

Conflict Resolution Algorithms for Deep Coalescence Phylogenetic Networks

Marcin Wawerka ✉

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland

Dawid Dąbkowski ✉

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland

Natalia Rutecka ✉

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland

Agnieszka Mykowiecka ✉ 

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland

Paweł Górecki ✉ 

Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Poland

Abstract

We address the problem of inferring an optimal tree displayed by a network, given a gene tree G and a tree-child network N , under the deep coalescence cost. We propose an $O(|G||N|)$ -time dynamic programming algorithm (DP) to compute a lower bound of the optimal displayed tree cost, where $|G|$ and $|N|$ are the sizes of G and N , respectively. This algorithm has the ability to state whether the cost is exact or is a lower bound. In addition, our algorithm provides a set of reticulation edges that correspond to the obtained cost. If the cost is exact, the set induces an optimal displayed tree that yields the cost. If the cost is a lower bound, the set contains pairs of conflicting edges, that is, edges sharing a reticulation node. Next, we show a conflict resolution algorithm that requires $2^{r+1} - 1$ invocations of DP in the worst case, where r is a number of reticulations. We propose a similar $O(2^k |G||N|)$ -time algorithm for level- k networks and a branch and bound solution to compute lower and upper bounds of optimal costs. We also show how our algorithms can be extended to a broader class of phylogenetic networks. Despite their exponential complexity in the worst case, our solutions perform significantly well on empirical and simulated datasets, thanks to the strategy of resolving internal dissimilarities between gene trees and networks. In particular, experiments on simulated data indicate that the runtime of our solution is $\Theta(2^{0.543k} |G||N|)$ on average. Therefore, our solution is an efficient alternative to enumeration strategies commonly proposed in the literature and enables analyses of complex networks with dozens of reticulations.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Applied computing → Computational genomics

Keywords and phrases Phylogenetic Network, Gene Tree, Species Tree, Deep Coalescence, Reticulation, Optimal Displayed Tree

Digital Object Identifier 10.4230/LIPIcs.WABI.2021.17

Funding The support was provided by National Science Centre grant #2019/33/B/ST6/00737.

1 Introduction

Evolutionary networks are mathematical models of evolutionary processes with reticulate events such as hybridization, recombination, or horizontal gene transfer [1, 24]. Hybridization is a common phenomenon in plants and is often used in agriculture to create new breeds [18]. Recombination and reassortment are two shuffling processes in which variants of genetic material are created from pairs of highly similar DNA sequences. For example, many viruses have segmented genomes, including influenza viruses and rotaviruses [32], while horizontal



© Marcin Wawerka, Dawid Dąbkowski, Natalia Rutecka, Agnieszka Mykowiecka, and Paweł Górecki; licensed under Creative Commons License CC-BY 4.0

21st International Workshop on Algorithms in Bioinformatics (WABI 2021).

Editors: Alessandra Carbone and Mohammed El-Kebir; Article No. 17; pp. 17:1–17:21

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

gene transfer is common in bacteria [3]. In the last decades, mathematical and computational properties of phylogenetic networks have been intensively studied (see books [20, 24]). One of the most classic notions is a *tree displayed by a network*, obtained from a network by removing a set of *reticulation edges*. Displayed trees may represent an evolutionary history of a gene family if the incongruence is due to a reticulation event [24]. Alternative approaches include embedding a gene tree into a displayed tree [27, 31, 43] or using a parental species tree as a generalization of a displayed tree [25, 43, 49].

The pioneering work by Maddison [28] introduced the deep coalescence (DC) cost, which measures the *extra* gene lineages of a gene tree when embedded into a species tree. When a gene is embedded into its species tree, each edge of the species contains several mapped gene lineages. In the perfect situation, when both trees have the same topology, this number is one. Therefore, the cost is defined by the number of *extra* gene lineages required to embed a gene tree into a species tree. DC and general coalescent-based methods are popular in classical problems of computational biology, e.g. estimation of species trees [33, 41, 48], tree reconciliation [15, 10, 44, 48], or gene tree error correction [8].

There are two main general approaches to embed a gene tree into a network using the parsimony principle: (1) choosing the tree displayed by the network with the lowest cost, i.e. solving the *optimal displayed tree (ODT)* problem, in which a reticulation node can be reached only from one fixed parent, or (2) a *direct tree-network* embedding, without the above constraint. These approaches are present in relevant articles concerning inferences of networks under Robinson-Foulds (RF) embedding cost [31], the duplication-loss cost [43], the deep coalescence cost [27], including the general parsimony framework using the concept of parental species trees [25]. Alternative studies are based on minimizing deep coalescence criterion [47] or probabilistic models on coalescent histories [46]. Model-based approaches are usually computationally demanding since they often require enumeration of all possible coalescence histories [42, 46]. Finally, perhaps one of the most prominent applications of the above methods is the problem of network inference (e.g., [24, 25, 31, 43, 45] and tools e.g., [42]).

From the theoretical point of view, ODT under DC corresponds to NP-hard problems: (1) best switching (i.e., choosing the set of reticulation edges) for the duplication-loss model [43], and (2) the computation of RF-embedding cost [31]. In [43] the problem is solved in $O(|N| + p2^k|G|)$ time, where G is a gene tree and p is the number of biconnected components in a level- k network N . [31] proposed an $O(2^r|N|)$ -time optimized algorithm to compute RF-embedding cost. Another relevant contribution is from [27] with an $O(4^k|G||N|^2)$ -time tree vs. level- k network reconciliation algorithm under DC events. However, the latter cannot be directly compared to ours since we solve a different problem. In all of the above contributions, the complexity coming from 2^r (or 2^k) is reached due to exhaustive enumeration strategies. In this article, we show how to avoid such strategies by proposing an efficient in practice method to infer optimal displayed trees despite the theoretical intractability of ODT in general.

Our contribution: We address the problem of inference of an optimal tree displayed by a network (ODT), given a gene tree G and a network N under the deep coalescence cost (DC). We propose a novel approach in which we define scenarios for embedding G into N using sets of reticulation edges from N , with a property that the score of a scenario approximates the displayed tree cost. In particular, we prove that the score of a scenario is a lower bound of the cost of the optimal displayed tree. In a particular case, when a scenario induces a non-conflicting set of reticulation edges, we provide the correspondence between a score of

this scenario and a cost of a displayed tree. Next, we propose an $O(|G||N|)$ time dynamic programming (DP) algorithm to compute an optimal scenario. We show that an optimal scenario with no conflicts corresponds to a solution of ODT. Based on DP, we design a recursive algorithm to ODT by resolving conflicts in sets of reticulation edges. This algorithm has exponential time complexity $O(2^r|G||N|)$, where r is the number of reticulation nodes in N . We propose a similar $O(2^k|G||N|)$ -time algorithm for level- k networks. We also show how the algorithms can be extended to a broader class of phylogenetic networks. Finally, we show experimental studies on random, simulated, and empirical datasets. We show that our algorithm has significantly improved runtime on simulated datasets by reducing the exponent from r to $0.543r$ on average.

2 Definitions

A *network* on a set of species X is a directed acyclic graph $N = (V(N), E(N))$ with a single root such that: (1) its leaves, i.e., nodes of indegree 1 and outdegree 0, are labeled by the species from X , and (2) there is a path from the root to any other vertex. A network is *binary* if its leaves, root, and the remaining nodes have degrees 1, 2 and 3, respectively. A node is called a *reticulation* if it has indegree two and outdegree one, and a *tree node* if it has indegree at most one and outdegree two. A network is *semi-binary*, if additionally, it may contain *semi-binary nodes* of indegree at most one and outdegree one, which includes the root having exactly one child. We can *contract* a semi-binary node v of indegree one as follows: (1) remove v , (2) remove both edges incident with v , and (3) insert a new edge connecting the unique parent of v with the only child of v . Similarly, if v has indegree zero we remove v , and the child of v becomes a new root.

If $\langle v, w \rangle \in E(N)$, then v is a parent of w and w is a child of v , denoted $w.\text{parent} = v$ if w is a non-root tree node or a leaf. We write $v.\text{sibling} = w$ if $v \neq w$ have the same parent. We write $v \succeq w$ if there is a directed path from v to w , and $v \succ w$ if $v \succeq w$ and $v \neq w$. The set of all leaves in a network is denoted $L(N)$, by $R(N) \subset V(N)$ we denote the set of reticulation nodes in N , and by $E_R(N) \subset E(N)$ we denote the set of all reticulation edges in N , that is, edges $\langle v, r \rangle \in E(N)$ with $r \in R(N)$. We say that a reticulation edge e is a *sibling* of a reticulation edge e' if they share the same bottom reticulation node. By $\text{outdeg}_N(v)$ we denote the outdegree of v in N .

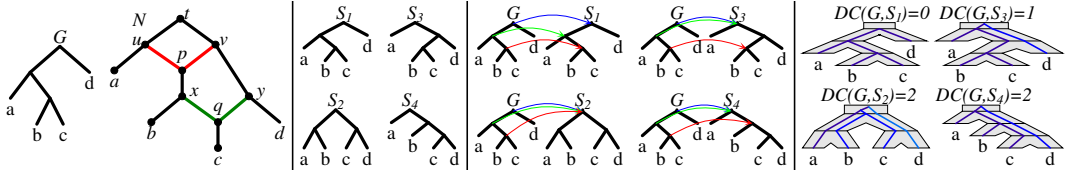
A *phylogenetic network* is a binary network on X in which the leaves are labeled one-to-one with the species from X ¹. A *species tree* is a phylogenetic network without reticulation nodes. Similarly, a *gene tree*, or in short *a tree*, is a binary network without reticulation nodes. Note that the leaf labeling in a gene tree does not have to be one-to-one.

