Consensus Clusters in Robinson-Foulds Reticulation Networks

Alexey Markin
Department of Computer Science, Iowa State University, Ames, IA, USA
amarkin@iastate.edu

Oliver Eulenstein
Department of Computer Science, Iowa State University, Ames, IA, USA
https://www.cs.iastate.edu/people/oliver-eulenstein
oeulenst@iastate.edu

Abstract

Inference of phylogenetic networks – the evolutionary histories of species involving speciation as well as reticulation events – has proved to be an extremely challenging problem even for smaller datasets easily tackled by supertree inference methods. An effective way to boost the scalability of distance-based supertree methods originates from the Pareto (for clusters) property, which is a highly desirable property for phylogenetic consensus methods. In particular, one can employ strict consensus merger algorithms to boost the scalability and accuracy of supertree methods satisfying Pareto; cf. SuperFine. In this work, we establish a Pareto-like property for phylogenetic networks. Then we consider the recently introduced RF-Net method that heuristically solves the so-called *RF-Network problem* and which was demonstrated to be an efficient and effective tool for the inference of hybridization and reassortment networks. As our main result, we provide a constructive proof (entailing an explicit refinement algorithm) that the Pareto property applies to the RF-Network problem when the solution space is restricted to the popular class of tree-child networks. This result implies that strict consensus merger strategies, similar to SuperFine, can be directly applied to boost both accuracy and scalability of RF-Net significantly. Finally, we further investigate the optimum solutions to the RF-Network problem; in particular, we describe structural properties of all optimum (tree-child) RF-networks in relation to strict consensus clusters of the input trees.

2012 ACM Subject Classification Applied computing → Computational biology; Mathematics of computing → Graph theory

Keywords and phrases Phylogenetics, phylogenetic tree, phylogenetic network, reticulation network, Robinson-Foulds, Pareto, RF-Net

Digital Object Identifier 10.4230/LIPIcs.WABI.2019.12

Funding This material is based upon work supported by the National Science Foundation under Grant No. 1617626.

Acknowledgements The authors want to thank the reviewers for their valuable and constructive comments.

1 Introduction

Inferring evolutionary histories of species is a crucial area of study in the biological sciences [10]. The inference of such histories as phylogenetic trees, while still an immensely challenging problem, is gradually becoming tractable on the scale of thousand(s) of species genomes [26, 4].

On the other hand, today, it is well known that evolutionary histories of many species involve complex evolutionary events such as hybridization, reassortment, recombination, and horizontal gene transfer [14]. In this case, evolutionary histories are modeled using *phylogenetic networks* that contain *reticulation vertices* in addition to classical speciation

vertices. Unfortunately, whereas phylogenetic networks potentially represent significantly more powerful tools than phylogenetic trees, the reticulate evolutionary model is of greater complexity than the standard speciation model. In this regard, the current state-of-the-art tools for the estimation of phylogenetic networks cannot yet compete with the inference methods for trees in terms of accuracy and scalability.

Recently, a novel method for the inference of hybridization and reassortment networks has been introduced, called RF-Net [17]. RF-Net was shown to outperform its closest counterparts in terms of scalability and be able to estimate networks with more than 150 taxa credibly. However, there are still significant limitations, particularly in terms of the maximum tractable number of reticulation vertices.

RF-Net follows the *extended hybridization framework* established in [17]. More precisely, the classical *hybridization framework*, formulated in the influential work of Baroni et al. [2], seeks a network that would display each of the input phylogenetic trees exactly. The extended framework then additionally accounts for *errors* that are typically present in input trees for supertree/super-network studies [3, 25]. That is, it defines a cost of embedding an input tree into a candidate network that measures how close that input tree is to be displayed in the network. More formally, the *embedding cost* is defined as the minimum distance between the input tree and each tree displayed in the candidate network (Figure 1 illustrates the concept of displayed trees). Whereas, generally, any established distance measurement for phylogenetic trees can be used to define the embedding cost, RF-Net uses the popular Robinson-Foulds (RF) metric [24]. A related concept was also explored by Yu et al. [29], where the parsimonious ILS principle is combined with the hybridization framework.

Given a collection of input trees, RF-Net attempts to solve the problem of finding a phylogenetic network with at most r reticulation vertices that minimizes the overall embedding cost of the input trees, i.e., the overall sum of the individual embedding costs, as well as the number of reticulations. We refer to this problem as the *RF-network problem*. The RF-network problem is NP-hard [17], and it is similar to the standard supertree (median tree) problem formulations that seek supertrees minimizing the overall distance towards the input trees. Such distance-based supertree methods are widely used for a task of large-scale species tree reconstruction [3]. For example, ASTRAL, a popular supertree software package, seeks a tree minimizing the *quartet distance* towards the input trees [19]. Further, RF-based supertree methods, e.g. [1, 27], as well as gene tree parsimony (GTP) supertree methods, e.g., [9, 12], have been successfully applied by the phylogenetic community (cf. [11, 22]).

