

Cherry-Picking Distance Between Binary Orchards Parameterized by Level

Manuel Lafond¹, Kaari Landry², Olivier Tremblay-Savard³, Hamza Wahed⁴,
Christopher Whidden⁴, and Norbert Zeh⁴

¹ Université de Sherbrooke, Canada

² Université de Montréal, Canada

³ University of Manitoba, Canada

⁴ Dalhousie University, Canada

manuel.lafond@USherbrooke.ca,

kaari.landry@umontreal.ca,

olivier.tremblay-savard@umanitoba.ca,

{hm682264,cwhidden,nzeh}@dal.ca

Abstract. Phylogenetic networks capture complex evolutionary relationships, including hybridization and gene flow, that are not easily represented by tree-based models. Orchards are a class of phylogenetic networks that have proven valuable for reconstructing evolution from ancestral profiles, and in which reticulation arcs model lateral gene transfers. Several methods exist to infer orchards, and distance measures between networks can be used to evaluate them against simulations or gold-standard datasets. One such measure is the maximum agreement cherry-reduced subnetwork (MACRS) of two orchards. Recently, Landry et al. proved that finding such a MACRS of two binary level-1 orchards is fixed-parameter tractable (FPT) with respect to the number of reticulations in both networks. In this paper, we show that this problem can be solved on arbitrary binary orchards, in FPT time with respect to the maximum level of the two networks. In particular, a MACRS of two binary level-1 orchards can be found in polynomial time.

Keywords: Orchards · Level-k networks · Fixed-parameter tractability · Cherry covers · Network distances

1 Introduction

Phylogenetic trees are the traditional model for representing the evolutionary history of a set of taxa by depicting speciation events. However, there are other events that can also drive evolution, such as hybridization, lateral gene transfer, and recombination. For instance, hybridization can produce new species like Triticale, a wheat-rye hybrid [9], while recombination can generate new viral strains through genetic reassortment, as seen with influenza viruses mixing RNA segments in co-infected cells [2]. To represent such non-tree-like events in the evolution of a set of taxa, phylogenetic networks have been introduced, which can include vertices with more than one parent, called *reticulations*.

A cherry in a network is a pair of leaves that either have a common parent (a *proper cherry*) or whose parents are connected by a reticulation arc (a *reticulated cherry*). A *cherry-picking operation* removes one cherry from the network, perhaps exposing others. A phylogenetic network reducible to a single leaf by a sequence of such operations is called an orchard. This network class is mathematically attractive, for example, because it takes linear time to decide whether a network is an orchard [6] or whether two orchards are isomorphic [14].

Orchards are also biologically relevant, as they were initially introduced because they can be uniquely reconstructed from their ancestral profiles (see [6]). They can also model horizontal gene transfer, and have in fact been characterized as “trees with additional horizontal arcs” [10]. There are thus several algorithms and software that construct orchards. For instance, methods that add time-consistent transfers to a species tree based on reconciled gene trees [13,1], or based on the presence/absence of character traits [22,21], result in an orchard.

Evaluating the accuracy of such methods typically consists of simulating networks and comparing them against the constructed networks, but there are no established metrics to compare networks, unlike for trees, which have the RF metric [23,20]. There has thus been research on designing (dis)similarity measures between orchards specifically, one example being the extended μ -distance from [5]. In this paper, we focus on another measure, the *maximum agreement cherry-reduced subnetwork* (MACRS), which is the largest common subnetwork that can be obtained from both networks through cherry-picking operations [17]. The distance itself is the quantity of such operations, of which there is a correspondence with the size of the MACRS (Theorem 4 of [17]).

Landry et al. [17] proved that finding the MACRS of two orchards is NP-hard, which raises the question whether it is fixed-parameter tractable (FPT) with respect to common parameters, i.e., whether it can be solved in a running time that depends exponentially only on that parameter. A popular network parameter is the *level*, defined as the maximum number of reticulations in any 2-edge-connected component (2ECC) of the network. The level can be expected to be small in practice [7], and many NP-hard problems are FPT in the level (see for instance [16,25]). Other parameters of interest include the treewidth [12,24], scanwidth [3,4,15], or just the total number of reticulations.

In this paper, we show that the MACRS problem on binary orchards is FPT with respect to level. This solves an open problem by Landry et al. [19], who proved that finding a MACRS of two binary level-1 orchards is FPT with respect to the number of reticulations in the networks. This algorithm was shown to be empirically efficient [18], but practical use is currently limited to level-1 networks.

In comparison, our algorithm applies to *arbitrary* binary orchards and is exponential only in the level ℓ , which is upper-bounded by r and is often much smaller. In particular, our result shows that a MACRS of two level-1 orchards can be found in polynomial time. The algorithm by Landry et al. guessed the reticulated cherries globally, for the entire network, which resulted in a running time of $O(3^r \cdot \text{poly}(n))$. Our contributions can be summarized as follows:

- We show that the reticulated cherries to pick can be guessed one 2ECC at

a time, which combined with dynamic programming leads to 3^ℓ guesses per 2ECC instead of 3^r for the whole network.

- For a pair of guesses on two corresponding 2ECCs in the two networks, we must also guess an isomorphism between them to know how vertices are “matched” in an agreement subnetwork. A naive analysis would result in $O(\ell!)$ such isomorphisms, but we show that there are at most $2^{3\ell/2}$ isomorphisms between two 2ECCs of arbitrary networks. This result may be of independent interest for the computation of other (dis)similarity measures.
- To prove the correctness of our algorithm, we generalize the notion of a cherry cover of an orchard [11] to define *partial cherry covers*, which allow only part of the network to be covered with cherry paths. We prove important properties of these partial cherry covers that may be of independent interest.

By combining these techniques, we obtain an $O(14.5^\ell \cdot n^3)$ -time algorithm, as developed in Theorem 6 and improved in Appendix A.

2 Preliminaries

In this section, we introduce the necessary terminology and notation regarding networks, cherry picking sequences, and cherry partitions, including preliminary results. Due to space limitations, all proofs can be found in the appendix.

Phylogenetic Networks. A *phylogenetic network* \mathcal{N} (*network* for short) is a directed acyclic graph with a single vertex of in-degree 0, called the *root*, whose out-degree-0 vertices, called *leaves*, have in-degree 1, and whose non-root, non-leaf vertices either have in-degree 1 and out-degree at least 2 (*tree vertices*) or out-degree 1 and in-degree at least 2 (*reticulations*). In the literature, the leaves of a network are labelled bijectively with the elements of some label set X , generally viewed to represent extant taxa. For reasons that will become clear shortly, leaves are labelled with *disjoint non-empty subsets* of X in this paper, as was already done in [17,19]. For a leaf v , we use $X(v)$ to denote its label set.

A network is *binary* if all vertices have in-degree at most 2 and out-degree at most 2. This paper is concerned only with binary networks.

We denote the sets of vertices, arcs, and leaves of a network \mathcal{N} by $V(\mathcal{N})$, $E(\mathcal{N})$, and $\mathcal{L}(\mathcal{N})$, respectively, and extend this notation in the obvious way to subgraphs, sets of arcs, paths, and sets of paths in a network. A vertex v is a *child*, *parent*, *descendant* or *ancestor* of another vertex u in a network \mathcal{N} if (u, v) is an arc of \mathcal{N} , (v, u) is an arc of \mathcal{N} , there exists a directed path from u to v in \mathcal{N} or there exists a directed path from v to u in \mathcal{N} , respectively. When v is a non-root leaf or tree vertex, then its parent is unique, and we refer to it as $p(v)$. We extend the definition of $X(v)$ to whole networks and internal vertices of networks by defining $X(v)$ to be the union of the label sets of all descendant leaves of v , and $X(\mathcal{N}) = X(\rho)$, where ρ is the root of \mathcal{N} . We call an arc $(u, v) \in E(\mathcal{N})$ a *tree arc* if v is a tree vertex or leaf, and a *reticulation arc* if v is a reticulation.

An undirected graph is *2-edge-connected* if it cannot be disconnected by removing a single edge. A *2-edge connected component* (2ECC) of an undirected graph G is a maximal 2-edge-connected subgraph. An edge of G is a *bridge* if its endpoints belong to different 2ECCs of G . The 2ECCs of a network \mathcal{N} are

the 2ECCs of its underlying undirected graph, which is the graph obtained from \mathcal{N} by forgetting the arc directions. We call a 2ECC *trivial* if it consists of a single vertex, and *non-trivial* otherwise. Note that every 2ECC in a network has a single root [8]. For any such root v of a 2ECC, let $\mathcal{N}(v)$ be the subnetwork of \mathcal{N} induced by all descendants of v . (If v is not a root of a 2ECC, then the subgraph induced by v 's descendants is not a network, as it contains vertices of in-degree 1 and out-degree 1.) The *level* of a network \mathcal{N} is the maximum number of reticulations in its 2ECCs. The top-left network in Fig. 2.1 is level-2.

In this paper, a *blob* refers to a subgraph of a network \mathcal{N} consisting of a 2ECC of \mathcal{N} and all its pendant bridges (excluding the bridge formed by the root of the 2ECC and its parent). So, blobs are edge disjoint but not necessarily vertex disjoint. A blob can be viewed as a network in which each endpoint v of a pendant bridge is labelled by $X(v)$. A blob is trivial or non-trivial if the 2ECC from which it is constructed is trivial or non-trivial. Note that a leaf of \mathcal{N} forms a trivial 2ECC, which has no pendant bridges. Each such 2ECC still defines a blob, which is identical to its 2ECC. The blobs of \mathcal{N} form a partition of the arcs of \mathcal{N} . Every non-leaf vertex of a blob has all its child arcs in the blob, and every non-root vertex has all its parent arcs in the blob.

An isomorphism $\phi = (\phi_V, \phi_E)$ between two directed graphs G_1 and G_2 consists of two set isomorphisms $\phi_V : V(G_1) \rightarrow V(G_2)$ and $\phi_E : E(G_1) \rightarrow E(G_2)$ such that $\phi_E((u, v)) = (\phi_V(u), \phi_V(v))$ for every arc $(u, v) \in E(G_1)$. In the remainder of this paper, we refer to both ϕ_V and ϕ_E simply as ϕ if no confusion can arise. An isomorphism between two networks \mathcal{N}_1 and \mathcal{N}_2 is a graph isomorphism that additionally respects leaf labels. Since leaves of networks are labelled with subsets of X in this paper, we require a network isomorphism to satisfy the weaker condition that $X(v) \cap X(\phi(v)) \neq \emptyset$ for every leaf of \mathcal{N}_1 . Landry et al. [19] called this a *weak isomorphism* and showed that using it in the definition of a MACRS, to be recalled shortly, captures exactly the idea that a MACRS can be obtained from two networks whose elements are labelled with individual elements of X by applying sequences of cherry picking operations to them.

Cherry-Picking Sequences, Orchards, and (Agreement) Cherry-Reduced Subnetworks. A *proper cherry* in a network \mathcal{N} is a pair of leaves (a, b) with a common parent. A *reticulated cherry* is a pair of leaves (a, b) such that $(p(b), p(a))$ is a reticulation arc in \mathcal{N} . We say *cherry* to refer to a proper or reticulated cherry. Picking a proper cherry (a, b) of \mathcal{N} consists of removing the leaves a and b along with their parent arcs. This makes their common parent a leaf, which is assigned the label set $X(a) \cup X(b)$. Picking a reticulated cherry (a, b) of \mathcal{N} consists of removing the leaves a and b along with their parent arcs and the arc $(p(b), p(a))$. This makes $p(a)$ and $p(b)$ leaves which we assign the label sets $X(a)$ and $X(b)$, respectively.

We call an ordered sequence of cherries $S = \langle (a_1, b_1), \dots, (a_n, b_n) \rangle$ a *cherry-picking sequence* (CPS) of a network \mathcal{N} if there exists a sequence of networks $\mathcal{N}_0, \mathcal{N}_1, \dots, \mathcal{N}_n$ with the following properties:

- $\mathcal{N}_0 = \mathcal{N}$
- (a_i, b_i) is a cherry of \mathcal{N}_{i-1} , for all $i \in [n]$.

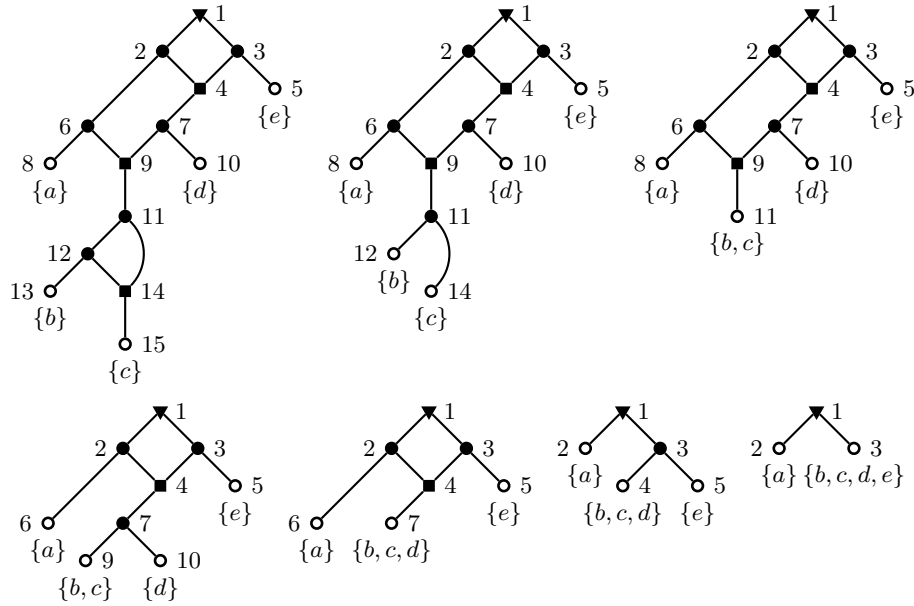


Fig. 2.1. An illustration of a CPS operating on a level-2 orchard. Leaves are shown as open circles; non-root tree vertices, as filled circles; reticulations, as filled squares; and the root, as a filled, inverted triangle. The sequence of networks produced from the top-left level-2 network by the CPS $\langle (15, 13), (12, 14), (11, 8), (9, 10), (7, 6), (4, 5) \rangle$. Picking the final cherry $(2, 3)$ will result in a network consisting of a single vertex 1 labelled by $\{a, b, c, d, e\}$. Thus, the top-left network is orchard.

– Picking (a_i, b_i) in \mathcal{N}_{i-1} produces \mathcal{N}_i , for all $i \in [n]$.

In this case, we refer to the final network \mathcal{N}_n in this sequence as $\mathcal{N} \wedge S$. We call a CPS *complete* if \mathcal{N}_n has a single leaf, and *partial* otherwise. A network is an *orchard* if it has a complete CPS, an example of which is shown in Figure 2.1.

A network \mathcal{N}' is a *cherry-reduced subnetwork* (CRS) of another network \mathcal{N} if there exists a CPS S of \mathcal{N} such that there is a network isomorphism between \mathcal{N}' and $\mathcal{N} \wedge S$. A network is an *agreement cherry reduced subnetwork* (ACRS) of two networks \mathcal{N}_1 and \mathcal{N}_2 if it is a CRS of both \mathcal{N}_1 and \mathcal{N}_2 . It is a *maximum agreement cherry-reduced subnetwork* (MACRS) of \mathcal{N}_1 and \mathcal{N}_2 if it has the maximum number of arcs among all ACRSs of $(\mathcal{N}_1, \mathcal{N}_2)$.

MAXIMUM AGREEMENT CHERRY-REDUCED SUBNETWORK PROBLEM

Input: Two networks \mathcal{N}_1 and \mathcal{N}_2

Find: A MACRS of \mathcal{N}_1 and \mathcal{N}_2

Cherry Partitions. For a network \mathcal{N} , we define two collections of *cherry paths* $C(\mathcal{N})$ and $R(\mathcal{N})$. A *proper cherry path* $\langle u, v, w \rangle$ consists of two arcs connecting two vertices u and w to their common parent v . In particular v is a tree vertex. A *reticulated cherry path* $\langle u, v, w, z \rangle$ consists of two arcs connecting two vertices v and w to their children u and z , respectively, and an arc from w to v . In partic-

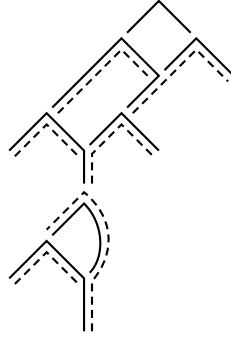


Fig. 2.2. The complete collection of cherry paths in the top-left network in Fig. 2.1 represented by both the solid and dashed paths. The solid paths form an acyclic cherry partition corresponding to the complete CPS described in Fig. 2.1.

ular, v is a reticulation, w is a tree vertex, and (w, v) is a reticulation arc. $C(\mathcal{N})$ is the collection of all cherry paths in \mathcal{N} , and $R(\mathcal{N}) \subseteq C(\mathcal{N})$ is the collection of all reticulated cherry paths in \mathcal{N} . The definition of $C(\mathcal{N})$ is illustrated in Fig. 2.2.

