A New Paradigm for Identifying Reconciliation-Scenario Altering Mutations Conferring Environmental Adaptation

Roni Zoller
Ben Gurion University of the Negev, Israel
ronizo@post.bgu.ac.il

Meirav Zehavi[1]
Ben Gurion University of the Negev, Israel
meiravze@bgu.ac.il

Michal Ziv-Ukelson[1]
Ben Gurion University of the Negev, Israel
michaluz@cs.bgu.ac.il

Abstract

An important goal in microbial computational genomics is to identify crucial events in the evolution of a gene that severely alter the duplication, loss and mobilization patterns of the gene within the genomes in which it disseminates. In this paper, we formalize this microbiological goal as a new pattern-matching problem in the domain of Gene tree and Species tree reconciliation, denoted "Reconciliation-Scenario Altering Mutation (RSAM) Discovery". We propose an $O(m \cdot n \cdot k)$ time algorithm to solve this new problem, where m and n are the number of vertices of the input Gene tree and Species tree, respectively, and k is a user-specified parameter that bounds from above the number of optimal solutions of interest. The algorithm first constructs a hypergraph representing the k highest scoring reconciliation scenarios between the given Gene tree and Species tree, and then interrogates this hypergraph for subtrees matching a pre-specified RSAM Pattern. Our algorithm is optimal in the sense that the number of hypernodes in the hypergraph can be lower bounded by $\Omega(m \cdot n \cdot k)$. We implement the new algorithm as a tool, denoted RSAM-finder, and demonstrate its application to the identification of RSAMs in toxins and drug resistance elements across a dataset spanning hundreds of species.

2012 ACM Subject Classification Applied computing → Bioinformatics

Keywords and phrases Gene tree, Species tree, Reconciliation

Digital Object Identifier 10.4230/LIPIcs.WABI.2019.9

Supplement Material https://github.com/ronizoller/RSAM

Funding Supported by ISF grants no. 1176/18 and no. 939/18. This work was supported by the Lynn and William Frankel Center for Computer Science.

Acknowledgements We thank the anonymous WABI reviewers for very helpful comments.

1 Introduction

Prokaryotes can be found in the most diverse and severe ecological niches of the planet. Adaptation of prokaryotes to new niches requires expanding their repertoire of protein families, via two evolutionary processes: first, by selection of novel gene mutants carrying stable genetic alterations that confer adaptation, and second, by dissemination of an adaptively mutated gene. These two processes are correlated: an adaptation-conferring mutation in a

[1] Corresponding authors.

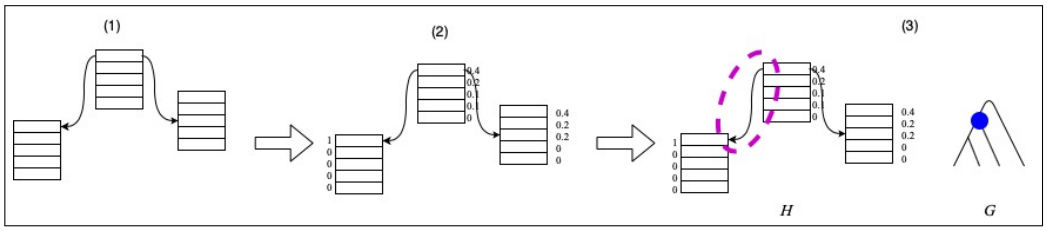

Figure 1 High-level overview of the RSAM-finder algorithm.

gene could accelerate its mobilization across bacterial lineages populating the corresponding environmental niche [25], and vice-versa, the mobilization of a gene by transposable elements increases its chances to mutate or "pick up" novel genomic context. Thus, an important research goal is to identify gene-level mutations that affect the spreading pattern of the mutated gene within and across the genomes harboring it.

For example, consider mutations conferring adaptation of bacteria to a human-pathogenesis environment. Here, a mutation to a resistance or virulence factor could enhance pathogenic adaptation, thus increasing the horizontal mobilization of the mutated gene within other human pathogens inhibiting this niche [25]. In this case, we say that the mutation has a *causal association* with the observed dissemination pattern of the mutated gene (i.e. the increased mobilization of the gene among pathogenic bacteria). Identifying such mutations could inform infectious disease monitoring and outbreak control, and assist in identifying potential drug targets.

The co-evolution of genes and their host species is classically described by computing the most parsimonious reconciliation scenario between a given Gene tree G and the corresponding Species tree S, that is, a mapping of each vertex $u \in G$ to a vertex $x \in S$. Three major evolutionary processes, traditionally considered by reconciliation approaches, are horizontal gene transfer, gene duplication, and gene loss [36]. Each mapping of a vertex $u \in G$ to a vertex $x \in S$ is associated with one of these evolutionary events, and assigned a cost, accordingly. The optimization problem of computing a least-cost reconciliation between G and S, where the total cost is computed as the sum of the costs assigned to each of the mappings, is denoted *Duplication-Loss-Transfer (DLT) Reconciliation*.

Motivated by examples such as the one given above, we formalize a new pattern-matching problem in the domain of DLT reconciliation. Given are a Gene tree G, a corresponding Species tree S, a mapping σ from the leaves of G to the leaves of S, and (optional) an environmental annotation labeling the leaves of the input trees. Let $\mathcal{H}$ denote some data structure, to be defined later in the paper, that models the space of reconciliations between G and S. A *DLT Reconciliation Scenario Pattern* denotes a mapping between a vertex $u \in G$ to a vertex $x \in S$, which obeys a set of user-defined specifications regarding the corresponding reconciliation event, the labels on the paired vertices, and other features associated with the mapping. Mappings between pairs of vertices ($u \in G$, $x \in S$) that abide by the requirements specified by P are denoted *instances of P in $\mathcal{H}$*. Given a pre-specified DLT Reconcilation Scenario pattern P and a data structure $\mathcal{H}$ modeling the space of reconciliations between G and S, a *Reconciliation Scenario Altering Mutation (RSAM) of P in $\mathcal{H}$* is a vertex $v \in G$ representing a gene mutation with a putative causal association to instances of P in $\mathcal{H}$. The *RSAM Discovery problem* is to identify RSAMs in G.