From now on, all phylogenetic networks are *tree-child* [4, 5, 6, 35] (except for Section 5.4 and 6.3), i.e., each non-leaf node has a child that is either a tree node or a leaf.

2.1 Deep Coalescence Cost: embedding a tree into a (displayed) tree

Given a gene tree G and a species tree S on X , the *lca-mapping* $M: V(G) \rightarrow V(S)$ is defined as follows: (1) if g is a leaf labeled $x \in X$ then $M(g)$ is the unique leaf labeled x in S , and (2) if g has two children g' and g'' , then $M(g)$ is the lowest common ancestor of $M(g')$ and $M(g'')$ in S . Embedding G into S is performed by mapping each edge $\langle v, w \rangle \in E(G)$ to a

¹ Note that in a binary network two leaves can be labeled by the same label, which is not allowed in a phylogenetic network.



■ **Figure 1** From the left: a gene tree G and a phylogenetic network N with two reticulations, four trees displayed by N , lca-mappings between G and S_i 's, visualization of embeddings of G into S_i 's. Here, S_1 is the optimal tree displayed by N with the DC cost of 0.

path connecting $M(v)$ and $M(w)$ in S . We say that the gene edge *visits* edges from that path. Let $\|v, w\|$ denote the number of edges on the path connecting v and w . Then, the visited edges contribute to the deep coalescence cost, denoted $DC(G, S)$, as follows:

$$DC(G, S) = \sum_{(v,w) \in E(G)} (\|M(v), M(w)\| - 1). \quad (1)$$

A set $Y \subseteq E_R(N)$ is called *perfect* if, for each $r \in R(N)$, Y contains exactly one edge whose bottom node is r . Given a perfect Y , the graph denoted N/Y , obtained from N by removing all edges from $E_R(N) \setminus Y$ is a semi-binary tree, i.e., semi-binary network with no reticulations. Further, contracting all semi-binary nodes yields a species tree on X , denoted N_Y . We say that a species tree T is *displayed* by a phylogenetic network N , if $N_Y = T$ for some perfect Y . We also say that the perfect set Y is induced by a tree T displayed by N if $N_Y = T$.

We now define the *Optimal Displayed Tree* problem (ODT) in the parsimony framework:

► **Problem 1** (ODT). *Given a tree G and a phylogenetic network N . Find an optimal tree S^* displayed by N that minimizes $DC(G, S)$ in the set of all trees S displayed by N .*

The *cost* of an optimal displayed tree, we denote $DC(G, N)$. While the complexity of ODT remains unknown for the class of tree-child networks, we claim that the problem is NP-hard in a general class of networks. The proof is similar to the NP-hardness proof of the best switching problem from [43]. See also [31] for the related problem of RF-embedding. Fig. 1 depicts an example of DC costs.

2.2 Scenarios between gene trees and phylogenetic networks

In the previous Section, we showed how a gene tree is embedded into a species tree. Here, we propose to embed a gene tree into a phylogenetic network using a more general approach than embedding through a displayed tree. We start with the notion of *unfolded network* (see also [22, 23]), then we define *scenarios* between gene trees and unfolded networks.

For a phylogenetic network N on X with k reticulations, the *unfolded network* \hat{N} is the tree N_k obtained from N by a sequence of k unfolding operations defined on pairs (N_i, σ_i) , such that N_i is a semi-binary network on X and $\sigma_i: V(N_i) \rightarrow V(N)$ defines the origin of a node from N_i . Let (N_0, σ_0) be a pair such that $N_0 = N$ and $\sigma_0(v) = v$ for each $v \in V(N)$. Then, for a sequence of all reticulation nodes r_1, r_2, \dots, r_k from N in a reversed topological order, (N_i, σ_i) is obtained from (N_{i-1}, σ_{i-1}) by unfolding the reticulation r_i as follows:

- Let S_i be a copy of the subtree of N_{i-1} rooted at r_i .
- $V(N_i) := V(N_{i-1}) \cup V(S_i)$ and $E(N_i) := (E(N_{i-1}) \setminus \{\langle p, r_i \rangle\}) \cup E(S_i) \cup \{\langle p, r'_i \rangle\}$, where p is an arbitrary parent of r_i and r'_i is the root of S_i .
- $\sigma_i(v)$ is $\sigma_{i-1}(v)$ if $v \in V(N_{i-1})$; otherwise, it is $\sigma_{i-1}(t)$, if v is a copy of t from N_{i-1} .

Informally, for each reticulation node, we copy its subtree, detach the original subtree from one parent, and attach the copy to the same parent, without changing the labels. To avoid using k directly, we set σ to be σ_k .² Fig. 2 depicts an unfolded network.

► **Lemma 2** (Correctness of unfolding). *The unfolded network \hat{N} of N is a semi-binary tree.*

Let a *leaf-root* path be a directed path connecting a leaf with the root in a network.

► **Theorem 3** (Unfolding Soundness). *There is a one-to-one correspondence between leaf-root paths in N and leaf-root paths in \hat{N} .*

It follows from Thm. 3 that N and \hat{N} have the same structure of leaf-root paths. A *scenario* for G and N is a function $\xi: L(G) \rightarrow L(\hat{N})$ that preserves the leaf labeling: for every $g \in L(G)$, the labels of g and $\xi(g)$ are equal. A scenario ξ can be extended to the lca-mapping $M_\xi: V(G) \rightarrow V(\hat{N})$ such that for $g \in V(G)$, $M_\xi(g)$ is the lowest node v in \hat{N} such that $\xi(g') \preceq v$, for each leaf $g' \preceq g$. Note that $M_\xi(g)$ is either a leaf or a tree node with two children.

2.3 Deep Coalescence Score of Scenarios

Having the lca-mapping determined by a scenario, we are ready to define the deep coalescence *score*, denoted \tilde{DC} , to approximate deep coalescence events induced by scenarios in phylogenetic networks. Our goal is to deduce properties allowing us to approximate the DC cost to solve ODT. In particular, our approach differs from the approaches from [25, 27, 43], e.g., in the way how a cost of a path is defined, although the general concept of mapping a gene tree into a network is analogous.

For a scenario ξ for G and N , we say that $\langle v, w \rangle \in E(G)$ *visits* $\langle a, b \rangle \in E(\hat{N})$ if $M_\xi(v) \succeq a \succ b \succeq M_\xi(w)$. Then, $\langle a, b \rangle$ has exactly one of the following *types*.

- Type I: $M_\xi(v) = a$, i.e., it is the first edge.
- Type II: $M_\xi(v) \succ a$, $\text{outdeg}_{\hat{N}}(a) = 2$ and $\sigma(b.\text{sibling}) \notin R(N)$;
- Type III: $M_\xi(v) \succ a$, $\text{outdeg}_{\hat{N}}(a) = 2$ and $\sigma(b.\text{sibling}) \in R(N)$; we say that ξ *bypasses* the reticulation edge $\sigma(\langle a, b.\text{sibling} \rangle)$
- Type IV: $\text{outdeg}_{\hat{N}}(a) = 1$ (only if $\sigma(a) \in R(N)$).

In the above definition, type (I) is only for the first (i.e., the closest to the root) edge visited by a given edge from G , while for the remaining visited edges from \hat{N} an edge has: Type (II) if the sibling of its bottom node is a tree node, Type (III) if the sibling of its bottom node is a reticulation, and Type (IV) if the top node of the edge is a reticulation.

By $\kappa_\xi(v, w)$ we denote the set of all edges of Type I or II visited by $\langle v, w \rangle$. Then, the *deep coalescence score* for G , N and a scenario ξ is

$$\tilde{DC}(G, N, \xi) = \sum_{\langle v, w \rangle \in E(G)} (|\kappa_\xi(v, w)| - 1). \quad (2)$$

Examples of scenarios and \tilde{DC} scores are depicted in Fig. 2. Finally, we can define *Optimal Scenario Inference* problem, DC-MinRec.

► **Problem 4** (DC-MinRec). *Given a gene tree G and a phylogenetic network N . Find an optimal scenario ξ^* that minimizes $\tilde{DC}(G, N, \xi)$ in the set of all scenarios ξ for G and N .*

² We also use σ with edges, e.g., $\sigma(\langle v, w \rangle)$ denotes $\langle \sigma(v), \sigma(w) \rangle$.

In Section 4, we propose a dynamic programming algorithm that solves DC-MinRec in $O(|G||N|)$ time. Note that the complexity depends on the size of N (not on the potentially exponential size of \hat{N}).

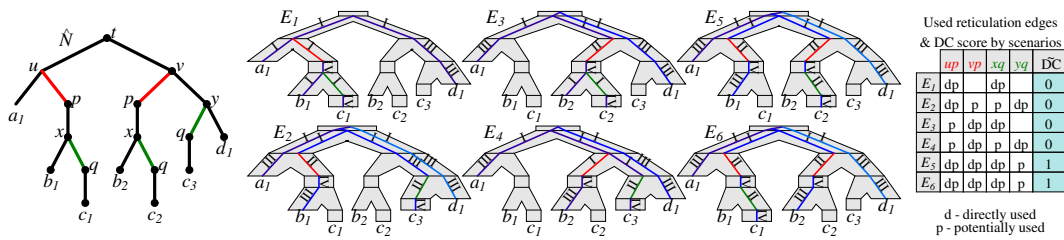
In a trivial case, the solution to DC-MinRec is induced by the classical DC cost.