A complete CPS $\langle (a_1, b_1), \dots, (a_n, b_n) \rangle$ of a binary orchard \mathcal{N} defines a partition $\mathcal{P} = \{P_1, \dots, P_n\} \subseteq C(\mathcal{N})$ of the arcs of \mathcal{N} into arc-disjoint paths, where each path P_i consists of the arcs removed when picking the cherry (a_i, b_i) . As illustrated in Fig. 2.2, P_1, \dots, P_n are cherry paths. If (a_i, b_i) is a proper cherry of \mathcal{N}_{i-1} , then P_i is a proper cherry path; if (a_i, b_i) is a reticulated cherry of \mathcal{N}_{i-1} , then P_i is a reticulated cherry path. We call \mathcal{P} a *complete cherry partition* of \mathcal{N} . We can define a graph $G_{\mathcal{P}}$ whose vertices are the paths in \mathcal{P} and with an arc (P_i, P_j) if one of the endpoints of P_i is an internal vertex of P_j . Van Iersel et al. [11] proved that a network is an orchard if and only if $G_{\mathcal{P}}$ is acyclic. In this case, we call \mathcal{P} an *acyclic complete cherry partition*. In particular, Van Iersel et al. proved that a complete CPS of \mathcal{N} can be built from an acyclic complete cherry partition by picking cherries according to a topological ordering of $G_{\mathcal{P}}$. They also proved that orchards can be recognized in linear time by picking cherries greedily, i.e., the order in which cherries are picked is unimportant.

This notion extends naturally to partial cherry picking sequences: Given a partial cherry picking sequence $\langle (a_1, b_1), \dots, (a_n, b_n) \rangle$ of a network \mathcal{N} , we can collect the associated cherry paths as for a complete cherry picking sequence to obtain a *partial cherry partition* $\mathcal{P} = \{P_1, \dots, P_n\}$. This partition consists of arc-disjoint cherry paths such that each endpoint of a path P_i in \mathcal{P} is either a leaf of \mathcal{N} or is an internal vertex of another path $P_j \in \mathcal{P}$. Just as for complete cherry partitions, we can define the graph $G_{\mathcal{P}}$ and say that \mathcal{P} is acyclic if and only if $G_{\mathcal{P}}$ is acyclic. For a network \mathcal{N} and a partial cherry partition \mathcal{P} of \mathcal{N} , let $\mathcal{N} - \mathcal{P}$ be the network obtained from \mathcal{N} by deleting all arcs that belong to the paths in \mathcal{P} , as well as all isolated vertices this creates. For every leaf v of $\mathcal{N} - \mathcal{P}$, we define $X(v)$ as the union of the label sets of all leaves of \mathcal{N} reachable from v via directed paths that do not contain any of the internal arcs of reticulated cherry paths in \mathcal{P} . By the next proposition, finding cherry reduced subnetworks

of \mathcal{N} amounts to finding appropriate partial cherry partitions of \mathcal{N} .

Proposition 1. *Given two binary networks \mathcal{N} and \mathcal{N}' , there exists a cherry picking sequence S with $\mathcal{N} \wedge S = \mathcal{N}'$ if and only if there exists an acyclic partial cherry partition \mathcal{P} of \mathcal{N} such that $\mathcal{N} - \mathcal{P} = \mathcal{N}'$.*

The following lemma shows that the blobs of every orchard \mathcal{N} are orchards and that, therefore, every acyclic cherry partition of \mathcal{N} is the union of acyclic cherry partitions of its blobs.

Lemma 2. *Let \mathcal{N} be a network with blobs B_1, \dots, B_t . Then*

- (i) $C(\mathcal{N}) = C(B_1) \cup \dots \cup C(B_t)$ and $C(B_1), \dots, C(B_t)$ are pairwise disjoint.
- (ii) For any subset $\mathcal{P} \subseteq C(\mathcal{N})$, let $\mathcal{P}_i = \mathcal{P} \cap C(B_i)$, for all $i \in [t]$. Then \mathcal{P} is an acyclic cherry partition of \mathcal{N} if and only if each \mathcal{P}_i is an acyclic cherry partition of B_i and, for each blob B_i whose root is part of some cherry path in \mathcal{P} , \mathcal{P}_i is a complete cherry partition of B_i .
- (iii) \mathcal{N} is an orchard if and only if B_1, \dots, B_t are orchards.
- (iv) If \mathcal{P} is an acyclic cherry partition of \mathcal{N} , $\mathcal{N}' := \mathcal{N} - \mathcal{P}$, x is the root of a blob of \mathcal{N} , and $\mathcal{P}' := \mathcal{P} \cap C(\mathcal{N}(x))$, then \mathcal{P}' is an acyclic cherry partition of $\mathcal{N}(x)$. If $x \in \mathcal{N}'$, then $\mathcal{N}'(x) = \mathcal{N}(x) - \mathcal{P}'$; if $x \notin \mathcal{N}'$, then \mathcal{P}' is a complete cherry partition of $\mathcal{N}(x)$, so $\mathcal{N}(x) - \mathcal{P}'$ has x as its only vertex.

3 MACRS Algorithm on Level- k Orchards

In this section, we present our MACRS algorithm for level- k orchards. First, we introduce some necessary background and intuition. This is followed by the algorithm and a detailed description. Finally, we analyze the running time of the algorithm. The correctness proof is deferred to Section 4, which develops the recurrence relations used by the algorithm.

Background. A *reticulation selector* is a function $\sigma : R \rightarrow \{0, 1, \perp\}$, for some $R \supseteq R(\mathcal{N})$. We also use \perp to denote the reticulation selector that maps every cherry path in R to \perp . A reticulation selector σ defines a family $\mathcal{R}_\sigma(\mathcal{N})$ of all subsets $R' \subseteq R(\mathcal{N})$ such that $\sigma^{-1}(1) \cap R(\mathcal{N}) \subseteq R'$ and $\sigma^{-1}(0) \cap R' = \emptyset$. We say that a cherry partition \mathcal{P} is σ -consistent if $\mathcal{P} \cap R(\mathcal{N}) \in \mathcal{R}_\sigma(\mathcal{N})$. In words, a reticulation selector σ specifies reticulated cherry paths that must ($\sigma(P) = 1$) or must not ($\sigma(P) = 0$) be contained in any σ -consistent cherry partition, and may leave the choice for other paths unspecified ($\sigma(P) = \perp$). A σ -CRS of \mathcal{N} is a CRS $\mathcal{N} - \mathcal{P}$, where \mathcal{P} is σ -consistent. Given two networks \mathcal{N}_1 and \mathcal{N}_2 and two reticulation selectors σ_1 and σ_2 in \mathcal{N}_1 and \mathcal{N}_2 , respectively, a (σ_1, σ_2) -ACRS of $(\mathcal{N}_1, \mathcal{N}_2)$ is an ACRS \mathcal{N}^* of $(\mathcal{N}_1, \mathcal{N}_2)$ that is a σ_1 -CRS of \mathcal{N}_1 and a σ_2 -CRS of \mathcal{N}_2 . It is possible that no such ACRS exists. If it does, we define a (σ_1, σ_2) -MACRS to be a (σ_1, σ_2) -ACRS of maximum size (i.e., maximum number of vertices).

Clearly, every partial cherry partition of \mathcal{N} is \perp -consistent, so every CRS of \mathcal{N} is a \perp -CRS of \mathcal{N} , and every ACRS of a pair of networks $(\mathcal{N}_1, \mathcal{N}_2)$ is also a (\perp, \perp) -ACRS. Thus, our algorithm needs to search for a (\perp, \perp) -MACRS of $(\mathcal{N}_1, \mathcal{N}_2)$. As the search progresses, it will use reticulation selectors that fix more

and more cherry paths to guide the search for such a (\perp, \perp) -MACRS of $(\mathcal{N}_1, \mathcal{N}_2)$.

We say that a reticulation selector σ *fixes* a path $P \in R(\mathcal{N})$ if $\sigma(P) \in \{0, 1\}$. We say that σ is *blob-respecting* if, for every blob B of \mathcal{N} , σ fixes either all or none of the paths in $R(B)$. We use \mathcal{N}^σ to denote the σ -CRS such that *every* σ -CRS of \mathcal{N} is also a σ -CRS of \mathcal{N}^σ . Intuitively, every σ -CRS of \mathcal{N} can be obtained by first constructing \mathcal{N}^σ and then picking additional cherries. We show that if σ is blob-respecting and any σ -CRS exists at all, then \mathcal{N}^σ exists and is unique.

Lemma 3. *Let σ be a blob-respecting reticulation selector of a network \mathcal{N} . If there exists a σ -CRS of \mathcal{N} , then there exists a σ -CRS \mathcal{N}^σ such that every σ -CRS of \mathcal{N} is also a σ -CRS of \mathcal{N}^σ . This immediately implies that \mathcal{N}^σ is unique. If all reticulated cherry paths fixed by σ belong to the same blob, then one can decide in linear time whether \mathcal{N}^σ exists and, if so, construct it.*

We need notation to refer to the blob that contains each vertex v of a network \mathcal{N} . We use B_v to denote the unique blob of \mathcal{N} such that v belongs to the underlying 2ECC of B_v . Analogously, for a reticulation selector σ such that \mathcal{N}^σ exists and $v \in \mathcal{N}^\sigma$, we use B_v^σ to denote the unique blob of \mathcal{N}^σ such that v belongs to the underlying 2ECC of B_v^σ . Note that $B_v^\sigma \subseteq B_v$ in this case. For a blob B of \mathcal{N} , we call a reticulation selector σ a *B-reticulation selector* if σ fixes no cherry path in $R(B')$, for any proper descendant blob B' of B , and it fixes all or none of the cherry paths in $R(B)$.

Algorithm. Our algorithm for finding a MACRS of a pair of networks $(\mathcal{N}_1, \mathcal{N}_2)$ is shown in Algorithm 1. Given two vertices $u \in \mathcal{N}_1$ and $v \in \mathcal{N}_2$, a B_u -reticulation selector σ_1 , and a B_v -reticulation selector σ_2 such that u and v are roots of blobs in $\mathcal{N}_1^{\sigma_1}$ and $\mathcal{N}_2^{\sigma_2}$, respectively, the algorithm returns the size of a (σ_1, σ_2) -MACRS of $(\mathcal{N}_1^{\sigma_1}(u), \mathcal{N}_2^{\sigma_2}(v))$. Thus, when called with $u = \rho_1$, $v = \rho_2$, and $\sigma_1 = \sigma_2 = \perp$, it returns the size of a MACRS of $(\mathcal{N}_1, \mathcal{N}_2)$. An actual MACRS can be recovered using a standard backtracking approach. We defer a formal correctness proof of the algorithm to Section 4. The following gives its intuition.

The algorithm uses memoization by storing the result of each invocation in a table entry $D[u, v, \sigma_1, \sigma_2]$. If it is called with the same argument again, then line 1 of the algorithm simply returns the result computed before.

If σ_1 fixes none of the cherry paths in $R(B_u)$ or σ_2 fixes none of the cherry paths in $R(B_v)$, then it can be shown that a (σ_1, σ_2) -MACRS of $(\mathcal{N}_1^{\sigma_1}(u), \mathcal{N}_2^{\sigma_2}(v))$ is a (σ'_1, σ'_2) -MACRS of $(\mathcal{N}_1^{\sigma_1}(u), \mathcal{N}_2^{\sigma_2}(v))$, for two reticulation selectors σ'_1 and σ'_2 obtained from σ_1 and σ_2 by fixing all those paths in $R(B_u)$ and $R(B_v)$ appropriately that are not already fixed by σ_1 and σ_2 . Thus, lines 2–7 consider all such “extensions” σ'_1 and σ'_2 of σ_1 and σ_2 and compute $D[u, v, \sigma_1, \sigma_2]$ as the maximum of all corresponding table entries $D[u, v, \sigma'_1, \sigma'_2]$ (returned by the recursive calls σ -MACRS($u, v, \sigma'_1, \sigma'_2$)). The $\text{fix}(\sigma, B)$ subroutine (not shown) returns all reticulation selectors that can be obtained from σ by fixing the reticulation paths in B not already fixed by σ .

If σ_1 and σ_2 fix all cherry paths in $R(B_u)$ and $R(B_v)$, then a (σ_1, σ_2) -MACRS \mathcal{N}^* of $(\mathcal{N}_1^{\sigma_1}(u), \mathcal{N}_2^{\sigma_2}(v))$ exists only if $\mathcal{N}_1^{\sigma_1}$ and $\mathcal{N}_2^{\sigma_2}$ exist. $\text{reduce}(\mathcal{N}, \sigma)$ returns the network \mathcal{N}^σ if it exists, or null otherwise. Lines 9–11 use this procedure to

Algorithm 1 σ -MACRS**Data:** $u \in \mathcal{N}_1, v \in \mathcal{N}_2, B_u$ - and B_v -reticulation selectors σ_1, σ_2 **Results:** (σ_1, σ_2) -MACRS

```

1: if  $D[u, v, \sigma_1, \sigma_2]$  is defined then return  $D[u, v, \sigma_1, \sigma_2]$ 
2: if  $\sigma_1 = \perp$  or  $\sigma_2 = \perp$  then
3:    $M \leftarrow -\infty$ 
4:   for all  $\sigma'_1$  in  $\text{fix}(\sigma_1, B_u)$  do
5:     for all  $\sigma'_2$  in  $\text{fix}(\sigma_2, B_v)$  do
6:        $M \leftarrow \max(M, \sigma\text{-MACRS}(u, v, \sigma'_1, \sigma'_2))$ 
7:    $D[u, v, \sigma_1, \sigma_2] \leftarrow M$ 
8: else
9:    $\mathcal{N}_1^{\sigma_1} \leftarrow \text{reduce}(\mathcal{N}_1, \sigma_1); \mathcal{N}_2^{\sigma_2} \leftarrow \text{reduce}(\mathcal{N}_2, \sigma_2)$ 
10:  if  $\mathcal{N}_1^{\sigma_1}$  is null or  $\mathcal{N}_2^{\sigma_2}$  is null then
11:     $D[u, v, \sigma_1, \sigma_2] = -\infty$ ; return  $D[u, v, \sigma_1, \sigma_2]$ 
12:  Let  $B_u^{\sigma_1}, B_v^{\sigma_2}$  be the blobs containing  $u$  and  $v$  in  $\mathcal{N}_1^{\sigma_1}$  and  $\mathcal{N}_2^{\sigma_2}$ 
13:  if subnets-are-reducible-and-overlap $(u, v, \sigma_1, \sigma_2)$  then
14:     $D_1[u, v, \sigma_1, \sigma_2] \leftarrow 1$ 
15:  else
16:     $D_1[u, v, \sigma_1, \sigma_2] \leftarrow -\infty$ 
17:   $\Phi \leftarrow \text{generate-isomorphisms}(B_u^{\sigma_1}, B_v^{\sigma_2})$ 
18:   $M \leftarrow -\infty$ 
19:  for all  $\phi \in \Phi$  do
20:     $M' \leftarrow |B_u^{\sigma_1}|$ 
21:    for all  $x \in \mathcal{L}(B_u^{\sigma_1})$  do
22:       $\sigma'_1 \leftarrow \sigma_1; \sigma'_2 \leftarrow \sigma_2$ 
23:      if  $x \in \mathcal{L}(B_u)$  then  $\sigma'_1 \leftarrow \perp$ 
24:      if  $\sigma(x) \in \mathcal{L}(B_v)$  then  $\sigma'_2 \leftarrow \perp$ 
25:       $M' \leftarrow M' + \sigma\text{-MACRS}(x, \phi(x), \sigma'_1, \sigma'_2)$ 
26:     $M \leftarrow \max(M, M')$ 
27:   $D_2[u, v, \sigma_1, \sigma_2] \leftarrow M$ 
28:   $D[u, v, \sigma_1, \sigma_2] \leftarrow \max(D_1[u, v, \sigma_1, \sigma_2], D_2[u, v, \sigma_1, \sigma_2])$ 
29: return  $D[u, v, \sigma_1, \sigma_2]$ 

```

construct $\mathcal{N}_1^{\sigma_1}$ and $\mathcal{N}_2^{\sigma_2}$, and return $-\infty$ if one of these networks does not exist.

If \mathcal{N}^* exists, then it either has a single vertex or its root blob is isomorphic to the root blobs of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$. The former is possible only if $\mathcal{N}_1^{\sigma_1}$ and $\mathcal{N}_2^{\sigma_2}$ have σ_1 - and σ_2 -consistent complete acyclic cherry partitions and their label sets are non-disjoint. ~~subnets-are-reducible-and-overlap~~ $(u, v, \sigma_1, \sigma_2)$ tests this, and we set $D_1[u, v, \sigma_1, \sigma_2]$ to 1 or $-\infty$ accordingly in line 14 or 16.

