TRINETS ENCODE ORCHARD PHYLOGENETIC NETWORKS

CHARLES SEMPLE AND GERRY TOFT

Abstract. Rooted triples, rooted binary phylogenetic trees on three leaves, are sufficient to encode rooted binary phylogenetic trees. That is, if $\mathcal{T}$ and $\mathcal{T}'$ are rooted binary phylogenetic X-trees that infer the same set of rooted triples, then $\mathcal{T}$ and $\mathcal{T}'$ are isomorphic. However, in general, this sufficiency does not extend to rooted binary phylogenetic networks. In this paper, we show that trinets, phylogenetic network analogues of rooted triples, are sufficient to encode rooted binary orchard networks. Rooted binary orchard networks naturally generalise rooted binary tree-child networks. Moreover, we present a polynomial-time algorithm for building a rooted binary orchard network from its set of trinets. As a consequence, this algorithm affirmatively answers a previously-posed question of whether there is a polynomial-time algorithm for building a rooted binary tree-child network from the set of trinets it infers.

1. Introduction

The evolutionary relationships of a collection of present-day species are typically represented by a rooted phylogenetic (evolutionary) tree. Over recent decades, a wide variety of methods for building rooted phylogenetic trees from genomic data have been developed [6] and these methods are routinely used by computational biologists. However, it is now well recognised [11] that, as the result of non-treelike (reticulate) processes, rooted phylogenetic networks provide a more accurate representation of the evolutionary relationships for many such collections. These processes include hybridisation and lateral gene transfer. Consequently, a central current task in computational biology is the development of methods for building rooted phylogenetic networks.

A canonical and practical approach to building rooted phylogenetic networks is to amalgamate smaller networks (or trees) on overlapping leaf sets

Date: August 17, 2021.

2020 *Mathematics Subject Classification.* 05C85, 68R10.

Key words and phrases. Level-k networks, tree-child networks, orchard networks, trinets.

The first author was supported by the New Zealand Marsden Fund.

into a single rooted phylogenetic network. In the context of building rooted phylogenetic trees, which amalgamate smaller trees, these approaches are collectively called supertree methods and they have been very successful in the inference of rooted phylogenetic trees (for example, see [3]). A desirable property of any supertree method is that if the smaller trees are consistent, then the returned supertree infers each of the smaller trees. The theorem that underlies this property is the following. Loosely speaking, a set $\mathcal{P}$ of (smaller) rooted phylogenetic networks "encodes" a rooted phylogenetic network $\mathcal{N}$ if $\mathcal{N}$ is the only rooted phylogenetic network to infer each of the networks in $\mathcal{P}$. A rooted binary phylogenetic tree is encoded by the set of all rooted triples it infers, where a rooted triple is a rooted binary phylogenetic tree on three leaves (see, for example, [1, 19]). In this paper, we are interested in analogues of this theorem for phylogenetic networks.

With a necessary mild restriction, Gambette and Huber [7] showed that rooted binary level-1 networks, that is, rooted binary phylogenetic networks whose underlying cycles are vertex disjoint, are encoded by the set of all rooted triples they infer. However, this result does not generalise to rooted binary level-2 networks [7]. On the other hand, generalising level-1 networks in a different direction, Linz and Semple [17] showed recently that rooted binary normal networks [20], while not encoded by the set of rooted triples they infer, are encoded by the set of all rooted binary caterpillars on three and four leaves they infer. This improves upon a result of Willson [21] who showed that a rooted binary normal network on n leaves is encoded by the set of all rooted phylogenetic trees on n leaves it infers. Note that a rooted caterpillar on three leaves is the same as a rooted triple. However, analogous results for rooted binary tree-child networks [4], a slight generalisation of rooted binary normal networks, do not hold. In particular, rooted binary tree-child networks on n leaves are not necessarily encoded by the set of all rooted binary phylogenetic trees on n leaves they infer (for an example, see [17]). Thus, if we want to build the correct rooted phylogenetic network using a supertree-type approach, doing so with trees is limiting.