In what follows, we propose a three-stage solution to the RSAM Discovery problem defined above (illustrated in Fig. 1). The first stage constructs a hypergraph $\mathcal{H}$ that recursively aggregates all the k-best reconciliations of G and S. Each supernode in $\mathcal{H}$ consists of k

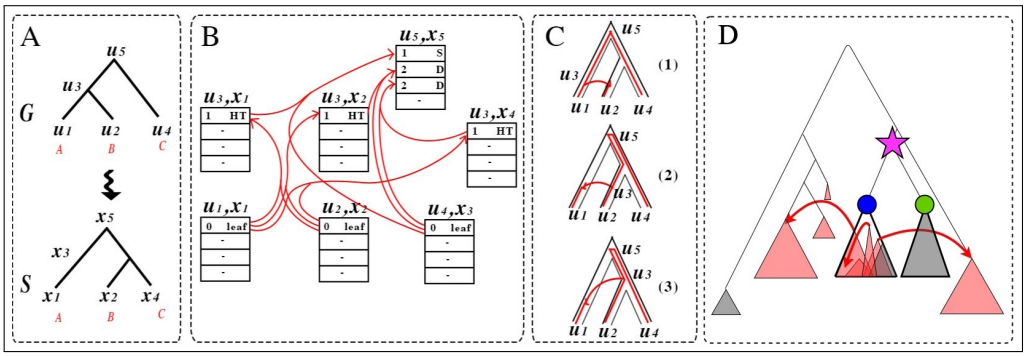

Figure 2 Various aspects of the problem addressed in this paper. (A) The input trees G and S. (B) An example of a hypergraph constructed based on the input trees and parameter k. (C) Three top-scoring solutions. (D) A putative RSAM (blue vertex) with causal association to the mobilization pattern of the gene among species labeled with the red environmental annotation.

hypernodes, where each hypernode represents a partial solution for the DLT-reconciliation problem. Our hypergraph-ensemble approach is based on a model proposed by [24] for network evolution, where here we extend and adapt it to the DLT model. This hypergraph of k-best reconciliations, intended to provide some robustness to the noise typical of this data, will serve as the search-space for the pattern-matching stage. The second stage of our proposed solution consists of assigning a probability to each partial solution, that is, to each hypernode of $\mathcal{H}$. Finally, in the third stage, instances of the sought RSAM-pattern P are identified within $\mathcal{H}$, and RSAM-ranking scores are assigned accordingly to the vertices of G. Based on these scores, vertices representing putative RSAMs are identified in G and subjected to biological interpretation.

The construction of $\mathcal{H}$, in the first stage, is the computational bottleneck of the RSAM-Discovery pipeline mentioned above. Here, we adapt the approach proposed by Bansal et al. [3] for the basic, one-best variant of DLT reconciliation, extending it to an efficient k-best variant. This yields an $O(m \cdot n \cdot k)$ time algorithm for the problem, where m and n are the number of vertices of the input Gene tree and Species tree, respectively, and k is a user-specified parameter that bounds from above the number of optimal solutions of interest. Our algorithm is optimal in the sense that the number of hypernodes in the hypergraph can be lower bounded by $\Omega(m \cdot n \cdot k)$.

Our proposed solution to the problem defined in this paper is implemented as a tool called RSAM-finder, publicly available on GitHub. We assert the performance of RSAM-finder in large scale simulations, and exemplify its application to the identification of RSAMs in toxins and drug resistance elements across a dataset spanning hundreds of species.

Previous Related Works. The DLT Reconciliation problem has been extensively studied. In particular, two main DLT variants have been considered: (1) the *undated DLT-reconciliation* where the species are undated, and (2) the *fully-dated DLT-reconciliation* where either each vertex in the Species (and Gene) tree is associated with an estimated date or the vertices of the Species (and Gene) tree are associated with a total order, and any reconciliation must respect these dates (i.e. an HT event can occur only between co-existing species).

In the acyclic version of these variants, there cannot exist two genes such that one is a descendant of the other, yet the descendant is mapped (in the Species tree S) to an ancestor of the other. Tofigh et al. [36] showed that the acyclic undated version is NP-Hard. However, the acyclic dated version becomes polynomially solvable [19].

Tofigh et al. [36, 35] and David et al. [12] studied a version of the undated (cyclic) problem that ignores losses and proposed an $O(mn^2)$ dynamic programming algorithm for it. They also gave a fixed-parameter tractable algorithm for enumerating all optimal solutions. The time complexity of the algorithm was improved to $O(mn)$ in [35] (under a restricted model that ignores the losses) and in [3] (which does not ignore losses).

It is well-known that the biological data used as input to the DLT Reconciliation problem could be inaccurate, whether due to a sequencing problem, a problem in the reconstruction of G or S [5], or due to some other problem caused by noise. To overcome this problem, previous works try to examine more than one optimal solution, for example [13, 27]. A probabilistic method for exploring the space of optimal solutions was suggested in [4, 14], where the latter was improved in [15]. Additional studies considered a space of candidate co-optimal scenarios within special variants of the DLT problem, some of which employed special constraints to drive the search [30, 34, 22, 10]. Although all of the previous works reviewed in this paragraph compute a space of candidate reconciliation scenarios, none of these works mentioned considered the application of pattern matching on this space, as we do in this work.

DLT Reconcilation algorithm variants, where the reconcilation computation is guided by constraints derived from vertex-coloring information, were proposed in applications studying host-parasite co-evolution, such as [6], where the vertex coloring (in both G and S) represents the geographical area of residence. However, the applied constraints were "hard-wired" to the specific problem addressed in that paper. In contrast, the approach proposed in this paper is more general, supporting a pattern-search that is guided by a user-defined pattern. Our tool RSAM-finder provides the users with a query language able to express more robust patterns, according to the various applications where the pattern-search is to be employed.