If \mathcal{N}^* has the same root blob as $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$, then there must exist an isomorphism ϕ between the root blobs $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$, and the pendant network attached to each leaf x of $B_u^{\sigma_1}$ in \mathcal{N}^* must be a (σ_1, σ_2) -

MACRS of $(\mathcal{N}_1^{\sigma_1}(x), \mathcal{N}_2^{\sigma_2}(\phi(x)))$. We call **generate-isomorphisms** $(B_u^{\sigma_1}, B_v^{\sigma_2})$ in line 17 to enumerate the set of all isomorphisms between $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$. For each isomorphism ϕ , lines 19–26 find the size of a corresponding ACRS by calling σ -MACRS $(x, \phi(x), \sigma_1, \sigma_2)$ recursively for each leaf x of $B_u^{\sigma_1}$ and adding the returned sizes of (σ_1, σ_2) -MACRS of $(\mathcal{N}_1^{\sigma_1}(x), \mathcal{N}_2^{\sigma_2}(\phi(x)))$ to $|B_u^{\sigma_1}|$. Thus, $D_2[u, v, \sigma_1, \sigma_2]$, defined in line 27, is the size of the biggest (σ_1, σ_2) -ACRS of $(\mathcal{N}_1^{\sigma_1}(u), \mathcal{N}_2^{\sigma_2}(v))$ that has a root blob isomorphic to $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$. Line 28 then defines $D[u, v, \sigma_1, \sigma_2] = \max(D_1[u, v, \sigma_1, \sigma_2], D_2[u, v, \sigma_1, \sigma_2])$ to record the size of the largest of all (σ_1, σ_2) -ACRS of $(\mathcal{N}_1^{\sigma_1}(u), \mathcal{N}_2^{\sigma_2}(v))$.

The attentive reader will have noticed that the recursive call in line 25 does not use σ_1 and σ_2 as arguments but reticulation selectors σ'_1 and σ'_2 that are reset to \perp whenever $B_x \neq B_u$ or $B_{\phi(x)} \neq B_v$. As we show in Section 4, we have $D[u, v, \sigma_1, \sigma_2] = D[u, v, \sigma'_1, \sigma'_2]$, so this does not affect the result computed by the algorithm, but it ensures that the reticulation selectors passed to each invocation fix cherry paths in at most one blob each. This limits the number of entries of D that need to be computed and is crucial for achieving a running time that is FPT in the level of the networks.

Complexity. Let n be the total size of the two input networks, counting vertices and arcs. The following two lemmas state the costs of key steps in the algorithm.

Lemma 4. *Let \mathcal{N} be a network, let σ be a blob-respecting reticulation selector of \mathcal{N} , and let u be the root of a blob of \mathcal{N}^σ . Then there exists a complete acyclic cherry partition of $\mathcal{N}^\sigma(u)$ that is σ -consistent if and only if there is no path $P \in R(\mathcal{N}^\sigma(u))$ such that $\sigma(P) = 0$. Consequently, each entry $D_1[u, v, \sigma_1, \sigma_2]$ can be computed in linear time.*

Lemma 5. *There are at most $\max(2, 2^{3r/2})$ isomorphisms between two binary blobs B_1 and B_2 with r reticulations. They can be enumerated in $O(2^{3r/2} \cdot |B_1|)$ time. This bound holds even if these blobs are not orchards.*

Theorem 6. *Let \mathcal{N}_1 and \mathcal{N}_2 be two networks of level at most ℓ . Then the MACRS problem can be solved in $O(14.5^\ell \cdot n^3)$ time.*

Section 4 proves the correctness of the algorithm, and the detailed complexity analysis can be found in Appendix A. Here, we sketch a simpler but weaker analysis that provides the intuition. Consider all possible inputs u, v, σ_1, σ_2 on which the algorithm can be called. There are $O(n^2)$ combinations of u and v . Then, σ_1 is either \perp , or it fixes all reticulated cherry paths in B_u and in no other blob (since line 23 resets σ_1 to \perp whenever we exit a blob). In a blob with ℓ reticulations, there are 2ℓ reticulated cherry paths, two per reticulation, and so, naively, there are 4^ℓ ways to fix these paths in a blob. However, for a reticulation v , there is no point in fixing both its reticulated cherry paths to 1, since no partial cherry partition can contain both. Hence, the **fix** routine considers, for each reticulation, at most three ways of fixing its reticulated cherry paths. This gives $O(3^\ell)$ possible values for σ_1 on input u . Likewise, σ_2 can take $O(3^\ell)$ values on input v , and so the total number of possible inputs is $O(9^\ell \cdot n^2)$.

The running time is dominated by the calls in which $\sigma_1, \sigma_2 \neq \perp$. In this case,

computing the entry $D_1[u, v, \sigma_1, \sigma_2]$ takes linear time, by Lemma 4. To compute $D_2[u, v, \sigma_1, \sigma_2]$, we need to enumerate all the isomorphisms, which takes time $O(2^{3\ell/2} \cdot |B_u^{\sigma_1}|)$ time, by Lemma 5. We then spend $O(|B_u^{\sigma_1}|) = O(n)$ time per isomorphism to make the recursive calls on the leaves of the blobs. The number of possible inputs multiplied by this complexity results in a running time of $O(9^\ell \cdot n^2 \cdot 2^{3\ell/2} \cdot n) = O(25.46^\ell \cdot n^3)$. Appendix A improves the first term to 14.5^ℓ .

4 Correctness of the Algorithm

In this section, we develop the recurrence relations for the D , D_1 , and D_2 tables used by Algorithm 1 to prove its correctness. Similar to the algorithm of [19] for level-1 networks, the idea is to guess the reticulated cherries to be included in a partial cherry cover that produces a MACRS and then to complete the partial cherry cover by including the necessary proper cherries. The correctness of this approach is easiest to establish when guessing the reticulated cherries globally, for the whole network, which is what the algorithm of [19] does. Therefore, we start by developing recurrence relations for tables D^* , D_1^* , and D_2^* that are analogous to the tables D , D_1 , and D_2 but are indexed by reticulation selectors for the whole networks \mathcal{N}_1 and \mathcal{N}_2 . While this helps with proving the correctness of these recurrence relations, the resulting tables are too big to achieve a running time that is FPT with respect to the level of the input networks. We obtain the tables D , D_1 , and D_2 used by the algorithm as compressed versions of these bigger tables based on the observation that the entries $D^*[u, v, \sigma_1, \sigma_2]$ that we care about depend only on the choices made by σ_1 and σ_2 in B_u and B_v .

4.1 Recurrence Relations for Network-Wide Reticulation Selectors

We say a reticulation selector σ_i *exposes* v if

- σ_i either fixes every cherry path in $R(B_v)$ or none of them,
- There exists at least one partial cherry partition that is σ_i -consistent, and
- v is the root of a blob in $\mathcal{N}_i^{\sigma_i}$

In particular, by the last condition, if σ_i fixes none of the cherry paths in $R(B_v)$, then v must be the root of B_v . By the second condition, the network $\mathcal{N}_i^{\sigma_i}$ exists. We can now define our D^* table.

Definition 7. For two vertices u and v of \mathcal{N}_1 and \mathcal{N}_2 , respectively, and two reticulation selectors σ_1 and σ_2 that expose u and v , respectively, we define $D^*[u, v, \sigma_1, \sigma_2]$ to be the size of a (σ_1, σ_2) -MACRS of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$.

If ρ_1 and ρ_2 are the roots of \mathcal{N}_1 and \mathcal{N}_2 , respectively, then $\mathcal{N}_1^{\sigma_1}(\rho_1) = \mathcal{N}_1$ and $\mathcal{N}_2^{\sigma_2}(\rho_2) = \mathcal{N}_2$. Thus, $D^*[\rho_1, \rho_2, \perp, \perp]$ is the size of a MACRS of \mathcal{N}_1 and \mathcal{N}_2 .

Now we call a reticulation selector σ'_i in \mathcal{N}_i a *B-extension*, for a blob B of \mathcal{N}_i if $\sigma_i^{-1}(0) \subseteq (\sigma'_i)^{-1}(0)$, $\sigma_i^{-1}(1) \subseteq (\sigma'_i)^{-1}(1)$, and $\sigma'_i(P) \in \{0, 1\}$, for all $P \in R(B)$. In words, σ'_i and σ_i agree on the selection of cherry paths fixed by σ_i , and σ'_i fixes all cherry paths in $R(B)$. Let $\text{Ext}(\sigma_i, B)$ be the set of all B -extensions of σ_i .

Observation 8. For a reticulation selector σ_i of \mathcal{N}_i and a blob B of \mathcal{N}_i , a partial cherry partition of \mathcal{N}_i is σ_i -consistent if and only if it is σ'_i -consistent for some $\sigma'_i \in \text{Ext}(\sigma_i, B)$.

Thus, for any two vertices $u \in \mathcal{N}_1$ and $v \in \mathcal{N}_2$, and reticulation selectors σ_1 and σ_2 that expose u and v , respectively, we have

$$D^*[u, v, \sigma_1, \sigma_2] = \max_{\substack{\sigma'_1 \in \text{Ext}(\sigma_1, B_u) \\ \sigma'_2 \in \text{Ext}(\sigma_2, B_v)}} D^*[u, v, \sigma'_1, \sigma'_2]. \quad (1)$$

If σ_1 and σ_2 fix all reticulated cherry paths in $R(B_u)$ and $R(B_v)$, respectively, then this equation holds trivially because $\text{Ext}(\sigma_1, B_u) = \{\sigma_1\}$ and $\text{Ext}(\sigma_2, B_v) = \{\sigma_2\}$, but it defines $D^*[u, v, \sigma_1, \sigma_2]$ in terms of itself and results in an infinite recursion. In this case, we use a different recurrence relation for $D^*[u, v, \sigma_1, \sigma_2]$.

So assume that σ_1 fixes all reticulated cherry paths in $R(B_u)$, and σ_2 fixes all reticulated cherry paths in $R(B_v)$. Assume further that there exists a (σ_1, σ_2) -ACRS \mathcal{N}^* of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$. Then we can view \mathcal{N}^* as a σ_1 -CRS of $\mathcal{N}_1^{\sigma_1}(u)$ together with an isomorphism ϕ between \mathcal{N}^* and a σ_2 -CRS $\phi(\mathcal{N}^*)$ of $\mathcal{N}_2^{\sigma_2}(v)$. The following lemma helps to characterize what such a network \mathcal{N}^* looks like:

Lemma 9. *If $v \in \mathcal{N}$ and σ is a reticulation selector in \mathcal{N} that exposes v and fixes all reticulated cherry paths in B_v , then for any σ -consistent acyclic cherry partition \mathcal{P} of $\mathcal{N}^\sigma(v)$, the following holds:*

- If B_v^σ is non-trivial, then it is also the root blob of $\mathcal{N}^\sigma(v) - \mathcal{P}$.
- If B_v^σ is trivial, then either it is the root blob of $\mathcal{N}^\sigma(v) - \mathcal{P}$ or $\mathcal{N}^\sigma(v) - \mathcal{P}$ has v as its only vertex.

By Lemma 9, either the two blobs $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ are both trivial and u is the only vertex of \mathcal{N}^* , or the two blobs are preserved and $\phi|_{B_u^{\sigma_1}}$ is an isomorphism between $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ (in the latter case, the blobs could be trivial or not). We capture these two cases in two auxiliary tables D_1^* and D_2^* :

- $D_1^*[u, v, \sigma_1, \sigma_2] = 1$ if there exists a (σ_1, σ_2) -ACRS of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$ that has u as its only vertex, and $D_1^*[u, v, \sigma_1, \sigma_2] = -\infty$ otherwise.
- $D_2^*[u, v, \sigma_1, \sigma_2]$ is the size of a (σ_1, σ_2) -MACRS of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$ that has more than one vertex, and $D_2^*[u, v, \sigma_1, \sigma_2] = -\infty$ if no such ACRS exists.

Then, by Lemma 9,

$$D^*[u, v, \sigma_1, \sigma_2] = \max(D_1^*[u, v, \sigma_1, \sigma_2], D_2^*[u, v, \sigma_1, \sigma_2]). \quad (2)$$

Note that, if u or v is a leaf, then $D_2^*[u, v, \sigma_1, \sigma_2] = -\infty$. Thus, $D^*[u, v, \sigma_1, \sigma_2] = D_1^*[u, v, \sigma_1, \sigma_2]$ in this case. This forms the base case of the recurrence.

The entries in the table D_1^* are defined without depending on any other table entries. We do not worry about its computation, as we use D^* , D_1^* , and D_2^* only as vehicles to arrive at the definitions of the tables D , D_1 , and D_2 . Next, we develop a recurrence relation for the entries in D_2^* :

If \mathcal{N}^* is a (σ_1, σ_2) -ACRS with more than one vertex, then we know by Lemma 9 that $B_u^{\sigma_1}$ must be the root blob of \mathcal{N}^* and must be isomorphic to $B_v^{\sigma_2}$. In particular, the leaves of $B_u^{\sigma_1}$ are mapped onto the leaves of $B_v^{\sigma_2}$. Since these leaves are also vertices of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$, respectively, and their parent arcs are also bridges of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$, this implies that, for every such leaf x of $B_u^{\sigma_1}$, $\mathcal{N}^*(x)$ is a subnetwork of $\mathcal{N}_1^{\sigma_1}(x)$ and $\mathcal{N}_2^{\sigma_2}(\phi(x))$.

The following observation follows immediately from Lemma 2 after observing that, for any pendant subnetwork $\mathcal{N}(x)$ of \mathcal{N} and any σ -consistent partial cherry partition \mathcal{P} of \mathcal{N} , $\mathcal{P} \cap C(\mathcal{N}(x))$ is also σ -consistent.

Observation 10. *Let \mathcal{N}' be a σ -CRS of \mathcal{N} , and let x be the bottom endpoint of a bridge in \mathcal{N} . Then $\mathcal{N}'(x)$ is a σ -CRS of $\mathcal{N}(x)$.*

Observation 10 implies that $\mathcal{N}^*(x)$ is not only a subnetwork of $\mathcal{N}_1^{\sigma_1}(x)$ and $\mathcal{N}_2^{\sigma_2}(\phi(x))$ but a (σ_1, σ_2) -ACRS of these two networks, for every leaf x of $B_u^{\sigma_1}$. This immediately implies that

$$D_2^*[u, v, \sigma_1, \sigma_2] \leq |B_u^{\sigma_1}| + \max_{\phi} \left(\sum_{x \in \mathcal{L}(B_u^{\sigma_1})} D^*[x, \phi(x), \sigma_1, \sigma_2] \right), \quad (3)$$

where the maximum is taken over all isomorphisms ϕ between $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$. If $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ are non-isomorphic, then $D_2^*[u, v, \sigma_1, \sigma_2] = -\infty$.

Conversely, for every isomorphism ϕ between $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ and every collection of (σ_1, σ_2) -ACRSs $\mathcal{N}^*(x)$ of $\mathcal{N}_1^{\sigma_1}(x)$ and $\mathcal{N}_2^{\sigma_2}(\phi(x))$, for all $x \in \mathcal{L}(B_u^{\sigma_1})$, $\mathcal{N}^* := B_u^{\sigma_1} \cup \bigcup_{x \in \mathcal{L}(B_u^{\sigma_1})} \mathcal{N}^*(x)$ is a common subnetwork of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$. The next lemma shows that it is in fact a (σ_1, σ_2) -ACRS of these two networks.

Lemma 11. *If ϕ is an isomorphism between $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ and $\mathcal{N}^*(x)$ is a (σ_1, σ_2) -ACRS of $\mathcal{N}_1^{\sigma_1}(x)$ and $\mathcal{N}_2^{\sigma_2}(\phi(x))$, for all $x \in \mathcal{L}(B_u^{\sigma_1})$, then $\mathcal{N}^* := B_u^{\sigma_1} \cup \bigcup_{x \in \mathcal{L}(B_u^{\sigma_1})} \mathcal{N}^*(x)$ is a (σ_1, σ_2) -ACRS of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$.*

This immediately implies the converse of eq. (3):

$$D_2^*[u, v, \sigma_1, \sigma_2] \geq |B_u^{\sigma_1}| + \max_{\phi} \left(\sum_{x \in \mathcal{L}(B_u^{\sigma_1})} D^*[x, \phi(x), \sigma_1, \sigma_2] \right). \quad (4)$$

Together, eqs. (3) and (4) imply that

$$D_2^*[u, v, \sigma_1, \sigma_2] = |B_u^{\sigma_1}| + \max_{\phi} \left(\sum_{x \in \mathcal{L}(B_u^{\sigma_1})} D^*[x, \phi(x), \sigma_1, \sigma_2] \right), \quad (5)$$

unless $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ are non-isomorphic, in which case $D_2^*[u, v, \sigma_1, \sigma_2] = -\infty$.

The table D^* can be filled in using eqs. (1), (2) and (5), and it is not hard to show that the cost of doing so is FPT with respect to the total number r of reticulations in \mathcal{N}_1 and \mathcal{N}_2 : There are at most 3^r pairs of reticulation selectors (σ_1, σ_2) that have consistent partial cherry partitions. Thus, there are $O(3^r \cdot n^2)$ table entries in D^* , and each can be shown to take $O(f(\ell))$ time to compute, where ℓ is the maximum level of \mathcal{N}_1 and \mathcal{N}_2 .

