}}, P^D_{\Theta_{2w}}, P^D_{\Theta_{2a}}\}$
5: Compute an optimal DTL reconciliation of D and the gene trees from $\mathcal{G}$ using duplication, transfer, and loss event costs P^D_{Δ}, P_{max}, *and* P^D_{loss}, respectively, to obtain $\mathcal{M}^D, \Sigma^D, \Delta^D, \Xi^{\mathcal{G}}$, and $\tau^{\mathcal{G}}$.
6: Let Θ^D be the set of domain-transfer nodes inferred in the previous step. Classify each node of Θ^D as belonging to either Θ^D_1 or Θ^D_2 based on the gene-species reconciliations computed in Steps 2 and 3.
7: **return** the computed Gen-DGS reconciliation $\langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta^D_1, \Theta^D_2, \Theta^{\mathcal{G}}, \Xi^D, \Xi^{\mathcal{G}}, \tau^D, \tau^{\mathcal{G}} \rangle$

Lemma 3.1. *Algorithm* SimpleApprox *can be implemented to run in* $O(|\mathit{Le}(D))| \cdot |\mathit{Le}(\mathcal{G})|^2 + |\mathit{Le}(\mathcal{G})| \cdot |\mathit{Le}(S)|^2)$ *time.*

Proof. The time complexity of Algorithm *SimpleApprox* is dominated by Steps 3, 3, and 5, in which optimal DTL reconciliations are computed. For our implementation of this algorithm, we wish to sample optimal DTL reconciliations uniformly at random from the space of all optimal DTL reconciliations. This can be accomplished in $O(mn^2)$ time, where m denotes the number of leaves in the "guest" tree and n denotes the number of leaves in the "host" tree, using existing algorithms [3] and their trivial extensions (i.e., to multiple host trees) [19]. This translates into time complexities of $O(\sum_{G \in \mathcal{G}} |\mathit{Le}(G)| \cdot |\mathit{Le}(S)|^2)$ to compute all gene-species reconciliations and $O(|\mathit{Le}(D))| \cdot |\mathit{Le}(\mathcal{G})|^2)$ to compute the domain-gene reconciliation. Summing these up, we get a total time complexity of $O(|\mathit{Le}(D))| \cdot |\mathit{Le}(\mathcal{G})|^2 + |\mathit{Le}(\mathcal{G})| \cdot |\mathit{Le}(S)|^2)$. □

We point out that the above algorithm can, in fact, be implemented to run in $O(|\mathit{Le}(D))| \cdot |\mathit{Le}(\mathcal{G})| + |\mathit{Le}(\mathcal{G})| \cdot |\mathit{Le}(S)|)$ time if using faster, $O(mn)$-time DTL reconciliation algorithms [2]. However, this faster implementation would be unable to sample uniformly at random from the space of optimal DTL reconciliations, which is useful for estimating uncertainty in inferred reconciliations.

Theorem 3.1. *Let* P_{max} *denote the maximum and* P_{min} *denote the minimum of the four event costs* $P^D_{\Theta_{1w}}$, $P^D_{\Theta_{1a}}$, $P^D_{\Theta_{2w}}$, *and* $P^D_{\Theta_{2a}}$. *Algorithm* SimpleApprox *is a* $\frac{P_{max}}{P_{min}}$*-approximation algorithm for the O-Gen-DGS problem.*

Proof. Let c denote the reconciliation cost of an optimal Gen-DGS reconciliation, denoted α, for $D, \mathcal{G}$, and S under the given event costs. Further, let $c_{D\mathcal{G}}$ and $c_{\mathcal{G}S}$ denote the domain-gene and gene-species reconciliation costs for this optimal reconciliation α. Thus, $c = c_{D\mathcal{G}} + c_{\mathcal{G}S}$. Now, let c' denote the reconciliation cost of the Gen-DGS reconciliation, denoted α', computed by Algorithm *SimpleApprox*. Finally, as before, let $c'_{D\mathcal{G}}$ and $c'_{\mathcal{G}S}$ denote the domain-gene and gene-species reconciliation costs for this computed reconciliation α'. Again, we must have $c' = c'_{D\mathcal{G}} + c'_{\mathcal{G}S}$.

Since Algorithm *SimpleApprox* computes an optimal DTL reconciliation between each gene tree and S, we must have $c'_{\mathcal{G}S} \leq c_{\mathcal{G}S}$. Thus, it suffices to show that $c'_{D\mathcal{G}} \leq \frac{P_{max}}{P_{min}} \cdot c_{D\mathcal{G}}$.

Suppose $c''_{D\mathcal{G}}$ denotes the reconciliation cost of an optimal DTL reconciliation, denoted by α'', of D and $\mathcal{G}$ with duplication, transfer, and loss event costs set to P^D_{Δ}, P_{min}, and P^D_{loss}, respectively. Since the corresponding Gen-DGS reconciliation uses the same costs for domain-duplication and domain-loss, and the same or higher costs for domain-transfer events, $c''_{D\mathcal{G}}$ must be a lower bound on $c_{D\mathcal{G}}$, i.e., $c''_{D\mathcal{G}} \leq c_{D\mathcal{G}}$. Now, consider the reconciliation cost, $\hat{c}''_{D\mathcal{G}}$ of α'' if we assign each transfer event in α'' a cost of P_{max} instead of P_{min}. Clearly, $\hat{c}''_{D\mathcal{G}} \leq \frac{P_{max}}{P_{min}} \cdot c''_{D\mathcal{G}}$. Finally, observe that $c'_{D\mathcal{G}} \leq \hat{c}''_{D\mathcal{G}}$ since α' is an optimal DTL reconciliation under those event costs.