As a consequence of the work in [7], Huber and Moulton [8] considered building rooted phylogenetic networks from smaller networks, in particular, rooted phylogenetic networks on three leaves which they called trinets. In contrast to above, van Iersel and Moulton [13] showed that rooted binary level-2 networks as well as rooted binary tree-child networks are encoded by the set of all trinets they infer. In this paper, we generalise this result for tree-child networks to the recently introduced class of rooted binary orchard networks [5, 14]. That is, we show that a rooted binary orchard network is encoded by the set of trinets it infers. Unlike rooted binary tree-child networks whose total number of vertices is bounded linearly in the size of its leaf set [4], for a fixed set of leaves, the total number of vertices in a rooted binary orchard network can be arbitrarily large. Also, for any positive integer k, there exists a rooted binary orchard network whose level is at least k. Furthermore, we present a polynomial-time algorithm for building a rooted binary orchard networks from the set of trinets it infers. As a consequence, this answers a question of van Iersel and Moulton [13] of whether it is possible to build a rooted binary tree-child network from the set of trinets it infers in polynomial time. We remark here that this question about tree-child networks was independently answered by van Bemmelen [2]. We next formally state the main result of the paper.

Throughout the paper, X denotes a non-empty finite set and all paths are directed.

Phylogenetic networks. A *binary phylogenetic network* $\mathcal{N}$ *on* X is a rooted acyclic directed graph with no arcs in parallel satisfying the following properties:

(i) the (unique) root has in-degree zero and out-degree two;
(ii) the set of vertices with out-degree zero is X and all such vertices have in-degree one;
(iii) all other vertices either have in-degree one and out-degree two, or in-degree two and out-degree one.

Additionally, if $|X| = 1$, we allow $\mathcal{N}$ to consist of the single vertex in X. The vertices in X are called *leaves*, and so we refer to X as the *leaf set* of $\mathcal{N}$. Furthermore, vertices of in-degree one and out-degree two are *tree vertices*, while vertices of in-degree two and out-degree one are *reticulations*. Arcs directed into a reticulation are called *reticulation arcs*. If $\mathcal{N}$ has no reticulations, then $\mathcal{N}$ is a *rooted binary phylogenetic* X*-tree*. Since we only consider rooted binary phylogenetic trees and binary phylogenetic networks, we will abbreviate such trees and networks to rooted phylogenetic trees and phylogenetic networks, respectively. To illustrate, a phylogenetic network $\mathcal{N}_1$ on $\{x_1, x_2, \ldots, x_6\}$ is shown in Fig. 1. In this figure, as in all other figures in the paper, all arcs are directed down the page.

Let $\mathcal{N}$ be a phylogenetic network on X. If u and v are vertices of $\mathcal{N}$ and there is a path from u to v, we say u is an *ancestor* of v or, equivalently, v is a *descendant* of u. Note that every vertex is an ancestor, and thus a descendant, of itself. Furthermore, if $|X| \geq 2$, for each leaf $x \in X$, we denote the (unique) parent of x by p_x.

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two phylogenetic networks on X with vertex and arc sets V_1 and E_1, and V_2 and E_2, respectively. We say $\mathcal{N}_1$ is *isomorphic* to $\mathcal{N}_2$ if there exists a bijection $\varphi : V_1 \rightarrow V_2$ such that $\varphi(x) = x$ for all $x \in X$, and $(u, v) \in E_1$ if and only if $(\varphi(u), \varphi(v)) \in E_2$ for all $u, v \in V_1$.

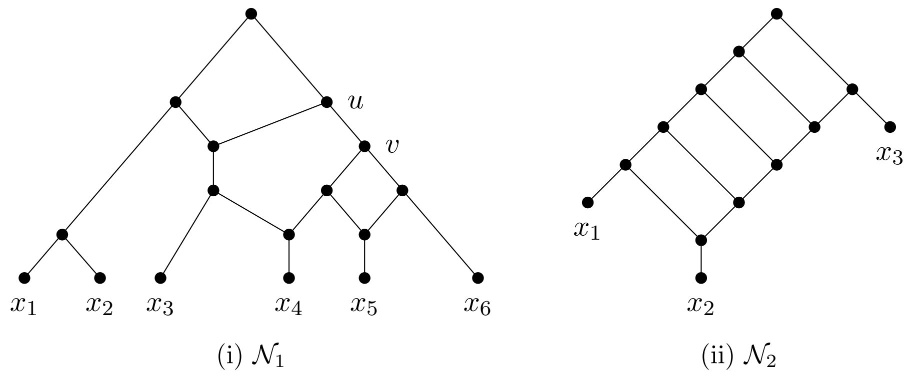


FIGURE 1. A phylogenetic network $\mathcal{N}_1$ on $\{x_1, x_2, \ldots, x_6\}$ and a phylogenetic network $\mathcal{N}_2$ on $\{x_1, x_2, x_3\}$. In $\mathcal{N}_1$, both u and v are stable ancestors of $\{x_5, x_6\}$.