2 Preliminaries

For a (binary) rooted tree T, let $L(T)$, $V(T)$, $I(T)$ and $E(T)$ denote the sets of leaves, vertices, internal vertices and edges, respectively, of T. Additionally, let $V(T)^\star$ denote the set of finite (ordered) vectors over $V(T)$, i.e. $V(T)^\star = \{(v_1, v_2, \ldots, v_\ell) \mid v_i \in V(T) \text{ for all } i \in \{1, \ldots, \ell\}, \ell \in \mathbb{N}\}$. When T is clear from context, let $V^\star = V(T)^\star$. Throughout, we treat any (binary) rooted tree T as a directed graph whose edges are directed from root to leaves. Then, if $(u, v) \in E(T)$, we say that v is a *child* of u, and u is the *parent* of v. For $u, v \in V(T)$, the notation $v \leq_T u$ signifies that v is a *descendant* of u (alternatively, u is an *ancestor* of v), i.e. there is a directed path from u to v or $u = v$. We say that v is a proper descendent (proper ancestor) of u if $v \leq_T u$ ($v \geq_T u$) and $u \neq v$, denoted $<_T$ ($>_T$). When both $u \not\leq_T v$ and $v \not\leq_T u$, we say that u and v are *incomparable*.

For any $u, v \in V(T)$, let $d_T(u, v)$ denote the number of edges in the (unique simple undirected) path between u and v in T. When T is clear from context, we drop it from the notations $v \leq_T u$ and $d_T(u, v)$. For any $u \in V(T)$, let T_u denote the subtree of T rooted in u (then, $V(T_u) = \{v \in V(T) \mid v \leq u\}$).

DLT Scenario. A *DLT scenario* for two binary trees G (the *Gene tree*) and S (the *Species tree*) is a tuple $\langle \sigma, \gamma, \Sigma, \Delta, \Theta, \Xi \rangle$ where $\sigma : L(G) \to L(S)$ is a mapping of the leaves of G to the leaves of S, $\gamma : V(G) \to V(S)$ is a mapping of the vertices of G to the vertices of S, and (Σ, Δ, Θ) is a partition of $I(G)$ (the set of internal vertices of G) into three event classes: *Speciation* (Σ), *Duplication* (Δ) and *Horizontal Transfer* (Θ). The subset $\Xi \subseteq E(G)$ specifies which edges are involved in horizontal transfer events. Additionally, the following constraints should be satisfied.

1. **Consistency of σ and γ.** For each leaf $u \in L(G)$, $\gamma(u) = \sigma(u)$. *This constraint ensures that γ respects σ – that is, each leaf of G is mapped to the species where it is found.*
2. **Consistency of γ and ancestorship relations in S.** For each $u \in I(G)$ with children v and w:
   a. $\gamma(u) \not<_S \gamma(v)$ and $\gamma(u) \not<_S \gamma(w)$. *This constraint ensures that each of the two children (in G) of the gene u is mapped by γ to a species that is* not *a proper ancestor (in S) of the species to which the gene u is mapped; thus, it can be either a descendant of u or incomparable to u.*
   b. At least one of $\gamma(v)$ and $\gamma(w)$ is a descendant of $\gamma(u)$. *This constraint ensures that at least one of the two children (in G) of the gene u is mapped by γ to a species that is a descendant (in S) of the species to which the gene u is mapped.*
3. **Identifying horizontal transfer edges.** For each edge $(u, v) \in E(G)$, it holds that $(u, v) \in \Xi$ if and only if $\gamma(u) \not\leq_S \gamma(v)$ and $\gamma(v) \not\leq_S \gamma(u)$. *This constraint identifies which edges are horizontal transfer edges – specifically, a horizontal transfer edge is an edge $(u, v) \in E(G)$ from a gene u to a gene v that are mapped to species $\gamma(u)$ and $\gamma(v)$ that are incomparable.*
4. **Associating events with internal vertices.** For each $u \in I(G)$ with children v, w:
   a. **Speciation.** $u \in \Sigma$ only if both *(i)* $\gamma(u) = \mathsf{lca}(\gamma(v), \gamma(w))$ and *(ii)* $\gamma(v)$ and $\gamma(w)$ are incomparable (i.e. $\gamma(v) \not\leq_S \gamma(w)$ and $\gamma(w) \not\leq_S \gamma(v)$).
   b. **Duplication.** $u \in \Delta$ only if $\gamma(u) \geq_S \mathsf{lca}(\gamma(v), \gamma(w))$.
   c. **Horizontal transfer.** $u \in \Theta$ if and only if either *(i)* $(u, v) \in \Xi$ or *(ii)* $(u, w) \in \Xi$

Fig. S1 demonstrates a DLT scenario.

Costs. We let c_Δ and c_Θ denote the costs of a duplication event and a horizontal transfer event, respectively. Accordingly, the cost of a DLT scenario is defined as $|\Delta| \cdot c_\Delta + |\Theta| \cdot c_\Theta$. When seeking a "best" DLT scenario, the goal is to find one that minimizes this cost. It is straightforward to extend the cost model, and the corresponding algorithm, to take losses into account, yet for lack of space, these details are omitted throughout the paper.

3 Hypergraph of k-Best Scenarios

To represent *k-best solutions*,[2] we use a directed hypergraph denoted by $\mathcal{H}$ based on the notation in [17]. The hypergraph is a tuple $\mathcal{H} = \langle V, E \rangle$ where V is a finite set of vertices, and E is a finite set of (directed) hyperedges defined as follows. Each $e \in E$ is a pair $\langle T(e), h(e) \rangle$, where $h(e) \in V$ is the *head* of e and $T(e) \in V^*$ (i.e. $T(e)$ is a vector of vertices in V) is its *tail*. In our settings, $|T(e)| = 2$ for every $e \in E$. In what follows, we define the hypernodes and hyperedges of $\mathcal{H}$ with respect to our problem. In Fig. 2.B, we illustrate the hypergraph corresponding to the input G and S given in Fig. 2.A, where $k = 4$. Each hypernode (u, x, i) represents a DLT scenario, annotated with its cost and with the event that occurred between u and x in this scenario (where 'S', 'D' and 'HT' stand for speciation, duplication and horizontal transfer, respectively). For additional details, see the Supplementary Materials.