Putting all the inequalities in the above paragraph together, we get $c'_{D\mathcal{G}} \leq \frac{P_{max}}{P_{min}} \cdot c''_{D\mathcal{G}} \leq \frac{P_{max}}{P_{min}} \cdot c_{D\mathcal{G}}$, as was to be shown. □

The following corollary identifies a special case of the O-Gen-DGS problem that can be solved exactly in polynomial time (despite the overall problem being NP-hard).

Corollary 3.1. *The O-Gen-DGS problem can be solved exactly in polynomial time if* $P^D_{\Theta_{1w}} = P^D_{\Theta_{1a}} = P^D_{\Theta_{2w}} = P^D_{\Theta_{2a}}$.

Proof. From Theorem 3.1, we know that Algorithm *SimpleApprox* becomes an exact algorithm for the O-Gen-DGS problem if $P^D_{\Theta_{1w}} = P^D_{\Theta_{1a}} = P^D_{\Theta_{2w}} = P^D_{\Theta_{2a}}$, and Algorithm *SimpleApprox* has polynomial time complexity (Lemma 3.1). □

We note that, when analyzing microbial gene families, it makes sense to assign similar costs to the four types

of domain-transfer events; for example, one may assign $P^D_{loss} = 1$, $P^D_{\Delta} = 2$, $P^D_{\Theta_{1w}} = 3$, $P^D_{\Theta_{1a}} = 4$, $P^D_{\Theta_{2w}} = 4$, and $P^D_{\Theta_{2a}} = 5$. The *SimpleApprox* algorithm would become a 5/3-approximation algorithm in this case.

3.2 An Improved Approximation Algorithm

We now provide an improved version of the *SimpleApprox* algorithm in which, instead of using the existing DTL reconciliation framework to reconcile the domain tree with the gene tree(s), we use a modified version of DTL reconciliation that can separately consider the four types of domain-transfer events and account for differences in their costs. We refer to the resulting improved approximation algorithm as the *ImprovedApprox* algorithm and prove that it has the same approximation ratio as the *SimpleApprox* algorithm. We also demonstrate later that the *ImprovedApprox* algorithm has better empirical performance than *SimpleApprox*.

The *ImprovedApprox* algorithm is very similar to the algorithm used by Stolzer et al. [28] for domain event inference, but has three key enhancements: First, it can reconcile the domain tree with multiple gene trees. Second, it allows for across-gene-family domain-transfers. And third, it uses a different version of the DTL reconciliation problem [2] than that used by Stolzer et al. [28], which, as we demonstrate later, significantly improves the scalability and applicability of the algorithm.

Algorithm *ImprovedApprox*$(D, \mathcal{G}, S, \mathcal{L}^D, \mathcal{L}^{\mathcal{G}})$

1: Let $P^{\mathcal{G}}_{\Delta}, P^{\mathcal{G}}_{\Theta}, P^{\mathcal{G}}_{loss}, P^D_{\Delta}, P^D_{\Theta_{1w}}, P^D_{\Theta_{1a}}, P^D_{\Theta_{2w}}, P^D_{\Theta_{2a}}, P^D_{loss}$ denote the assigned event costs.
2: **for** each gene tree $G \in \mathcal{G}$ **do**
3: Compute an optimal DTL reconciliation of G and S using duplication, transfer, and loss event costs $P^{\mathcal{G}}_{\Delta}, P^{\mathcal{G}}_{\Theta}$, and $P^{\mathcal{G}}_{loss}$, respectively, to obtain $\mathcal{M}^{\mathcal{G}}, \Sigma^{\mathcal{G}}, \Delta^{\mathcal{G}}, \Theta^{\mathcal{G}}, \Xi^{\mathcal{G}}$, and $\tau^{\mathcal{G}}$
4: Compute an optimal modified-DTL reconciliation of D and the gene trees from $\mathcal{G}$ using duplication and loss event costs P^D_{Δ} and P^D_{loss}, respectively, and the four domain-transfer type costs $P^D_{\Theta_{1w}}, P^D_{\Theta_{1a}}, P^D_{\Theta_{2w}}$, and $P^D_{\Theta_{2a}}$, to obtain $\mathcal{M}^D, \Sigma^D, \Delta^D, \Theta^D_1, \Theta^D_2, \Xi^{\mathcal{G}}$, and $\tau^{\mathcal{G}}$.
5: **return** the computed Gen-DGS reconciliation $\langle \mathcal{M}^D, \mathcal{M}^{\mathcal{G}}, \Sigma^D, \Sigma^{\mathcal{G}}, \Delta^D, \Delta^{\mathcal{G}}, \Theta^D_1, \Theta^D_2, \Theta^{\mathcal{G}}, \Xi^D, \Xi^{\mathcal{G}}, \tau^D, \tau^{\mathcal{G}} \rangle$

Observe that, in Step 4 of the above algorithm, we compute an optimal *modified-DTL* reconciliation of D with the gene trees from $\mathcal{G}$. A modified-DTL reconciliation is the same as a DTL reconciliation except that a distinction is made between the four types of domain-transfer events. It is easy to adapt existing algorithms for computing optimal DTL reconciliations [2], [3] to explicitly consider all four domain-transfer types during computation and account for any differences in their assigned costs. Specifically, such a modification only requires a trivial adjustment (inclusion of new cases) in the dynamic programming equation that forms the basis of algorithms for computing optimal DTL reconciliations and consideration of the given (previously computed) gene-species reconciliation to distinguish between inter-species and intra-species domain-transfers. These adjustments can be made without affecting the optimality or asymptotic time complexity of the underlying dynamic programming algorithm. Since nearly identical modifications to DTL reconciliation algorithms have already been previously described [19], [28], we omit a formal description of the algorithm used to compute optimal modified-DTL reconciliations.