Orchard networks. Let $\mathcal{N}$ be a phylogenetic network on X. Let $\{a, b\}$ be a 2-element subset of X. We say $\{a, b\}$ is a *cherry* if $p_a = p_b$. Furthermore, (a, b) is a *reticulated cherry* if (p_a, p_b) is a reticulation arc of $\mathcal{N}$ where, necessarily, p_a is a tree vertex and p_b is a reticulation. The arc (p_a, p_b) is called the *reticulation arc* of (a, b). As an example, consider the phylogenetic network $\mathcal{N}_1$ shown in Fig. 1. The set $\{x_1, x_2\}$ is a cherry, while (x_3, x_4) and (x_6, x_5) are reticulated cherries of $\mathcal{N}_1$. We next describe two reduction operations associated with cherries and reticulated cherries. First, suppose that $\{a, b\}$ is a cherry of $\mathcal{N}$. Let $\mathcal{N}'$ be the phylogenetic network on $X - \{b\}$ obtained from $\mathcal{N}$ by deleting b and its incident arc, and suppressing the resulting degree-two vertex p_a. We say that $\mathcal{N}'$ has been obtained from $\mathcal{N}$ by *reducing* b. Note that if p_a is the root of $\mathcal{N}$, the operation of reducing b corresponds to replacing $\mathcal{N}$ with the phylogenetic network consisting of the single vertex a. Second, suppose that (a, b) is a reticulated cherry of $\mathcal{N}$. Now let $\mathcal{N}'$ be the phylogenetic network on X obtained from $\mathcal{N}$ by deleting (p_a, p_b) and suppressing the two resulting degree-two vertices p_a and p_b. We say that $\mathcal{N}'$ has been obtained from $\mathcal{N}$ by *cutting* (a, b). For ease of reading, we sometimes refer to these operations as *picking a cherry* or *picking a reticulated cherry*, respectively.

A phylogenetic network $\mathcal{N}$ is *orchard* if there is a sequence

$$\mathcal{N} = \mathcal{N}_0, \mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$$

of phylogenetic networks such that, for each $i \in \{1, 2, \ldots, k\}$, the phylogenetic network $\mathcal{N}_i$ is obtained from $\mathcal{N}_{i-1}$ by either reducing a leaf of a cherry or cutting a reticulated cherry, and $\mathcal{N}_k$ consists of a single vertex. It is easily checked that both $\mathcal{N}_1$ and $\mathcal{N}_2$ in Fig. 1 are orchard networks. For $\mathcal{N}_2$, we can obtain a sequence by repeatedly cutting the reticulated cherry (x_1, x_2) until there are no more reticulations, and then reducing x_3 of the cherry

$\{x_2, x_3\}$, and reducing x_2 of the cherry $\{x_1, x_2\}$. It may appear that the order in which we pick a cherry or a reticulated cherry is important, but this is not the case as the following lemma [5, 14] shows.

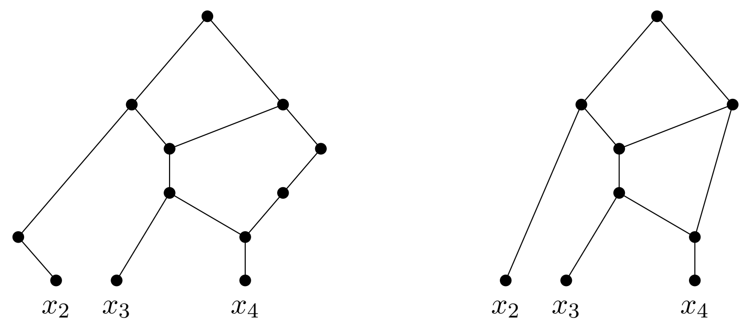
Lemma 1.1. *Let $\mathcal{N}$ be an orchard network, and suppose that $\mathcal{N}'$ is obtained from $\mathcal{N}$ by picking either a cherry or a reticulated cherry. Then $\mathcal{N}'$ is an orchard network.*

Orchard networks were introduced independently in [5] and [14], and generalise the more familiar class of tree-child networks. A phylogenetic network is *tree-child* if every non-leaf vertex is the parent of a tree vertex or a leaf [4]. However, not all phylogenetic networks are orchard. For example, neither of the two phylogenetic networks shown in Fig. 3 is orchard.