One of the most important properties for the distance-based supertree methods is the so-called *Pareto for clusters* [1] property [23]. Pareto for clusters is a highly desired property, both from theoretical as well as application perspectives [6, 20]. This property was introduced in the context of *consensus methods* that seek to "summarize" a collection of input trees over the same species set. A consensus method is Pareto if every cluster that appears in *all* input trees (a *strict consensus cluster*) also appears in the consensus of these trees obtained by that method. This notion was then naturally adopted for distance-based supertree methods when restricted to the consensus setting (i.e., input trees have the same leaf-sets) [15]. Given that there could be multiple optimum solutions to a distance-based supertree problem, a respective distance measurement is said to (i) *satisfy Pareto* if *each* optimum solution contains all the strict consensus clusters from the input trees and (ii) *satisfy weak Pareto* if at least one optimum solution has that property [21]. If a distance measurement satisfies (weak) Pareto then the respective supertree problem, in a consensus setting, can be solved using a

[1] This Pareto property introduced by Neumann [23] in the context of phylogenetic consensus methods is not to be confused with the Pareto efficiency/optimality term originating from the field of economics [18].

parameterized approach, where, first, a strict consensus tree of the input trees is constructed, and then each polytomy in the strict consensus tree is resolved using the original supertree method. In case the input trees are relatively similar, this approach yields a much more efficient and accurate supertree method [21].

Typically, however, the input trees can be unrooted and can have incomplete sets of taxa. In this case, the scalability boost can be achieved using the *strict consensus merger (SCM)* methods [13, 26]. Such strategy was first formulated and evaluated by Swenson et al. [26]; their method, SuperFine, was shown to improve the scalability and accuracy of supertree methods. Generally, any distance-based supertree method with the respective distance measurement satisfying at least weak Pareto can largely benefit from using SCM algorithms. For example, recently, the NP-hard gene duplication supertree problem (GD) [16] was efficiently and effectively approached [20] by combining the greedy SCM method [13] with the exact dynamic programming GD solution [8] and the popular GD heuristic, DupTree [28].

Our contribution. In this work, we establish a Pareto-like property in the context of phylogenetic networks and prove that the Robinson-Foulds embedding cost satisfies it. This result immediately implies that SCM methods can be used to improve the scalability and accuracy of the RF-Net method significantly. We demonstrate this result for the commonly used (restricted) class of networks called *tree-child networks* [7].

More precisely, we define the notion of C-separated networks for some cluster C. A network is C-separated if one can remove a single edge (u, v) from a network and obtain two disconnected subnetworks/subgraphs, such that the subnetwork rooted at v has the leaf-set C. For example, the network N from Figure 1 is $\{b, c\}$-separated, but not $\{e, b, c\}$-separated. Intuitively, this notion implies that the evolutionary history of the species in C (up to their common ancestor) is not inter-related with lineages of any other species in the network.

Founded on this notion, our main result is as follows: when the solution space is constrained to the class of *tree-child networks*, there always exists an *optimum* RF-network that is C-separated for every strict consensus cluster C. Adopting the above-discussed terminology for distance-based supertree methods, this result implies that the RF embedding cost satisfies weak Pareto for tree-child networks. It is important to note that, since our proof is by construction, it explicitly provides an algorithm that can bring a non-C-separated network to the C-separated form with the same or an improved overall embedding cost. To achieve our result, we introduce three edit operations on networks and prove that they preserve specific non-trivial properties. Then we use these operations to design the said algorithm and complete the proof.

Further, we describe conditions for the existence of non-C-separated optimum RF-networks. We prove that such networks can appear only due to *uncertainty in the ordering of some reticulation events* that is not resolved by the input trees. That way, when uncertainty is not present, the RF embedding cost satisfies (strong) Pareto for tree-child networks, rather than weak Pareto. In fact, we prove that even if for some collection of input trees a non-C-separated optimum RF-network N exists, it must be equivalent to another RF-network N' *up to a reordering of reticulation events* (this notion is formally defined in Section 3.2), such that N' is both optimum and C-separated.

2 Preliminaries

A *(phylogenetic) network* is a directed acyclic graph (DAG) with a designated root and with all other vertices either of in-degree one and out-degree two (*tree vertices*), in-degree two and out-degree one (*reticulation vertices*), or in-degree one and out-degree zero (*leaves*). All

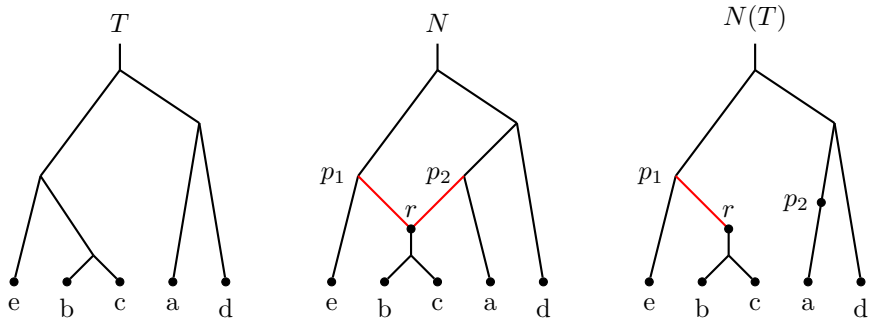


Figure 1 An example of tree T displayed in network N with one reticulation vertex, r. T is displayed in N by removing the reticulation edge (p_2, r) and obtaining a subdivision $N(T)$.

leaves are bijectively labeled by a label-set X and are identified with the elements of X. For convenience, networks are *planted*, i.e., the root has in-degree zero and out-degree one. For a network N the root and leaves are denoted by $\rho(N)$ and $\mathsf{L}(N)$ respectively. For every vertex v we denote the set of children, parent(s), and sibling(s) by $\mathsf{Ch}(v)$, $\mathsf{Pa}(v)$, and $\mathsf{Sb}(v)$ respectively. Note that if v is the child of a reticulation w then, we define $\mathsf{Sb}(v)$ to be the two siblings of w.