The time complexity of *ImprovedApprox* remains the same as that of *SimpleAprox* since time complexity is again dominated by the time required to compute the gene-species and domain-gene reconciliations, yielding the following lemma.

Lemma 3.2. *Algorithm* ImprovedApprox *can be implemented to run in* $O(|\,Le(D))| \cdot |\,Le(\mathcal{G})|^2 + |\,Le(\mathcal{G})| \cdot |\,Le(S)|^2)$ *time.*

We can use the fact that the *SimpleApprox* algorithm is a $\frac{P_{max}}{P_{min}}$-approximation algorithm (Theorem 3.1) to prove that the *ImprovedApprox* algorithm must also then be a $\frac{P_{max}}{P_{min}}$-approximation algorithm.

Theorem 3.2. *Let* P_{max} *denote the maximum and* P_{min} *denote the minimum of the four event costs* $P^D_{\Theta_{1w}}$, $P^D_{\Theta_{1a}}$, $P^D_{\Theta_{2w}}$, *and* $P^D_{\Theta_{2a}}$. *Algorithm* ImprovedApprox *is a* $\frac{P_{max}}{P_{min}}$*-approximation algorithm for the O-Gen-DGS problem.*

Proof. From Theorem 3.1, we know that the *SimpleApprox* algorithm is a $\frac{P_{max}}{P_{min}}$-approximation algorithm for the O-Gen-DGS problem irrespective of which optimal DTL reconciliation is computed by the algorithm at the gene-species level. It therefore suffices to prove that if the same optimal gene-species reconciliation is used by *ImprovedApprox*, then the domain-gene reconciliation cost for *ImprovedApprox* cannot be greater than the domain-gene reconciliation cost for *SimpleApprox*. Accordingly, let $\alpha_{\mathcal{G}S}$ denote an optimal gene-species reconciliation computed during a corresponding run of the *SimpleApprox* algorithm, let $c_{\mathcal{G}S}$ denote the reconciliation cost of $\alpha_{\mathcal{G}S}$, and let $c_{D\mathcal{G}}$ denote the domain-gene reconciliation cost. Thus, the total reconciliation cost of this Gen-DGS reconciliation computed using *SimpleApprox* is $c = c_{D\mathcal{G}} + c_{\mathcal{G}S}$.

Now, suppose that our run of the *ImprovedApprox* algorithm uses the same gene-species reconciliation $\alpha_{\mathcal{G}S}$, and let $c'_{D\mathcal{G}}$ denote the cost of the resulting domain-gene reconciliation. Thus, the total reconciliation cost of the Gen-DGS reconciliation computed using *ImprovedApprox* is $c' = c'_{D\mathcal{G}} + c_{\mathcal{G}S}$.

Since the domain-gene DTL reconciliation computed by the *SimpleApprox* algorithm must be a valid candidate solution under modified-DTL reconciliation as well, the cost of an optimal modified-DTL reconciliation computed by *ImprovedApprox* at the domain-gene level must have reconciliation cost no greater than that of the domain-gene DTL-reconciliation computed by *SimpleApprox*. Thus, $c'_{D\mathcal{G}} \leq c_{D\mathcal{G}}$, which immediately implies that $c' \leq c$. □

4 Experimental Evaluation

We used simulated data to assess the accuracies of the two approximation algorithms and compare against the two most relevant previous approaches, SEADOG, which implements a heuristic for estimating optimal DGS reconciliations [19], and Notung-DM, which implements the domain event inference approach of Stolzer et al. [28]. We also applied our algorithms to a large real dataset of 11 cyanobacterial species to demonstrate their impact in practice and further compare against Notung-DM.

4.1 Results on Simulated Data

Dataset description. We used SaGePhy [17] to simulate 100 sets of species tees, gene trees, and domain trees. We first simulated 100 species trees each with exactly 100 leaves (taxa) and height 1, using a birth-death process. For each species tree, we then simulated 3 gene trees under the probabilistic duplication-transfer-loss model implemented in SaGePhy. Following previous literature, e.g. [5], we used (gene) duplication, transfer, and loss rates of 0.3, 0.6, and 0.6, respectively, resulting in gene trees that had an average of 101.91 leaves, 6.03 gene duplications, 12.32 gene transfers, and 18.64 gene losses, on average. We then used each species tree and its three gene trees to simulate the evolution of a domain tree in the three gene trees under a duplication-transfer-loss based domain evolution model [17] that allows for intra-species and inter-species, as well as intra-gene-family and inter-gene-family, domain-transfer. As before, we used (domain) duplication, transfer, and loss rates of 0.3, 0.6, and 0.6, respectively, resulting in domain trees that had an average of 129.04 leaves, 5.69 domain duplications, 9.8 domain transfers, and 16.06 domain losses. Among the domain transfers, an average of 7.9 were across-gene-family transfers and 1.9 were within-gene-family transfers. We used these 100 sets of domain, gene, and species trees to evaluate the relative accuracies of *SimpleApprox*, *ImprovedApprox*, and SEADOG in inferring the evolutionary histories of domain families. (A separate simulated dataset had to be used for comparison against Notung-DM, as discussed later.) Specifically, we evaluated the ability of these algorithms/methods to correctly infer domain tree mappings as well as infer the correct event type for each internal domain tree node.