Trinets. A *trinet* is a phylogenetic network on three leaves. Observe that trinets generalise the more familiar concept of *rooted triples*, rooted (binary) phylogenetic trees on three leaves.

Let $\mathcal{N}$ be a phylogenetic network on X, and let X' be a subset of X. A *stable ancestor* of X' is a vertex u of $\mathcal{N}$ having the property that, for all $x \in X'$, every path from the root of $\mathcal{N}$ to x traverses u. In the literature, a stable ancestor of X' is also referred to as a "visible" ancestor of X'. Since the root itself satisfies this property, such a vertex always exists. Furthermore, we say u is a *lowest stable ancestor* of X' if no distinct stable ancestor of X' is a descendant of u. Note that if u and v are stable ancestors of X', then there is either a path from u to v, or a path from v to u. It follows that the lowest stable ancestor of X' is unique. We denote the lowest stable ancestor of X' by $\mathrm{lsa}(X')$. In Fig. 1, u and v are stable ancestors of $\{x_5, x_6\}$ in $\mathcal{N}_1$, but v is the lowest stable ancestor of $\{x_5, x_6\}$ in $\mathcal{N}_1$.

For a directed graph G, the *full simplification* of G is the directed graph obtained from G by repeatedly suppressing vertices of in-degree one and out-degree one, and deleting exactly one arc of any pair of arcs in parallel until neither of these operations are applicable. Now, let $\mathcal{N}$ be a phylogenetic network on X, and let X' be a subset of X. Suppose that u is the lowest stable ancestor of X'. The *path graph of $\mathcal{N}$ on X'* is the directed subgraph of $\mathcal{N}$ obtained by deleting all vertices and arcs not on a path from u to a leaf in X'. That is, the path graph of $\mathcal{N}$ on X' consists of all paths of $\mathcal{N}$ starting at u and ending at a vertex in X'. The phylogenetic network *exhibited by $\mathcal{N}$ on X'* is the full simplification of the path graph of $\mathcal{N}$ on X'. We denote the phylogenetic network exhibited by $\mathcal{N}$ on X' by $\mathcal{N}_{X'}$. In the special case $|X'| = 3$, this process constructs the *trinet exhibited by $\mathcal{N}$ on X'*. The set of all trinets exhibited by $\mathcal{N}$ is denoted by $Tn(\mathcal{N})$. Again consider the phylogenetic network $\mathcal{N}_1$ shown in Fig. 1. Noting that the root is the lowest stable ancestor of $\{x_2, x_3, x_4\}$, the path graph of $\mathcal{N}_1$ on $\{x_2, x_3, x_4\}$



(i) The path-graph exhibited by $\mathcal{N}_1$ on $\{x_2, x_3, x_4\}$.

(ii) The trinet exhibited by $\mathcal{N}_1$ on $\{x_2, x_3, x_4\}$.

FIGURE 2. The path graph of the phylogenetic network $\mathcal{N}_1$, shown in Fig. 1, on $\{x_2, x_3, x_4\}$, and the trinet exhibited by $\mathcal{N}_1$ on $\{x_2, x_3, x_4\}$.

is shown in Fig. 2(i), while the full simplification of this path graph, that is, the trinet exhibited by $\mathcal{N}_1$ on $\{x_2, x_3, x_4\}$, is shown in Fig. 2(ii).

A phylogenetic network $\mathcal{N}$ on X is *recoverable* if it has no arc (u, v) whose deletion disconnects $\mathcal{N}$ and v is an ancestor of every element in X, that is, v is a stable ancestor of X (and v is not the root). Equivalently, $\mathcal{N}$ is recoverable if $\mathrm{lsa}(X)$ is the root of $\mathcal{N}$. We say a recoverable phylogenetic network $\mathcal{N}$ is *encoded* by $Tn(\mathcal{N})$ if it has the following property: If $\mathcal{N}'$ is a recoverable phylogenetic network and, up to isomorphism, $Tn(\mathcal{N}) = Tn(\mathcal{N}')$, then $\mathcal{N}$ is isomorphic to $\mathcal{N}'$. Observe that if a phylogenetic network $\mathcal{N}$ is not recoverable, then $Tn(\mathcal{N})$ provides no information of the structure of $\mathcal{N}$ between the root of $\mathcal{N}$ and an arc (u, v) whose deletion disconnects $\mathcal{N}$ and in which every leaf is descendant of v.