► **Lemma 5.** *If N is a phylogenetic network with no reticulation node then there is only one scenario ξ for G and N . Moreover, $\tilde{\text{DC}}(G, N, \xi) = \text{DC}(G, N)$.*

3 Scores of scenarios vs. costs of displayed trees

In this Section, we present several theoretical results connecting our scoring functions. Note that the notion of a *cost* will be used only with the DC cost defined in (1) for trees and for phylogenetic networks in Problem 1, while for scenarios, we will use the notion of a ($\tilde{\text{DC}}$) *score*. To establish the correspondence, we first show that each perfect set Y determines a scenario. Recall that N_Y is obtained from N/Y by contracting semi-binary nodes. Let \hat{N}_Y be the graph obtained from \hat{N} by removing all edges e such that $\sigma(e) \in E_R(N) \setminus Y$ and all subtrees whose root is the bottom node of e .

► **Lemma 6.** *\hat{N}_Y and N/Y are isomorphic, and the isomorphism is established by $\sigma|_{V(\hat{N}_Y)}$.*



■ **Figure 2** *Left:* The unfolded network \hat{N} of N from Fig. 1 is shown with σ values attached to nodes, where for the leaves, the index is inserted to distinguish leaves with the same labels/mappings. *Middle:* 6 scenarios for $G = ((a, (b, c)), d)$ shown as embeddings of G to \hat{N} . Numbers I-IV denote the type of a visited edge. Only E_1 is regular, while $E_1 - E_4$ are optimal. *Right:* DC score and types of used reticulation edges for each scenario ($\Upsilon_{[1]}$).

For a perfect set Y , we define a scenario ξ_Y such that for each gene leaf g labeled x , $\xi_Y(g)$ is the only leaf in $L(\hat{N}_Y) \subseteq L(\hat{N})$ labeled x . Correctness follows from Lemma 6. For example, in Fig. 2, if $Y = \{\langle u, p \rangle, \langle x, q \rangle\}$, then Y is perfect and $N_Y = S_1$ from Fig. 1. Moreover, for $G = ((a, (b, c)), d)$, ξ_Y maps a to a_1 , b to b_1 , c to c_1 and d to d_1 as depicted in E_1 .

We say that $e \in E_R(N)$ is *directly used* by scenario ξ if there is a visited edge e' of Type I or II such that $\sigma(e') = e$. Similarly, we say that reticulation edge e is *potentially used* by ξ if the sibling edge of e is bypassed by ξ . By $\Upsilon_\xi \subseteq E_R(N)$ we denote the set of reticulation edges used directly or potentially by ξ (see Fig. 2).

We say that $Y \subseteq E_R(N)$ has a *conflict* if Y contains two sibling edges. We say that ξ is *regular* if Υ_ξ has no conflict. For instance, Υ_{E_4} for E_4 from Fig. 2 has two possible conflicts in N . Observe that Υ_ξ may not be perfect in general, even if ξ is regular. For instance, if $G = (c, d)$ and ξ maps c to c_3 in the network from Fig. 2, then $\Upsilon_\xi = \{\langle y, q \rangle\}$.

Now, we can state the crucial proposition that establishes a correspondence between regular scenarios and embedding to trees displayed by a network.

► **Proposition 7 (Scenario-Displayed Tree Correspondence).** *A scenario ξ is regular, if and only if for every perfect set Y such that $\Upsilon_\xi \subseteq Y$, $\tilde{\text{DC}}(G, N, \xi) = \text{DC}(G, N_Y)$.*

In the following proposition, we show that the cost of a tree displayed by a network using a perfect set is bounded from below by the cost of its corresponding scenario.

► **Proposition 8.** *If Y is perfect, then $\text{DC}(G, N_Y) \geq \tilde{\text{DC}}(G, N, \xi_Y)$.*

Finally, we show that the equality between the score and the cost holds only if the induced scenario is regular.

► **Proposition 9.** *If Y is perfect, then $\text{DC}(G, N_Y) = \tilde{\text{DC}}(G, N, \xi_Y)$ if and only if ξ_Y is regular.*

The next theorem states that the cost of an optimal tree displayed by a network is bounded from below by the score of an optimal scenario.

► **Theorem 10 (Lower Bound Property).** *If S^* is an optimal tree displayed by N , and ξ^* is an optimal scenario of N then $\text{DC}(G, S^*) \geq \tilde{\text{DC}}(G, N, \xi^*)$.*

In our example from Fig. 1 and Fig. 2, the cost of S_1 and the score of E_1 are equal. However, in general, a regular scenario may not exist. For instance, if $G = (a, d)$, there is only one scenario ξ for N from Fig. 2, where a and d are mapped to a_1 and d_1 , respectively. Then, ξ is not regular, and $0 = \tilde{\text{DC}}(G, N, \xi) < \text{DC}(G, N) = 1$ (for S_1 or S_2).

Finally, we present a crucial theoretical property used to solve ODT using solutions to instances of DC-MinRec.

► **Theorem 11 (Regularity).** *Let d be the score of an optimal scenario of N . A tree S displayed by N with $\text{DC}(G, S) = d$ exists, if and only if there is an optimal regular scenario of N .*

4 Dynamic Programming (DP) Algorithm to solve DC-MinRec

Dynamic programming algorithms are commonly used in tree reconciliation, including models based on directed acyclic graphs (DAGs) [10, 16, 27, 37, 43], where a gene tree is mapped to a tree or a DAG through the lca-mapping or general mapping based on concepts close to our scenarios. Such approaches often lead to polynomial time solutions with square time complexity in the best case. Here, we present a dynamic programming solution to Problem 4 by providing formulas to compute the score of an optimal scenario.

Additional notation: By v' and v'' , we denote the children of a tree node v , and by r' the child of a reticulation node r . For simplicity, instead of $\sigma(M_\xi(g))$ for a gene tree node g , we write ξ_g (i.e., $\xi_{[\cdot]}$ is a mapping from G to N).

By $G|g$, we denote the subtree of G rooted at g . The main component of dynamic programming is δ such that for $g \in V(G)$ and $s \in V(N)$, $\delta(g, s)$ is the minimum score for $G|g$ in the set of all scenarios ξ between $G|g$ and \hat{N} such that $\xi_g = s$. For simplicity, we ignore -1 from the $\tilde{\text{DC}}$ formula in the partial costs in δ as this yields a constant term dependent on the size of G . Let $\tau(s)$ be equal 0 if s is a reticulation, and 1 otherwise. Then, we have the following dynamic programming formula that solves DC-MinRec:

$$\delta(g, s) = \begin{cases} \delta^f(g', s) + \delta^f(g'', s) & g \text{ and } s \text{ are tree nodes,} & (3) \\ \delta(g', s) + \delta(g'', s) & g \text{ is a tree node and } s \text{ is a leaf,} & (4) \\ 0 & g \text{ and } s \text{ are leaves labeled by the same species,} & (5) \\ +\infty & \text{otherwise,} & (6) \end{cases}$$

$$\delta^f(g, s) = \min(\delta(g, s), \tau(s') + \delta^\dagger(g, s'), \tau(s'') + \delta^\dagger(g, s'')), \quad (7)$$

$$\delta^\uparrow(g, s) = \begin{cases} \min(\delta(g, s), \tau(s')\tau(s'') + \min(\delta^\uparrow(g, s'), \delta^\uparrow(g, s''))) & s \text{ is a tree node,} \\ 1 + \delta^\uparrow(g, s') & s \in R(N), \\ \delta(g, s) & s \text{ is a leaf.} \end{cases} \quad (8)$$

$$1 + \delta^\uparrow(g, s') \quad s \in R(N), \quad (9)$$

$$\delta(g, s) \quad s \text{ is a leaf.} \quad (10)$$

In the next Lemma, we express properties satisfied by the above formulas.

► **Lemma 12.** *Let $g \in V(G)$, $s \in V(N)$ and all scenarios below are for G and N .*

D1 $\delta(g, s)$ is equal to minimum number of Type I/II edges visited by edges from $E(G|g)$ among scenarios ξ satisfying $\xi_g = s$.

D2 If c is a child of g and t is not a reticulation. Then, $\delta^\uparrow(c, t)$ is equal to minimum number of Type I/II edges visited by edges from $E(G|c)$ plus the number of edges $e' = \langle a, b \rangle$ of Type II visited by $\{c, g\}$ with $t \succeq \sigma(a)$ among scenarios ξ such that $\xi_g = s \succ t \succeq \xi_c$.

D3 If c is a child of g and s is a tree node. Then, $\delta^f(c, s)$ is equal to minimum number of Type I/II edges visited by edges from $E(G|c) \cup \{c, g\}$ among scenarios ξ satisfying $\xi_g = s$.

The optimal score is given by the following theorem, whose proof follows immediately from the definitions of δ , DC and Lemma 12.

► **Theorem 13.** *Given a gene tree G and a phylogenetic network N . The score of an optimal scenario ξ^* is $\text{DC}(G, N, \xi^*) = -|E(G)| + \min_{s \in V(N)} \delta(G.\text{root}, s)$.*

To infer an optimal scenario, we apply standard backtracking based on values of δ array. Since there are three arrays, each of size $|G||N|$ and every cell of an array can be computed in $O(1)$ time, DP has $O(|G||N|)$ time and space complexity. Note that in implementation δ^f can be embedded into δ computation. Thus, the space may be reduced to two arrays.