All methods were executed on this simulated dataset with the same event costs, based on previous literature [5], [11], [19]: $P^{\mathcal{G}}_{\Delta} = 2, P^{\mathcal{G}}_{\Theta} = 3, P^{\mathcal{G}}_{loss} = 1, P^{D}_{\Delta} = 2, P^{D}_{\Theta_{1w}} = 3, P^{D}_{\Theta_{1a}} = 4, P^{D}_{\Theta_{2w}} = 4, P^{D}_{\Theta_{2a}} = 5, P^{D}_{loss} = 1$.

Comparison of *SimpleApprox* and *ImprovedApprox*. We first compared the relative accuracies of *SimpleApprox* and *ImprovedApprox* at inferring domain-level evolutionary histories. These results are shown in Table 1. As the table shows, the overall event inference accuracy (i.e., correct labeling of each internal node of the domain tree as either co-divergence, domain-duplication, or domain-transfer) is almost the same for both algorithms, at 99.67% for *SimpleApprox* and 99.71% for *ImprovedApprox*. However, *ImprovedApprox* shows significantly higher accuracy at distinguishing between the four types of domain-transfer events. For example, for within-gene-family intra-species domain-transfers, the event inference accuracy for *SimpleApprox* and *ImprovedApprox* are 81.48% and 88.89%, respectively. Results on mapping accuracy of the two algorithms show similar trends, with both methods showing near-perfect overall mapping accuracy and roughly 90% mapping accuracy for domain-transfers, but *ImprovedApprox* showing higher mapping accuracies for three out of the four types of domain-transfer events. Thus, while both algorithms show high event inference and mapping accuracies, *ImprovedApprox* consistently shows greater accuracy than *SimpleApprox*.

Running times of *SimpleApprox* and *ImprovedApprox*. Our implementations of *SimpleApprox* and *ImprovedApprox* are highly efficient and scalable. On the above simulated dataset, our implemenation of *SimpleApprox* took an average of 0.17 seconds per domain tree (i.e., to reconcile one domain tree, three gene trees, and one species tree). For the *ImprovedApprox* algorithm this average time increased slightly to 0.21 seconds. These runs were executed on a commodity laptop computer with a 2.8 GHz Intel i7 processor and 12 GB of RAM, using a single core.

Comparison with SEADOG. We applied SEADOG, which estimates optimal DGS reconciliations, to the above simulated dataset and again evaluated its ability to accurately infer domain-level evolutionary histories. These results are also shown in Table 1. As the table shows, both event inference and mapping accuracies are significantly lower for SEADOG than for *SimpleApprox* and *ImprovedApprox*. This is not surprising since SEADOG does not model gene-transfer or inter-species domain-transfer, both of which are present in the simulated dataset. In particular, the event and mapping accuracies for domain-transfer events are only 11.63% and 36.05%, respectively. This demonstrates the inapplicability of SEADOG, and of DGS reconciliation in general, in the presence of inter-species horizontal transfer. It is worth noting that while the overall event and mapping accuracies (for all internal nodes) for SEADOG are significantly lower than those for *SimpleApprox* and *ImprovedApprox*, at 90.76% and 94.98%, respectively, they are still quite high. This is because most of the internal nodes in the domain trees are co-divergence events, which, as these results suggest, SEADOG is able to identify and map with high accuracy despite the presence of inter-species domain transfer.

Comparison with Notung-DM. Notung-DM, which implements the approach of Stolzer et al. [28] is unable to consider the evolution of a domain family in more than one gene tree. Thus, Notung-DM cannot be applied to our previously described simulated dataset. We therefore created a new, restricted simulated dataset in which only one gene tree was simulated per species tree and the domain tree was made to evolve only within this single gene tree (and otherwise using similar simulation parameters as before). We applied *SimpleApprox*, *ImprovedApprox*, and Notung-DM to the 100 domain, gene, and species trees in this restricted dataset and, as before, measured the domain event inference and domain mapping accuracies for these methods. Surprisingly, we found that Notung-DM could only be successfully applied to 47 out of the 100 sets of trees. For 48 of the 100 sets of trees, Notung-DM failed to compute either the domain-gene reconciliation or gene-species reconciliation (or both) using DTL reconciliation, resorting to using the Duplication-Loss model instead. This happens because Notung-DM uses a slightly different formulation of DTL reconciliation that requires temporal consistency of inferred transfer events. This makes the problem of computing optimal DTL reconciliations more challenging and Notung-DM can fail to find any temporally consistent solution for the given tree pair. For the remaining 5 sets of trees, Notung-DM failed to terminate with an output within one hour and appeared to crash/freeze.

Table 2 shows the results of our analysis for the 47 sets of trees on which Notung-DM could be executed successfully. As the table shows, Notung-DM shows similar event and mapping accuracy as *ImprovedApprox*. This is

TABLE 1
Domain tree event inference and mapping accuracies for *SimpleApprox*, *ImprovedApprox*, and SEADOG on the simulated dataset. Accuracies are averaged over all 100 domain trees and are reported in percentages. They are calculated to be the percentage of internal nodes of the specified type in the true evolutionary history of the simulated domain tree whose event type or mapping are inferred correctly by the chosen method.

Criteria	Event inference accuracy			Mapping Accuracy		
	SimpleApprox	*ImprovedApprox*	SEADOG	*SimpleApprox*	*ImprovedApprox*	SEADOG
All internal nodes	99.67	99.71	90.76	99.66	99.64	94.98
All domain-transfer events	94.96	96.9	11.63	90.31	91.86	36.05
Within-gene-family intra-species domain-transfer events	81.48	88.89	81.48	77.78	85.19	40.74
Across-gene-family intra-species domain-transfer events	87.5	100	100	87.5	100	50
Within-gene-family inter-species domain-transfer events	92.79	94.59	0	86.49	87.39	38.74
Across-gene-family inter-species domain-transfer events	99.11	99.11	0	97.32	97.32	31.25

to be expected given the previously discussed similarity between the *ImprovedApprox* algorithm and the algorithm of Stolzer et al. [28]. Note, however, that Notung-DM shows much lower accuracy in identifying inter-species domain-transfer events, at 60.83% compared to 88.33% and 89.17% for *SimpleApprox* and *ImprovedApprox*. We found that this is because Notung-DM sometimes incorrectly labels inter-species domain-transfer events as intra-species domain-transfer events.