4.2 Smaller Tables via Reticulation Selectors Restricted to Blobs

To obtain a running time that depends only on the size and maximum level of \mathcal{N}_1 and \mathcal{N}_2 , and not on the total number of reticulations, we need to reduce the sizes of D^* , D_1^* , and D_2^* . Intuitively, for an entry $D^*[u, v, \sigma_1, \sigma_2]$ where u is a blob root of \mathcal{N}_1 , we can prune away from σ_1 any selection $\sigma_1(P) \in \{0, 1\}$ where $P \notin R(\mathcal{N}_1(u))$, and use \perp instead. The same holds for σ_2 , which helps reduce

the table sizes significantly. To formalize this, we need two lemmas:

Lemma 12. *Let $u \in \mathcal{N}_1$, $v \in \mathcal{N}_2$, σ_1 and σ'_1 reticulation selectors in \mathcal{N}_1 that expose u , σ_2 and σ'_2 reticulation selectors in \mathcal{N}_2 that expose v , and r_u and r_v the roots of B_u and B_v . If $\sigma_1(P) = \sigma'_1(P)$ for every reticulated cherry path $P \in R(\mathcal{N}_1(r_u))$, and $\sigma_2(P) = \sigma'_2(P)$ for every reticulated cherry path $P \in R(\mathcal{N}_2(r_v))$, then $D^*[u, v, \sigma_1, \sigma_2] = D^*[u, v, \sigma'_1, \sigma'_2]$.*

Consider the dependency graph between all entries in D^* , with an arrow from an entry $D^*[u, v, \sigma_1, \sigma_2]$ to an entry $D^*[x, y, \sigma'_1, \sigma'_2]$ if the computation of $D^*[u, v, \sigma_1, \sigma_2]$ using eq. (1) or eqs. (2) and (5) depends on $D^*[x, y, \sigma'_1, \sigma'_2]$. We say that the computation of $D^*[\rho_1, \rho_2, \perp, \perp]$ *requires* an entry $D^*[u, v, \sigma_1, \sigma_2]$ if there exists a path from $D^*[\rho_1, \rho_2, \perp, \perp]$ to $D^*[u, v, \sigma_1, \sigma_2]$ in this graph. Clearly, only entries required by $D^*[\rho_1, \rho_2, \perp, \perp]$ need to be computed.

Lemma 13. *For every entry $D^*[u, v, \sigma_1, \sigma_2]$ required by $D^*[\rho_1, \rho_2, \perp, \perp]$, σ_1 fixes all or none of the reticulated cherry paths in $R(B_u)$ and fixes none of the reticulated cherry paths in proper descendant blobs of B_u , and σ_2 fixes all or none of the reticulated cherry paths in $R(B_v)$ and fixes none of the reticulated cherry paths in proper descendant blobs of B_v .*

By Lemma 13, we only need to know $D^*[u, v, \sigma_1, \sigma_2]$ for tuples $(u, v, \sigma_1, \sigma_2)$ such that σ_1 and σ_2 do not fix any cherry paths in proper descendant blobs of B_u and B_v . Thus, for two such entries $D^*[u, v, \sigma_1, \sigma_2]$ and $D^*[u, v, \sigma'_1, \sigma'_2]$, σ_1 and σ'_1 agree on all reticulated cherry paths reachable from the root of B_u if and only if they agree on the reticulated cherry paths in $R(B_u)$, and σ_2 and σ'_2 agree on all reticulated cherry paths reachable from the root of B_v if and only if they agree on the reticulated cherry paths in $R(B_v)$. By Lemma 12, $D^*[u, v, \sigma_1, \sigma_2] = D^*[u, v, \sigma'_1, \sigma'_2]$ in this case. This allows us to replace the entries in eqs. (1), (2) and (5) with entries in much smaller tables D , D_1 , and D_2 indexed by tuples $(u, v, \sigma_1, \sigma_2)$, where σ_1 and σ_2 are reticulation selectors defined only on the paths in $R(B_u)$ and $R(B_v)$ respectively.

We define $\bar{\sigma}_1$ to be the completion of σ_1 to a reticulation selector on all reticulated cherry paths in $R(\mathcal{N}_1)$ by setting

$$\bar{\sigma}_1(P) = \begin{cases} \sigma_1(P) & \text{for all } P \in R(B_u) \\ \perp & \text{for all } P \in R(\mathcal{N}_1) \setminus R(B_u). \end{cases}$$

We define $\bar{\sigma}_2$ analogously. Then the entries of the tables D , D_1 , and D_2 are defined as

$$\begin{aligned} D[u, v, \sigma_1, \sigma_2] &= D^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2], \\ D_1[u, v, \sigma_1, \sigma_2] &= D_1^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2], \\ D_2[u, v, \sigma_1, \sigma_2] &= D_2^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2]. \end{aligned}$$

By Lemmas 12 and 13, if σ_1 does not fix the reticulated cherry paths in B_u or

σ_2 does not fix the reticulated cherry paths in B_v , then we have

$$\begin{aligned}
D[u, v, \sigma_1, \sigma_2] &= D^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2] \\
&= \max_{\substack{\sigma'_1 \in \text{Ext}(\bar{\sigma}_1, B_u) \\ \sigma'_2 \in \text{Ext}(\bar{\sigma}_2, B_v)}} D^*[u, v, \sigma'_1, \sigma'_2] \\
&= \max_{\substack{\sigma'_1 \in \text{Ext}(\sigma_1, B_u) \\ \sigma'_2 \in \text{Ext}(\sigma_2, B_v)}} D[u, v, \sigma'_1, \sigma'_2]. \tag{6}
\end{aligned}$$

If σ_1 and σ_2 fix the reticulated cherry paths in B_u and B_v (which includes cases where B_u , B_v or both are trivial because there are no cherry paths to fix then), then

$$\begin{aligned}
D[u, v, \sigma_1, \sigma_2] &= D^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2] \\
&= \max(D_1^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2], D_2^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2]) \\
&= \max(D_1[u, v, \sigma_1, \sigma_2], D_2[u, v, \sigma_1, \sigma_2]). \tag{7}
\end{aligned}$$

The computation of the D_1 entries was already discussed in Lemma 4.

As for D_2 , we have $D_2[u, v, \sigma_1, \sigma_2] = -\infty$ if $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$ are non-isomorphic. Otherwise, we have

$$\begin{aligned}
D_2[u, v, \sigma_1, \sigma_2] &= D_2^*[u, v, \bar{\sigma}_1, \bar{\sigma}_2] \\
&= |B_u^{\sigma_1}| + \max_{\phi} \left(\sum_{x \in \mathcal{L}(B_u^{\sigma_1})} D^*[x, \phi(x), \bar{\sigma}_1, \bar{\sigma}_2] \right) \\
&= |B_u^{\sigma_1}| + \max_{\phi} \left(\sum_{x \in \mathcal{L}(B_u^{\sigma_1})} D[x, \phi(x), \sigma_1, \sigma_2] \right), \tag{8}
\end{aligned}$$

where the maximum is taken over all isomorphisms $\phi : B_u^{\sigma_1} \rightarrow B_v^{\sigma_2}$. It follows that the value of $D[\rho_1, \rho_2, \perp, \perp]$ obtained from the above recurrences is equal to the desired value of $D^*[\rho_1, \rho_2, \perp, \perp]$. Moreover, one easily verifies that the above recurrences for D are equivalent to those that are computed by the algorithm, and the latter is therefore correct.

5 Discussion

In this paper, we have answered the open problem of [19] by showing that the MACRS problem is FPT with respect to level. This required new ideas on efficient 2ECC isomorphism enumeration and partial cherry partitions that could be useful for other problems. This enumeration makes use of a generator, a graph that represents the internal topology of a 2ECC. In the future, we will aim to implement our algorithm and evaluate it empirically. The algorithm needs to store $O(9^\ell \cdot n^2)$ entries, which could be a practical limitation due to its space requirements. On the other hand, it appears reasonable to expect most entries to have few isomorphisms to enumerate between 2ECCs, with some entries having no isomorphism at all. In this manner, the number of entries that the algorithm needs to compute on practical inputs could be many fewer than our worst-case upper bound, although this remains to be established.

We also believe that the complexity analysis of our algorithm can be im-

proved. In particular, it should be possible to improve the n^3 factor to n^2 with appropriate data structures. It will also be interesting to verify whether our dissimilarity measure can be applied to networks that are not necessarily orchards. Although an ACRS may not exist in this case, our techniques could be generalized to detect whether two non-orchards can be made identical using cherry-picking operations, while doing so in a minimal way.

References

1. Bansal, M.S., Kellis, M., Kordi, M., Kundu, S.: Ranger-dtl 2.0: rigorous reconstruction of gene-family evolution by duplication, transfer and loss. *Bioinformatics* **34**(18), 3214–3216 (2018)
2. Bean Jr, W.J., Cox, N.J., Kendal, A.P.: Recombination of human influenza a viruses in nature. *Nature* **284**(5757), 638–640 (1980)
3. Berry, V., Scornavacca, C., Weller, M.: Scanning phylogenetic networks is np-hard. In: *International Conference on Current Trends in Theory and Practice of Informatics*. pp. 519–530. Springer (2020)
4. Bruchhold, S., Weller, M.: Exploiting low scanwidth to resolve soft polytomies. In: Kozik, J., Wolff, A. (eds.) *SOFSEM 2026: Theory and Practice of Computer Science*. pp. 332–346. Springer Nature Switzerland, Cham (2026)
5. Cardona, G., Pons, J.C., Ribas, G., Coronado, T.M.: Comparison of orchard networks using their extended μ -representation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **21**(3), 501–507 (2024)
6. Erdős, P.L., Semple, C., Steel, M.: A class of phylogenetic networks reconstructable from ancestral profiles. *Mathematical biosciences* **313**, 33–40 (2019)
7. Gambette, P., Berry, V., Paul, C.: The structure of level-k phylogenetic networks. In: *Annual Symposium on Combinatorial Pattern Matching*. pp. 289–300. Springer (2009)
8. Hellmuth, M., Schaller, D., Stadler, P.F.: Clustering systems of phylogenetic networks. *Theory in Biosciences* **142**(4), 301–358 (2023)
9. Hills, M.J., Hall, L.M., Messenger, D.F., Graf, R.J., Beres, B.L., Eudes, F.: Evaluation of crossability between triticales (x triticosecale wittmack) and common wheat, durum wheat and rye. *Environmental biosafety research* **6**(4), 249–257 (2007)
10. van Iersel, L., Janssen, R., Jones, M., Murakami, Y.: Orchard networks are trees with additional horizontal arcs. *Bulletin of Mathematical Biology* **84**(8), 76 (2022)
11. van Iersel, L., Janssen, R., Jones, M., Murakami, Y., Zeh, N.: A unifying characterization of tree-based networks and orchard networks using cherry covers. *Advances in Applied Mathematics* **129**, 102222 (2021)
12. van Iersel, L., Jones, M., Weller, M.: Embedding phylogenetic trees in networks of low treewidth. *Discrete Mathematics & Theoretical Computer Science* **25**(Discrete Algorithms) (2023)
13. Jacox, E., Chauve, C., Szöllősi, G.J., Ponty, Y., Scornavacca, C.: eccetera: comprehensive gene tree-species tree reconciliation using parsimony. *Bioinformatics* **32**(13), 2056–2058 (2016)
14. Janssen, R., Murakami, Y.: On cherry-picking and network containment. *Theoretical Computer Science* **856**, 121–150 (2021)
15. Jones, M., Schestag, J.: Parameterized Algorithms for Diversity of Networks with Ecological Dependencies. In: *20th International Symposium on Parameterized and Exact Computation (IPEC 2025)*. *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 358, pp. 11:1–11:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2025)

16. Kelk, S., Scornavacca, C.: Constructing minimal phylogenetic networks from soft-wired clusters is fixed parameter tractable. *Algorithmica* **68**(4), 886–915 (2014)
17. Landry, K., Teodocio, A., Lafond, M., Tremblay-Savard, O.: Defining phylogenetic network distances using cherry operations. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **20**(3), 1654–1666 (2022)
18. Landry, K., Tremblay-Savard, O.: Fast calculation of cherry distance on level-1 orchard networks: Optimization, heuristic and implementation. In: *RECOMB International Workshop on Comparative Genomics*. pp. 157–177. Springer (2025)
19. Landry, K., Tremblay-Savard, O., Lafond, M.: A fixed-parameter tractable algorithm for finding agreement cherry-reduced subnetworks in level-1 orchard networks. *Journal of Computational Biology* **31**(4), 360–379 (2024)
20. Lin, Y., Rajan, V., Moret, B.M.: A metric for phylogenetic trees based on matching. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **9**(4), 1014–1022 (2011)
21. López Sánchez, A., Lafond, M.: Predicting horizontal gene transfers with perfect transfer networks. *Algorithms for Molecular Biology* **19**(1), 6 (2024)
22. Nakhleh, L., Ringe, D.A., Warnow, T.: Perfect phylogenetic networks: A new methodology for reconstructing the evolutionary history of natural languages. *Language* **81**(2), 382–420 (2005)
23. Robinson, D.F., Foulds, L.R.: Comparison of phylogenetic trees. *Mathematical biosciences* **53**(1-2), 131–147 (1981)
24. Scornavacca, C., Weller, M.: Treewidth-based algorithms for the small parsimony problem on networks. *Algorithms for Molecular Biology* **17**(1), 15 (2022)
25. Van Iersel, L., Kelk, S., Lekic, N., Whidden, C., Zeh, N.: Hybridization number on three rooted binary trees is EPT. *SIAM Journal on Discrete Mathematics* **30**(3), 1607–1631 (2016)

A Running Time Analysis

The simple analysis given in the main text of the paper only shows a running time of $O((18\sqrt{2})^\ell \cdot \text{poly}(n)) \approx O(25.5^\ell \cdot \text{poly}(n))$ for our algorithm. In this appendix, we prove the $O((6(1+\sqrt{2}))^\ell \cdot n^3) \approx O(14.5^\ell \cdot n^3)$ running time claimed in Theorem 6. To prove it, let $\alpha = 6(1+\sqrt{2})$. Then we assign $O(\alpha^\ell \cdot n^2)$ credits to every vertex in \mathcal{N}_1 and show that these credits can pay for the cost of the entire algorithm. As there are $O(n)$ vertices in \mathcal{N}_1 , this shows that the cost of the algorithm is $O(\alpha^\ell \cdot n^3)$.

Algorithm 1 is a recursive algorithm that computes only table entries that the computation of $D[\rho_1, \rho_2, \perp, \perp]$ relies on. It is easily verified that this means that the only entries that are computed are those entries $D[u, v, \sigma_1, \sigma_2]$ for which u is the root of $B_u^{\sigma_1}$ and v is the root of $B_v^{\sigma_2}$. Consider the cost of computing a single such entry $D[u, v, \sigma_1, \sigma_2]$. If σ_1 does not fix all reticulated cherry paths in B_u and σ_2 does not fix all reticulated cherry paths in B_v , then calculating this entry using eq. (6) takes $O(9^\ell)$ time, as there are at most 3^ℓ B_u -extensions of σ_1 in $\text{Ext}(\sigma_1, B_u)$ and at most 3^ℓ B_v -extensions of σ_2 in $\text{Ext}(\sigma_2, B_v)$. Summed over all such entries for a fixed choice of u , the cost is $O(9^\ell \cdot n)$ because there are n choices for v and only one choice for each of σ_1 and σ_2 .

If σ_1 fixes all reticulated cherry paths in B_u but σ_2 does not fix all reticulated cherry paths in B_v , then calculating $D[u, v, \sigma_1, \sigma_2]$ using eq. (6) takes $O(3^\ell)$ time, as there are at most 3^ℓ B_v -extensions of σ_2 in $\text{Ext}(\sigma_2, B_v)$. Summed over all such

entries for a fixed choice of u , the cost is $O(9^\ell \cdot n)$ because there are n choices for v , at most 3^ℓ choices for σ_1 , and only one choice for σ_2 . By an analogous argument, computing all entries $D[u, v, \sigma_1, \sigma_2]$ such that σ_1 does not fix all reticulated cherry paths in B_u but σ_2 fixes all reticulated cherry paths in B_v , takes $O(9^\ell \cdot n)$ time. Thus, the total cost of computing all entries $D[u, v, \sigma_1, \sigma_2]$ such that σ_1 does not fix all reticulated cherry paths in B_u or σ_2 does not fix all reticulated cherry paths in B_v is $O(9^\ell \cdot n)$, which we pay from the credits assigned to u .