The next theorem is one of two main results in [13]. It generalises the well-known result mentioned earlier that says a rooted phylogenetic tree $\mathcal{T}$ is encoded by the set of all rooted triples exhibited by $\mathcal{T}$ (see, for example, [1, 19]). All tree-child networks are recoverable since, provided the leaf set has size at least two, the root has out-degree two and every non-leaf vertex is the parent of a tree vertex or a leaf.

Theorem 1.2. *Let $\mathcal{N}$ be a tree-child network on X, where $|X| \geq 3$. Then $Tn(\mathcal{N})$ encodes $\mathcal{N}$.*

The first part of Theorem 1.3, the main result of this paper, generalises Theorem 1.2 to the class of orchard networks. The second part of Theorem 1.3 shows that orchard networks can be reconstructed from the set of trinets they exhibit in polynomial time which, as a consequence, answers a question of [13] of whether such a reconstruction is possible for tree-child networks.

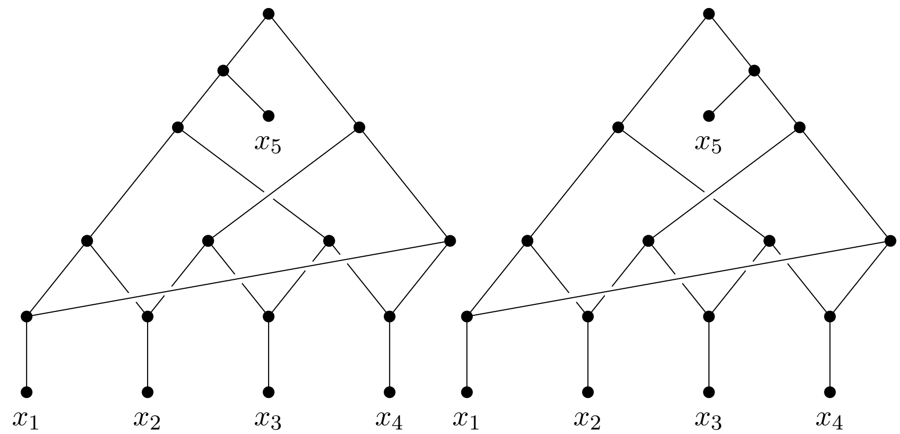
Theorem 1.3. *Let $\mathcal{N}$ be an orchard network on X, where $|X| \geq 3$. Then*

(i) *$Tn(\mathcal{N})$ encodes $\mathcal{N}$, and*
(ii) *up to isomorphism, $\mathcal{N}$ can be reconstructed from $Tn(\mathcal{N})$ in time $O(|V|^6)$, where V is the vertex set of $\mathcal{N}$.*

As we show in the next section, like tree-child networks, orchard networks are recoverable. However, unlike tree-child networks whose total number of reticulations is at most linear in the size of their leaf sets, and so the total number of vertices in a tree-child network is bounded (see [18]), the total number of reticulations in an orchard network is not bounded by the size of its leaf set. For example, by extending $\mathcal{N}_2$ in Fig. 1 in the obvious way, it follows that, even with three leaves, the total number of reticulations in an orchard network is not bounded. Moreover, this extension also shows that, for each non-negative integer k, there exists an orchard network whose level is at least k. A phylogenetic network is *level-k* if each biconnected component contains at most k reticulations.

In addition to Theorem 1.2, the second main result in [13] establishes that recoverable level-2 phylogenetic networks are also encoded by their sets of exhibited trinets. These results, together with Theorem 1.3, support the conjecture in [8], and restated in [13], that if a phylogenetic network $\mathcal{N}$ is recoverable, then $Tn(\mathcal{N})$ encodes $\mathcal{N}$. However, Huber et al. [9] construct a family of counterexamples to this conjecture, where the level of the phylogenetic network is exponential in the size of the leaf set. In particular, for all $n \geq 4$, if the size of the leaf set is n, then the level of the phylogenetic network is $(2^{n-2} - 1)n$. Thus if $n = 4$, then the level of the counterexample is 12. This raises the problem of determining the largest value of k for which all recoverable level-k phylogenetic networks are encoded by their sets of trinets. In the last section of the paper, we show that this value is at most 3 by showing that the two non-isomorphic recoverable phylogenetic networks $\mathcal{N}_1$ and $\mathcal{N}_2$, each of level-4, shown in Fig. 3 have the property that, up to isomorphism, $Tn(\mathcal{N}_1) = Tn(\mathcal{N}_2)$. Note that each of the counterexamples $\mathcal{N}$ in [9] have the much stronger property that the set of all phylogenetic networks exhibited by $\mathcal{N}$ on all proper subsets of the leaf set of $\mathcal{N}$ does not encode $\mathcal{N}$.