4.1 Inferring used reticulations edges from DP

An optimal scenario can be inferred from DP formulas using standard backtracking. However, this scenario may not be perfect. To further utilize the results of DP, we infer the set of used reticulation edges. For two nodes v and w , let $\rho(v, w) = \{\langle v, w \rangle\}$ denote the one-element set with $\langle v, w \rangle$ if the edge is a reticulation edge in N , and $\rho(v, w) = \emptyset$ otherwise. Similarly, by $\bar{\rho}(v, w)$ we denote the one-element set with the sibling edge of $e = \langle v, w \rangle$ if e is a reticulation edge in N , and $\bar{\rho}(v, w) = \emptyset$, otherwise. Then, DP components δ , δ^f and δ^\uparrow are associated with reticulation edge usage rules u , u^f , and u^\uparrow , resp., as follows:

$$u(g, s) = \begin{cases} u^f(g', s) \cup u^f(g'', s) & \text{in (3),} \\ \emptyset & \text{in (4)-(6),} \end{cases}$$

$$u^f(g, s) = \begin{cases} u(g, s) & \text{if } \delta^f(g, s) = \delta(g, s) \text{ in (7),} \\ u^\uparrow(g, c) \cup \rho(s, c) & \text{if } \delta^f(g, s) = \tau(c) + \delta^\uparrow(g, c) \text{ for some } c \in \{s', s''\} \text{ in (7),} \end{cases}$$

$$u^\uparrow(g, s) = \begin{cases} u(g, s) & \text{if } \delta^\uparrow(g, s) = \delta(g, s) \text{ in (8) or (10),} \\ u^\uparrow(g, c) & \text{in (9),} \\ u^\uparrow(g, c) \cup \rho(s, c) \cup \bar{\rho}(s, c.\text{sibling}) & \text{if } \delta^\uparrow(g, s) = \tau(s')\tau(s'') + \delta^\uparrow(g, c) \\ & \text{for some } c \in \{s', s''\} \text{ in (8).} \end{cases}$$

The correctness of above formulas follows from the next lemma.

► **Lemma 14.** *If the backtracking of DP results in a scenario ξ , then $\Upsilon_\xi = u(G.\text{root}, \xi_{G.\text{root}})$.*

5 Inferring Optimal Displayed Trees

In this Section, we propose algorithms to solve ODT and its variants. We also answer whether the problem can be analogously solved for level- k networks and for broader classes of phylogenetic networks.

5.1 Solution to ODT

Thm. 11 motivates the following general branching algorithm to solve ODT. Suppose DP returns a solution with a conflict. Then, such a conflict can be resolved by branching and solving two sub-instances of the problem with phylogenetic networks induced from the input phylogenetic network by removing exactly one edge from the conflict. Let N_e be the phylogenetic network obtained from $N/\{e\}$ by contracting all semi-binary nodes³. Alg. 1 details the procedure to infer an optimal tree displayed by a given network. Here, branching occurs when there is a conflict in the set of used reticulation edges. Thus, if the number of conflicts is low, e.g., when G and N are similar, we expect a small number of DP invocations.

■ **Algorithm 1** Optimal displayed tree inference: a solution to ODT.

-
- 1: **Input:** a gene tree G and a phylogenetic network N .
 - 2: **Output:** an optimal tree displayed by a network N
 - 3: Compute Υ_{ξ^*} (see Thm. 14) for some optimal scenario ξ^* for G and N inferred by DP.
 - 4: Optimization: **Return** the empty tree with $+\infty$ score if the score of ξ^* is larger or equal to the currently known best score of a regular scenario (if already computed in a recursive call).
 - 5: **If** Υ_{ξ^*} has no conflict **Then Return** N_Y for some perfect set $Y \supseteq \Upsilon_{\xi^*}$.
 - 6: **Otherwise** Υ_{ξ^*} has a conflict, i.e., there are two sibling edges, say e and e' in Υ_{ξ^*} .
 - 7: Recursively compute two optimal trees displayed by networks N_e and $N_{e'}$, resp.
 - 8: **Return** the tree with the lower cost.
-

Correctness of Algorithm 1 follows from Thm. 11 and the following theorem.

► **Theorem 15.** *If $e, e' \in E_R(N)$ are sibling edges then $\text{DC}(G, N) = \min\{\text{DC}(G, N_e), \text{DC}(G, N_{e'})\}$. Moreover, T is an optimal tree displayed by N if and only if T is an optimal tree displayed by a network N_e or $N_{e'}$ with minimum cost.*

In the worst case, we need to branch for every reticulation twice, which gives $2^{r+1} - 1$ invocations of DP. Thus, Alg. 1 has time complexity $O(2^r |G| |N|)$ in the worst case. However, as mentioned previously, we expect Alg. 1 to behave better than worst complexity in practice. See also our experimental evaluation in Section 6.

5.2 Lower and upper bounds of the optimal cost of a displayed tree

In applications where only the optimal cost is needed, for instance, in problems of network inference, we can use the Lower Bound Theorem 10. As the cost of an optimal displayed tree is bounded below by the score from DP, we can also compute the upper bound using regular scenarios returned from multiple invocations of DP. See details in Alg. 2.

► **Lemma 16.** *For G and N , Alg. 2 returns l and u such that $l \leq \text{DC}(G, N) \leq u$.*

³ Recall that N/X is the network obtained from N by removing all edges from X .

■ **Algorithm 2** Branch and bound algorithm to approximate $\text{DC}(G, N)$.

-
- 1: **Input:** a gene tree G , a phylogenetic network N , the maximal depth of recursion $d \geq 0$.
 - 2: **Output:** a pair: lower and upper bounds of the cost of an optimal displayed tree for G and N .
 - 3: Compute Υ_{ξ^*} and the optimal score c^* for some optimal scenario ξ^* for G and N .
 - 4: **If** Υ_{ξ^*} has no conflict **Then Return** $\langle c^*, c^* \rangle$ // *Regular scenario; report the exact cost*
 - 5: **If** the depth d of recursive invocations is reached **Then Return** $\langle c^*, +\infty \rangle$ // *Lower bound only*
-
- 6: Recursively compute bounds $\langle l, u \rangle$ and $\langle l', u' \rangle$ for G and N_e and $N_{e'}$, respectively, where e and e' are two conflicted sibling edges from Υ_{ξ^*} .
 - 7: **Return** $\langle \min(l, l'), \min(u, u') \rangle$ // *Combine results from two invocations*
-

5.3 Inferring optimal trees displayed by level-k networks

Our results can also be extended to level- k networks. The definition and properties are adopted from [9, 12, 43]. A level- k network is a phylogenetic network in which every biconnected component has at most k reticulation nodes [9]. If B is a biconnected component of N , then by $B.\text{root}$ we denote the unique node in B with no ancestors in B . Using the notation from [43], by $bc(N)$ we denote the tree obtained from N by contracting all its biconnected components. Let $\text{Lab}(N)$ denote the set of species present in N as leaf labels.

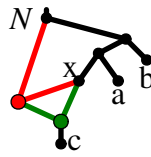
■ **Algorithm 3** Optimal displayed tree for level- k network.

-
- 1: **Input:** a gene tree G , a phylogenetic network N .
 - 2: **Output:** the cost of optimal tree displayed by N .
 - 3: $c = 0$ // *The cost variable*
 - 4: **For** each biconnected component B from $bc(N)$ in postfix order:
 - 5: **If** $\text{Lab}(G) \subseteq \text{Lab}(B)$ **Then Return** $c + \text{DC}(G, B)$ // *The case of root and the top component*
 - 6: Let F be a collection of all maximal subtrees of G rooted at g such that $\text{Lab}(G|g) \subseteq \text{Lab}(B)$.
 - 7: $c := c + \sum_{G' \in F} \text{DC}^\dagger(G', B)$ // DC^\dagger is a “ δ^\dagger ” variant of DC ; see def. in Section 5.3
 - 8: Let s_B be a new species label representing the component B .
 - 9: Replace B in N , and every $G' \in F$, by a leaf labeled s_B . // *Contract B and subtrees of G*
 - 10: **Return** c
-

In Alg. 3, edges visited by subtrees of G have to be connected in the embedding. Therefore, for each non-root component B in $b(N)$, we minimize the score using the additional costs of a path to the root of B . Formally, $\text{DC}^\dagger(G, N)$ is the minimum value of $\text{DC}(G, S) + \|\text{M}(G.\text{root}), S.\text{root}\|$ in the set of all displayed trees S of N . Computing the value (almost) does not require modification of our algorithms. Here, instead of the formula from Thm. 13, we compute $\text{DC}^\dagger(G, N)$ using $-|E(G)| + \delta^\dagger(G.\text{root}, S.\text{root})$. The correctness follows from Lemma 12 case D2. The formula can be easily embedded into Alg. 1. The time complexity of Alg. 3 is $O(2^k |G||N|)$.

5.4 Beyond tree-child networks

DP can be extended to analyse a broader class of networks, which is more beneficial from a practical point of view. Assume that instead of a tree-child network condition, our class of networks satisfies a relaxed condition: *a node has at most one reticulation child*. This assumption admits the child of reticulation to be a reticulation, which is not allowed in tree-child networks. Then, in DP, we have the following modification in (9): $\tau(s') + \delta^\dagger(g, s')$, and in usage rules in the 2nd case of u^\dagger referring to (9): $u^\dagger(g, s) \cup \rho(s, c)$, which is needed when the child is also a reticulation. Under this modification, Alg. 1 returns a correct optimal displayed tree. We omit details for brevity.



■ **Figure 3** A non tree-child network.