- **Hypernodes.** For every vertex u in G, a vertex x in S and an integer $i \in \{1, \ldots, k\}$, we have a node (u, x, i) in $\mathcal{H}$. Such a node (u, x, i) is associated with the i^{th} best (where ties are broken arbitrarily) solution mapping the subtree of G rooted at u to the subtree of S

[2] That is, k DLT scenarios of the highest score(s), where ties (if any exist) can be broken arbitrarily.

rooted in x that is a DLT scenario. In addition, for every integer $i \in \{1, \ldots, k\}$ we have a node $(root, i)$ in the hypergraph $\mathcal{H}$. Such a node $(root, i)$ is associated with the i^{th} best solution of mapping G (entirely) to any subtree of S. Each node (u, x, i) has a score $\mathsf{c}(u, x, i)$, and each node $(root, i)$ has a score $\mathsf{c}(root, i)$. Moreover, each node (u, x, i) is associated with the event corresponding to the mapping of u and x in the DLT scenario of (u, x, i) (speciation, duplication and horizontal transfer), denoted $\mathsf{event}(u, x, i)$.

- **Supernodes.** For any vertex $u \in V(G)$ and vertex $x \in V(S)$, we define the *supernode* (u, x) as the list $\{(u, x, i) : 1 \leq i \leq k\}$ (i.e. (u, x) is the set of k hypernodes corresponding to the mapping of the subtree of G rooted in u to the subtree of G rooted in x). This notation will simplify our presentation.
- **Hyperedges.** Recall that each hypernode $(u, x, i) \in V(\mathcal{H})$ describes a DLT scenario. Each hypernode has exactly one incoming hyperedge, but it can have multiple outgoing hyperedges. In particular, for each hypernode $(u, x, i) \in V(\mathcal{H})$, the (only) incoming hyperedge $e = \langle T(e), h(e) \rangle = \langle [(v, y, j), (w, z, r)], (u, x, i) \rangle$ describes the mapping of the subtrees of the children of u, namely, v and w, in the scenario of (u, x, i); here, the subtree of v is mapped to the subtree of y as in the scenario of (v, y, j), and the subtree of w is mapped to the subtree of z as in the scenario of (w, z, r).

4 Framework and Algorithms

In this section, we elaborate on each of the three stages of the workflow in Section 1.

4.1 Stage 1: Hypergraph Construction

The first stage of our framework constructs the hypergraph described in Section 3. To this end, we develop an efficient algorithm that runs in time $O(m \cdot n \cdot k)$. The technical details (including pseudocode) are given in the Supplementary Materials.

The Algorithm. We iterate over all $u \in V(G)$ in postorder, as well as over all $x \in V(S)$ in postorder. (However, as explained immediately, when we consider a vertex $u \in V(G)$, after iterating over all vertices $x \in V(S)$ in postorder, we also iterate over all vertices $x \in V(S)$ in preorder.) In each iteration, corresponding to a pair (u, x), we construct three lists: p_Σ (speciation), p_Δ (duplication) and p_Θ (horizontal transfer). Specifically, p_Σ should be a list of k-best solutions that are DLT scenarios where the subtree of G rooted in u is mapped to the subtree of S rooted in x under the restriction that the event corresponding to matching u and x is speciation. The meaning of the lists p_Δ and p_Θ is similar, where the restriction of speciation is replaced by duplication or horizontal transfer, respectively. Having these three lists suffices to construct the hypernode (u, x).

To avoid repetitive computation, we maintain two additional lists: $\mathsf{subtree}$ and incomp. Intuitively, $\mathsf{subtree}(u, x, i)$ is the i^{th} best cost of a reconciliation of the subtree of G rooted in u with some subtree of S whose root is a vertex y that is a descendant of x, and $\mathsf{incomp}(u, x, i)$ is the i^{th} best cost of a reconciliation of the subtree of G rooted in u with some subtree of S whose root is a vertex y incomparable to x. We use $\mathsf{subtree}$ to speed-up the computation of p_Σ, p_Δ and p_Θ, and incomp to speed-up the computation of p_Θ. The notations $\mathsf{subtree}(u, x)$ and $\mathsf{incomp}(u, x)$ refer to the lists of the k-best scores $\{\mathsf{subtree}(u, x, i)\}_{i=1}^{k}$ and $\{\mathsf{incomp}(u, x, i)\}_{i=1}^{k}$, respectively, similarly to our usage of the notation of a supernode.

The efficient computation of p_Σ, p_Δ and p_Θ, along with the maintenance of subtree and incomp themselves, is highly non-trivial. For space constraints, the technical details (in particular, pseudocode) are delegated to the Supplementary Materials Section 2.1. On a high-level, we first initialize all five lists to contain only costs of ∞; then, still in the initialization phase, we add hypernodes that match between leaves of G and S in accordance with σ and update subtree consequently. After the initialization, the main computation considers each $u \in V(G)$ in postorder, and performs two steps. In the first step, we consider each $x \in V(S)$ in postorder. Then, for each $i \in \{1, \ldots, k\}$, we compute $p_\Sigma(u, x, i)$, $p_\Delta(u, x, i)$ and $p_\Theta(u, x, i)$ based on somewhat involved recursive formulas. Afterwards, we construct the surpernode (u, x), as well as compute the list $\mathsf{subtree}(u, x)$. In the second step, we consider each $x \in I(S)$ with children y and z in preorder, and compute the lists $\mathsf{incomp}(u, y)$ and $\mathsf{incomp}(u, z)$.

Having constructed all hypernodes of the form (u, x, i) along with their ingoing hyperedges, it is trivial to construct the hypernodes of the form $(root, i)$ and their ingoing edges. Thus, we conclude the outline with statements of correctness and running time proved in Supplementary Materials Section 2.1.

▶ **Lemma 1.** *Given an instance (G, S, σ) of the DLT problem and a positive integer k, the efficient algorithm correctly constructs a hypergraph $\mathcal{H}$ that represents k-best solutions for (G, S, σ).*

▶ Remark 2. Given an instance (G, S, σ) of the DLT problem and a positive integer k, the efficient algorithm runs in time $O(m \cdot n \cdot k)$.