Now assume that σ_1 fixes all reticulated cherry paths in B_u and σ_2 fixes all reticulated cherry paths in B_v . Then for each such entry, computing the value of $D_1[u, v, \sigma_1, \sigma_2]$ takes time $O(n)$ by Lemma 4. We charge this time to u , for each possible v, σ_1 , and σ_2 , adding a total charge of $O(9^\ell \cdot n^2)$ to u . Then, the rest of the computation for this entry using eqs. (7) and (8) takes $O(2^{3\ell_1/2} \cdot |B_u^{\sigma_1}|)$ time, where ℓ_1 is the number of reticulations in $B_u^{\sigma_1}$. By Lemma 5, there are at most $2^{3\ell_1/2}$ isomorphisms between $B_u^{\sigma_1}$ and $B_v^{\sigma_2}$, and they can be found in $O(2^{3\ell_1/2} \cdot |B_u^{\sigma_1}|)$ time. For each isomorphism, we need to evaluate the sum in eq. (8), which takes $O(|B_u^{\sigma_1}|)$ time. We pay for this cost by charging an $O(2^{3\ell_1/2})$ portion of it to each vertex in the 2ECC of \mathcal{N}^{σ_1} contained in $B_u^{\sigma_1}$. This is indeed sufficient, as it is easily verified that at least a constant fraction of the vertices in $B_u^{\sigma_1}$ belong to this 2ECC. We need to prove that no vertex is charged for a total cost greater than $O(\alpha^\ell \cdot n)$ in this fashion.

Fix such a vertex x in some blob B of \mathcal{N}_1 , and consider all tuples $(u, v, \sigma_1, \sigma_2)$ such that we charge x for a portion of the cost of computing $D[u, v, \sigma_1, \sigma_2]$. Then x belongs to the 2ECC of B^{σ_1} contained in $B_u^{\sigma_1}$. Let $r(\sigma_1)$ be the number of paths $P \in R(B)$ such that $\sigma_1(P) = 1$. Then $\ell_1 \leq \ell - r(\sigma_1)$, that is, we charge x for an $O(2^{3(\ell - r(\sigma_1))/2})$ portion of the cost of computing $D[u, v, \sigma_1, \sigma_2]$. For a fixed choice of σ_1 , there exists at most one vertex $u \in B$ such that x is charged for part of the cost of computing entries $D[u, v, \sigma_1, \sigma_2]$ because x is part of the 2ECC of B^{σ_1} contained in $B_u^{\sigma_1}$ and the 2ECCs of B^{σ_1} are vertex-disjoint. Thus, the total cost for which we charge x is

$$O\left(3^\ell \cdot n \cdot \sum_{r(\sigma_1)=0}^{\ell} \binom{\ell}{r(\sigma_1)} 2^{r(\sigma_1)} 2^{3(\ell - r(\sigma_1))/2}\right). \quad (9)$$

Indeed, there are 3^ℓ choices for σ_2 and n choices for v and, as just argued, only one choice for u . Each choice of σ_1 charges x for a cost of at most $O(2^{3(\ell - r(\sigma_1))/2})$. For each value of $r(\sigma_1)$, there are at most $\binom{\ell}{r(\sigma_1)} 2^{r(\sigma_1)}$ choices of σ_1 : First, we choose $r(\sigma_1)$ of the at most ℓ reticulations in B as the reticulations that are part of reticulated cherry paths P with $\sigma_1(P) = 1$. Then, for each such reticulation z , we choose one of its parent arcs as the interior arc of such a cherry path P . There are 2 choices of this arc for every reticulation, $2^{r(\sigma_1)}$ choices for all chosen reticulations. Equation (9) simplifies to

$$O(3^\ell \cdot n \cdot (2 + 2^{3/2})^\ell) = O((6(1 + \sqrt{2}))^\ell n) = O(\alpha^\ell n).$$

This proves that no vertex in \mathcal{N}_1 is charged for more than an $O(9^\ell \cdot n^2 + \alpha^\ell \cdot n)$ cost of filling in the table D , which is $O(\alpha^\ell \cdot n^2)$. Since these charges pay for the cost of the algorithm and there are $O(n)$ vertices in \mathcal{N}_1 , the cost of the algorithm

is $O(\alpha^\ell \cdot n^3)$, as claimed.

B Proofs for Section 2 (Preliminaries)

Proof of Proposition 1. First, assume that $\mathcal{N}' = \mathcal{N} \wedge S$, for some cherry picking sequence $S = \langle (a_1, b_1), \dots, (a_n, b_n) \rangle$. We use induction on n to prove that there exists an acyclic cherry partition \mathcal{P} of \mathcal{N} such that $\mathcal{N}' = \mathcal{N} - \mathcal{P}$.

If $n = 0$, then $\mathcal{N} = \mathcal{N}' = \mathcal{N} - \emptyset$, and $\mathcal{P} = \emptyset$ is clearly acyclic. So assume that $n > 0$, let P_1 be the cherry path in \mathcal{N} corresponding to (a_1, b_1) , let $\mathcal{N}'' = \mathcal{N} \wedge (a_1, b_1)$, and let $S' = \langle (a_2, b_2), \dots, (a_n, b_n) \rangle$. Then $\mathcal{N}'' = \mathcal{N} - \{P_1\}$ and $\mathcal{N}' = \mathcal{N}'' \wedge S'$. Thus, by the induction hypothesis, there exists an acyclic cherry partition \mathcal{P}' of \mathcal{N}'' such that $\mathcal{N}' = \mathcal{N}'' - \mathcal{P}'$. Thus, for $\mathcal{P} = \mathcal{P}' \cup \{P_1\}$, we have $\mathcal{N}' = \mathcal{N} - \mathcal{P}$. Since \mathcal{P}' is a cherry partition of \mathcal{N}'' and every leaf of \mathcal{N}'' is either a leaf of \mathcal{N} or an internal vertex of P_1 , \mathcal{P} is a cherry partition of \mathcal{N} . Since \mathcal{P}' is acyclic, any cycle in $G_{\mathcal{P}}$ would have to involve P_1 . Since both endpoints of P_1 are leaves of \mathcal{N} , no such cycle can exist. Thus, \mathcal{P} is acyclic.

Now, assume that $\mathcal{N}' = \mathcal{N} - \mathcal{P}$, for some acyclic cherry partition $\mathcal{P} = \{P_1, \dots, P_n\}$ of \mathcal{N} . Assume that the cherry paths in \mathcal{P} are indexed according to a topological ordering of $G_{\mathcal{P}}$. We use induction on n to prove that there exists a partial cherry picking sequence S such that $\mathcal{N}' = \mathcal{N} \wedge S$.

If $n = 0$, then $\mathcal{N}' = \mathcal{N} = \mathcal{N} \wedge \langle \rangle$. So assume that $n > 0$. Then, since \mathcal{P} is a cherry partition of \mathcal{N} , each endpoint of P_n is either a leaf of \mathcal{N} or an internal vertex of some other cherry path $P_j \in \mathcal{P}$. In the latter case, $G_{\mathcal{P}}$ would contain the arc (P_n, P_j) , but this is impossible because the cherry paths in \mathcal{P} are indexed according to a topological ordering of $G_{\mathcal{P}}$. Thus, the two endpoints a_1 and b_1 of P_n are leaves of \mathcal{N} and form a proper or reticulated cherry (a_1, b_1) of \mathcal{N} such that $\mathcal{N}'' := \mathcal{N} \wedge \langle (a_1, b_1) \rangle = \mathcal{N} - P_n$. Now, $\mathcal{N}' = \mathcal{N}'' - \{P_1, \dots, P_{n-1}\}$. Thus, by the induction hypothesis, there exists a cherry picking sequence S' such that $\mathcal{N}' = \mathcal{N}'' \wedge S'$. Therefore, since $\mathcal{N}'' = \mathcal{N} \wedge \langle (a_1, b_1) \rangle$, $\mathcal{N}' = \mathcal{N} \wedge (\langle (a_1, b_1) \rangle \circ S')$. \square

Proof of Lemma 2. (i) The claim that $C(B_1), \dots, C(B_t)$ are pairwise disjoint is obvious. To prove that $C(\mathcal{N}) = C(B_1) \cup \dots \cup C(B_t)$, we need to prove that every cherry path $P \in C(\mathcal{N})$ belongs to $C(B_i)$, for some blob B_i of \mathcal{N} . So consider such a cherry path P , and let (u, v) be the arc in P such that u is a tree vertex and v is an endpoint of P . Such an arc exists whether P is a proper or reticulated cherry path. Since the blobs of \mathcal{N} partition the arcs of \mathcal{N} , there must exist a blob B_i that contains this arc. We need to prove that B_i also contains the other arcs of P . Since $(u, v) \in B_i$, u is not a leaf of B_i . Since it is a tree vertex, B_i contains both child arcs of u . Let (u, w) be the other child arc of u . If P is a proper cherry path, then $P = \langle v, u, w \rangle$, that is, both arcs of P belong to B_i . If P is a reticulated cherry path, then w is a reticulation whose child is some vertex z , and $P = \langle z, w, u, v \rangle$. Both parent arcs of a reticulation belong to the same 2ECC of \mathcal{N} . Thus, neither of these two parent arcs is a bridge, which implies that w is not a leaf of B_i . Thus, its child z belongs to B_i , as does the arc (w, z) . Since (u, v) and (u, w) also belong to B_i , this shows that all arcs of P belong to B_i also when P is a reticulated cherry path.

(ii) Consider any subset $\mathcal{P} \subseteq C(\mathcal{N})$ and subsets $\mathcal{P}_1, \dots, \mathcal{P}_t \subseteq \mathcal{P}$ as defined

in the lemma. If \mathcal{P} is an acyclic cherry partition of \mathcal{N} , then every subset of \mathcal{P} is acyclic and the paths in every such subset are arc-disjoint. Thus, each set \mathcal{P}_i is an acyclic cherry partition of B_i if every endpoint v of a path in \mathcal{P}_i is a leaf of B_i or an internal vertex of another path in \mathcal{P}_i . If v is a leaf of \mathcal{N} , then it is also a leaf of B_i . Otherwise, since \mathcal{P} is an acyclic cherry partition of \mathcal{N} , there exists a path $P \in \mathcal{P}$ that has v as an internal vertex. If $P \in \mathcal{P}_i$, then v is an internal vertex of another path in \mathcal{P}_i . If $P \notin \mathcal{P}_i$, then P must belong to another blob of \mathcal{N} , which makes the parent arc of v a bridge in \mathcal{N} . Thus, v is a leaf of B_i . This shows that each set \mathcal{P}_i is a partial cherry partition of B_i . If the root of some blob B_i is part of cherry path $P \in \mathcal{P}$, then all arcs in B_i are reachable from P and, therefore, must be covered by cherry paths in \mathcal{P} . Since these cherry paths are all part of B_i , by (i), they belong to \mathcal{P}_i , that is, \mathcal{P}_i is a complete cherry partition of B_i , as claimed. Conversely, assume that $\mathcal{P}_1, \dots, \mathcal{P}_t$ are acyclic cherry partitions of B_1, \dots, B_t and that \mathcal{P}_i is a complete cherry partition of B_i if the root of B_i is contained in some cherry path in \mathcal{P} . Then the paths in \mathcal{P} are arc-disjoint because the paths in each of the sets $\mathcal{P}_1, \dots, \mathcal{P}_t$ are arc-disjoint and the paths in different sets belong to different blobs of \mathcal{N} . For any endpoint v of some path $P \in \mathcal{P}_i$, either v is a leaf of B_i or it is an internal vertex of another path $P' \in \mathcal{P}_i \subseteq \mathcal{P}$. If v is a leaf of B_i , then v is either a leaf of \mathcal{N} or the root of another blob B_j . In the latter case, \mathcal{P}_j is a complete cherry partition of B_j , that is, it contains a path P' that has v as an internal vertex. This shows that the endpoint of every path in \mathcal{P} is either a leaf of \mathcal{N} or an internal vertex of another path in \mathcal{P} . Since the paths in \mathcal{P} are arc-disjoint, this shows that \mathcal{P} is a cherry partition of \mathcal{N} .

It remains to show that \mathcal{P} is acyclic. Assume the contrary. Then there exists a cycle $C = \langle P_0, \dots, P_k = P_0 \rangle$ in $G_{\mathcal{P}}$. Since each set \mathcal{P}_i is acyclic, this cycle must contain paths from at least two blobs of \mathcal{N} . For each path P_j , let B_j be the blob that contains P_j , let $S = \langle B_0, \dots, B_k \rangle$ be the sequence of blobs corresponding to C , and let $S' = \langle B_{j_0}, \dots, B_{j_r} = B_{j_0} \rangle$ be the subsequence of S obtained from S by replacing every maximal subsequence of consecutive occurrences of the same blob in S with a single occurrence of this blob. Then, for all, $h \in [r]$, the root of B_{j_h} is a leaf of $B_{j_{h-1}}$. In particular, the root of $B_{j_{h-1}}$ can reach the root of B_{j_h} in \mathcal{N} . Since $B_{j_r} = B_{j_0}$, this implies that \mathcal{N} contains a cycle, which is the desired contradiction. Thus, \mathcal{P} is an acyclic cherry partition of \mathcal{N} .

(iii) If \mathcal{N} is orchard, then it has a complete acyclic cherry partition \mathcal{P} . By (ii) the restrictions $\mathcal{P}_1, \dots, \mathcal{P}_t$ to the blobs of \mathcal{N} are acyclic cherry partitions of these blobs, and they must be complete by (i) and because \mathcal{P} is complete. Thus, each blob of \mathcal{N} is orchard. Conversely, if the blobs of \mathcal{N} are orchard, then they have complete acyclic cherry partitions. By (ii), their union \mathcal{P} is an acyclic cherry partition of \mathcal{N} . By (i), \mathcal{P} is complete, so \mathcal{N} is orchard.

(iv) If x is the root of a blob, then $\mathcal{N}(x)$ is a union of blobs B_{i_1}, \dots, B_{i_k} . If \mathcal{P} is an acyclic cherry partition of \mathcal{N} , then the restrictions $\mathcal{P}_{i_1}, \dots, \mathcal{P}_{i_k}$ of \mathcal{P} to these blobs are acyclic cherry partitions of these blobs, by (ii). Since $\mathcal{N}(x)$ is the union of these blobs, we conclude that $\mathcal{P}' := \mathcal{P} \cap C(\mathcal{N}(x)) = \mathcal{P}_{i_1} \cup \dots \cup \mathcal{P}_{i_k}$ is an acyclic cherry partition of $\mathcal{N}(x)$, again by (ii). If $x \notin \mathcal{N}'$, then \mathcal{P} must cover all descendant arcs of x in \mathcal{N} . But these are exactly the arcs in $\mathcal{N}(x)$.

Thus, \mathcal{P}' , being the restriction of \mathcal{P} to $\mathcal{N}(x)$, must be a complete acyclic cherry partition of $\mathcal{N}(x)$, and $\mathcal{N}(x) - \mathcal{P}'$ has x as its only vertex. If $x \in \mathcal{N}'$, then $\mathcal{N}'(x)$ is the subnetwork of $\mathcal{N}(x)$ obtained by removing exactly those paths in \mathcal{P} from $\mathcal{N}(x)$ that belong to $\mathcal{N}(x)$. But those are exactly the paths in \mathcal{P}' , that is, $\mathcal{N}'(x) = \mathcal{N}(x) - \mathcal{P}'$. \square

C Proofs for Section 3 (MACRS Algorithm on Level- k Orchards)

Proof of Lemma 3. We call a σ -consistent acyclic cherry partition \mathcal{P} *minimal* if, for every σ -consistent acyclic cherry partition \mathcal{P}' , $E(\mathcal{P}) \subseteq E(\mathcal{P}')$. Proposition 14 below proves that, if there exists a σ -consistent acyclic cherry partition, then there exists a minimal σ -consistent acyclic cherry partition \mathcal{P} , and such a partition can be found in linear time. By definition, the set $E(\mathcal{P})$ is unique.

Next, we consider two arbitrary minimal σ -consistent partial cherry partitions \mathcal{P} and \mathcal{P}' . Our goal is to prove that $\mathcal{N} - \mathcal{P} = \mathcal{N} - \mathcal{P}'$. Let $\langle B_1, \dots, B_t \rangle$ be the sequence of blobs of \mathcal{N} , arranged bottom-up in \mathcal{N} . By Lemma 2, we can partition \mathcal{P} into acyclic cherry partitions $\mathcal{P}_1, \dots, \mathcal{P}_t$ of the blobs B_1, \dots, B_t . We arrange these acyclic cherry partitions in the sequence $\langle \mathcal{P}_1, \dots, \mathcal{P}_t \rangle$ matching the chosen bottom-up ordering of the corresponding blobs. We obtain a similar sequence $\langle \mathcal{P}'_1, \dots, \mathcal{P}'_t \rangle$ of acyclic cherry partitions of B_1, \dots, B_t whose union is \mathcal{P}' . Next we define two network sequences $\mathcal{N}_0, \dots, \mathcal{N}_t$ and $\mathcal{N}'_0, \dots, \mathcal{N}'_t$ defined as $\mathcal{N}_0 = \mathcal{N}'_0 = \mathcal{N}$, and $\mathcal{N}_i = \mathcal{N}_{i-1} - \mathcal{P}_i$ and $\mathcal{N}'_i = \mathcal{N}'_{i-1} - \mathcal{P}'_i$, for all $i \in [t]$. Then $\mathcal{N}_t = \mathcal{N} - \mathcal{P}$ and $\mathcal{N}'_t = \mathcal{N} - \mathcal{P}'$. Thus, our goal is to prove that $\mathcal{N}_i = \mathcal{N}'_i$. We use induction on i to prove that $\mathcal{N}_i = \mathcal{N}'_i$, for all $i \in [t]_0$.