We distinguish *reticulation edges* – edges entering a recitulation vertex – and *tree edges* – edges entering a tree vertex. A *tree-path* is a directed path that consists of tree edges. We say that *vertex v has a tree-path to vertex w* if either $v = w$ or there is a tree-path from v to w.

A vertex v is a *descendant* of w if there is a directed path from w to v (we consider each vertex to be a descendant of itself); w is also called an *ancestor* of v. Alternatively, we say that v is *below* w or w is *above* v. A *(hardwired) cluster* of vertex v, denoted by C_v, is the set of all leaves that are descendants of v. Generally, we call any set of leaves a *cluster*.

A *(phylogenetic) tree* is a network with no reticulation vertices. A *least common ancestor (LCA)* of two vertices v, w in a tree, denoted by $\mathsf{lca}(v, w)$, is the lowest vertex x such that v and w are descendants of x. Two clusters (leaf-sets) C_1 and C_2 are called compatible if there exists a tree that contains both of them. Equivalently, the clusters are compatible if either $C_1 \subseteq C_2$, $C_2 \subseteq C_1$, or $C_1 \cap C_2 = \emptyset$.

Displayed trees. A tree T is *displayed* in a network N (with the same leaf set), if one can remove exactly one reticulation edge from each reticulation vertex, then remove all potentially appearing non-labeled vertices with out-degree zero, and obtain a subdivision of T – denoted $N(T)$. See an example in Figure 1. We say that an edge of a network is used to display tree T if there is a way to obtain such subdivision of T without removing this edge. Note that in general there could be several ways to display tree T.

Tree-child networks. A network is called *tree-child* if each vertex has at least one outgoing tree edge. It is easy to see that each vertex in a tree-child network must have a tree-path going to some leaf. This is a crucial property that we will employ in this work.

For the convenience of further discussion, we define a surjective mapping from the vertices of a tree-child network N, denoted here V_N, to the vertices of a tree T displayed in N. Let us fix a way to display T in N. It is not difficult to observe that for a tree-child network removing reticulation edges in order to display a tree will not result in out-degree 0 vertices. That is, the fixed subdivision $N(T)$ of T contains all vertices of N. Since $N(T)$ is a subdivision of T, each vertex w from V_N that has two children in $N(T)$ must have a unique equivalent

in T. We then map each such w to the corresponding equivalent vertex in T. Further, we map leaves of N to the leaves of T with the same label. Finally, for each v in V_N that has exactly one child in $N(T)$, let w denote the highest descendant of v in $N(T)$ that was already mapped (i.e., w is either a leaf or has two children); we then map v to the same vertex in T that w is mapped to. For example, in Figure 1 vertices a and p_2 from N are both mapped to leaf a of T.

We then say that a vertex $v \in N$ *corresponds to* vertex $v' \in T$ if v is mapped to v'.

Embedding cost. We define the cost of embedding a tree T in a network N on the same leaf set using the standard Robinson-Foulds (RF) distance [24]. The cost should be zero, when the tree is displayed in the network and positive otherwise. Then the cost is defined as follows: let $\mathcal{P}_N$ be a set of all trees displayed in N, then

$$\delta(T, N) := \min_{S \in \mathcal{P}_N} RF(T, S),$$

where $RF(T, S)$ is the Robinson-Foulds (cluster) distance defined as the size of the symmetric difference between the cluster representations of two trees.

A tree S, displayed in N, that has the minimum RF distance to T is called *an embedding of T (in N)*. Note that in general T can have multiple embeddings.

3 Consensus clusters in networks

We now establish a definition central to this work.

▶ **Definition 1** (C-separated networks). *Given a cluster C, network N is called C-separated if it contains an edge (u, v) such that removing this edge disconnects the network into two networks (components) N_1 and N_2 with*

- $\mathsf{L}(N_1) = C$; $\mathsf{L}(N_2) = \mathsf{L}(N) \setminus C$;
- *and no edges going across the components in the underlying undirected graph.*

For example, network N from Figure 1 is $\{b, c\}$-separated.

Let $\mathcal{T}$ be a set of trees over the same leaf set. A cluster C is said to be a *consensus cluster (of $\mathcal{T}$)* if for each $T \in \mathcal{T}$ there is a vertex v such that $C_v = C$.

Given a network N over the leaf set as $\mathcal{T}$, we define the distance between $\mathcal{T}$ and N as follows:

$$\mathsf{d}(\mathcal{T}, N) := w_e \cdot \sum_{T \in \mathcal{T}} \delta(T, N) + w_r \cdot R(N),$$

where $R(N)$ is the number of reticulation vertices in N and $w_e, w_r > 0$ are integer coefficients that weigh the overall embedding cost and the number of reticulations respectively. Note that, for convenience, we define the distance slightly differently from the original definition in [17], where a limit on the number of reticulations was given instead of a weight; however, it does not affect the results stated in this work.