We also analysed a general class of binary networks, i.e., in which a tree node may have two reticulation children. However, DP cannot correctly analyse such networks. When embedding a gene tree (a, b) into the network N from Fig. 3, we see that the optimal displayed tree is $S = ((a, b), c)$ with the cost 0. Here, S is constructed by removing a node x and all three incident edges, and a tree node x .parent with two children is also contracted. In the current DP, when a gene edge $\langle a$.parent, $a \rangle$ from G visits x .parent DP will increase the cost. Therefore, the lower bound property is not satisfied in this case unless a solution in which such removed tree nodes are detected is implemented. It remains open whether it can be done in polynomial time without checking all variants of displayed trees.

6 Experimental evaluation

In this Section, we present the experimental evaluation using our prototype implementation of DP, Alg. 1 and Alg. 2 called **EmbRetNet** written in Python 3. The algorithms were extended to analyse the class of networks described in Section 5.4. The software package is available from <https://bitbucket.org/pgor17/embretnet>.

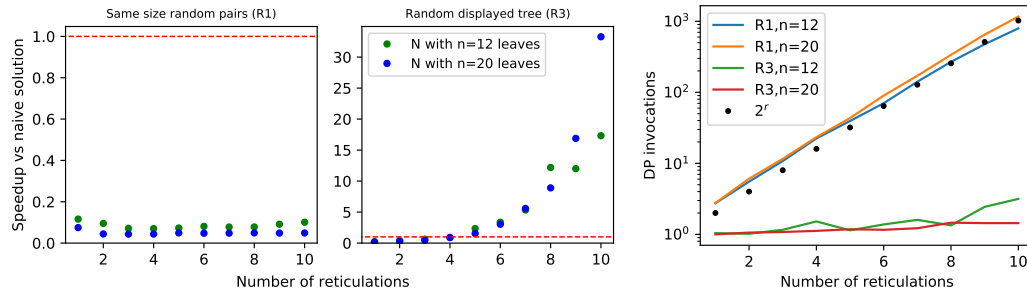
Performance of inferring optimal displayed trees

Our Alg. 1 is exponential in the worst case; however, we expect better performance when the gene tree is similar to the network, which is expected for empirical data. In this Section, we summarize the results of several experiments to compare the performance of our implementation of Alg. 1 to the naïve implementation in which all trees displayed by a given network are generated, and then, the costs are computed using a linear time solution from [48]. Note that both algorithms have exponential time complexity; however, the naïve algorithm always has the same number of steps, proportional to $2^r(|G| + |N|)$, where r is the number of reticulations in N . Experiments were conducted on a Ubuntu server with Intel(R) Xeon(R) CPU E5-2698 v4@2.20GHz (80 cores) and 500 GB of RAM.

6.1 Evaluation on random datasets

Data preparation: To generate random tree-child phylogenetic networks, we used an algorithm from [26] and its implementation in Python from GitHub with a slight modification to generate only binary networks. Random gene trees with one-to-one labeling of leaves were generated using the Yule-Harding model. Then, we generated datasets R1, R2, and R3, each consisting of $2 \cdot 10 \cdot 100$ pairs of random gene trees and networks. For each $n \in \{12, 20\}$ and $r \in \{1, 2, \dots, 10\}$, we generated 100 pairs $\langle G, N \rangle$, such that N is a network with n leaves and r reticulations and in dataset R1 $|L(G)| = n$, in R2 the number of leaves in G is sampled uniformly from the interval $[2, n]$, and in R3 G is a randomly chosen tree displayed by N .

Discussion: In the two left panels of Fig. 4, we summarize the experimental evaluation results of R1 and R3, where G and N were generated independently. Since these datasets represent extremes, in which G and N are highly different, our algorithm frequently infers



■ **Figure 4** *Left*: Performance of Alg. 1 vs naïve approach for random datasets R1 and R3. Each dot represents the average speedup computed from the runtimes of 100 pairs of gene trees and phylogenetic networks. *Right*: Average number of DP invocations necessary to calculate an answer for random datasets R1 and R3. Since the results of R2 are similar to R1, we moved them to Appendix A.1.

conflicted sets of reticulation edges by visiting almost every possible scenario, thus achieving nearly the pessimistic exponential complexity. In consequence, it is noticeably slower than the naïve one for R1. On the contrary, with data from R3 the algorithm rarely branches, and its average runtime matches the complexity of DP, i.e., $O(|G||N|)$. Even with a larger constant factor, we outperform the naïve algorithm for $r > 4$, achieving > 15 times speedup for $r = 10$.

6.2 Evaluation on simulated datasets

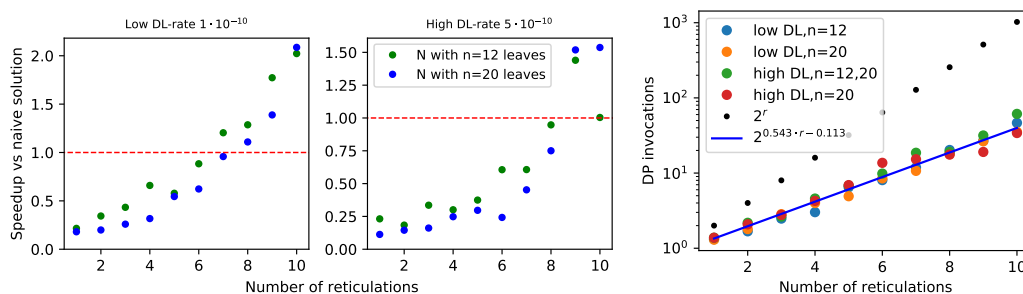
Our simulation procedure can be divided into three major phases: (i) simulating tree-child phylogenetic networks, (ii) simulating gene trees, and (iii) introducing errors to the gene trees. The selection of the parameters in all three phases is mainly based on the simulation study conducted by Molloy and Warnow [34], which uses parameters derived from a fungal dataset presented by Rasmussen and Kellis [36]. See Appendix A.2 for more details.

(i) Simulating tree-child phylogenetic networks. First, we simulated species trees using a general sampling approach implemented in R package TreeSim version 2.4 [21] with the parameters from [34]. The number of leaves was set to 12 or 20. Next, we inferred a network with r reticulations from each of the simulated species trees, where r was uniformly sampled from $[1, 10]$. We followed a popular study by Solis-Lemus and Ané [39]. To add r reticulations, we started by randomly choosing r pairs of edges in the species tree. We then subdivided both edges in a pair, making two new vertices. Finally, we added a reticulation edge between the vertices. This method generates networks belonging to the tree-based class [14]. Similarly to [31], the networks were constrained to be tree-child.

(ii) Simulating gene trees. For each phylogenetic network, we randomly chose one of its displayed trees, obtaining one of the possible trees, along which gene families evolve. We simulated one gene tree per displayed tree using SimPhy version 1.0.2 [30]. Similarly to [34], we used three rates of duplication/loss (DL) $\{10^{-10}, 2 \cdot 10^{-10}, 5 \cdot 10^{-10}\}$ and two values of the effective population size $\{10^7, 5 \cdot 10^7\}$, corresponding respectively to a low and a medium level of incomplete lineage sorting (ILS). Altogether, we used six sets of simulation parameters, which allowed us to obtain a diversified set of gene trees. Additionally, we set the minimum number of leaves to 3.

(iii) **Simulating sequences and estimating gene trees.** To introduce errors to the generated gene trees, we simulated sequences along with them and estimated gene trees from multiple sequence alignments using the maximum-likelihood method (MLE). DNA sequences were simulated by INDELible v1.03 [13] and SimPhy [30]. Again, we followed the parameters proposed in [34]. The alignment length was set to 1000 bp. To estimate gene trees, we used a true alignment returned by INDELible. Then, we inferred ML-trees by PhyML v.3.1 [19] using GTR+ Γ model. Finally, to obtain a rooted gene tree from an unrooted ML-tree, we conducted midpoint-plateau rooting implemented in URec [17] using the corresponding displayed tree inferred in step (ii) of our pipeline.

Finally, for each set of parameters of duplication-loss rates, population sizes, reticulation values, and leaf-set sizes we simulated 100 networks and 100 corresponding gene trees. The simulations were run in parallel on 10 cores and the total simulation time was under 8 hours. The algorithm took 2 hours to process all datasets, and it took, on average, 45 seconds to run 100 instances with 20 leaves and 10 reticulations for low ILS and low DL.



■ **Figure 5** *Left*: Performance of Alg. 1 vs. naïve approach for datasets with low ILS ($1 \cdot 10^7$). See Appendix A.3 for a complete set. *Right*: Average number of DP invocations necessary to calculate an answer for datasets with low ILS. The blue line is calculated using least squares linear regression, that best approximates results for all simulated datasets.

Discussion. In Fig. 5, we present diagrams showing the results of evaluations for two datasets. The results for the remaining datasets are similar (please refer to Appendix). The way we simulated data makes trees and networks more similar to each other. Thus, we can see significant improvements vs. random datasets. Regardless of parameter choices, we start to outperform the naïve solution for $r > 9$. For simulated data closest to reality (low ILS, low DL), we achieved better results for $r > 7$. The results suggest a hybrid approach in Alg. 1: enumerate all displayed trees to compute DC costs directly if the network has a low number of reticulations (e.g., $r < 9$).