4.2 Stage 2: Assigning Probabilities

In the second stage, we assign a probability to each hypernode so that a hypernode with best score has the highest probability, and hypernodes with score ∞ (the worst possible score) have probability 0.

Weight Computation. Let $\gamma \in \mathbb{R}^+$ be a user-specified parameter. As γ grows lower, hypernodes with higher (worse) scores are assigned probabilities much lower than hypernodes with lower scores.

Denote $r = root$, and let $\mathsf{m}(r)$ be the largest integer $i \in \{1, \ldots, k\}$ such that $\mathsf{c}(r, i) \neq \infty$ (the notation $(root, i)$ was defined in Section 3). For a node (r, i) where $i \in \{1, \ldots, \mathsf{m}(r)\}$, define $\mathsf{w}'(r, i) = e^{\gamma \frac{\mathsf{c}(r,1) - \mathsf{c}(r,i)}{\mathsf{c}(r,1) - \mathsf{c}(r,\mathsf{m}(r))}}$. Then, the weight of a node (r, i), which stands for the (unconditional) probability that the scenario described by (r, i) happens, is defined as follows: if $i \in \{1, \ldots, \mathsf{m}(r)\}$, then $\mathsf{w}(r, i) = \frac{\mathsf{w}'(r,i)}{\sum_{j=1}^{\mathsf{m}(r)} \mathsf{w}'(r,j)}$; otherwise (i.e. if $i \in \{\mathsf{m}(r) + 1, \mathsf{m}(r) + 2, \ldots, k\}$), $\mathsf{w}(r, i) = 0$.

We now turn to define the weight of a hypernode (u, x, i), which should stand for the (unconditional) probability that the scenario described by (u, x, i) happens. The definition is recursive. In the basis, where u is the root of G, we define $\mathsf{w}(u, x, i)$ (for any $x \in V(S)$ and $i \in \{1, \ldots, k\}$) as follows: if there exists an index $j \in \{1, \ldots, k\}$ such that (r, j) is derived from (u, x, i) (here, it means that they represent the same scenario), then $\mathsf{w}(u, x, i) = \mathsf{w}(r, j)$; otherwise, $\mathsf{w}(u, x, i) = 0$.

Now, consider v that is not the root of G. We define $\mathsf{w}(v, y, i)$ (for any $y \in V(S)$ and $i \in \{1, \ldots, k\}$) as follows. First, let $\mathsf{D}(v, y, i)$ denote the collection of nodes (u, x, j) such that $\mathsf{c}(u, x, j)$ was derived from $\mathsf{c}(v, y, i)$ – in other words, the hypergraph has an hyperedge directed from (v, y, i) (and some other node) to (u, x, j). In particular, u is the parent of v in G, hence the weight $\mathsf{w}(u, x, j)$ is calculated before the weight $\mathsf{w}(v, y, i)$. Then, define $\mathsf{w}(v, y, i) = \sum_{(u,x,j) \in \mathsf{D}(v,y,i)} \mathsf{w}(u, x, j)$.

Note that $\sum_{i\in\{1,\ldots,k\}} \mathsf{w}(r,i) = 1$. As an additional check, we prove the following in Supplementary Materials Section 3.

▶ **Lemma 3.** *For any two compatible leaves $u \in L(G)$ and $x \in L(S)$, $\mathsf{w}(u,x,1) = 1$.*

Time Complexity. Iterating the hypergraph in $O(|V(\mathcal{H})|) = O(m \cdot n \cdot k)$ time.

4.3 Stage 3: Pattern Discovery

The current version of RSAM-finder allows pattern queries to be specified as follows. A pattern specification consists of a tuple $(\mathsf{EV}, \mathsf{color}, \mathsf{distance})$ where:

1. $\mathsf{EV} \subseteq \{\mathsf{S}, \mathsf{D}, \mathsf{HT}\}$ specifies the evolutionary event of the pattern (S for speciation, D for duplication and HT for horizontal transfer).
2. $\mathsf{color} \in \{\mathsf{red}, \mathsf{black}, \mathsf{None}\}$ specifies the color representing the environmental niche to which the sought RSAM confers adaptation.
3. $\mathsf{distance} \in \{\mathsf{True}, \mathsf{False}\}$ is a boolean indicator specifying whether or not to consider edge lengths (representing evolutionary distances) in the pattern specification.

For a colored query (having the second parameter in the specification set to red or black), the user is expected to provide, as part of the input, a function $\mathsf{colors} : L(X) \to \Upsilon$ where X specifies whether the pattern refers to a subtree of S or a subtree of G, and $\Upsilon = \{\mathsf{red}, \mathsf{black}\}$. Here, colors represent a binary environmental annotation of the leaves. Then, a preprocessing step is applied, in which the nodes of S and G are colored based on the colors assigned to the leaves of the subtree they root. We omit the technical details entailing the implementation of this preprocessing step to Supplementary Materials Section 4.1.

In addition to the settings described above, the user can select one of two modes:

1. **Single-pattern mode.** In this mode, the user specifies a single pattern and a threshold, and the sought RSAMs are identified as nodes $u \in I(G)$ such that G_u is enriched in the pattern, and $|V(G_u)|$ is bounded from below by the specified threshold.
2. **Dual-pattern (contrasting) mode.** In this mode, the user specifies two patterns and one threshold, and the sought RSAMs are identified as nodes $u \in I(G)$ with children $v, w \in V(G)$ such that $|V(G_u)|$ is enriched with one pattern while $|V(G_w)|$is enriched with the other pattern. Here, the threshold that bounds (from below) the subtree-size refers to $|G_v|$ and $|G_w|$.