3.1 Consensus clusters in embeddings

We now formulate our core result. To prove it we introduce our main machinery that is also used later to demonstrate stronger results.

▶ **Theorem 2.** *Let N be a tree-child network with the minimum $\mathsf{d}(\mathcal{T}, N)$ distance; then for each consensus cluster C of $\mathcal{T}$ and each $T_i \in \mathcal{T}$, every embedding of T_i in N has cluster C.*

The remainder of the section is dedicated to the proof of the theorem.

Proof strategy. Assume, for the purpose of contradiction, that an embedding of some T_i, $S_i \in \mathcal{P}_N$, does not contain a consensus cluster C. Observe that this implies that N is not C-separated, since each tree displayed in a C-separated network must have cluster C.

We are going to transform N into a C-separated network with a smaller distance to $\mathcal{T}$. In order to do that, we define three transformations on networks that preserve displayed clusters that are compatible with C.

Formally, the transformations should satisfy the property defined in Definition 3.

▶ **Definition 3** (C-compatible transformation). *A network transformation operation that can transform N into a network N' is said to be C-compatible if for each tree T displayed in N there exists a tree T' displayed in N', such that T' contains **all** clusters of T that are compatible with C.*

Let V_C be the set of vertices in N that have a tree-path to some leaf in C. For convenience we classify the *tree vertices* in V_C as follows.

▶ **Definition 4** (V_C vertices classification). *A tree vertex $v \in V_C$ is exactly of one of the following types:*

- ***Type 1:** if v is a leaf; or if children of v both belong to V_C and both children are tree vertices.*
- ***Type 2:** if both children of v belong to V_C but one of the children is a reticulation vertex.*
- ***Type 3:** If only one child of v, u_1, belongs to V_C, while the other, u_2, does not. Note that in this case u_1 has to be a tree vertex, while u_2 could be a tree vertex or a reticulation vertex.*

We now define our first network rearrangement operation and prove that it is C-compatible.

▶ **Definition 5** (Network transformation T1). *If N contains an edge (w, z) such that $w, z \in V_C$, w, z are tree vertices, w is of Type 2, and z is of Type 3 then transformation T1 proceeds as follows:*

(i) *Let c denote the sibling of z and u denote the child of z such that $u \notin V_C$;*

(ii) *Remove edges (w, c) and (z, u) and add edges (w, u) and (z, c).*

That is, the transformation swaps specific children of z and w. Note that z becomes a Type 2 vertex and w – Type 3; that way T1 "moves" a Type 2 vertex down and a Type 3 vertex up. See the illustration in Figure 2a.

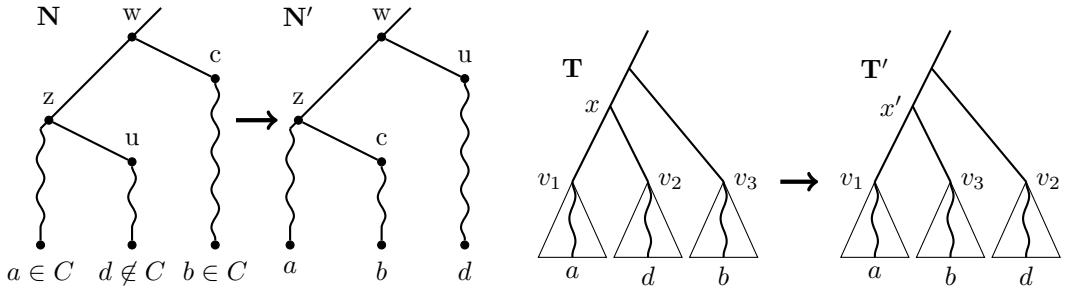
We now prove that this transformation could not increase the embedding distance of an input tree with cluster C to the network.

▶ **Lemma 6.** *T1 is C-compatible.*

Proof. Assume that T1 was performed as depicted in Figure 2a. Note that, by definition, z and c belong to V_C and therefore have tree-paths to some leaves a and b in C respectively (as shown in the figure as well). On the other hand, $u \notin V_C$ and therefore u has a tree-path to some $d \notin C$ (it could be that $u = d$ and/or $c = b$).

Consider some T displayed in N. We need to show that exists T' displayed in N' that contains all clusters of T compatible with C. First of all, observe that if T can be displayed in N by removing either (or both) of the edges (w, c) and (z, u) then T is also displayed in N' and the statement is trivially true.

Otherwise, let x denote $\mathsf{lca}_T(a, d)$ (as shown in Figure 2b). Since each edge (w, z), (z, u), and (w, c) are used for displaying T (as well as all edges on the tree-paths shown in Figure 2a, since they are tree edges and cannot be removed) then vertex z in N should correspond to x



(a) T1 transformation. **(b)** Displayed tree transformation.

Figure 2 (a): A network N transformed into network N' via transformation T1 as per Definition 5; waved lines indicate tree-paths to leaves. (b): A respective transformation of a tree T displayed in N (assuming that both edges (z, u) and (w, c) are used to display T) to a tree T' displayed in N'.