The paper is organised as follows. The next section consists of some preliminary lemmas which are used in the proof of Theorem 1.3. The proof of Theorem 1.3 is by induction on the sum of the number of leaves and the number of reticulations of an orchard network. The approach taken is to initially pick either a cherry, thereby reducing the number of leaves,



(i) $\mathcal{N}_1$ (ii) $\mathcal{N}_2$

FIGURE 3. Two non-isomorphic level-4 phylogenetic networks $\mathcal{N}_1$ and $\mathcal{N}_2$. Both $\mathcal{N}_1$ and $\mathcal{N}_2$ are recoverable and, up to isomorphism, $Tn(\mathcal{N}_1) = Tn(\mathcal{N}_2)$.

or a reticulated cherry, thereby reducing the number of reticulations. In Section 3, we establish various lemmas concerning the notion of exhibit and that of cherries and reticulated cherries. The proof of Theorem 1.3 is given in Section 4. The last section, Section 5, verifies the above-mentioned level-4 example.

We end the introduction with two remarks. First, a concept in mathematical phylogenetics that is similar to exhibit is that of "display". In particular, let $\mathcal{N}$ be a phylogenetic network on X and let $\mathcal{T}$ be a rooted phylogenetic X'-tree, where $X' \subseteq X$. We say $\mathcal{N}$ *displays* $\mathcal{T}$ if $\mathcal{T}$ can be obtained from $\mathcal{N}$ by deleting arcs and vertices, and suppressing any resulting vertices of in-degree one and out-degree one. If $\mathcal{N}$ is a rooted phylogenetic tree, then the concepts of exhibit and display are equivalent. For clarification, in the initial part of the introduction, whenever we said, for example, a phylogenetic network "infers" a rooted phylogenetic tree, we really meant a phylogenetic network displays a rooted phylogenetic tree.

Second, Theorem 1.3 and other analogous theorems are a step towards developing supertree-type methods for building phylogenetic networks. In practice, it is unlikely that the input to such a method is the entire set $Tn(\mathcal{N})$ of trinets exhibited by a phylogenetic network $\mathcal{N}$. A more realistic task is when the input is an arbitrary subset of trinets and the goal is to decide whether or not there is a phylogenetic network that exhibits each of the trinets in this set. This has been considered previously for when the input is a set of rooted triples and we are asked to find a level-1 network that displays each of the rooted triples in the set [12, 15, 16] and, more recently, when the input is a set of trinets and we are asked to find a level-1 network that exhibits each of the trinets in the set [10]. As an intermediate step towards developing a supertree-type method for building orchard networks, we leave it as an open problem to develop an algorithm that takes an arbitrary collection of trinets on overlapping leaf sets and decides whether or not there is an orchard network that exhibits each trinet in the collection.

2. Exhibiting Lemmas

In this section, we establish some general lemmas in relation to the notion of exhibiting that will be used in the proof of Theorem 1.3. The first two lemmas are used in several places.

Lemma 2.1. *Let $\mathcal{N}$ be a phylogenetic network on X, and let $A \subseteq X$. Let G_A be the path graph of $\mathcal{N}$ on A, and let*

$$G_A = G_0, G_1, G_2, \ldots, G_k = \mathcal{N}_A$$

be a sequence of directed graphs such that, for all $i \in \{1, 2, \ldots, k\}$, the directed graph G_i is obtained from G_{i-1} by ether suppressing a vertex of in-degree one and out-degree one, or deleting an arc in parallel. Let u and v be vertices of G_i for some i. Then

- (i) *If there is a path from u to v in G_i, then there is a path from u to v in G_A.*
- (ii) *If u and v are vertices of G_i for some i, then every path from u to a (fixed) leaf ℓ traverses v in G_A if and only if every path from u to ℓ traverses v in G_i.*