Overall, these results show that while Notung-DM can yield results that are almost as accurate as those of *ImprovedApprox*, its applicability is severely limited both by its inability to be applied to domain families found in multiple gene families and by its inability to use DTL reconciliation for either domain-gene or gene-species reconciliation (i.e., inability to account for gene-transfer or inter-species domain-transfer or both) in many cases.

4.2 Results on Real Data

Dataset assembly and description. We further evaluated the performance and applicability of *SimpleApprox*, *improvedApprox*, and Notung-DM on domain and gene families from 11 cyanobacterial species [37] (Supplementary Table S1). We used Cyanobase [15] to identify all annotated genes in these 11 genomes. Out of the total of 42,999 genes identified, 41,651 genes could be assigned a Uniprot ID [10]. We used these Uniprot IDs to query Pfam [6] and identify annotated domains within each gene. Out of the 41,651 genes with Uniprot IDs, 31,138 had at least one annotated domain. In total, we identified 50,280 domains belonging to 3413 Pfam domain families in these 31,138 genes. On average, each gene contained 1.61 domain sequences and each domain family consisted of 14.73 domain sequences. Next, we used eggNOG-mapper v2 [9] to cluster the 41,651 gene sequences into gene families. We could assign 38,654 gene sequences to a gene family using eggNOG-mapper v2 and so we removed the other 2,997 gene sequences. After removing the domains from these 2,997 gene sequence, we were left with a total of 49,486 domain sequences from 3,386 domain families. Related details appear in Supplementary Table S1.

To perform more meaningful analysis, we pruned out all domain families that has fewer than 4 domain sequences or more than 500 domain sequences. After this pruning step, we also filtered out any gene families that no longer contained any of the remaining domain families. This resulted in our final set of 2587 gene families and 2347 domain families. On average, each domain family consisted of 18.83 domain sequences and each gene family consisted of 12.56 gene sequences. We observed that, out of the 2347 domain families, 1367 were found in (i.e., mapped to) only one gene family and the remaining 980 were found in more than one gene family. On average, each domain family was found in 2.1 gene families. Figure 2 shows the distribution of the number of gene families represented in each domain family. We aligned the gene family and domain family amino-acid sequences using MUSCLE [13] and constructed maximum likelihood domain trees and gene trees using RAxML [26] under the JTT substitution model with gamma-distributed rates and thorough search parameters. We used MAD rooting [31] to root all domain trees and all gene trees with at least four leaves (i.e., all gene trees reconstructed by RAxML). While gene trees with only one or two leaves are trivially rooted, the 125 gene trees with exactly three leaves were rooted using the reconciliation-based OptRoot approach [4]. (We did not use MAD rooting for gene trees with three leaves because such gene trees could not be "estimated" using RAxML and therefore did not have associated branch lengths.) Both MAD rooting and OptRoot have been shown to be effective at rooting microbial gene families [33]. As our species tree, we used the tree reported in [37].

Experimental setup. We used this real dataset to (i) evaluate the applicability of Notung-DM on real data, (ii) compare differences between the reconciliations inferred using *SimpleApprox*, *improvedApprox*, and Notung-DM, and (3) shed light on the relative prevalence of intra-species vs inter-species and within-gene-family vs across-gene-family domain-transfers on this dataset. Recall that Notung-DM can only be applied to domain trees associated with a single gene tree. Thus, we separately analyzed the 1367 domain trees found in only one gene family and the 980 domain trees found in more than one gene family. We refer to these two separate datasets as the *uni-gene* and *multi-gene* datasets, respectively. Furthermore, since Notung-DM only offers a graphical interface, making it difficult to automate the analysis, we randomly sampled 100 domain trees from the uni-gene dataset for the uni-gene dataset analysis.

All methods were executed using the same event costs as for the simulated datasets: $P^{\mathcal{G}}_{\Delta} = 2, P^{\mathcal{G}}_{\Theta} = 3, P^{\mathcal{G}}_{loss} =$

TABLE 2
Domain tree event inference and mapping accuracies for *SimpleApprox*, *ImprovedApprox*, and Notung-DM on the restricted simulated dataset. Results are shown only for the 47 sets of trees, out of 100, on which Notung-DM could be executed successfully. Accuracies are averaged over the 47 domain trees and are reported in percentages. They are calculated to be the percentage of internal nodes of the specified type in the true evolutionary history of the simulated domain tree whose event type or mapping are inferred correctly by the chosen method.