The following three queries will be exemplified in Section 5,

- $Q_1 : (\{\mathsf{D}\}, \mathsf{None}, \mathsf{False})$. Q_1 identifies vertices $u \in I(G)$, such that G_u is enriched in duplication events.
- $Q_2 : (\{\mathsf{S}, \mathsf{HT}\}, \mathsf{None}, \mathsf{True})$. Q_2 identifies vertices $u \in I(G)$, such that G_u is enriched in speciation and horizontal transfer events (at "the expense" of duplication events).
- $Q_3 : ((\{\mathsf{HT}\}, \mathsf{red}, \mathsf{True}), (\{\mathsf{S}, \mathsf{D}, \mathsf{HT}\}, \mathsf{black}, \mathsf{False}))$. Q_3 identifies vertices $u \in I(G)$ with children $v, w \in V(G)$, such that G_v is enriched in red-to-red horizontal transfer events and G_w is enriched in any black events (illustrated in Fig. 2.D).

The Pattern Identification algorithm proceeds as follows.

1. For each pattern $P = (\mathsf{EV}, \mathsf{color}, \mathsf{distance})$ and for each hypernode $(u, x, i) \in V(\mathcal{H})$, check whether both $\mathsf{event}(u, x, i) \in \mathsf{EV}$ and the colors obey the requirements derived from the color field of the pattern specification. (described in more details in Supplementary Materials Section 4.1.1). If so, mark (u, x, i) as interesting.

2. Reflect the interesting nodes identified in $\mathcal{H}$ to G, by assigning corresponding weights to $V(G)$; Each $u \in I(G)$ is assigned a score, which is the cumulative probabilities of instances of the pattern found in G_u, normalized by the number of possible events in G_u. Additional book-keeping details regarding how this score is computed are given in Supplementary Materials Section 4.2.
3. Based on the specified mode of the query (single pattern or dual pattern), identify the t top scoring vertices $u \in I(G)$. In case of a single-pattern mode, the scores are as defined in (2). In case of dual-pattern mode, let P_1 and P_2 be the patterns. To each $u \in V(G)$ with children $v, w \in V(G)$, we assign two scores: the first score of u is the score of v for P_1 (as defined in (2)) plus the score of w for P_2, and the second is the score of w for P_1 plus the score of v for P_2.

Time Complexity. Iterating over the hypergraph takes $O(|V(\mathcal{H})|) = O(m \cdot n \cdot k)$ time.

5 Applications

In this section we exemplify preliminary applications of RSAM-finder to genomic analysis. Mobile elements in prokaryotes contribute greatly to the process of gene duplication and dissemination. For example, toxins and antibiotic resistance factors, conferring adaptation to the pathogenesis environment, are often encoded on plasmids, prophages, transposons and other mobile elements in bacteria [25]. Thus, we exemplify two microbiological applications of RSAM-finder by applying query patterns Q_1, Q_2, and Q_3 (defined in Section 4.3), to the discovery of RSAMs in toxins and resistance factors.

Supplementary Materials Section 5.2 reports on large scale simulations, where we demonstrate the engine's tolerance to noise, and Supplementary Materials Section 5.3 measures the practical running times of the proposed hypergraph construction algorithm as a function of increasing input size.

Methods and Data Bases. Genes in our experiment are represented by their membership in a Cluster of Orthologous Genes [33]. The STRING database [32] was used to extract the chromosomal protein sequences for the COGs of interest, annotated with their corresponding species names as well as the corresponding NCBI IDs. Protein sequences were subjected to multiple sequence alignment and dendogram construction via Clustal Omega [28]. The list of NCBI IDs was used as input for the *NCBI Taxamony Browser* which provided a (non-binary) Species tree. Both Gene and Species trees were converted to binary trees via the Ape R package [26]. Habitat labels for the species were extracted from PATRIC, and missing tags were manually annotated by information from the GOLD database [23] and from literature. MEME motif discovery [2] was employed to identify sequence motifs distinguishing the RSAM-subtree gene sequences from the background. CD Search [20] was employed to seek statistically significant discriminating domain-level mutations (i.e. the gain or loss of a protein functional domain). The simulator and our algorithm were implemented in Python, using NetworkX package, DendroPy [31] and ETE Toolkit [18]. Visualizations of the trees and plots were created using Matpllotlib and Seaborn tools.

RSAM Discovery in a Chromosomally Acquired Toxin. Bacterial toxin-antitoxin (TA) systems are diverse and widespread in the prokaryotic kingdom. They are composed of closely linked genes encoding a *stable toxin* that can harm the host cell and its cognate *unstable antitoxin*, which protects the host from the toxin's deleterious effect. TA systems

invade bacterial genomes through horizontal gene transfer. Their role in stabilization and maintenance of plasmids or genomic islands by post-segregational killing has been thoroughly studied [25].

However, much is yet to be learned about how horizontally acquired TA systems are fixed within the population, and about the functions of the chromosomally encoded TA systems. Some TA systems, such as higBA [38], have integrated into host regulatory networks, controlling drug tolerance, growth arrest and programmed cell death.

Here we exemplify how RSAM-finder could be harnessed to advance such studies, e.g. by identifying chromosomally acquired variants of the same toxin gene, that exhibit distinct reconciliation patterns: One variant of the toxin follows a "deep and ongoing" invasion pattern (pattern Q_1), i.e. one involving abundant recent duplications within invaded genomes. The other variant follows a "wide and shallow" invasion pattern (pattern Q_2), i.e. the invaded genomes are far-apart in terms of phylogenetic distance, and a very slow rate of duplications within the invaded genomes is observed.

The Plasmid Maintenance System Killer Protein higB (represented by COG3549) is the toxin component of the TA module higBA. This toxin, which is abundant in Proteobacteria [11], is repressed by the Plasmid Maintenance System Antitoxin higA (represented by COG3093). Gene trees for COG3549, and for the chromosomes of Protobacterial species harboring it, were constructed (652 genes versus 493 species), and RSAM-Finder was applied to interrogate this dataset with queries Q_1 and Q_2. Parameters for Q_1 were set as follows: $k = 50$, and the minimum size required per sought subtree was set to 0.05 of the total number leaves of G. Parameters for Q_2 were set in the same manner, without bounding the size of sought subtrees and $c_\Delta = c_\Theta = 1$. Figures displaying G, where the top-ranking RSAM nodes are marked with a star, are provided in the supplementary materials, jointly with the corresponding sequences. Also provided is a figure displaying S, σ, and the full multiple alignment used for constructing G.