For $i = 0$, the claim is trivial. So assume that $i \in [t]$ and that $\mathcal{N}_{i-1} = \mathcal{N}'_{i-1}$. We distinguish two cases: (i) If \mathcal{P}_i and \mathcal{P}'_i are *complete* cherry partitions of B_i , then \mathcal{N}_i and \mathcal{N}'_i are obtained from $\mathcal{N}_{i-1} = \mathcal{N}'_{i-1}$ by removing all arcs in B_i . In particular, \mathcal{N}_i and \mathcal{N}'_i have the same vertices and arcs. Thus, to prove that $\mathcal{N}_i = \mathcal{N}'_i$, it suffices to prove that every leaf has the same label in \mathcal{N}_i as in \mathcal{N}'_i . If r_i is the root of B_i , then $\mathcal{L}(\mathcal{N}_i) = \mathcal{L}(\mathcal{N}'_i) = \mathcal{L}(\mathcal{N}_{i-1}) \setminus \mathcal{L}(B_i) \cup \{r_i\}$. Each leaf in $\mathcal{L}(\mathcal{N}_{i-1}) \setminus \mathcal{L}(B_i)$ has the same label in \mathcal{N}_{i-1} , \mathcal{N}_i , and \mathcal{N}'_i because picking cherries does not relabel leaves. By Proposition 15 below applied to B_i , r_i also has the same label in \mathcal{N}_i and \mathcal{N}'_i , so $\mathcal{N}_i = \mathcal{N}'_i$. (ii) If \mathcal{P}_i and \mathcal{P}'_i are *partial* cherry partitions of B_i , then σ must fix all cherry paths in $R(B_i)$. Indeed, as the proof of Proposition 14 shows, an arc e belongs to $E(\mathcal{P})$ if and only if it is reachable from a cherry path $P' \in R(\mathcal{N})$ such that $\sigma(P') = 1$. For an arc $e \in B_i$, this cherry path P' belongs to $R(B_i)$ or to $R(B)$, for some proper ancestor blob B of B_i in \mathcal{N} . In the latter case, all arcs in B_i are reachable from P' , that is, \mathcal{P} and, therefore, \mathcal{P}_i would cover all arcs in B_i , that is, \mathcal{P}_i would be a *complete* cherry partition of B_i . Thus, $P' \in R(B_i)$, that is, σ fixes a cherry path in $R(B_i)$. Since σ is blob-respecting, this implies that it fixes all cherry paths in $R(B_i)$. Therefore, Proposition 16 below shows that we have $\mathcal{P}_i = \mathcal{P}'_i$. Thus, $\mathcal{N}_i = \mathcal{N}_{i-1} - \mathcal{P}_i = \mathcal{N}'_{i-1} - \mathcal{P}'_i = \mathcal{N}'_i$ also in this case. This proves the first half of the lemma about the uniqueness of \mathcal{N}^σ .

Now we prove that we can decide in linear time whether \mathcal{N}^σ exists and, if so,

construct it. Assume that σ is fixing only the reticulated cherry paths in a given blob B of \mathcal{N} . By proposition 16, we can find a σ -consistent cherry partition \mathcal{P} for B in linear time, if it exists. Next, we find complete cherry partitions for the pendant subnetworks of B whose root is a leaf of a path in \mathcal{P} . This is done, in linear time, by picking cherries in any order they come available, and defining the cherry paths as they are picked, producing an acyclic cherry partition (as described by [11, Theorem 4.3]). Partitions produced in this manner are always σ -consistent since σ is unfixed for all pendant subnetworks, and every arbitrary partition is σ -consistent for \perp . Also note that cherry paths do not cross blobs (see Lemma 2), i.e. we can take the union of an arbitrary cherry partition of blobs to form a partition for \mathcal{N}^σ . This means that we can find \mathcal{N}^σ in linear time if it exists, and this completes the proof for the second part of the lemma. \square

Proposition 14. *If there exists a σ -consistent acyclic cherry partition of \mathcal{N} , then there exists a minimal such partition.*

Proof. Let $F = \{P \in R(\mathcal{N}) \mid \sigma(P) = 1\}$, and let D be the set of arcs of \mathcal{N} reachable from paths in F , where we call an arc e reachable from a path $P \in F$ if P contains an arc f such that there exists a directed path in \mathcal{N} that starts with f and ends with e . We prove that, if there exists a σ -consistent acyclic cherry partition at all, then there exists such a partition \mathcal{P} with $E(\mathcal{P}) = D$. We also prove that every σ -consistent acyclic cherry partition \mathcal{P}' satisfies $E(\mathcal{P}') \supseteq D$. Thus, \mathcal{P} is minimal.

So assume that \mathcal{P}' is a σ -consistent acyclic cherry partition of \mathcal{N} . Then $\mathcal{P}' \supseteq F$. Since every endpoint of a path in \mathcal{P}' is either a leaf of \mathcal{N} or an internal vertex of another path in \mathcal{P}' , an inductive argument shows that $E(\mathcal{P}') \supseteq D$, as claimed. Now, let \mathcal{P}' be an arbitrary σ -consistent acyclic cherry partition of \mathcal{N} , let G' be the maximal subgraph of $G_{\mathcal{P}'}$ such that every vertex in G' is reachable from a vertex in F , and let $\mathcal{P} \subseteq \mathcal{P}'$ be the vertex set of G' , that is, $G' = G_{\mathcal{P}}$. Then $F \subseteq \mathcal{P}$ and $G_{\mathcal{P}}$ is acyclic because $G_{\mathcal{P}'}$ is and $G_{\mathcal{P}} \subseteq G_{\mathcal{P}'}$. Moreover, \mathcal{P} is a cherry partition of \mathcal{N} . Indeed, since \mathcal{P}' is a cherry partition of \mathcal{N} and $\mathcal{P} \subseteq \mathcal{P}'$, all paths in \mathcal{P} are arc-disjoint. Thus, if \mathcal{P} is not a cherry partition of \mathcal{N} , then there exists a path $P_1 \in \mathcal{P}$ with an endpoint v that is not a leaf of \mathcal{N} nor an internal vertex of another cherry path in \mathcal{P} . Since \mathcal{P}' is a cherry partition of \mathcal{N} and $\mathcal{P}' \supseteq \mathcal{P}$, this implies that v is an internal vertex of a path $P_2 \in \mathcal{P}' \setminus \mathcal{P}$. This, however, implies that P_2 is reachable from some vertex in F because $P_1 \in \mathcal{P}$ implies that P_1 is. This is a contradiction. Therefore, \mathcal{P} is a cherry partition of \mathcal{N} . Since $F \subseteq \mathcal{P} \subseteq \mathcal{P}'$ and \mathcal{P}' is σ -consistent, so is \mathcal{P} . As already shown, this implies that $E(\mathcal{P}) \supseteq D$. Conversely, $E(\mathcal{P}) \subseteq D$ because, by the definition of \mathcal{P} , every path $P_2 \in \mathcal{P}$ is reachable from a path $P_1 \in F$, which implies that $E(P_2) \subseteq D$. Since this is true for every path in \mathcal{P} , this implies that $E(\mathcal{P}) \subseteq D$. \square

Proposition 15. *For a complete cherry partition \mathcal{P} of a network \mathcal{N} , $\mathcal{N} - \mathcal{P}$ is a network that consists of a single leaf with label set $X(\mathcal{N})$.*

Proof. As shown in [10], \mathcal{P} corresponds to a representation of \mathcal{N} as a base tree T plus exactly those reticulation arcs that are internal to reticulated cherry paths

in \mathcal{P} . For every proper cherry path $\langle u, v, w \rangle$ in \mathcal{P} , u and w are the children of v in T . This implies that when we pick a cherry that makes some vertex v a leaf, then v is labelled with the union of the labels of its children in T . Therefore, by a simple inductive argument, the label of every vertex is the union of the labels of its descendant leaves in T . In particular, the label of the root of T , which is the one vertex of $\mathcal{N} - \mathcal{P}$, is $\bigcup_{v \in \mathcal{L}(\mathcal{N})} X(v) = X(\mathcal{N})$. \square

Proposition 16. *Assume that σ is a reticulation selector that fixes all reticulated cherries in \mathcal{N} . Then every σ -consistent acyclic cherry partition \mathcal{P} of \mathcal{N} is uniquely determined by $E(\mathcal{P})$. In particular, the minimal σ -consistent acyclic cherry partition of \mathcal{N} is unique if it exists. It takes linear time to decide whether this partition exists and, if so, find it.*

Proof. First assume that \mathcal{P} is a σ -consistent acyclic cherry partition of \mathcal{N} . Then the reticulated cherry paths in \mathcal{P} are exactly those paths $P \in R(\mathcal{N})$ that satisfy $\sigma(P) = 1$, as there is no path $P \in R(\mathcal{N})$ with $\sigma(P) = \perp$ and \mathcal{P} is σ -consistent. Thus, $\mathcal{P} \setminus R(\mathcal{N})$ is a set of proper cherries. Since proper cherries do not contain reticulations as internal vertices and no two cherry paths in a cherry partition share an endpoint, this implies that the arcs in $E(\mathcal{P} \setminus R(\mathcal{N}))$ form a binary forest in \mathcal{N} in which every vertex is either a leaf (of the forest) or has two children in the forest. It is easily verified, via an inductive argument that picks cherries bottom-up in this forest, that any binary forest has a unique cherry partition. Thus, $\mathcal{P} = (\mathcal{P} \cap R(\mathcal{N})) \cup (\mathcal{P} \setminus R(\mathcal{N}))$ is also unique, as claimed.

To test whether there exists a minimal σ -consistent acyclic cherry partition \mathcal{P} of \mathcal{N} , we start with the set $F = \{P \in R(\mathcal{N}) \mid \sigma(P) = 1\}$ already defined in the proof of Proposition 14. This is the set of reticulated cherries in \mathcal{P} . If there are two paths in F that share an arc, which is easy to test in linear time, then there is no cherry partition that is a superset of F , that is, there is no σ -consistent cherry partition of \mathcal{N} . Otherwise, the set D of arcs reachable from the paths in F also already defined in the proof of Proposition 14 is easily identified in linear time using a simple graph traversal. By Proposition 14, D is exactly the set of arcs in $E(\mathcal{P})$. Given D and $E(F)$, we can in linear time compute their difference $D' = D \setminus E(F)$. Given D' , we can check whether every vertex of \mathcal{N} either has no child arcs in D' or it is a tree vertex with both its child arcs in D' . D' induces a binary forest in \mathcal{N} in which every vertex is either a leaf or has two children if and only if every vertex of \mathcal{N} passes this test. If D' induces such a forest, we can compute the unique partition \mathcal{P}' of the arcs in D' into proper cherry paths by picking cherries bottom-up in this forest. The union $\mathcal{P} = F \cup \mathcal{P}'$ is a σ -consistent cherry partition of \mathcal{N} . It remains to check whether this partition is acyclic, which we do by constructing $G_{\mathcal{P}}$ and using depth-first search to try to find a cycle. Since all of the steps in this procedure can be implemented in linear time, this proves the second part of the proposition. \square

Proof of Lemma 4. Suppose that $\sigma(P) \in \{1, \perp\}$ for every $P \in R(\mathcal{N}^\sigma(u))$. Then note that, in fact, $\sigma(P) = 1$ is not possible, because $\mathcal{N}^\sigma = \mathcal{N} - \mathcal{P}$, for some σ -consistent acyclic cherry partition \mathcal{P} of \mathcal{N} , and $\sigma(P) = 1$ implies that $P \in \mathcal{P}$, so $P \notin R(\mathcal{N}^\sigma) \supseteq R(\mathcal{N}^\sigma(u))$. Thus, $\sigma(P) = \perp$, for all $P \in R(\mathcal{N}^\sigma(u))$, that is, every

acyclic cherry partition of $\mathcal{N}^\sigma(u)$ is σ -consistent. Since $\mathcal{N}_1^{\sigma_1}(u)$ is an orchard, it has a complete acyclic cherry partition and, as just argued, this partition is σ -consistent. Conversely, assume that $\sigma(P) = 0$ for some $P \in R(\mathcal{N}^\sigma(u))$. Then P contains a reticulation v . Let P' be the other reticulated cherry path in $R(\mathcal{N}^\sigma(u))$ that contains v . Then $\sigma(P') \neq \perp$ because σ is blob-respecting, and $\sigma(P') \neq 1$ because, as argued above, \mathcal{P} contains all reticulated cherry paths $P \in R(\mathcal{N})$ that satisfy $\sigma(P) = 1$. Thus, $\sigma(P') = 0$. Thus, there is no σ -consistent cherry partition of $\mathcal{N}^\sigma(u)$ that covers v or its child arc, that is, no σ -consistent cherry partition of $\mathcal{N}^\sigma(u)$ is complete.

As for the cost of computing $D_1[u, v, \sigma_1, \sigma_2]$, we have $D_1[u, v, \sigma_1, \sigma_2] = 1$ if and only if $X(\mathcal{N}_1^{\sigma_1}(u)) \cap X(\mathcal{N}_2^{\sigma_2}(v)) \neq \emptyset$ and $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$ have complete cherry partitions that are σ_1 -consistent and σ_2 -consistent, respectively. The intersection property is checked in linear time and, by the previous arguments, we can check for complete cherry covers of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$ in linear time. \square

Proof of Lemma 5. If B_1 is a trivial blob, then it can be isomorphic to B_2 only if B_2 is also trivial. In this case, the lemma holds because there are exactly 2 isomorphisms between B_1 and B_2 corresponding to the two possible mappings of the leaves of B_1 onto the leaves of B_2 . These two isomorphisms can be found in constant time. For the rest of this proof, therefore, assume that B_1 and B_2 are both non-trivial. For $i \in \{1, 2\}$, let B'_i be the 2ECC obtained by deleting the leaves of B_i and their incident arcs, and let the *generator* G_i of B_i be the graph obtained from B'_i by suppressing all vertices with in-degree 1 and out-degree 1, i.e. we delete such a vertex and all its incident edges, adding a new edge between its former neighbours. Note that each arc e of G_i corresponds to a path $P_i(e)$ in B'_i whose internal vertices have one parent and one child, and G_i may have parallel arcs. Moreover, B_i and G_i have the same number of reticulation.

Restricting any isomorphism $\phi : B_1 \rightarrow B_2$ to the vertices of G_1 uniquely defines a corresponding isomorphism $\psi : V(G_1) \rightarrow V(G_2)$. Similarly, for every generator arc $e = (u, v) \in E(G_1)$, ϕ must map the path $P_1(e)$ onto $P_2(e')$ in B_2 , where $e' = (u', v')$ is an arc of G_2 with $u' = \phi(u)$ and $v' = \phi(v)$. Thus, by setting $\psi(e) = e'$ for every arc of G_1 , we obtain a unique isomorphism $\psi : G_1 \rightarrow G_2$. Conversely, for every isomorphism $\psi : G_1 \rightarrow G_2$, there exists at most one corresponding isomorphism $\phi : B'_1 \rightarrow B'_2$ defined by setting $\phi(v) = \psi(v)$, for every vertex $v \in V(G)$ and $\phi(P_1(e)) = P_2(\psi(e))$, for every arc $e \in E(G)$. This is an isomorphism exactly if $P_1(e)$ and $P_2(\psi(e))$ have the same length, for every arc $e \in E(G)$. This isomorphism ϕ has a natural extension to a unique isomorphism $B_1 \rightarrow B_2$, as no vertex in a non-trivial blob B_1 has more than one child that is a leaf and, therefore, the mapping of each leaf of B_1 is uniquely determined by the mapping of its parent in B'_1 . This shows that there exists a 1-to-1 correspondence between isomorphisms $B_1 \rightarrow B_2$ and isomorphisms between their generators. Moreover, the construction of an isomorphism $\phi : B_1 \rightarrow B_2$ from an isomorphism $\psi : G_1 \rightarrow G_2$ just described is easily carried out in linear time, as is the construction of G_1 and G_2 from B_1 and B_2 . Thus, it suffices to argue that there are at most $2^{3r/2}$ isomorphisms $\psi : G_1 \rightarrow G_2$ and that they can be enumerated in $O(2^{3r/2} \cdot |G_1|)$ time.