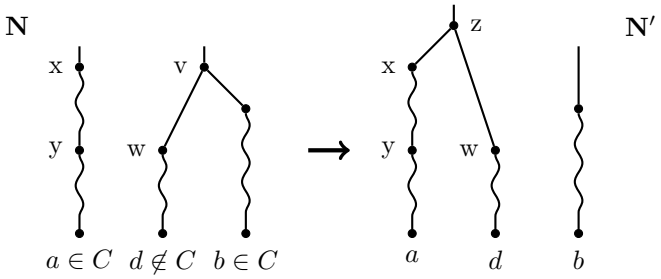


Figure 3 An illustration of transformation T2. Network N is transformed into N'. Potentially empty tree-paths are shown via waved lines.

in T. Further, vertex c should correspond to the sibling of x, v_3. Therefore, v_3 should have leaf b below it. It is then not difficult to see that a tree T' obtained by swapping subtrees rooted at v_3 and v_2 is displayed in N' (see Figure 2b).

Finally, observe that T differs from T' only by one cluster; namely, C_x, which is not present in T'. However, C_x is not compatible with C, since (i) C_x contains $a \in C$ and $d \notin C$ and (ii) $C \not\subset C_x$ because $b \notin C_x$. ◀

We now define two more C-compatible rearrangement operations.

▶ **Definition 7** (Network transformations T2 and T3). *Let x be a tree vertex in V_C of Type 2 or Type 1 such that there is no tree vertex above x of Type 2 or 1 with a path to x. Further, let y be the highest vertex of Type 1 with a tree-path from x (note that y could be equal to x); we then require that there is no vertex of Type 3 on the path from x to y. The transformations are defined as follows:*

T2. *Given edge (v, w), such that $v \in V_C$, $w \notin V_C$, and v is not an ancestor of x:*

- **(i)** *remove edge (v, w) and contract v;*
- **(ii)** *subdivide $(\mathsf{Pa}(x), x)$ with a new vertex z and add edge (z, w).*

T3. *Same as T2 with the difference that $v \notin V_C$, while $w \in V_C$, and w is not an ancestor of x (note that v could be an ancestor of x).*

▶ Remark 8. Note that T1, T2, and T3 could be seen as specifically constrained versions of the subnet prune and regraft operation (SNPR) defined and studied by Bordewich et al. [5].

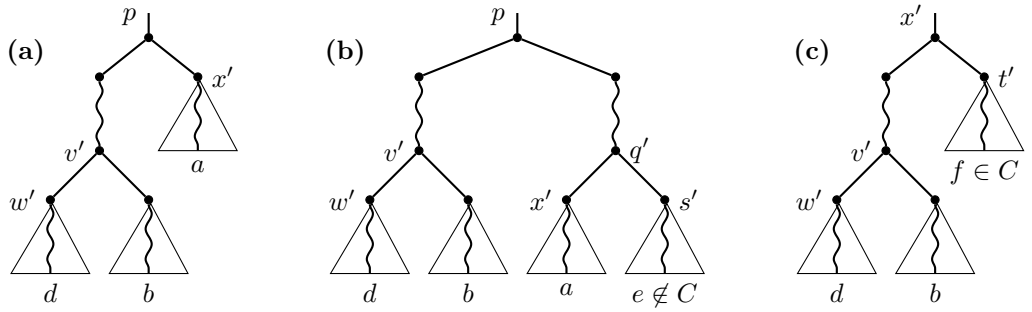


Figure 4 Three possible scenarios – (a), (b), and (c) – for a tree T displayed in N. Tree T' displayed in a transformed network N' (as shown in Figure 3) can be obtained in all three cases by regrafting the subtree rooted at w' just above vertex x'.

▶ **Lemma 9.** *T2 and T3 are C-compatible.*

Proof. The proofs for T2 and T3 are similar; therefore, we provide a proof for T2 only. Let N be transformed to N' as depicted in Figure 3.

Consider some T displayed in N. We need to show that exists T' displayed in N' that contains all clusters of T compatible with C. Similarly to the proof of Lemma 6, observe that if (v, w) is a a reticulation edge and there exists a way to display T in N without using this edge, then T is displayed in N' as well and the statement holds. We now fix a way to display T in N. Let x' then denote the vertex in T such that x (from N) corresponds to x'. Further, let $v' = \mathsf{lca}_T(b, d)$ be the vertex such that v corresponds to v' and let w' be the child of v' such that w corresponds to w'. Observe that a tree T' displayed in N' is obtained from T by (i) removing edge (v', w') and suppressing vertex v' and (ii) subdividing edge $(\mathsf{Pa}(x'), x')$ with a new vertex z' and adding an edge (z', v'). That is, subtree rooted at w' is regrafted above x'.

We now distinguish three cases for our proof depending on the location of x' relative to v' in T. Let p denote $\mathsf{lca}_T(v', x')$. Then the three cases are as follows (see also the illustration of these cases in Figure 4):

(a) x' is a child of p (see Figure 4(a)). Then T' does not have only those clusters of T that correspond to the intermediate vertices on the path from p to w'. However, each such cluster could not be compatible with C, since it contains $d \notin C$ and $b \in C$, but does not contain $a \in C$ (that is, it is neither a subset nor a superset of C, nor it is disjoint from C).