To estimate the average runtime of our algorithm, we first calculated the depths d of the recursive calls as \log_2 of the number of DP invocations from each experiment. Then, we found that for all points $\langle r, d \rangle$ from our experiments, $d = 0.543r - 0.1135$ is the fitted least squares regression line having the standard error of .011 (see the rightmost diagram in Fig. 5). We conclude that, despite the worst case theoretical complexity, i.e., $O((2^{r+1} - 1)|G||N|)$, the real runtime of our implementation on simulated data is proportional to $2^{0.543r}|G||N|$ and outperforms the naïve approach starting from small r 's. We claim that a similar statement holds for the algorithm with level- k networks. In other words, it is possible to analyse empirical networks even with $r = k = 40$, since the exponent can be reduced by half.

6.3 Empirical tests

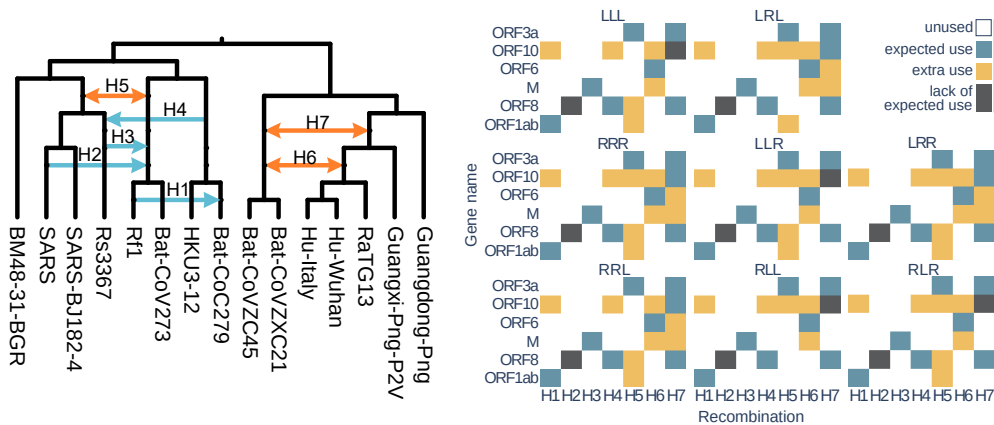
Our final experiment was conducted using real data. We revisited research presented in [29] concerning coronavirus (CoV) phylogeny. In the cited paper, the authors investigated the origins of the SARS-CoV-2 virus which causes a severe respiratory disease. They validated the hypothesis that the appearance of this new coronavirus is a consequence of several recombination events that occurred between some evolutionarily close CoV species. The results showed that both intergenic and intragenic recombination played a significant role in the SARS-CoV-2 evolution.

The goal of our study was to test whether scenarios with the lowest DC cost inferred for the phylogenetic network from [29] and individual gene trees confirm recombinations identified in the cited paper. In our experiment, we focused on intergenic recombinations, in which a whole gene is transferred from one species and integrated into another species genome.

Data preparation: In the gene trees inference step, we followed the cited work. We selected 15 out of 25 examined coronavirus species, omitting a few species to avoid multifurcations. As representatives of SARS-Cov-2, two variants were used. One was sampled from a patient from Wuhan (Hu-Wuhan), the origin of the pandemic spread of coronavirus, and the other was collected in Italy (Hu-Italy). Other selected species were RaTG13 bat CoV from *R. affinis* which, at first, was considered the only close relative of SARS-CoV-2, bat CoV ZC45 and ZXC21 strains from Zhejiang province of China (Bat-CoVZC45, and Bat-CoVZXC21), bat coronaviruses collected from species found in several provinces of China and from Bulgaria (Rf1, HKU3-12, BatCoV273, BatCoV279, and BM48-31 BGR), two CoV strains from Guangdong and Guangxi pangolins (Guangdong-Png, Guangxi-Png-P2V), and three SARS CoV related species (SARS, SARS-BJ182-4, and Rs3367). For the list of full names please refer to Appendix A.4. Coronavirus sequences were obtained from GenBank [2] at <https://www.ncbi.nlm.nih.gov/> and GISAID [38] databases. In the studied phylogenetic network, recombinations were found in the case of the genes M, ORF1ab, ORF3a, ORF6, ORF8, and ORF10; therefore, our research was focused on this set of genes. Multiple sequence alignments for the gene families were performed with MUSCLE [11] and corrected by GBlocks [7] with less stringent correction option. The ML gene trees were inferred using RAxML [40] with parameters described in [29]. All species were present in all gene families except ORF8, which lacks the BM48-31-BGR species.

Phylogenetic networks: The coronavirus tree with marked intergenic recombinations ($H1$ - $H7$) identified in [29] is depicted in Fig. 6. Since the direction of three out of seven recombinations was not certain, we prepared 8 networks corresponding to all combinations of the directions of gene transfers. Each network is named with three letters L/R responding to the direction of $H5$, $H6$ and $H7$, respectively, i.e. in the LRL network $H5$ and $H7$ are directed left and $H6$ is directed right. Please note that the inferred networks are not tree-child and therefore in this experiment we use the extended version of our algorithm described in Section 5.4.

Discussion: The results of the experiment are depicted in Fig 6. For each gene family, we checked whether the expected reticulation edge was used by the inferred scenario with the lowest DC cost. We can distinguish three possibilities for reticulation edge e : 1. e was used by the expected gene, 2. e was used by one or more extra genes, and 3. the expected gene didn't use e . We were able to confirm most of the reticulations except two: $H2$ that was reported in [29] with the lowest support wasn't confirmed by any of the networks, and $H7$



■ **Figure 6** *Left*: Coronavirus species tree with recombinations $H1-H7$ reported in [29]. Genes-recombinations assignment from [29]: $H1$:ORF1ab, $H2$:ORF8, $H3$:M, $H4$:ORF8, $H5$:ORF3a, $H6$:ORF6, $H7$:ORF3a, ORF8, ORF10. *Right*: Results showing which recombination edges were used by each gene tree in the scenarios with the lowest DC cost for each network variant.

was confirmed only by networks with $H6$ directed right. The most extra uses were found for $H5$ and gene $ORF10$ which gene tree had low support values. Further research might be performed for these cases. The least extra uses were present in $RLL(5)$ and $LLL(6)$, which may be some lead when investigating the direction of $H6$ and $H7$.

We observed that, for a fixed gene tree, the set of optimal displayed trees inferred by our algorithm and their cost, is independent of the network variant. This phenomenon needs further theoretical investigation. Costs for each gene tree have the following values: ORF3a: 0, ORF10: 5, ORF6: 1, M: 3, ORF8: 2, ORF1ab: 2. This observation may lead to discovering some important property and can be a subject for further investigation.

7 Conclusions

In this work, we have investigated the problem of inferring an optimal tree displayed by a network under the deep coalescence cost and proposed a new score to approximate the cost. We have shown that the score has nice mathematical and computational properties allowing us to bound the cost of an optimal displayed tree from below. We have proposed a polynomial-time dynamic programming (DP) algorithm to compute the score together with the set of used reticulation edges that yielded the score. Then, we have proposed a new way to infer a displayed tree by a recursive procedure resolving conflicts detected in multiple invocations of DP. In the worst case, our algorithm to infer an optimal tree requires $2^{r+1}-1$ DP invocations, where r is the number of reticulations. However, numerous tests on simulated data have indicated that the exponent may be reduced by half on average. This phenomenon is explained by similarity, i.e., we expect a low number of conflicts if a gene tree is more congruent with its network. In other words, the average runtime of $\Omega(2^{0.543r}|G||N|)$ can compete on empirical datasets with exhaustive enumeration strategies (either on the level of a whole network or each biconnected component independently) commonly used in alternative approaches to scoring tree-network pairs [25, 27, 43]. We also claim that the statement holds for level- k networks by replacing r by k in the formula. We conclude that our conflict resolution algorithm enables analyses of complex networks with dozens of reticulation events. We also claim that resolving conflicts returned by dynamic programming is a new alternative towards designing efficient algorithms that utilize internal similarities of empirical datasets.

Future Outlooks. Resolving conflicts in the usage of reticulation edges can be naturally generalized to other cost functions, e.g., gene duplication cost. Also, it is not difficult to extend DP to analyze unrooted gene trees. Another critical question is whether the runtime exponent can be further reduced, e.g., by choosing optimal scenarios with the smallest possible sets of conflicted reticulation edges. Furthermore, we would like to test the efficiency and accuracy of the branch and bound algorithm to approximate the optimal cost. Also, we plan to apply the methods in computationally demanding problems of network inference from sets of gene trees, which may require reimplementing in a low-level programming language (e.g., C/C++).