Consider a rooted spanning tree T_1 of G_1 and a topological ordering $\sigma = \langle v_1, \dots, v_n \rangle$ of its vertices. In other words, we have $i < j$ for every arc (v_i, v_j) in T_1 . For every isomorphism $\psi : G_1 \rightarrow G_2$, the restriction of $\psi|_{T_1}$ is an isomorphism between T_1 and a rooted spanning tree T_2 of G_2 . Since all vertices of G_1 are vertices of T_1 , $\psi|_{T_1}$ fully defines the action of ψ on the vertices of G_1 . In fact, $\psi|_{T_1}$ also fully defines the action of ψ on the arcs of G_1 . Clearly, it defines the action of ψ on the arcs of T_1 . For every arc $e = (u, v) \in E(G_1) \setminus E(T_1)$, $\psi(e)$ must be an arc with endpoints $\psi(u)$ and $\psi(v)$. If there is only one such arc in G_2 , then this arc is fully defined by these two endpoints. If there are two such arcs (because a generator can have parallel arcs, but a generator of a binary network cannot have more than two parallel arcs between two vertices), then, since ψ is an isomorphism, there are also two arcs with endpoints u and v in G_1 . This makes u the only parent of v in G_1 (because G_1 is binary), which implies that one of the arcs with endpoints u and v is in T_1 . Its image is in T_2 . The image of the arc $(u, v) \in E(G_1) \setminus E(T_1)$ is thus the unique arc $(\phi(u), \phi(v)) \in E(G_2) \setminus E(T_2)$. Thus, the image of (u, v) is once again uniquely defined by $\psi|_{T_1}$.

By the argument in the previous paragraph, we can use the following strategy for enumerating all isomorphisms from G_1 to G_2 : Conceptually, we enumerate all isomorphisms between T_1 and spanning trees of G_2 and then discard those that do not extend to isomorphisms between G_1 and G_2 . However, we will construct these isomorphisms between T_1 and spanning trees of G_2 incrementally, using a recursive search. This makes it more efficient to abort a branch in the search as soon as the choices made so far guarantee that no isomorphism between T_1 and a spanning tree of G_2 that includes these choices can be extended to an isomorphism between G_1 and G_2 . In particular, we inspect the vertices of G_1 in the order σ . For all $i \in [n]$, let $V_i = \{v_1, \dots, v_i\}$. Then, at every point in the recursive search, we will have constructed an isomorphism $\psi|_{V_i}$ between the subtree $T_1|_{V_i}$ and some subtree of G_2 whose root is the root of G_2 . We call such an isomorphism *valid* if

- Every vertex $v_j \in V_i$ has the same type in G_1 as $\psi(v_j)$ has in G_2 , where the type of a vertex is “tree vertex” or “reticulation” and
- For every pair of vertices $v_h, v_j \in V_i$, there exists an arc $(v_h, v_j) \in E(G_1) \setminus E(T_1)$ if and only if the arc $(\psi(v_h), \psi(v_j))$ exists in $E(G_2) \setminus E(\psi(T_1|_{V_i}))$.

We make recursive calls only on valid isomorphisms $\psi|_{V_i}$ because an invalid isomorphism clearly cannot be extended to an isomorphism ψ between G_1 and G_2 . Conversely, by the argument in the 3rd paragraph of this proof, if $\psi|_{V_n}$ is valid, then it has a unique extension to an isomorphism of G_1 and G_2 , and it is not hard to verify that this extension can be found in linear time.

There is exactly one valid isomorphism $\psi|_{V_1}$, which maps v_1 to the root of G_2 . This is the input to the top-level invocation of the search. Given a valid isomorphism $\psi|_{V_i}$, if $i = n$, then we complete $\psi|_{V_n}$ to an isomorphism between G_1 and G_2 as discussed in the fifth paragraph of this proof. If $i < n$, then we identify all valid isomorphisms $\psi|_{V_{i+1}}$ such $\psi|_{V_i}$ is the restriction of $\psi|_{V_{i+1}}$ to $T_1|_{V_i}$ and make one recursive call on each of them. In particular, let v_j be the parent of v_{i+1} in T_1 , and let e be the arc connecting them. Then $\psi(v_{i+1})$ must be

a child of $\psi(v_j)$ not in $\psi(T_1|_{V_i})$. We inspect each child arc $e' = (\psi(v_j), z)$ of $\psi(v_j)$, and set $\psi(v_{i+1}) = z$ and $\psi(e) = e'$ for each such arc that satisfies $z \notin \psi(T_1|_{V_i})$. This defines a (not necessarily valid) isomorphism between $T_1|_{V_{i+1}}$ and a subtree of G_2 . Next we check that v_{i+1} and $\psi(v_{i+1})$ have the same type and that, if v_{i+1} has an incident arc (v_h, v_{i+1}) or (v_{i+1}, v_h) in $E(G_1) \setminus E(T_1|_{V_{i+1}})$ with $v_h \in V_i$, then there exists the corresponding arc $(\psi(v_h), \psi(v_{i+1}))$ or $(\psi(v_{i+1}), \psi(v_h))$ in $E(G_2) \setminus E(\psi(T_1|_{V_{i+1}}))$. If $\psi|_{V_{i+1}}$ passes both of these tests, then it is valid, so we recurse on it. Otherwise, it is invalid, and we discard it.

The discussion so far shows that this procedure enumerates all isomorphisms between G_1 and G_2 . Now we bound its running time and the number of isomorphisms it finds. It is easily seen that the cost per recursive call is constant, as it only needs to find the parent v_j of v_{i+1} , and then it needs to inspect the neighbours of v_j , v_{j+1} , $\psi(v_j)$, and $\psi(v_{i+1})$, of which there are only a constant number. The depth of the recursive search is $O(|G_1|)$. Thus, to bound both the number of isomorphisms found by $2^{3r/2}$ and the time to find them by $O(2^{3r/2} \cdot |G_1|)$, it suffices to prove that there are at most $2^{3r/2}$ leaves in this search tree. To this end, we call a valid isomorphism $\psi|_{V_i}$ a *choice point* if the parent v_j of v_{i+1} in T_1 is a tree vertex in G_1 and neither of its children belongs to V_i . If $\psi|_{V_i}$ is not a choice point, then it is easily seen to have at most one valid extension $\phi|_{V_{i+1}}$. If $\psi|_{V_i}$ is a choice point, then it has at most two valid extensions corresponding to the two possible mappings of v_{i+1} to one of the children of $\psi(v_j)$. Thus, to prove that there are at most $2^{3r/2}$ leaves in the recursion tree, it suffices to prove that any root-to-leaf path in this recursion tree contains at most $3r/2$ choice points.

As an isomorphism $\psi|_{V_i}$ can be a choice point only if v_j is a tree vertex in G_1 , the number of tree vertices t in G_1 is an upper bound on the number of choice points on any root-to-leaf path of the recursion tree. Let r_0 be the number of reticulations without children, let r_1 be the number of reticulations with one child and with two distinct parents, and let r_2 be the number of reticulations that are the bottom endpoints of parallel arcs. Since B_1 is a blob, G_1 is 2-edge-connected, which implies that the bottom endpoint of every pair of parallel arcs must be a reticulation that has a child in G_1 . Let t_1 be the number of tree vertices that are not top endpoints of parallel arcs, and let t_2 be the number of tree vertices that are top endpoints of parallel arcs. Then $r = r_0 + r_1 + r_2$, $t = t_1 + t_2$, $r_2 = t_2$, and $2(t_1 + t_2) + r_1 + r_2 = t_1 + t_2 - 1 + 2(r_0 + r_1 + r_2)$ because the total number of in-arcs of all vertices must equal the total number of their out-arcs. Thus, since $r_2 = t_2$, we have $t_1 + 1 = 2r_0 + r_1$ and, therefore, $t + 1 = t_1 + t_2 + 1 = 2r_0 + r_1 + r_2 = r + r_0 \leq 2r$. This shows that there are at most $2r$ choice points on any root-to-leaf path in the recursion tree, i.e., the number of isomorphisms $< 4^r$ and they can be enumerated in $O(4^r \cdot |G_1|)$ time.

To obtain a better bound, we need to choose the spanning tree T_1 and the topological ordering σ more carefully. Consider an arbitrary reticulation v_i in G_1 with parents v_j and v_h . Then one of the arcs (v_j, v_i) and (v_h, v_i) belongs to T_1 , the other does not. Assume that $(v_j, v_i) \notin T_1$ and that v_j has a child v_k in T_1 . If $j > i$, then $k > i$ because $j < k$. Thus, $v_i \in V_{k-1}$ and $\psi|_{V_{k-1}}$ is not a choice point. Our goal now is to construct T_1 and σ so that there are at least $(r_0 - r_2)/2$

tree vertices v_j in G_1 that have an incident arc $(v_j, v_i) \notin T_1$ that satisfies $v_i < v_j$. These tree vertices cannot be choice points, so the number of choice points on any path in the recursion tree is at most $t - (r_0 - r_2)/2 < r + (r_0 + r_2)/2 < 3r/2$.

To construct T_1 and $\sigma = \langle v_1, \dots, v_n \rangle$, we start by adding the root ρ of G_1 to T_1 and setting $\sigma = \langle \rho \rangle$. Since G_1 is a DAG, there always exists a vertex v that has all its parents in T_1 . If v has a single parent (which includes the case when v is the bottom endpoint of two parallel arcs) or at least one of the parents of v is a reticulation, then we choose one of the parents of v — call it u — and make v a child of u in T_1 . We add v to the end of σ to ensure that σ remains a topological ordering of the tree constructed so far. If v is a reticulation whose parents u_1 and u_2 in G_1 are both tree vertices, then recall that $u_1, u_2 \in T_1$ at the time we add v to T_1 . Assume that u_1 precedes u_2 in σ . Then we make v a child of u_1 and insert v into σ immediately after u_1 . This ensures both that σ remains a topological ordering of T_1 , as v succeeds its parent, and that the non-tree arc (u_2, v) has the property we desire: that v precedes u_2 in σ . Thus, whether u_2 ends up having a child in T_1 or not, it does not correspond to a choice point.

Every reticulation with two tree vertices as parents eliminates one choice point, but if there is a tree vertex u whose children v_1 and v_2 are both reticulations, then v_1 and v_2 may both eliminate u as a tree vertex corresponding to a choice point. However, every tree vertex can be eliminated only by its children, that is, every tree vertex is eliminated at most twice. Therefore, the number of choice points eliminated by our construction is at least half the number of reticulations in G_1 that have two distinct tree vertices as parents. The number of parent arcs of reticulations with two distinct parents is $2(r_0 + r_1)$. At most $r_1 + r_2$ of them have reticulations as top endpoints. This leaves at least $2r_0 + r_1 - r_2$ arcs from tree vertices to reticulations with distinct parents. Since there are only $r_0 + r_1$ such reticulations, this implies that there must be at least $r_0 - r_2$ reticulations with two distinct tree vertices as parents. Thus, the chosen tree T_1 and topological ordering ensures that there are at most $t - (r_0 - r_2)/2 < 3r/2$ choice points along any path in the recursion tree. The number of isomorphisms from G_1 to G_2 is thus at most $2^{3r/2}$, and they can be found in $O(2^{3r/2} \cdot |G_1|)$ time. \square

D Proofs for Section 4 (Correctness of the Algorithm)

Proof of Lemma 9. Let \mathcal{P} be any σ -consistent acyclic cherry partition of $\mathcal{N}^\sigma(v)$. Suppose for now that B_v^σ is trivial. If B_v^σ has a single leaf, then that leaf is v , $\mathcal{P} = \emptyset$, and so $\mathcal{N}^\sigma(v) - \mathcal{P} = \mathcal{N}^\sigma(v)$ and has v as its only vertex. Otherwise, the root v of B_v^σ has two children u_1 and u_2 . If neither (v, u_1) nor (v, u_2) is in a path in \mathcal{P} , then $\mathcal{N}^\sigma(v) - \mathcal{P}$ contains both arcs and, thus, its root blob is B_v^σ , as claimed. If some path $P \in \mathcal{P}$ contains one of the two arcs (v, u_1) and (v, u_2) , it must also contain the other, as v is a tree vertex. Since every arc in $\mathcal{N}^\sigma(v)$ is reachable from v , any cherry partition \mathcal{P} of $\mathcal{N}^\sigma(v)$ that contains P must cover all arcs of $\mathcal{N}^\sigma(v)$, that is $\mathcal{N}^\sigma(v) - \mathcal{P}$ has v as its only vertex.

So assume from now on that B_v^σ is non-trivial. Then note that $R(B_v^\sigma) \subseteq R(B_v)$ because removing arcs from \mathcal{N} does not merge 2ECCs, that is, $B_v^\sigma \subseteq B_v$. Thus, since σ fixes all reticulated cherry paths in B_v , we have $\sigma(P) \in \{0, 1\}$, for all $P \in R(B_v^\sigma)$. Since \mathcal{N}^σ is a σ -CRS of \mathcal{N} , there exists a σ -consistent acyclic

cherry partition \mathcal{P}' of \mathcal{N} such that $\mathcal{N}^\sigma = \mathcal{N} - \mathcal{P}'$. In particular, \mathcal{P}' contains all paths $P' \in R(\mathcal{N})$ with $\sigma(P') = 1$, so $R(B_v^\sigma) \subseteq R(\mathcal{N} - \mathcal{P}')$ cannot contain any such path. This shows that $\sigma(P) = 0$ for every reticulated cherry path in $R(B_v^\sigma)$.

Now consider \mathcal{P} again. If no path in \mathcal{P} contains an arc of B_v^σ , then B_v^σ is the root blob of $\mathcal{N}^\sigma(v) - \mathcal{P}$, as claimed. So assume for the sake of contradiction that some path $P \in \mathcal{P}$ contains an arc of B_v^σ . Note that P cannot be a reticulated cherry path, as $\sigma(P') = 0$ for all $P' \in R(B_v^\sigma)$, and \mathcal{P} is σ -consistent. Thus, P is a proper cherry path $\langle x, y, z \rangle$. This implies that y and, w.l.o.g., x are non-leaf vertices of B_v^σ because no vertex in a non-trivial binary blob can have two leaves of the blob as children. Because removing the leaves of B_v^σ results in a 2ECC, there must be an undirected cycle $C = \langle v_0, \dots, v_t \rangle$ in B_v^σ with $v_0 = v_t = y$ and $v_1 = x$. As B_v^σ is acyclic, there exists an index i such that $\langle v_0, \dots, v_i \rangle$ is a directed path from y to v_i and v_i is a reticulation with v_{i-1} and v_{i+1} as its parents. Since P contains (y, x) , $P \in \mathcal{P}$, and \mathcal{P} is an acyclic cherry partition, every arc along this path must be covered by paths in \mathcal{P} , including (v_{i-1}, v_i) . This implies that some path $P' \in \mathcal{P}$ must cover the child arc of v_i . Since v_i is a reticulation, this path must be reticulated cherry path. Since $\sigma(P') = 0$ for any such path, this contradicts the σ -consistency of \mathcal{P} . \square

Proof of Lemma 11. Let $\mathcal{L}(B_u^{\sigma_1}) = \{x_1, \dots, x_t\}$ and, for all $i \in [t]$, let \mathcal{P}_i be a σ_1 -consistent acyclic cherry partition of $\mathcal{N}^{\sigma_1}(x_i)$ such that $\mathcal{N}^{\sigma_1}(x) = \mathcal{N}^*(x)$. These cherry partitions exist because $\mathcal{N}^*(x_i)$ is a σ_1 -CRS of $\mathcal{N}^{\sigma_1}(x_i)$, for all $i \in [t]$. As each of the subnetworks $\mathcal{N}_1^{\sigma_1}(x_1), \dots, \mathcal{N}_1^{\sigma_1}(x_t)$ is rooted in a distinct leaf of $B_u^{\sigma_1}$, no two of these subnetworks have a vertex in common. Thus, $\mathcal{P} := \mathcal{P}_1 \cup \dots \cup \mathcal{P}_t$ is an acyclic cherry partition of $\mathcal{N}_1^{\sigma_1}(u)$. Our definition of \mathcal{N}^* implies that $\mathcal{N}^* = \mathcal{N}^{\sigma_1}(u) - \mathcal{P}$, so \mathcal{N}^* is a CRS of $\mathcal{N}^{\sigma_1}(u)$. We also need to prove that \mathcal{P} is σ_1 -consistent to show that \mathcal{N}^* is a σ_1 -CRS of \mathcal{N}^{σ_1} . As in the proof of Lemma 9, we know that $\sigma_1(P) \in \{0, \perp\}$, for every $P \in R(\mathcal{N}_1^{\sigma_1}(u))$. Thus, it is sufficient to argue that \mathcal{P} contains no path $P \in R(\mathcal{N}_1^{\sigma_1}(u))$ with $\sigma(P) = 0$. This, however, follows from the σ_1 -consistency of $\mathcal{P}_1, \dots, \mathcal{P}_t$, as $\mathcal{P} = \mathcal{P}_1 \cup \dots \cup \mathcal{P}_t$.