Proof. We omit the proof of (i) as it takes the same approach as the proof of (ii) but is simpler. For the proof of (ii), it suffices to show that if $j \in \{0, 1, \ldots, i-1\}$, then every path from u to ℓ traverses v in G_j if and only if every path from u to ℓ traverses v in G_{j+1}. Clearly, this sufficiency holds if G_{j+1} is obtained from G_j by deleting an arc in parallel. Therefore assume that G_{j+1} is obtained from G_j by suppressing a vertex, say w, of in-degree one and out-degree one. Let e denote the new arc in G_{j+1} resulting from this suppression. Now, if P is a path from u to ℓ that traverses v and w in G_j, then the path obtained from P by replacing w and its incident arcs with e is a path from u to ℓ that traverses v and e in G_{j+1}. Since the analogous converse of this also holds, it follows that every path from u to ℓ traverses v in G_j if and only if every path from u to ℓ traverses v in G_{j+1}. This completes the proof of the lemma. □

Lemma 2.2. *Let $\mathcal{N}$ be a phylogenetic network on X, and let $A \subseteq X$. Let u and v be vertices of the path graph G_A of $\mathcal{N}$ on A, and let $\ell \in A$. Then*

(i) *If every path from u to ℓ traverses v in G_A, then every path from u to ℓ traverses v in $\mathcal{N}$.*
(ii) *If v is a stable ancestor of ℓ in $\mathcal{N}_A$, then v is a stable ancestor of ℓ in $\mathcal{N}$.*

Proof. Since u and v are vertices of G_A, the proof of (i) is an immediate consequence of the construction of G_A from $\mathcal{N}$. To prove (ii), suppose that v is a stable ancestor of ℓ in $\mathcal{N}_A$. Then, as the root of $\mathcal{N}_A$ is $\mathrm{lsa}(A)$, it follows by Lemma 2.1, that every path from $\mathrm{lsa}(A)$ to ℓ in G_A traverses v. As every path from the root of $\mathcal{N}$ to ℓ traverses $\mathrm{lsa}(A)$, it follows by (i) that v is a stable ancestor of ℓ in $\mathcal{N}$. □

The next three lemmas provide sufficient conditions for a vertex of a phylogenetic network $\mathcal{N}$ to be a vertex of the phylogenetic network exhibited by $\mathcal{N}$ on a given subset of leaves.

Lemma 2.3. *Let $\mathcal{N}$ be a phylogenetic network on X, and let $A \subseteq X$. Let v be a tree vertex of $\mathcal{N}$ with children c_1 and c_2, and suppose there exists $\ell_1, \ell_2 \in A$ such that*

(i) *ℓ_1 is a descendant of c_1,*
(ii) *ℓ_2 is a descendant of c_2, and*
(iii) *ℓ_2 is not a descendant of c_1.*

Then v is a vertex of $\mathcal{N}_A$.

Proof. We first show that v is a vertex of the path graph G_A of $\mathcal{N}$ on A. Since there is a path in $\mathcal{N}$ from v to a leaf in A, either v is a descendant of $\mathrm{lsa}(A)$ or $\mathrm{lsa}(A)$ is a descendant of v. If the latter holds, then there are paths from c_1 to ℓ_1 and from c_2 to ℓ_2, each of which traverses $\mathrm{lsa}(A)$. This implies that there is a path from c_1 to ℓ_2 via $\mathrm{lsa}(A)$, contradicting (iii). Hence v is a descendant of $\mathrm{lsa}(A)$, and so v is a vertex of G_A.

We complete the proof of the lemma by showing that v is not suppressed in the process of obtaining $\mathcal{N}_A$ from G_A. If v is suppressed, then at some stage in the process, v has one incoming arc and one outgoing arc, (v, w) say. By Lemma 2.1, every path in G_A from v to a leaf in A traverses w which, in turn, implies by Lemma 2.2 that every path in $\mathcal{N}$ from v to a leaf in A traverses w. In particular, every path in $\mathcal{N}$ from c_1 to ℓ_1 and from c_2 to ℓ_2 traverses w, in which case, there is a path in $\mathcal{N}$ from c_1 to ℓ_2 via w, contradicting (iii). It follows that v is a vertex of $\mathcal{N}_A$. □

Lemma 2.4. *Let $\mathcal{N}$ be a phylogenetic network on X, and let $A \subseteq X$. Let v be a reticulation of $\mathcal{N}$. If a parent of v is a vertex of $\mathcal{N}_A$, then v is a vertex of $\mathcal{N}_A$.*

Proof. Let p and q denote the parents of v in $\mathcal{N}$, and suppose that p is a vertex of $\mathcal{N}_A$. We begin by showing that v, as well as p and q, is a vertex of the path graph G_A of $\mathcal{N}$ on A. Now p lies on a path of $\mathcal{N}$ from $\operatorname{lsa}(A)$ to a leaf in A. If p is a reticulation of $\mathcal{N}$, then v also lies on this path. Furthermore, if p is a tree vertex of $\mathcal{N}$, then, as p is a vertex of $\mathcal{N}_A$, both children of p must also lie on such a path; otherwise, p has in-degree one and out-degree one in G_A. Thus v is a vertex of G_A, and so both parents of v are also vertices of G_A.