References

- 1 Eric Bapteste, Leo van Iersel, Axel Janke, Scot Kelchner, Steven Kelk, James O. McInerney, David A. Morrison, Luay Nakhleh, Mike Steel, Leen Stougie, and James Whitfield. Networks: expanding evolutionary thinking. *Trends in Genetics*, 29(8):439–441, 2013.
- 2 Dennis A Benson, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. Genbank. *Nucleic Acids Research*, 39(suppl_1):D32–D37, 2010.
- 3 Luis Boto. Horizontal gene transfer in evolution: facts and challenges. *Proceedings of the Royal Society B: Biological Sciences*, 277(1683):819–827, November 2009.
- 4 Gabriel Cardona, Francesc Rosselló, and Gabriel Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(4):552–569, 2008.
- 5 Gabriel Cardona, Francesc Rossello, and Gabriel Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(4):552–569, October 2009.
- 6 Gabriel Cardona and Louxin Zhang. Counting and enumerating tree-child networks and their subclasses. *Journal of Computer and System Sciences*, 114:84–104, 2020.
- 7 Jose Castresana. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Molecular Biology and Evolution*, 17(4):540–552, 2000.
- 8 Ruchi Chaudhary, J Gordon Burleigh, and Oliver Eulenstein. Efficient error correction algorithms for gene tree reconciliation based on duplication, duplication and loss, and deep coalescence. In *BMC Bioinformatics*, volume 13, pages 1–10. BioMed Central, 2012.
- 9 Charles Choy, Jesper Jansson, Kunihiko Sadakane, and Wing-Kin Sung. Computing the maximum agreement of phylogenetic networks. *Theoretical Computer Science*, 335(1):93–107, 2005.
- 10 Beatrice Donati, Christian Baudet, Blerina Sinimeri, Pierluigi Crescenzi, and Marie-France Sagot. EUCALYPT: efficient tree reconciliation enumerator. *Algorithms for Molecular Biology*, 10(1):3, 2015.
- 11 Robert C Edgar. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5(1):1–19, 2004.
- 12 Mareike Fischer, Leo Van Iersel, Steven Kelk, and Celine Scornavacca. On computing the maximum parsimony score of a phylogenetic network. *SIAM Journal on Discrete Mathematics*, 29(1):559–585, 2015.
- 13 William Fletcher and Ziheng Yang. Indelible: a flexible simulator of biological sequence evolution. *Molecular Biology and Evolution*, 26(8):1879–1888, 2009.
- 14 Andrew R. Francis and Mike Steel. Which phylogenetic networks are merely trees with additional arcs? *Systematic Biology*, 64(5):768–777, June 2015.
- 15 Paweł Górecki, Oliver Eulenstein, and Jerzy Tiuryn. Unrooted tree reconciliation: A unified approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(2):522–536, 2013.
- 16 Paweł Górecki and Jerzy Tiuryn. DLS-trees: A model of evolutionary scenarios. *Theoretical Computer Science*, 359(1-3):378–399, 2006.

- 17 Paweł Górecki and Jerzy Tiuryn. Urec: a system for unrooted reconciliation. *Bioinformatics*, 23(4):511–512, 2007.
- 18 Benjamin E. Goulet, Federico Roda, and Robin Hopkins. Hybridization in plants: Old ideas, new techniques. *Plant Physiology*, 173(1):65–78, November 2016.
- 19 Stéphane Guindon, Jean-François Dufayard, Lefort Vincent, Maria Anisimova, Wim Hordijk, and Olivier Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: Assessing the performance of phylml 3.0. *Systematic Biology*, 59(3):307–321, 2010.
- 20 Dan Gusfield. *ReCombinatorics: the Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks*. MIT Press, Boston, 2014.
- 21 Klaas Hartmann, Dennis Wong, and Tanja Stadler. Sampling trees from evolutionary models. *Systematic Biology*, 52(4):465–476, 2010.
- 22 Katharina T. Huber and Vincent Moulton. Phylogenetic networks from multi-labelled trees. *Journal of Mathematical Biology*, 52(5):613–632, 2006.
- 23 Katharina T. Huber, Vincent Moulton, Mike Steel, and Taoyang Wu. Folding and unfolding phylogenetic trees and networks. *Journal of Mathematical Biology*, 73(6-7):1761–1780, 2016.
- 24 Daniel H. Huson, Regula Rupp, and Celine Scornavacca. *Phylogenetic Networks: Concepts Algorithms and Applications*. Cambridge University Press, New York, 2010.
- 25 Leo Van Iersel, Mark Jones, and Celine Scornavacca. Improved maximum parsimony models for phylogenetic networks. *Systematic Biology*, 67(3):518–542, December 2017.
- 26 Remie Janssen and Yukihiro Murakami. Linear time algorithm for tree-child network containment. In *International Conference on Algorithms for Computational Biology*, pages 93–107. Springer, 2020.
- 27 Matthew LeMay, Ran Libeskind-Hadas, and Yi-Chieh Wu. A polynomial-time algorithm for minimizing the deep coalescence cost for level-1 species networks. *bioRxiv*, November 2020.
- 28 Wayne P. Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997.
- 29 Vladimir Makarenkov, Bogdan Mazouze, Guillaume Rabusseau, and Pierre Legendre. Horizontal gene transfer and recombination analysis of SARS-CoV-2 genes helps discover its close relatives and shed light on its origin. *BMC Ecology and Evolution*, 21(1):1–18, 2021.
- 30 Diego Mallo, Leonardo De Oliveira Martins, and David Posada. Simphy: Phylogenomic simulation of gene, locus, and species trees. *Systematic Biology*, 65(2):334–344, 2015.
- 31 Alexey Markin, Tavis K. Anderson, Venkata SKT Vadali, and Oliver Eulenstein. Robinson-foulds reticulation networks. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 77–86, 2019.
- 32 Sarah M. McDonald, Martha I. Nelson, Paul E. Turner, and John T. Patton. Reassortment in segmented RNA viruses: mechanisms and outcomes. *Nature Reviews Microbiology*, 14(7):448–460, May 2016.
- 33 Siavash Mirarab and Tandy Warnow. ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. *Bioinformatics*, 31(12):i44–i52, 2015.
- 34 Erin K Molloy and Tandy Warnow. Fastmulrfs: fast and accurate species tree estimation under generic gene duplication and loss models. *Bioinformatics*, 36(Supplement_1):i57–i65, 2020.
- 35 Yukihiro Murakami, Leo van Iersel, Remie Janssen, Mark Jones, and Vincent Moulton. Reconstructing tree-child networks from reticulate-edge-deleted subnetworks. *Bulletin of Mathematical Biology*, 81(10):3823–3863, 2019.
- 36 Matthew D. Rasmussen and Manolis Kellis. Unified modeling of gene duplication, loss, and coalescence using a locus tree. *Genome Research*, 22(4):755–765, 2012.
- 37 Celine Scornavacca, Joan Carles Pons Mayol, and Gabriel Cardona. Fast algorithm for the reconciliation of gene trees and lgt networks. *Journal of Theoretical Biology*, 418:129–137, 2017.
- 38 Yuelong Shu and John McCauley. Gisaids: Global initiative on sharing all influenza data—from vision to reality. *Eurosurveillance*, 22(13):30494, 2017.

- 39 Claudia Solís-Lemus and Cécile Ané. Inferring phylogenetic networks with maximum pseudo-likelihood under incomplete lineage sorting. *PLOS Genetics*, 12(3):1–21, 2016.
- 40 Alexandros Stamatakis. Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22(21):2688–2690, 2006.
- 41 Cuong Than and Luay Nakhleh. Species tree inference by minimizing deep coalescences. *PLoS Computational Biology*, 5(9):e1000501, 2009.
- 42 Cuong Than, Derek Ruths, and Luay Nakhleh. PhyloNet: a software package for analyzing and reconstructing reticulate evolutionary relationships. *BMC Bioinformatics*, 9(1), July 2008.
- 43 Thu-Hien To and Celine Scornavacca. Efficient algorithms for reconciling gene trees and species networks via duplication and loss events. *BMC Genomics*, 16(S10), 2015.
- 44 Yi-Chieh Wu, Matthew D Rasmussen, Mukul S Bansal, and Manolis Kellis. Most parsimonious reconciliation in the presence of gene duplication, loss, and deep coalescence using labeled coalescent trees. *Genome Research*, 24(3):475–486, 2014.
- 45 Yun Yu, R. Matthew Barnett, and Luay Nakhleh. Parsimonious inference of hybridization in the presence of incomplete lineage sorting. *Systematic Biology*, 62(5):738–751, July 2013.
- 46 Yun Yu, James H. Degnan, and Luay Nakhleh. The probability of a gene tree topology within a phylogenetic network with applications to hybridization detection. *PLoS Genetics*, 8(4):e1002660, April 2012.
- 47 Yun Yu, Tandy Warnow, and Luay Nakhleh. Algorithms for MDC-based multi-locus phylogeny inference: Beyond rooted binary gene trees on single alleles. *Journal of Computational Biology*, 18(11):1543–1559, November 2011.
- 48 Louxin Zhang. From gene trees to species trees ii: Species tree inference by minimizing deep coalescence events. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6):1685–1691, 2011.
- 49 Jiafan Zhu, Yun Yu, and Luay Nakhleh. In the light of deep coalescence: revisiting trees within networks. *BMC Bioinformatics*, 17(S14), November 2016.

A Experimental evaluation

A.1 Random datasets results

Figure 7 presents complete set of diagrams for random datasets.