Criteria	Event inference accuracy			Mapping Accuracy		
	SimpleApprox	*ImprovedApprox*	Notung-DM	*SimpleApprox*	*ImprovedApprox*	Notung-DM
All internal nodes	98.48	98.78	96.55	97.92	98.29	97.31
All domain-transfer events	91.15	91.67	91.15	78.13	82.29	79.69
Within-gene-family intra-species domain-transfer events	80.56	88.89	88.89	86.11	90.28	81.94
Within-gene-family inter-species domain-transfer events	88.33	89.17	60.83	73.33	77.5	78.33

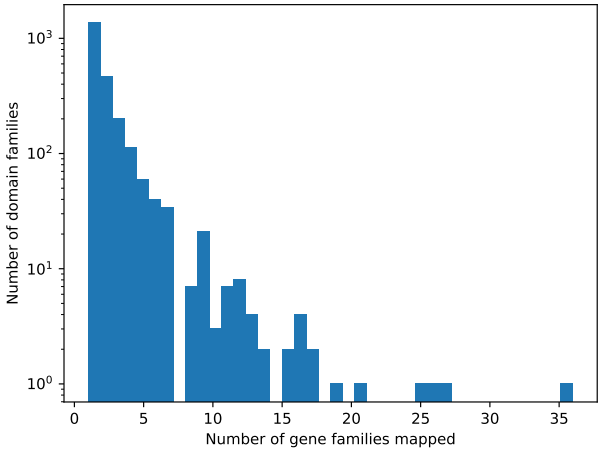

Fig. 2. **Distribution of the number of gene families represented in each of the 2347 domain families.**

$1, P^D_{\Delta} = 2, P^D_{\Theta_{1w}} = 3, P^D_{\Theta_{1a}} = 4, P^D_{\Theta_{2w}} = 4, P^D_{\Theta_{2a}} = 5, P^D_{loss} = 1$.

Comparison with Notung-DM. Out of the 100 randomly sampled sets of domain, gene, and species trees in the uni-gene dataset, we found that Notung-DM could only be successfully applied to 40. For 57 sets of trees, Notung-DM failed to compute either the domain-gene reconciliation or gene-species reconciliation, or both, using DTL reconciliation, resorting to using the Duplication-Loss model instead. For the remaining 3 sets of trees, Notung-DM appeared to crash/freeze and failed to terminate with an output within one hour. These results are consistent with what was previously observed on the simulated dataset, where Notung-DM could be successfully applied to only 47% of the input tree sets. In contrast, both *SimpleApprox* and *ImprovedApprox* could be successfully applied to all 100 sets of trees and required less than a second of running time per domain tree. On the 40 sets of trees on which Notung-DM could be applied successfully, it yielded reconciliation costs identical to that of *ImprovedApprox* for 39 of the 40 sets of trees.

Overall, these results highlight the limited applicability of Notung-DM in practice. This is both due to Notung-DM's inability to handle domain families evolving within multiple gene trees and due to its high rate of failure on even uni-gene datasets. For example, on our real dataset of 2347 domain trees, Notung-DM would be applicable to only about 550 domain families, or less than 24% of the real dataset.

Comparison of *SimpleApprox* and *ImprovedApprox*. On the uni-gene dataset, we found that *SimpleApprox* and *ImprovedApprox* computed Gen-DGS reconciliations with the same reconciliation costs on 93 of the 100 tree sets. On the 7 tree sets where there was a difference in reconciliation costs, *SimpleApprox* showed a 3.48% higher cost on average. We observed a significantly greater difference in performance on the multi-gene dataset, where we found that *SimpleApprox* and *ImprovedApprox* computed Gen-DGS reconciliations with the same reconciliation costs on only 607 (or 61.9%) of the 980 tree sets in that dataset. On the tree sets with different reconciliation costs, *SimpleApprox* showed a 2.97% higher cost on average.

We also compared the overall similarity between domain-gene reconciliations computed by *SimpleApprox* and *ImprovedApprox*. On the uni-gene dataset, we found that, on average, 80.72% of the internal nodes in a domain tree were mapped to the same gene tree node under both reconciliations. On the multi-gene dataset, an average of 72.05% of the internal nodes in each domain tree mapped to the same gene tree node under both reconciliations.

Overall, these results show that Gen-DGS reconciliations computed by *SimpleApprox* and *ImprovedApprox*, though similar, can have significant differences, especially when the domain family being analyzed evolves within multiple gene families. When coupled with results on simulated data, this suggests that *ImprovedApprox* should yield more accurate Gen-DGS reconciliations than *SimpleApprox* on real biological datasets.

Relative frequencies of different domain transfer types. We analyzed the Gen-DGS reconciliations computed by *ImprovedApprox* on the multi-gene dataset to gain insight into the relative frequencies of intra-species versus inter-species and within-gene-family versus across-gene-family domain-transfers. We found that intra-species domain-transfers comprised about 31.6% of all domain-transfers, with inter-species domain-transfers accounting for the remaining 68.4%. For within-gene-family and across-gene-family domain-transfers, these percentages were 76.5% and 24.5%, respectively. Over half of the inferred domain-transfer events, at 52.8%, were within-gene-family inter-species domain-transfers, while only 7.8% were across-gene-family intra-species domain-transfers. Note that these results are likely to have been affected by various confounding factors, such as errors in the domain and gene trees, and

should therefore be interpreted with caution. Still, this analysis demonstrates the potential of Gen-DGS reconciliation and of our new algorithms to shed light on fundamental questions related to the evolution of domain families in microbes.

5 Discussion

In this work, we extended the DGS reconciliation framework to include gene transfer events and inter-species domain transfer events, formulating the Generalised DGS reconciliation problem. We showed that a fast, relatively simple algorithm based on separately reconciling the domain tree with the gene trees and the gene trees with the species tree is an approximation algorithm for the corresponding optimization problem. We also provided an improved approximation algorithm with the same approximation ratio but better empirical performance. Experimental analysis using simulated and real microbial datasets shows that both algorithms perform well in practice and significantly outperform the two existing approaches, SEADOG and Notung-DM. Our analysis of the real dataset also sheds new light on the relative prevalence of intra-species versus inter-species and within-gene-family versus across-gene-family domain-transfers.