The highest-scoring RSAM identified for Q_2 roots a subtree with 23 leaf nodes (shown in Fig. S2.C), spanning classes α, β, γ and δ of proteobacteria. The sequences of the genes represented by the leaves of this subtree, denoted the "identified gene set", were subjected to distinguishing sequence motif search analysis [2], using the full set of input genes, denoted "background gene set", as background. This analysis identifies an insertion sequence represented by the motif logo given in Fig. S2.E (MEME e-value 4.5e-094). This insertion is validated by the multiple sequence alignment, based on which the gene tree was constructed (see Fig. S2.F and the corresponding multiple alignment file provided in the Supplementary Materials).

Most instances of the higB toxin (473/652) in the background gene set have the antitoxin higA in their immediate downstream position. In contrast, the instances of higA found immediately downstream the genes from the identified gene set are enriched in an additional domain: the Zn-dependent peptidase ImmA, belonging to the M78 peptidase family (14/23 vs. 14/652, hypergeometric p-value = 3.27e-23). Homologues of immA are found in many mobile genetic elements [7], and it is predicted to participate in a conserved mechanism for regulation of horizontal gene transfer.

Thus, RSAM-Finder identifies a *putative three-component variant of higBA*, consisting of a variant of the higB toxin with a conserved insertion sequence, coupled with an ImmA-higA fusion protein variant of the higA antitoxin. Further analysis of this result may reveal possible functional correlation between the insertion sequence identified within this higB toxin variant, and the additional ImmA domain characterizing the corresponding higA instances.

In contrast to the top-ranking result for pattern Q_2, the top-ranking result for pattern Q_1 identifies a subtree of G whose leaves encode members of a *two component variant of the higBA TA* (shown in Fig. S2.D). Further analysis of this result may help decipher whether the observed duplications of this invading higB variant are tolerated by the invaded genomes due to mere decay of its addictive properties, or due to some mutations conferring selective advantage to the host.

RSAM Discovery in a Beta Lactamase. Beta lactamases are versatile enzymes conferring resistance to the Beta lactam antibiotics, found in a diversity of bacterial sources. Their commonality is the ability to hydrolyze chemical compounds containing a Beta lactam ring [9]. The persistent exposure of bacterial strains to a multitude of Beta lactams has induced dynamic and continuous production and mutation of Beta lactamases in these bacteria, expanding their activity even against the newly developed Beta lactam antibiotics [29]. Thus, an important objective is to identify mutations in Beta lactamase genes conferring adaptation to human and animal hosts.

Among the known classes (A-D) of Beta lactamase, class D (represented by COG2602) is considered to be the most diverse [16]. Thus, we selected COG2602 (622 genes in 543 genomes) to exemplify the colored RSAM pattern Q_3, where colors represent a binary environmental annotation: human and animal host (219 species) were annotated "red", while species associated with all other habitats (324 species), such as soil, water and plant, were annotated "black". Parameters were set as follows: $k = 50$, the minimum size required per sought subtree was set to 0.1 of the total number leaves of G, and $c_\Delta = c_\Theta = 1$. A figure displaying G, where the top-ranking RSAM node is marked with a star, are given in the supplementary materials. Also provided are the corresponding sequences, a figure displaying the corresponding S, and σ.

Within the top-ranking result for this query, we were interested in the subtree matching the first part of pattern Q_3 (i.e. enrichment in red-to-red HT edges). The gene set represented by the leaves of this subtree, denoted "identified gene set", was found to be enriched in an additional domain, BlaR, a signal transducer membrane protein regulating Beta lactamase production (87/119 in the identified gene set versus 118/622 in the background, p-val = 3.94e-52). The only transcriptional regulator currently known for Beta lactamase genes is the repressor protein BlaI, previously predicted to operate in a two-component regulatory system together with BlaR in Class A Beta lactamase [1]. The positions adjacent to the instances of the identified gene set in the corresponding genomes were found to be enriched in BlaI (70/119 of the identified gene set instances versus 90/622 of the background gene set instances, hypergeometric p-value = 1.11e-41).

In contrast to the identified gene set, the genes represented by the subtree that matches the second part of Q_3 (frequent black HT, S and D events) are not enriched in the BlaR domain (2/36), nor is there contextual enrichment in BlaI (4/36) in positions immediately adjacent to instances of these genes. Applying RSAM-finder to this data with simpler queries that take into account only enrichment in environmental coloring does not yield this result, nor does the application of RSAM-finder to this data with any part of Q_3 on its own.

The identified gene set for this result spans a wide range of Firmicutes, including both pathogenic (e.g. staphylococcus) and non-pathogenic species (e.g. various gut microbes from the Clostridiales order). Homology between BlaR receptor proteins and the extra-cellular domain of Class D Beta-lactamases was previously observed [21, 8], mainly in *gram-negative bacteria* (with focus on clinical samples).

Thus, RSAM-finder identifies a putative Beta lactamase system in *gram positive bacteria*, consisting of a *COG2602-BlaR* Beta lactamase-receptor protein and its BlaI family repressor, *predicted to confer adaptation to animal and human host environment.* Further comparative sequence-level analysis [37] may reveal the affinity of this Beta lactamase system to specific Beta lactam drugs.

References

1 L E Alksne and B A Rasmussen. Expression of the AsbA1, OXA-12, and AsbM1 beta-lactamases in Aeromonas jandaei AER 14 is coordinated by a two-component regulon. *Journal of bacteriology*, 179(6):2006–2013, 1997.

2 Timothy L Bailey, Nadya Williams, Chris Misleh, and Wilfred W Li. MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic acids research*, 34(suppl_2):W369–W373, 2006.

3 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):i283–i291, 2012.

4 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.

5 E Bapteste, M A O'Malley, R G Beiko, M Ereshefsky, J P Gogarten, L Franklin-Hall, F-J Lapointe, J Dupré, T Dagan, Y Boucher, et al. Prokaryotic evolution and the tree of life are two different things. *Biology direct*, 4(1):34, 2009.