A.2 Simulated data preparation

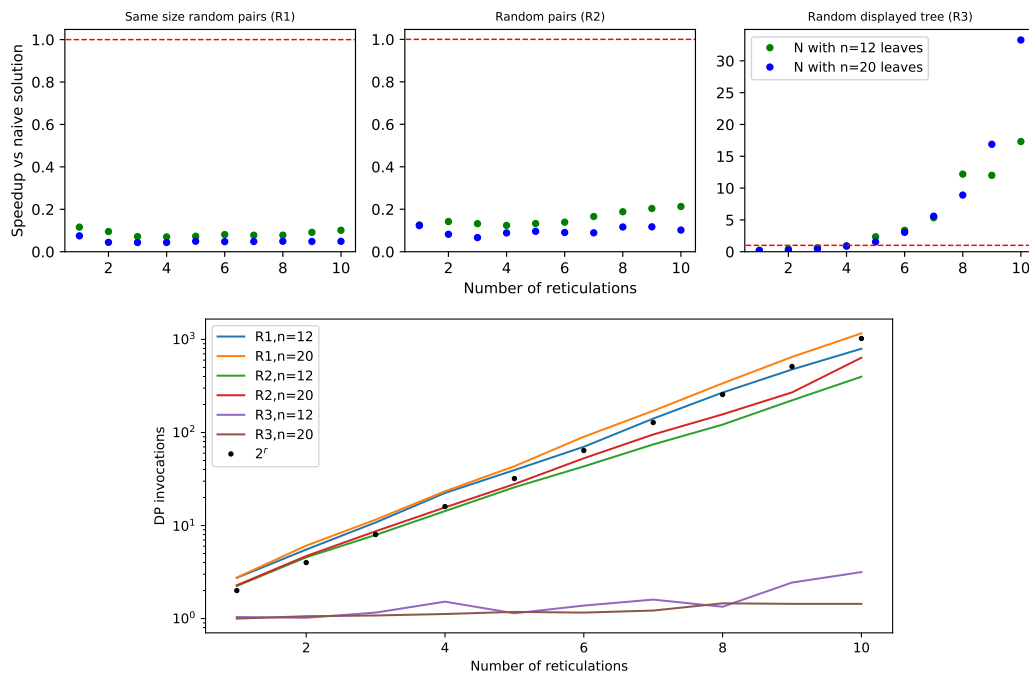
A.2.1 Species trees parameters

To simulate species trees, we ran `sim.bd.taxa.age` function implemented in R package `TreeSim` version 2.4 [21] with the following parameters: tree height = 1800000337.5 years, speciation rate = $1.8 * 10^{-9}$ events/year and extinction rate = 0 events/year. The number of taxa was set to either 12 or 20.

A.2.2 Inferring networks from species trees

After simulating a species tree using the general sampling approach, we assigned a time value to each of its nodes, corresponding to a length of a path connecting the root with the node. Note that the general sampling approach produces ultrametric species trees, therefore time values assigned to the leaves were equal.

Then we inferred a tree-child phylogenetic network with r reticulations from each of the species trees. We added r reticulations one by one by repeating the following procedure. To add a reticulation edge to a species tree/network, we start by randomly choosing a pair of branches and subdivide them, making two new vertices. We then sample a time value for each of the vertices from `uniform(vertex.parent.time, vertex.child.time)`. Finally, we add a



■ **Figure 7** *Top*: Performance of Alg. 1 vs naive approach for random datasets R1, R2, R3. *Bottom*: Average number of DP invocations necessary to calculate an answer for random datasets R1, R2 and R3.

reticulation edge from the vertex with the lower time value t_l to the vertex with the higher time value t_h . If the addition disturbs the tree-child property, we delete the reticulation edge and contract the vertices. Otherwise, we set the length of the reticulation edge to $t_h - t_l$.

Note that this way of introducing reticulation edges does not change time values of the leaves, hence all displayed trees of the resulting network are ultrametric and have equal heights.

A.2.3 Gene trees parameters

To simulate a gene tree from a randomly chosen displayed tree we ran SimPhy [30] with the following command:

```
simphy_lnx64 -sr <displayed tree> -rg 1 -rl 1 -si F:1 \
-sp F:$ps su F:0.0000000004 -lb F:$dl -ld F:lb -ll 3\
-hg LN:1.5,1 -oc 1 -o <output folder> -v 0 -cs 22
```

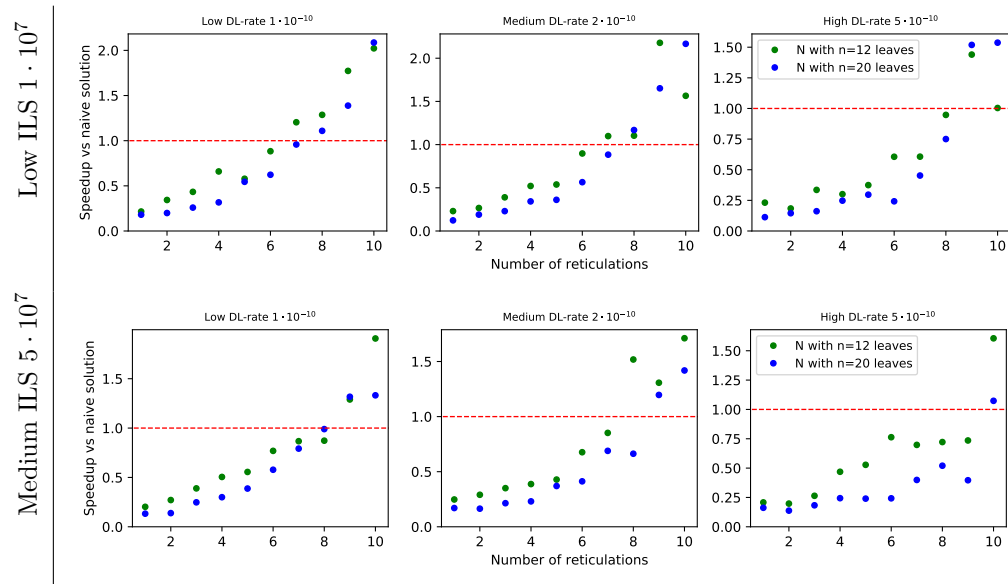
where <displayed tree> is a nexus file containing the randomly chosen displayed tree, \$ps is a population size and \$dl is a duplication/loss rate.

A.2.4 Sequences parameters

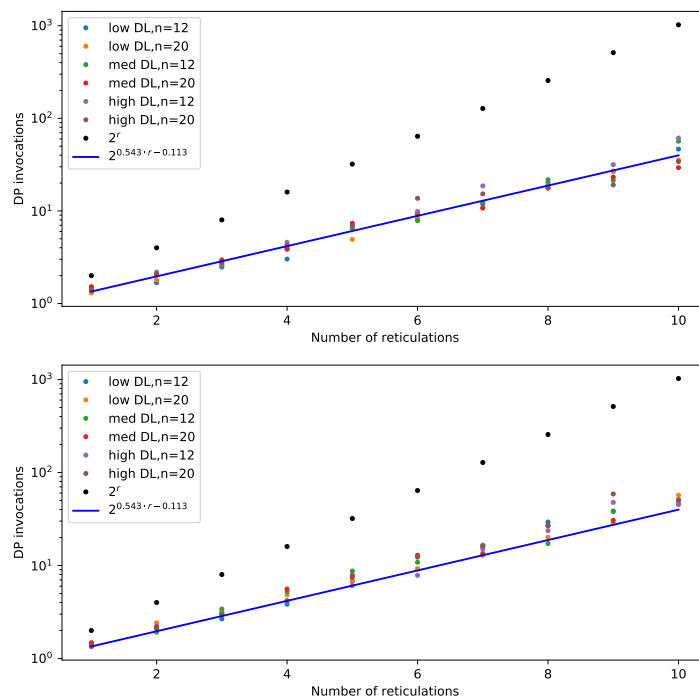
The sequences were simulated using INDELible v1.03 [13] by running a perl script INDELible_wrapper.pl included in SimPhy [30]. We used GTR model with substitution rates (AC, AG, AT, CG, CT and GT respectively) sampled for each gene tree from Dirichlet(12.776722, 20.869581, 5.647810, 9.863668, 30.679899, 3.199725). The nucleotide frequencies (T, C, A and G respectively) were sampled from Dirichlet(113.48869, 69.02545, 78.66144, 99.83793), whilst α parameter was sampled from Lognormal(-0.470703916, 0.348667224). The alignment length was set to 1000 bp.

A.3 Simulated datasets results

Figures 8 and 9 present the complete set of diagrams.



■ **Figure 8** Performance of Alg. 1 vs naive approach for six simulated datasets.



■ **Figure 9** *Top*: Average number of DP invocations necessary to calculate an answer for datasets with low ILS. *Bottom*: Average number of DP invocations necessary to calculate an answer for datasets with medium ILS. Recall that blue line represents coefficients calculated by linear regression for data with low and medium ILS combined.

A.4 Empirical experiment dataset

Organism names and database accession numbers of all species used in our research are listed in Table 1.

■ **Table 1** List of the full names and database accession numbers of coronavirus species used in our research. Species were chosen from the dataset studied in [29].

Abbreviated name	Organism name	Accession Number (GenBank/GISAID)	Host organism
Hu-Wuhan	BetaCoV Wuhan-Hu-1	NC_045512.2	Human
Hu-Italy	hCoV-19/Italy/ABR-IZSGC-TE4836/2020	EPI_ISL_418260	Human
RaTG13	Bat CoV RaTG13	MN996532.1	Bat
Guangdong-Png	hCoV-19/pangolin/Guangdong/1/2019	EPI_ISL_410721	Pangolin
Guanxi-Png-P2V	Pangolin CoV isolate PCoV_GX-P2V	MT072864	Pangolin
Bat-CoVZC45	Bat SARS-like CoV isolate bat-SL-CoVZC45	MG772933.1	Bat
Bat-CoVZXC21	Bat SARS-like CoV isolate bat-SL-CoVZXC21	MG772934.1	Bat
Bat-CoV273	Bat CoV BtCoV/273/2005	DQ648856.1	Bat
Bat-CoV 279	Bat CoV BtCoV/279/2005	DQ648857.1	Bat
HKU3-12	Bat SARS CoV HKU3-12	GQ153547.1	Bat
Rf1	Bat SARS CoV Rf1	DQ412042.1	Bat
SARS	SARS CoV BJ01	AY278488.2	Human
SARS-BJ182-4	SARS CoV BJ182-4	EU371562	Human
Rs3367	Bat SARS-like CoV Rs3367	KC881006.1	Bat
BM48-31-BGR	Bat CoV BM48-31/BGR/2008	GU190215.1	Bat