(b) There is at least one vertex on the path from p to x' (see Figure 4(b)). Let q' and s' denote the parent and sibling of x' respectively (as depicted on the figure). We argue that there must exist leaf $e \notin C$ below s'.

Consider a fixed subdivision $N(T)$ and let t be the lowest ancestor of x in $N(T)$ with two children (that is, t corresponds to t'). It is not difficult to observe that there could be at most one reticulation edge on the path from t to x in $N(T)$ and that edge has to originate from t (due to the tree-child property). Recall now that there is no vertex of Type 1 or 2 above x.

If there is a reticulation edge on the path from t to x, let s denote the child of t that is a tree vertex. Note that s could not belong to V_C, since in that case t would be a vertex of Type 2 (note that the other child of t, that is a reticulation, must have a tree-path to x and therefore a tree-path to $a \in C$ – that is, it belongs to V_C). Hence, there exists $e \notin C$ with a tree-path from s. Given that s corresponds to s' in T, leaf e must be present below s' as well.

Further, if there is no reticulation edge on the path from t to x, then s, defined as the child of t that is not on this path, is not in V_C as well. This is the case, since otherwise t would be a vertex of Type 1 or 2. Therefore, leaf e exists in that case as well.
We use these observations to finish the proof for (b). Note that tree T' does not have the clusters of T that correspond to the intermediate vertices on the paths from p to w' and from p to x'. However, using the argument similar to (a), it is not difficult to see that none of these clusters is compatible with C.

(c) $p = x'$ (see Figure 4(c)). Let t' be the child of x' that is not an ancestor of v'. To prove that the lemma holds in that case as well, we are going to show that there must exist leaf $f \in C$ below t'. Note that f could be equal to a.
Consider a fixed subdivision $N(T)$ and let u be the vertex in $N(T)$ that corresponds to x' and has two children. Observe that by our constraints u could be any vertex on the path from x to y (including x and y). Therefore, by the constraints in the definition of T2, u must be of Type 2 or 1 and hence both its children must be in V_C. Clearly, one of the children of u corresponds to t' and therefore such leaf f must exist.
Similarly to (a), tree T' does not contain only those clusters of T that correspond to the intermediate vertices on the path from x' to w'. It is then not difficult to see that these clusters are not compatible with C (via the same argument as in (a)).

These cases cover all possible scenarios, since by constraints given in Definition 7 v' cannot be an ancestor of x'. Note that for transformation T3 such a case (i.e., v' is an ancestor of x') is possible; however, using the same type of arguments as above, it is not difficult to check that for this case C-compatibility is preserved as well. ◀

▶ **Lemma 10.** *A not C-separated network N can be transformed into a C-separated network N' only using operations T1, T2, and T3.*

Proof. Consider the set of connected components (in the undirected sense) induced by the vertices in V_C. Let $A \in \mathcal{C}$ be such component; that is, all vertices in A are in V_C and every edge incident to A connects to a vertex that is not in V_C. We are going to potentially exclude some "redundant" vertices from A, while keeping it connected. Let v be a *maximal* vertex in A, i.e., its parent(s) is(are) not in A. If v has exactly one child that is in A, then exclude v from set A. Keep iterating this procedure until all maximal vertices in A are either leaves or have both children in A.

Now, let $\mathcal{C}'$ be the set of components obtained from $\mathcal{C}$ by performing the described above pruning procedure on each component. Note that after the pruning all maximal vertices in $\mathcal{C}'$ components are tree-vertices of Type 1 or 2.

Next, we are going to use transformations T1, T2, and T3 on N in order to aggregate all components in $\mathcal{C}'$ together and separate them from the rest of the network.

For each $A \in \mathcal{C}'$ let $I(A)$ be the set of vertices in A that are incident to an edge outside of A. Note that $I(A)$ includes all maximal vertices in A. Further, define $I(\mathcal{C}') := \cup_{A \in \mathcal{C}'} I(A)$. Since N is a DAG, there a vertex x in $I(\mathcal{C}')$, such that no other vertex in this set is an ancestor of x. Clearly, x must be a maximal vertex of some component in $\mathcal{C}'$.

Assume that x is a vertex of Type 2 (if x is of Type 1, then this step is not needed). Let y be the highest vertex of Type 1 with a tree path from x. Consider now all vertices on the tree path from x to y (excluding y); let us denote these vertices as $(v_1 = x, v_2, \ldots, v_k)$. Note that each v_i is either of Type 2 or 3. Transformation T1 then allows us to rearrange the types of these vertices in the way that there will be an index $1 \leq j \leq k$ such that for all $i \geq j$, v_i is a Type 2 vertex and for all $i < j$, v_i is a Type 3 vertex. That way we modified the component $C_x \in \mathcal{C}'$ that contains x. Since $v_1, \ldots, v_{j-1}$ (if $j > 1$) vertices have only one

child in V_C by the definition of Type 3 vertices, then applying the pruning procedure again will exclude $v_1, \ldots, v_{j-1}$ from C_x. Therefore, we re-define $x := v_j$. Observe that x is still a vertex with the property that no other vertex in $I(\mathcal{C}')$ is an ancestor of x. Further, there is no vertex of Type 3 between x and y.