6 V Berry, F Chevenet, J-P Doyon, and E Jousselin. A geography-aware reconciliation method to investigate diversification patterns in host/parasite interactions. *Molecular ecology resources*, 18(5):1173–1184, 2018.

7 Baundauna Bose, Jennifer M Auchtung, Catherine A Lee, and Alan D Grossman. A conserved anti-repressor controls horizontal gene transfer by proteolysis. *Molecular microbiology*, 70(3):570–582, 2008.

8 C Brandt, S D Braun, C Stein, P Slickers, R Ehricht, M W Pletz, and O Makarewicz. In silico serine β-lactamases analysis reveals a huge potential resistome in environmental and pathogenic species. *Scientific reports*, 7:43232, 2017.

9 K Bush. Past and present perspectives on β-lactamases. *Antimicrobial agents and chemotherapy*, 62(10):e01076–18, 2018.

10 MA Charleston. Jungles: a new solution to the host/parasite phylogeny reconciliation problem. *Mathematical biosciences*, 149(2):191–223, 1998.

11 R Kumar Chaudhary, G Singh, R Naraian, and S Ram. Structural and functional in-silicoanalysis of toxin-antitoxin proteins in persister cells of pseudomonas aeruginosa. *Plant Archives*, 18(2):1643–1651, 2018.

12 L A David and E J Alm. Rapid evolutionary innovation during an Archaean genetic expansion. *Nature*, 469(7328):93, 2011.

13 B Donati, C Baudet, B Sinaimeri, P Crescenzi, and MF Sagot. EUCALYPT: efficient tree reconciliation enumerator. *Algorithms for Molecular Biology*, 10(1):3, 2015.

14 J-P Doyon, C Chauve, and S Hamel. Space of gene/species trees reconciliations and parsimonious models. *Journal of Computational Biology*, 16(10):1399–1418, 2009.

15 J-P Doyon, S Hamel, and C Chauve. An efficient method for exploring the space of gene tree/species tree reconciliations in a probabilistic framework. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(1):26–39, 2011.

16 B A Evans and S GB Amyes. OXA β-lactamases. *Clinical microbiology reviews*, 27(2):241–263, 2014.

17 L Huang and D Chiang. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics, 2005.

18 J Huerta-Cepas, F Serra, and P Bork. ETE 3: reconstruction, analysis, and visualization of phylogenomic data. *Molecular biology and evolution*, 33(6):1635–1638, 2016.

19 R Libeskind-Hadas and M A Charleston. On the computational complexity of the reticulate cophylogeny reconstruction problem. *Journal of Computational Biology*, 16(1):105–117, 2009.

20 A Marchler-Bauer and S H Bryant. CD-Search: protein domain annotations on the fly. *Nucleic acids research*, 32(suppl_2):W327–W331, 2004.

21 O Massidda, M P Montanari, M Mingoia, and P E Varaldo. Borderline methicillin-susceptible Staphylococcus aureus strains have more in common than reduced susceptibility to penicillinase-resistant penicillins. *Antimicrobial agents and chemotherapy*, 40(12):2769–2774, 1996.

22 D Merkle, M Middendorf, and N Wieseke. A parameter-adaptive dynamic programming approach for inferring cophylogenies. *BMC bioinformatics*, 11(1):S60, 2010.

23 S Mukherjee, D Stamatis, J Bertsch, G Ovchinnikova, O Verezemska, M Isbandi, A D Thomas, R Ali, K Sharma, N C Kyrpides, et al. Genomes OnLine Database (GOLD) v. 6: data updates and feature enhancements. *Nucleic acids research*, page gkw992, 2016.

24 R Patro and C Kingsford. Predicting protein interactions via parsimonious network history inference. *Bioinformatics*, 29(13):i237–i246, 2013.

25 L Poirel, A Carrër, J D Pitout, and P Nordmann. Integron mobilization unit as a source of mobility of antibiotic resistance genes. *Antimicrobial agents and chemotherapy*, 53(6):2492–2498, 2009.

26 A-A Popescu, K T Huber, and E Paradis. ape 3.0: New tools for distance-based phylogenetics and evolutionary analysis in R. *Bioinformatics*, 28(11):1536–1537, 2012.

27 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.

28 F Sievers and D G Higgins. Clustal Omega for making accurate alignments of many protein sequences. *Protein Science*, 27(1):135–145, 2018.

29 P JM Stapleton, M Murphy, N McCallion, M Brennan, R Cunney, and R J Drew. Outbreaks of extended spectrum beta-lactamase-producing Enterobacteriaceae in neonatal intensive care units: a systematic review. *Archives of Disease in Childhood-Fetal and Neonatal Edition*, 101(1):72–78, 2016.

30 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):i409–i415, 2012.

31 J Sukumaran and M T Holder. DendroPy: a Python library for phylogenetic computing. *Bioinformatics*, 26(12):1569–1571, 2010.

32 D Szklarczyk, J H Morris, H Cook, M Kuhn, S Wyder, M Simonovic, A Santos, N T Doncheva, A Roth, P Bork, et al. The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic acids research*, page gkw937, 2016.

33 R L Tatusov, M Y Galperin, D A Natale, and E V Koonin. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic acids research*, 28(1):33–36, 2000.

34 TH To, E Jacox, V Ranwez, and C Scornavacca. A fast method for calculating reliable event supports in tree reconciliations via Pareto optimality. *BMC bioinformatics*, 16(1):384, 2015.

35 A Tofigh. *Using trees to capture reticulate evolution: lateral gene transfers and cancer progression*. PhD thesis, KTH, 2009.

36 A Tofigh, M Hallett, and J Lagergren. Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(2):517–535, 2011.

37 M Toth, N T Antunes, N K Stewart, H Frase, M Bhattacharya, C A Smith, and S B Vakulenko. Class D β-lactamases do exist in Gram-positive bacteria. *Nature chemical biology*, 12(1):9, 2016.

38 L Van Melderen and M S De Bast. Bacterial toxin–antitoxin systems: more than selfish entities? *PLoS genetics*, 5(3):e1000437, 2009.