It is worth noting that the new approximation algorithms only work well in the microbial setting, i.e., when inter-species transfer costs are low. Thus, even though the Gen-DGS reconciliation generalises DGS reconciliation, our current algorithms for the Gen-DGS reconciliation problem should not be used for the DGS reconciliation problem. Thus, the two algorithms, SimpleApprox and ImprovedApprox, are not good substitutes for SEADOG when analyzing domain families and gene families from multi-cellular eukaryotes. Going forward, it would be worthwhile to develop new algorithms for Gen-DGS reconciliation, based on co-optimizing domain-gene and gene-species reconciliations, that could work well in both the microbial (high rate of horizontal transfer) and non-microbial (low rate of horizontal transfer) settings.

A limitation of the current Gen-DGS reconciliation framework is that it is only defined for single domain families, i.e., it only reconciles one domain tree at a time. Since multiple domain families often co-evolve within gene families, it is desirable to simultaneously reconcile all co-occuring domain families, the entire collection of gene families in which one or more of these domain families are present, and the species tree. This multi-domain multi-gene reconciliation problem has been previously studied in the context of the DGS reconciliation framework [20] and it would be useful to define a similar extension for Gen-DGS reconciliation. The resulting extended framework will also require new algorithm development since our proposed algorithms for Gen-DGS reconciliation are unlikely to work well for the extended problem.

The Gen-DGS reconciliation framework can be leveraged to potentially improve the accuracy of domain trees and gene trees and this application is worth exploring further. While gene tree construction has been shown to be relatively robust to small-scale subgene recombination (such as through domain-transfer) [36], it can be greatly affected by larger-scale subgene level events. Likewise, domain trees can be difficult to estimate accurately due to their short sequence lengths. By co-estimating gene trees and domain trees and assessing their "fit" with each other and with the species tree under a Gen-DGS framework, the reconstruction accuracy of both types of trees could be improved.

The Gen-DGS reconciliation framework and our new algorithms may also be applicable in other contexts, such as to host-symbiont-gene reconciliation [22] or other kinds of reconciliation problems [21], and such applications are worth exploring further.

Finally, it may be useful to develop improved probabilistic generative models and inference approaches for Gen-DGS reconciliation. The recently published approach of Menet et al. [22], developed independently in parallel with the current work, introduces such a probabilistic framework for host-symbiont-gene phylogenetic reconciliation. This framework is also applicable to domain-gene-species reconciliation, with the host tree serving as the species tree, symbiont trees serving as gene trees, and gene trees serving as domain trees, and the underlying reconciliation model is almost identical to Gen-DGS reconciliation. Probabilistic inference of reconciliations, though less scalable, has the potential to be more accurate than parsimony-based inference, especially when the underlying probabilistic generative model sufficiently resembles the actual evolutionary process. It would be informative to systematically compare the accuracies of our parsimony based approach and the probabilistic approach of Menet et al. [22] under different evolutionary conditions. More generally, it would likely be useful to develop more biologically realistic probabilistic generative models for improved Gen-DGS reconciliation accuracy on biological datasets.

Funding

This work was supported in part by NSF award IIS 1553421 to MSB.

References

[1] S. Arena, S. Benvenuti, and A. Bardelli. Genetic analysis of the kinome and phosphatome in cancer. *Cellular and Molecular Life Sciences*, 62(18):2092–2099, 2005.

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

[3] M. S. Bansal, E. J. Alm, and M. Kellis. Reconciliation revisited: Handling multiple optima when reconciling with duplication, transfer, and loss. *Journal of Computational Biology*, 20(10):738–754, 2013.

[4] M. S. Bansal, M. Kellis, M. Kordi, and S. Kundu. Ranger-dtl 2.0: rigorous reconstruction of gene-family evolution by duplication, transfer and loss. *Bioinformatics*, 34(18):3214–3216, 2018.

[5] M. S. Bansal, Y.-C. Wu, E. J. Alm, and M. Kellis. Improved gene tree error correction in the presence of horizontal gene transfer. *Bioinformatics*, 31(8):1211–1218, 2015.

[6] A. Bateman, L. Coin, R. Durbin, R. D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E. L. Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl_1):D138–D141, 2004.

[7] 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.

[8] M. Blum, H.-Y. Chang, S. Chuguransky, T. Grego, S. Kandasaamy, A. Mitchell, G. Nuka, T. Paysan-Lafosse, M. Qureshi, S. Raj, L. Richardson, G. A. Salazar, L. Williams, P. Bork, A. Bridge, J. Gough, D. H. Haft, I. Letunic, A. Marchler-Bauer, H. Mi, D. A. Natale, M. Necci, C. A. Orengo, A. P. Pandurangan, C. Rivoire, C. J. A. Sigrist, I. Sillitoe, N. Thanki, P. D. Thomas, S. C. E. Tosatto, C. H. Wu, A. Bateman, and R. D. Finn. The InterPro protein families and domains database: 20 years on. *Nucleic Acids Research*, 49(D1):D344–D354, 11 2020.

[9] C. P. Cantalapiedra, A. Hernández-Plaza, I. Letunic, P. Bork, and J. Huerta-Cepas. eggnog-mapper v2: functional annotation, orthology assignments, and domain prediction at the metagenomic scale. *Molecular biology and evolution*, 38(12):5825–5829, 2021.

[10] U. Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 43(D1):D204–D212, 2015.

[11] L. A. David and E. J. Alm. Rapid evolutionary innovation during an archaean genetic expansion. *Nature*, 469:93–96, 2011.

[12] J.-P. Doyon, C. Scornavacca, K. Y. Gorbunov, G. J. Szöllosi, V. Ranwez, and V. Berry. An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. In E. Tannier, editor, *RECOMB-CG*, volume 6398 of *Lecture Notes in Computer Science*, pages 93–108. Springer, 2010.