Consider now all edges of type (v, w), such that v is in some component $A \in \mathcal{C}'$ and $w \notin V_C$. We perform transformation T2 (in any order) by reconnecting each such w above x. Further, consider all maximal vertices in $\mathcal{C}'$ except for x, we use T3 to reconnect those vertices above x (note that after each such operation T3 we need to re-assign $x = \mathsf{Pa}(x)$). It is not difficult to see that as the result of this procedure all vertices in V_C now lie in the same connected component. Finally, consider all reticulation vertices v in that component, such that one of the edges (w, v) is outside of the component. We perform T3 to relocate the source of such reticulation edges above x. ◀

The proof of Theorem 2 then follows from the above results. That is, Lemma 10 allows us to obtain a C-separated network N' with the property that for each S_j displayed in N exists S'_j displayed in N', such that S'_j has all the clusters of S_j compatible with C (guaranteed by Lemmas 6 and 9). Note now that all clusters in input trees T_j are compatible with C; hence, $RF(T_j, S'_j) \leq RF(T_j, S_j)$. Moreover, we assumed in the beginning that S_i does not contain C, while the respective tree S'_i displayed in N' must contain it. Therefore:

$$\mathsf{d}(\mathcal{T}, N') < \mathsf{d}(\mathcal{T}, N).$$

3.2 Consensus clusters in RF-networks

Theorem 2 says that for a tree-child network N with minimum $\mathsf{d}(\mathcal{T}, N)$ distance (for convenience, we refer to such networks as *minimum RF-networks*), each embedding of the input trees must contain *all* the consensus clusters. While this does not directly imply that each minimum RF-network is C-separated for each consensus cluster, it is not difficult to observe that at least one minimum RF-network has this property.

▶ **Theorem 11.** *For a collection of trees $\mathcal{T}$ there exists a minimum RF-network N such that N is C-separated for each consensus cluster C of $\mathcal{T}$.*

Proof. Take any minimum RF-network N. If N is not C-separated for some consensus cluster C, the transformations T1-T3 defined in the previous section can be used to bring N to the C-separated form (using the procedure from the proof of Lemma 10). As was demonstrated, T1-T3 do not increase the embedding distance for any $T \in \mathcal{T}$; thus the resulting network N' is also a minimum RF-network.

Further, it is not difficult to observe that if N is C_1-separated and the procedure from Lemma 10 is used to make it C_2-separated – where C_1 and C_2 are two distinct compatible clusters – the resulting network will remain C_1-separated. Hence, the theorem holds. ◀

While this theorem has important implications for practitioners on its own, we now show a much stronger result claiming that all minimum RF-networks are "almost" C-separated for each consensus cluster C. More precisely, we show that a minimum RF-network is not C-separated only if there is uncertainty induced by the set of input trees.

▶ **Definition 12.** *We say that two networks N_1 and N_2 are equivalent up to an ordering of reticulation events, if there is a tree-path $(u_1, \ldots, u_k)$ in N_1 such that each u_i has one reticulation child and N_2 can be obtained from N_1 by re-ordering the vertices on that path along with their outgoing reticulation edges. That is, N_2 can be obtained from N_1 by transformations similar to T1.*

▶ **Theorem 13.** *Let N be a tree-child network that displays a set of trees $\mathcal{Q}$ with the minimal number of reticulations (i.e., if at least one reticulation edge is removed from N, it will not display at least one of the trees from $\mathcal{Q}$). Then for a consensus cluster C of $\mathcal{Q}$ either N is C-separated or N is equivalent to a C-separated network N' up to an ordering of reticulation events, such that N' displays all trees from $\mathcal{Q}$ as well.*

The proof is omitted for brevity.

Using Theorem 2 we obtain the following corollary.

▶ **Corollary 14.** *Given an input tree-set $\mathcal{T}$, consider a minimum RF-network N for $\mathcal{T}$ that is not C-separated for some consensus cluster C. Let $\mathcal{Q}$ be the set of all embeddings for all trees $T \in \mathcal{T}$. Then there exists a C-separated minimum RF-network N' that displays each tree in $\mathcal{Q}$ and N' is equivalent to N up to an ordering of reticulation events.*

This corollary implies that a minimum RF-network is not C-separated *only if there is uncertainty in the ordering of some reticulation events that is not resolved by the embeddings of the input trees.*

4 Conclusion

We make the first step towards boosting the scalability of methods for the inference of hybridization and reassortment networks. We establish a Pareto-like property for phylogenetic networks and prove that the recently introduced *RF embedding cost* satisfies it for tree-child networks. This result allows one to use strict consensus merger strategies to significantly magnify the scalability of the RF-Net method, particularly, when input trees are relatively similar. Our results arise from studying the structure of optimum tree-child RF-Networks in relation to the strict consensus clusters from the input trees. We anticipate that future work would shed light on whether the Pareto properties that we discovered in this work could be generalized to other classes of networks, e.g., reticulation-visible networks.

References

1 Mukul S. Bansal, J. Gordon Burleigh, Oliver Eulenstein, and David Fernández-Baca. Robinson-Foulds supertrees. *Algorithms Mol Biol*, 5:18, 2010.

2 Mihaela Baroni, Charles Semple, and Mike Steel. A framework for representing reticulate evolution. *Annals of Combinatorics*, 8(4):391–408, 2005.

3 Olaf R.P. Bininda-Emonds, editor. *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, volume 4 of *Computational Biology*. Springer Verlag, 2004.