Since $\mathcal{N}^*(x_i)$ is a σ_2 -CRS of $\mathcal{N}_2^{\sigma_2}(\phi(x_i))$, for all $i \in [t]$, there exist a network $\mathcal{N}^{**}(\phi(x_i))$, an isomorphism $\psi_i : \mathcal{N}^*(x_i) \rightarrow \mathcal{N}^{**}(\phi(x_i))$, and a σ_2 -consistent acyclic cherry partition \mathcal{P}'_i of $\mathcal{N}_2^{\sigma_2}(\phi(x_i))$ such that $\mathcal{N}_2^{\sigma_2}(\phi(x_i)) - \mathcal{P}'_i = \mathcal{N}^{**}(\phi(x_i))$. By the same argument as above, $\mathcal{P}' := \mathcal{P}'_1 \cup \dots \cup \mathcal{P}'_t$ is a σ_2 -consistent acyclic cherry partition of $\mathcal{N}_2^{\sigma_2}(v)$, and $\mathcal{N}^{**} := B_v^{\sigma_2} \cup \bigcup_{x \in \mathcal{L}(B_u^{\sigma_1})} \mathcal{N}^{**}(\phi(x_i))$ is a σ_2 -CRS of $\mathcal{N}_2^{\sigma_2}(v)$, as $\mathcal{N}^{**} = \mathcal{N}_2^{\sigma_2}(v) - \mathcal{P}'$. Since $\phi, \psi_1, \dots, \psi_t$ are isomorphisms that agree on the mappings of the vertices in $\mathcal{L}(B_u^{\sigma_1})$, $B_u^{\sigma_1}, \mathcal{N}^{\sigma_1}(x_1), \dots, \mathcal{N}^{\sigma_1}(x_t)$ share no other vertices nor any arcs, and the same is true for $B_v^{\sigma_2}, \mathcal{N}_2^{\sigma_2}(\phi(x_1)), \dots, \mathcal{N}_2^{\sigma_2}(\phi(x_t))$, the mapping $\psi : \mathcal{N}^* \rightarrow \mathcal{N}^{**}$ that maps vertices and arcs according to $\phi, \psi_1, \dots, \psi_t$ is an isomorphism. Thus, since \mathcal{N}^* is a σ_1 -CRS of $\mathcal{N}_1^{\sigma_1}(u)$ and \mathcal{N}^{**} is a σ_2 -CRS of $\mathcal{N}_2^{\sigma_2}(v)$, \mathcal{N}^* is a (σ_1, σ_2) -ACRS of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$. \square

Proof of Lemma 12. We prove that every σ_1 -CRS of $\mathcal{N}_1^{\sigma_1}(u)$ is a σ'_1 -CRS of $\mathcal{N}_1^{\sigma'_1}(u)$. An analogous argument shows that every σ'_1 -CRS of $\mathcal{N}_1^{\sigma'_1}(u)$ is a σ_1 -CRS of $\mathcal{N}_1^{\sigma_1}(u)$, that every σ_2 -CRS of $\mathcal{N}_2^{\sigma_2}(v)$ is a σ'_2 -CRS of $\mathcal{N}_2^{\sigma'_2}(v)$, and that every σ'_2 -CRS of $\mathcal{N}_2^{\sigma'_2}(v)$ is a σ_2 -CRS of $\mathcal{N}_2^{\sigma_2}(v)$. This implies that every (σ_1, σ_2) -ACRS of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$ is a (σ'_1, σ'_2) -ACRS of $\mathcal{N}_1^{\sigma'_1}(u)$ and $\mathcal{N}_2^{\sigma'_2}(v)$,

and vice versa. This in turn implies that every (σ_1, σ_2) -MACRS of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_2^{\sigma_2}(v)$ is a (σ'_1, σ'_2) -MACRS of $\mathcal{N}_1^{\sigma'_1}(u)$ and $\mathcal{N}_2^{\sigma'_2}(v)$, and vice versa, that is, that $D^*[u, v, \sigma_1, \sigma_2] = D^*[u, v, \sigma'_1, \sigma'_2]$.

So let \mathcal{N} be a σ_1 -CRS of $\mathcal{N}_1^{\sigma_1}(u)$, let \mathcal{P}_1 be a σ_1 -consistent acyclic cherry partition of \mathcal{N}_1 such that $\mathcal{N}_1^{\sigma_1} = \mathcal{N}_1 - \mathcal{P}_1$, let \mathcal{P}_2 be a σ'_1 -consistent acyclic cherry partition of \mathcal{N}_1 such that $\mathcal{N}_1^{\sigma'_1} = \mathcal{N}_1 - \mathcal{P}_2$, and let \mathcal{P}_3 be a σ_1 -consistent acyclic cherry partition of $\mathcal{N}_1^{\sigma_1}(u)$ such that $\mathcal{N} = \mathcal{N}_1^{\sigma_1}(u) - \mathcal{P}_3$. Next, let $\mathcal{P}'_1 = \mathcal{P}_1 \cap C(\mathcal{N}_1(r_u))$, $\mathcal{P}'_2 = \mathcal{P}_2 \setminus C(\mathcal{N}_1(r_u))$, and $\mathcal{P} = \mathcal{P}'_1 \cup \mathcal{P}'_2 \cup \mathcal{P}_3$. We prove that

- (i) \mathcal{P} is a σ'_1 -consistent partial cherry partition of \mathcal{N}_1 and
- (ii) \mathcal{N} is a pendant subnetwork of $\mathcal{N}' := \mathcal{N}_1 - \mathcal{P}$, that is, u is the bottom endpoint of a bridge in \mathcal{N}' , and $\mathcal{N}'(u) = \mathcal{N}$.

By the definition of $\mathcal{N}_1^{\sigma'_1}$, (i) implies that there exists a σ'_1 -consistent partial cherry partition \mathcal{P}' of $\mathcal{N}_1^{\sigma'_1}$ such that $\mathcal{N}' = \mathcal{N}_1^{\sigma'_1} - \mathcal{P}'$. Since σ'_1 exposes u , u is the bottom endpoint of a bridge in $\mathcal{N}_1^{\sigma'_1}$. Thus, by Lemma 2, $\mathcal{P}'' := \mathcal{P}' \cap C(\mathcal{N}_1^{\sigma'_1}(u))$ is a partial cherry partition of $\mathcal{N}_1^{\sigma'_1}(u)$ and $\mathcal{N}_1^{\sigma'_1}(u) - \mathcal{P}'' = \mathcal{N}'(u)$. Since $\mathcal{P}'' \subseteq \mathcal{P}'$ and \mathcal{P}' is σ'_1 -consistent, so is \mathcal{P}'' . This makes $\mathcal{N}'(u)$ a σ'_1 -CRS of $\mathcal{N}_1^{\sigma'_1}(u)$. By (ii), $\mathcal{N}'(u) = \mathcal{N}$, that is, \mathcal{N} is a σ'_1 -CRS of $\mathcal{N}_1^{\sigma'_1}(u)$, as claimed.

To prove (i), we need to show that \mathcal{P} is an acyclic cherry partition of \mathcal{N}_1 and that it is σ'_1 -consistent. To prove that \mathcal{P} is an acyclic cherry partition of \mathcal{N}_1 , we need to prove that the paths in \mathcal{P} are arc-disjoint, that every endpoint of a path in \mathcal{P} is either a leaf of \mathcal{N}_1 or an internal vertex of another path in \mathcal{P} , and that $G_{\mathcal{P}}$ is acyclic. To see that the paths in \mathcal{P} are arc-disjoint, observe that no path $P_1 \in \mathcal{P}'_1 \cup \mathcal{P}_3$ can share an arc with a path $P_2 \in \mathcal{P}'_2$ because $E(P_1) \subseteq E(\mathcal{N}_1(r_u))$ and $E(P_2) \cap E(\mathcal{N}_1(r_u)) = \emptyset$. No path $P_1 \in \mathcal{P}_3$ shares an arc with a path $P_2 \in \mathcal{P}'_1$ because $\mathcal{P}'_1 \subseteq \mathcal{P}_1$, $\mathcal{N}_1^{\sigma_1} = \mathcal{N}_1 - \mathcal{P}_1$, and \mathcal{P}_3 is an acyclic cherry partition of $\mathcal{N}_1^{\sigma_1}(u) \subseteq \mathcal{N}_1^{\sigma_1}$. Thus, all paths in \mathcal{P} are arc-disjoint.

Next consider an endpoint v of a path $P \in \mathcal{P}$. If $P \in \mathcal{P}'_1$, then note that, by Lemma 2, \mathcal{P}'_1 is an acyclic cherry partition of $\mathcal{N}_1(r_u)$. Thus, v is either a leaf of \mathcal{N}_1 or internal to another path in $\mathcal{P}'_1 \subseteq \mathcal{P}$. If $P \in \mathcal{P}_3$, then v is either a leaf of $\mathcal{N}_1^{\sigma_1}(u)$ or an internal vertex of a path in $\mathcal{P}_3 \subseteq \mathcal{P}$. If v is a leaf of $\mathcal{N}_1^{\sigma_1}(u)$, then it is either a leaf of \mathcal{N}_1 or internal to a path $P' \in \mathcal{P}_1$ because $\mathcal{N}_1^{\sigma_1} = \mathcal{N}_1 - \mathcal{P}_1$. Since $v \in \mathcal{N}_1^{\sigma_1}(u)$ and $u \in B_u$, this path P' belongs to $C(\mathcal{N}_1(r_u))$ and, thus, to $\mathcal{P}_1 \cap C(\mathcal{N}_1(r_u)) = \mathcal{P}'_1 \subseteq \mathcal{P}$. Finally, assume that $P \in \mathcal{P}'_2$. Since $\mathcal{P}'_2 \subseteq \mathcal{P}_2$ and \mathcal{P}_2 is a partial cherry partition of \mathcal{N}_1 , v is either a leaf of \mathcal{N}_1 or an internal vertex of a path $P' \in \mathcal{P}_2$. If $P' \in \mathcal{P}'_2$, then $P' \in \mathcal{P}$. Otherwise, we must have $v = r_u$ because r_u being the bottom endpoint of a bridge in \mathcal{N}_1 implies that it is the only vertex that can belong to cherry paths in $\mathcal{P}'_2 = \mathcal{P}_2 \setminus C(\mathcal{N}_1(r_u))$ and in $\mathcal{P}_2 \cap C(\mathcal{N}_1(r_u))$. However, since $P \in \mathcal{P}_2$ and \mathcal{P}_2 is a cherry partition of \mathcal{N}_1 , this implies that $\mathcal{N}_1^{\sigma'_1} = \mathcal{N}_1 - \mathcal{P}_2$ cannot contain any vertex in $\mathcal{N}_1(r_u)$, a contradiction because $u \in \mathcal{N}_1(r_u)$ and σ'_1 exposes u . Thus, the case when $P' \in \mathcal{P}_2 \setminus \mathcal{P}'_2$ cannot arise. This finishes the proof that every endpoint of a path in \mathcal{P} is a leaf of \mathcal{N}_1 or an internal vertex of another path in \mathcal{P} .

To see that $G_{\mathcal{P}}$ is acyclic, note that there are no cycles involving paths from only one of \mathcal{P}'_1 , \mathcal{P}'_2 , and \mathcal{P}_3 because $\mathcal{P}_1 \supseteq \mathcal{P}'_1$, $\mathcal{P}_2 \supseteq \mathcal{P}'_2$, and \mathcal{P}_3 are acyclic cherry partitions. There is no cycle involving paths in $\mathcal{P}'_1 \cup \mathcal{P}_3$ and in \mathcal{P}'_2 because no

endpoint of a path in $C(\mathcal{N}_1(r_u)) \supseteq \mathcal{P}'_1 \cup \mathcal{P}_3$ is an internal vertex of any path in $C(\mathcal{N}_1) \setminus C(\mathcal{N}_1(r_u)) \supseteq \mathcal{P}'_2$. Finally, there is no cycle involving cherry in \mathcal{P}'_1 and \mathcal{P}_3 but not in \mathcal{P}'_2 because, for every endpoint v of a path in \mathcal{P}'_1 , all descendant arcs of v are covered by paths in \mathcal{P}_1 and, thus, cannot belong to any path in \mathcal{P}_3 because \mathcal{P}'_1 and \mathcal{P}_3 are arc-disjoint. Thus, $G_{\mathcal{P}}$ is acyclic and \mathcal{P} is an acyclic cherry partition of \mathcal{N}_1 .

To finish the proof of (i), we need to prove that \mathcal{P} is σ'_1 -consistent. Since \mathcal{P}_1 and \mathcal{P}_3 are σ_1 -consistent and $\mathcal{P}'_1 \subseteq \mathcal{P}_1$, \mathcal{P}'_1 and \mathcal{P}_3 do not contain any path $P \in R(\mathcal{N}_1)$ with $\sigma_1(P) = 0$. Since $\mathcal{P}'_1 \cup \mathcal{P}_3 \subseteq C(\mathcal{N}_1(r_u))$ and σ_1 and σ'_1 agree on $R(\mathcal{N}_1(r_u))$, this shows that $\mathcal{P}'_1 \cup \mathcal{P}_3$ does not contain any path $P \in R(\mathcal{N}_1)$ with $\sigma'_1(P) = 0$. The same is true for \mathcal{P}'_2 because $\mathcal{P}'_2 \subseteq \mathcal{P}_2$ and \mathcal{P}_2 is σ'_1 -consistent. Thus, \mathcal{P} does not contain any path $P \in R(\mathcal{N}_1)$ with $\sigma'_1(P) = 0$. Since \mathcal{P}_2 is σ'_1 -consistent, it contains every path $P \in R(\mathcal{N}_1)$ with $\sigma'_1(P) = 1$. If this path belongs to $R(\mathcal{N}_1) \setminus R(\mathcal{N}_1(r_u))$, then it belongs to $\mathcal{P}'_2 \subseteq \mathcal{P}$. If it belongs to $R(\mathcal{N}_1(r_u))$, then $\sigma_1(P) = \sigma'_1(P) = 1$. Thus, as \mathcal{P}_1 is σ_1 -consistent, it contains P , as does $\mathcal{P}'_1 = \mathcal{P}_1 \cap C(\mathcal{N}_1(r_u))$. Thus, \mathcal{P} contains all paths in $R(\mathcal{N}_1)$ with $\sigma'_1(P) = 0$. Since it does not contain any paths $P \in R(\mathcal{N}_1)$ with $\sigma'_1(P) = 0$, this proves that \mathcal{P} is σ'_1 -consistent.

To prove (ii), observe that our proof of (i) shows in fact that $\mathcal{P}'_1 \cup \mathcal{P}'_2$ is an acyclic cherry partition of \mathcal{N}_1 . Since \mathcal{P}_3 is an acyclic cherry partition of $\mathcal{N}_1^{\sigma_1}(u)$ and $\mathcal{N}_1^{\sigma_1}(u) \setminus \mathcal{P}_3 = \mathcal{N}$, it suffices to prove that $\mathcal{N}_1^{\sigma_1}(u)$ is a pendant subnetwork of $\mathcal{N}_1 - (\mathcal{P}'_1 \cup \mathcal{P}'_2)$. Since σ_1 exposes u , $\mathcal{N}_1^{\sigma_1}(u)$ is a pendant subnetwork of $\mathcal{N}_1^{\sigma_1} = \mathcal{N}_1 - \mathcal{P}_1$. By Lemma 2, we have $\mathcal{N}_1(r_u) - \mathcal{P}'_1 = \mathcal{N}_1^{\sigma_1}(r_u)$, that is, $\mathcal{N}_1^{\sigma_1}(u)$ is a pendant subnetwork of $\mathcal{N}_1(r_u) - \mathcal{P}'_1$. Since $\mathcal{P}'_2 \cap C(\mathcal{N}_1(r_u)) = \emptyset$ and $\mathcal{N}_1(r_u)$ is a pendant subnetwork of \mathcal{N}_1 , this implies that $\mathcal{N}_1^{\sigma_1}(u)$ is a pendant subnetwork of $\mathcal{N}_1 - (\mathcal{P}'_1 \cup \mathcal{P}'_2)$, as required. This proves (ii) and, thus, finishes the proof. \square

Proof of Lemma 13. We use induction on the length of any path from $D^*[\rho_1, \rho_2, \perp, \perp]$ to $D^*[u, v, \sigma_1, \sigma_2]$ in the dependency graph to prove the lemma. For $D^*[\rho_1, \rho_2, \perp, \perp]$, the lemma clearly holds. For the inductive step, we consider a required entry $D^*[u, v, \sigma_1, \sigma_2]$ that satisfies the lemma, and prove that the lemma holds also for all its out-neighbours in the dependency graph.

So let $D^*[x, y, \sigma'_1, \sigma'_2]$ be such an out-neighbour. If $D^*[u, v, \sigma_1, \sigma_2]$ depends on it via eq. (1), then $x = u$ and σ'_1 is a B_u -extension of σ_1 . Thus, σ'_1 fixes all cherry paths in $R(B_u)$ and, since σ_1 does not fix any cherry paths in proper descendant blobs of B_u , neither does σ'_1 . By an analogous argument, $y = v$ and σ'_2 fixes all cherry paths in $R(B_v)$ and none of the cherry paths in proper descendant blobs of B_v . If $D^*[u, v, \sigma_1, \sigma_2]$ depends on $D^*[x, y, \sigma'_1, \sigma'_2]$ via eqs. (2) and (5), then $\sigma'_1 = \sigma_1$ and B_x is a descendant blob of B_u . Since σ_1 fixes all or no cherry paths in B_u and no cherry paths in any proper descendant blob of B_u , this implies that σ'_1 fixes all or no cherry paths in B_x and no cherry paths in any proper descendant blob of B_x . By an analogous argument, σ'_2 fixes all or no cherry paths in B_y and no cherry paths in any proper descendant blob of B_y . This finishes the inductive step and, thus, the proof of the lemma. \square