[13] R. C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[14] 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.

[15] T. Fujisawa, R. Narikawa, S.-i. Maeda, S. Watanabe, Y. Kanesaki, K. Kobayashi, J. Nomata, M. Hanaoka, M. Watanabe, S. Ehira, et al. Cyanobase: a large-scale update on its 20th anniversary. *Nucleic acids research*, 45(D1):D551–D554, 2017.

[16] 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.

[17] S. Kundu and M. S. Bansal. SaGePhy: An improved phylogenetic simulation framework for gene and subgene evolution. *Bioinformatics (in press)*, 2019.

[18] 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.

[19] 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.

[20] L. Li and M. S. Bansal. Simultaneous multi-domain-multi-gene reconciliation under the domain-gene-species reconciliation model. In Z. Cai, P. Skums, and M. Li, editors, *Bioinformatics Research and Applications*, pages 73–86, Cham, 2019. Springer International Publishing.

[21] H. Menet, V. Daubin, and E. Tannier. Phylogenetic reconciliation. *PLOS Computational Biology*, 18(11):1–29, 11 2022.

[22] H. Menet, A. N. Trung, V. Daubin, and E. Tannier. Host-symbiont-gene phylogenetic reconciliation. *Peer Community Journal*, 3, 2023.

[23] T. Miyata and H. Suga. Divergence pattern of animal gene families and relationship with the cambrian explosion. *BioEssays*, 23(11):1018–1027, 2001.

[24] S. A. Muhammad, B. Sennblad, and J. Lagergren. Species tree-aware simultaneous reconstruction of gene and domain evolution. *bioRxiv*, 2018.

[25] C. Scornavacca, W. Paprotny, V. Berry, and V. Ranwez. Representing a set of reconciliations in a compact way. *Journal of Bioinformatics and Computational Biology*, 11(02):1250025, 2013.

[26] A. Stamatakis. Raxml version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313, 2014.

[27] M. Stolzer, H. Lai, M. Xu, D. Sathaye, B. Vernot, and D. Durand. Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. *Bioinformatics*, 28(18):409–415, 2012.

[28] 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.

[29] A. Tofigh, M. T. Hallett, and J. Lagergren. Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 8(2):517–535, 2011.

[30] H. Tordai, A. Nagy, K. Farkas, L. Banyai, and L. Patthy. Modules, multidomain proteins and organismic complexity. *FEBS Journal*, 272(19):5064–5078, 2005.

[31] F. D. K. Tria, G. Landan, and T. Dagan. Phylogenetic rooting using minimal ancestor deviation. *Nature ecology & evolution*, 1(1):1–7, 2017.

[32] 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.

[33] T. Wade, L. T. Rangel, S. Kundu, G. P. Fournier, and M. S. Bansal. Assessing the accuracy of phylogenetic rooting methods on prokaryotic gene families. *PloS one*, 15(5):e0232950, 2020.

[34] 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.

[35] 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.

[36] S. Zaman and M. S. Bansal. On partial gene transfer and its impact on gene tree reconstruction. In L. Jin and D. Durand, editors, *Comparative Genomics*, pages 168–186, Cham, 2022. Springer International Publishing.

[37] O. Zhaxybayeva, J. P. Gogarten, R. L. Charlebois, W. F. Doolittle, and R. T. Papke. Phylogenetic analyses of cyanobacterial genomes: quantification of horizontal gene transfer events. *Genome research*, 16(9):1099–1108, 2006.

Abhijit Mondal received the Ph.D. degree in Computer Science and Engineering from the University of Connecticut, USA, in 2022. His primary areas of expertise are algorithm design, bioinformatics, and machine learning. He completed his B.Sc. degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology in 2016. He currently works as a software engineer in Seattle, USA.

Mukul S. Bansal is an associate professor with the Department of Computer Science and Engineering at the University of Connecticut, USA. His research interests are in computational biology and bioinformatics, with an emphasis on computational molecular evolution. He received the PhD degree in computer science from Iowa State University in 2009. He was an Edmond J. Safra postdoctoral fellow at the School of Computer Science at Tel Aviv University until December 2010, and a postdoctoral associate at the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology until August 2013.

SUPPLEMENTARY MATERIAL

TABLE S1
Basic statistics for real dataset. The table shows the numbers of genes, domains, domain families, and genes with at least one domain found within the 11 chosen cyanobacterial genomes after various annotation and filtering steps. The first filtering step was to filter out genes for which we could not find a Uniprot ID. The second filtering step (last row of table) was to remove all genes that could not be assigned to a gene family using eggNOG-mapper v2.

Species Name	# Genes	# Genes found in Uniprot	# Domains	# Domain families	# Genes with ≥ 1 domain
Nostoc punctiforme ATCC29133	6690	6690	9043	2549	5038
Crocosphaera watsonii WH8501	5958	5958	6693	2073	4570
Synechocystis PCC6803	3661	3212	3827	1823	2515
Trichodesmium erythraeum IMS101	4451	4451	6560	2056	3405
Gloeobacter violaceus PCC7421	4431	4111	4956	1926	3138
Anabaena PCC7120	6135	5843	7076	2324	4246
Thermosynechococcus elongatus BP1	2476	2196	2782	1492	1788
Synechococcus WH8102	2517	2514	2768	1618	1891
Prochlorococcus marinus MIT9313	2850	2846	2542	1529	1735
Prochlorococcus marinus CCMP1375	1882	1882	2033	1359	1407
Prochlorococcus marinus MED4	1948	1948	2000	1339	1405
Total	42999	41651	50280	3413	31138
Total after EggNOG mapping	38654	38654	49486	3386	30775