4 Olaf R.P. Bininda-Emonds, John L. Gittleman, and Mike A. Steel. The (Super)tree of life: Procedures, problems, and prospects. *The (Super)tree of life: Procedures, problems, and prospects*, 33:265–289, 2002.

5 Magnus Bordewich, Simone Linz, and Charles Semple. Lost in space? Generalising subtree prune and regraft to spaces of phylogenetic networks. *J Theor Biol*, 423:1–12, 2017.

6 David Bryant. A classification of consensus methods for phylogenetics. In M. Janowitz, F.-J. Lapointe, F. McMorris, B. Mirkin, and F. Roberts, editors, *Discrete Mathematics and Theoretical Computer Science*, volume 61, pages 163–185. Am Math Soc, Providence, RI, 2003.

7 Gabriel Cardona, Francesc Rossello, and Gabriel Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM TCBB*, 6(4):552–569, 2009.

8 Wen-Chieh Chang, Paweł Górecki, and Oliver Eulenstein. Exact solutions for species tree inference from discordant gene trees. *J Bioinform Comput Biol*, 11(5):1342005, 2013.

9 Cedric Chauve, Nadia El-Mabrouk, Laurent Guéguen, Magali Semeria, and Eric Tannier. *Duplication, Rearrangement and Reconciliation: A Follow-Up 13 Years Later*, pages 47–62. Springer, London, 2013.

10 Theodosius Dobzhansky. Nothing in biology makes sense except in the light of evolution. *The american biology teacher*, 75(2):87–91, 2013.

11 Juliane C Dohm, André E Minoche, Daniela Holtgräwe, Salvador Capella-Gutiérrez, Falk Zakrzewski, Hakim Tafer, Oliver Rupp, Thomas Rosleff Sörensen, Ralf Stracke, Richard Reinhardt, et al. The genome of the recently domesticated crop plant sugar beet (Beta vulgaris). *Nature*, 505(7484):546, 2014.

12 Oliver Eulenstein, Snehalata Huzurbazar, and David A Liberles. *Evolution after Gene Duplication*, chapter Reconciling Phylogenetic Trees, pages 185–206. John Wiley, 2010.

13 Daniel H Huson, Scott M Nettles, and Tandy J Warnow. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. *J Comput Biol*, 6(3-4):369–386, 1999.

14 Daniel H Huson, Regula Rupp, and Celine Scornavacca. *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press, 2010.

15 Harris T. Lin, J. Gordon Burleigh, and Oliver Eulenstein. Consensus properties for the deep coalescence problem and their application for scalable tree search. *BMC Bioinf*, 13 Suppl 10:S12, 2012.

16 Bin Ma, Ming Li, and Louxin Zhang. From Gene Trees to Species Trees. *SIAM J. Comput.*, 30(3):729–752, 2000. doi:10.1137/S0097539798343362.

17 Alexey Markin, Tavis K Anderson, Venkata SKT Vadali, and Oliver Eulenstein. Robinson-Foulds Reticulation Networks. *bioRxiv*, 2019. doi:10.1101/642793.

18 Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. *Microeconomic theory*, volume 1. Oxford university press New York, 1995.

19 Siavash Mirarab, Rezwana Reaz, Md S Bayzid, Théo Zimmermann, M Shel Swenson, and Tandy Warnow. ASTRAL: genome-scale coalescent-based species tree estimation. *Bioinformatics*, 30(17):541–548, 2014.

20 Jucheol Moon and Oliver Eulenstein. Synthesizing large-scale species trees using the strict consensus approach. *J Bioinform Comput Biol*, 15(03), 2017.

21 Jucheol Moon, Harris T Lin, and Oliver Eulenstein. Consensus properties and their large-scale applications for the gene duplication problem. *J Bioinform Comput Biol*, 14(03), 2016.

22 Thomas J Near, Ron I Eytan, Alex Dornburg, Kristen L Kuhn, Jon A Moore, Matthew P Davis, Peter C Wainwright, Matt Friedman, and W Leo Smith. Resolution of ray-finned fish phylogeny and timing of diversification. *PNAS*, 109(34):13698–13703, 2012.

23 DA Neumann. Faithful consensus methods for n-trees. *Math Biosci*, 63(2):271–287, 1983.

24 David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Math Biosci*, 53:131–147, 1981.

25 Mike Steel and Allen Rodrigo. Maximum likelihood supertrees. *Syst Biol*, 57(2):243–250, April 2008.

26 M Shel Swenson, Rahul Suri, C Randal Linder, and Tandy Warnow. SuperFine: fast and accurate supertree estimation. *Syst Biol*, 61(2):214, 2011.

27 Pranjal Vachaspati and Tandy Warnow. FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization. *Bioinformatics*, 33(5):631–639, 2016.

28 André Wehe, Mukul S Bansal, J Gordon Burleigh, and Oliver Eulenstein. DupTree: a program for large-scale phylogenetic analyses using gene tree parsimony. *Bioinformatics*, 24(13):1540–1541, 2008.

29 Yun Yu, R. Matthew Barnett, and Luay Nakhleh. Parsimonious Inference of Hybridization in the Presence of Incomplete Lineage Sorting. *Syst Biol*, 62(5):738–751, 2013.