It remains to show that v is not suppressed in the process of obtaining $\mathcal{N}_A$ from G_A. If v is suppressed in this process, then, as p is a vertex of $\mathcal{N}_A$, at subsequent stages in the process of obtaining $\mathcal{N}_A$ from G_A, the vertex q is suppressed, and v has two distinct incoming arcs in parallel, one of which is (p, v). Since p is a vertex of $\mathcal{N}_A$, this in turn implies that the other arc in parallel also connects p and v. But then p is a tree vertex of $\mathcal{N}$ and so, once one of these parallel arcs is deleted, p has in-degree one and out-degree one, a contradiction as p is a vertex of $\mathcal{N}_A$. Hence v is not suppressed in obtaining $\mathcal{N}_A$ from G_A, and so v is a vertex of $\mathcal{N}_A$. □

Lemma 2.5. *Let $\mathcal{N}$ be a phylogenetic network on X, and let $A \subseteq X$. Let v be a tree vertex of $\mathcal{N}$ that is a descendant of* $\operatorname{lsa}(A)$*. If A contains every leaf of $\mathcal{N}$ that is a descendant of v, then every descendant vertex of v in $\mathcal{N}$ is a vertex of $\mathcal{N}_A$.*

Proof. First observe that every vertex that is a descendant of v is a vertex of the path graph G_A of $\mathcal{N}$ on A. Furthermore, as A contains every leaf that is a descendant of v, it follows that no vertex that is a descendant of v has in-degree one and out-degree one in G_A. Suppose that at some stage of the process of obtaining $\mathcal{N}_A$ from G_A a descendant of v, say w, has in-degree one and out-degree one. Without loss of generality, choose w such that no descendant of v has in-degree one and out-degree one prior to w in this process. If w is a tree vertex of $\mathcal{N}$, then w has two distinct children and so, for w to have in-degree one and out-degree one, one if its children needs to have in-degree one and out-degree one prior to this happening. As both children of w are descendants of v, this contradicts the choice of w. Thus we may assume that w is a reticulation of $\mathcal{N}$.

At least one parent, p say, of w is a descendant of v in $\mathcal{N}$. Since w is suppressed in the process of obtaining $\mathcal{N}_A$ from G_A, it follows by the choice of w that, in this process prior to w having in-degree one and out-degree one, the parent of w that is not p, say q, is suppressed and w has two distinct incoming arcs in parallel, one of which is (p, w). By Lemma 2.1(i), this implies that q is a descendant of v in G_A, and so q is a descendant of v in $\mathcal{N}$, contradicting the choice of w. This completes the proof of the lemma. □

Lemma 2.6. *Let $\mathcal{N}$ be a phylogenetic network on X, and let $A \subseteq B \subseteq X$. Then $\mathcal{N}_A$ is the phylogenetic network exhibited by $\mathcal{N}_B$ on A.*

Proof. Let G_A and G_B be the path graphs of $\mathcal{N}$ on A and B, respectively. Since $A \subseteq B$, the vertex $\mathrm{lsa}(A)$ is a descendant of $\mathrm{lsa}(B)$, and so $\mathrm{lsa}(A)$ is a vertex of G_B. In turn, this implies that G_A is a subgraph of G_B. If v is a vertex of G_A, then the in-degree of v in G_A is at most the in-degree of v in G_B, and the out-degree of v in G_A is at most the out-degree of v in G_B. Therefore, every vertex of G_A that is not a vertex of $\mathcal{N}_B$ is also not a vertex of $\mathcal{N}_A$. Thus the directed graph G'_A, the path graph of $\mathcal{N}_B$ on A, can be obtained from G_A by repeated applications of suppressing vertices of in-degree one and out-degree one, and deleting exactly one arc of any pair of arcs in parallel. Note that we need not take the full simplification of G_A to get G'_A. Since $\mathcal{N}_A$ is the full simplification of G'_A, it follows that $\mathcal{N}_A$ is the phylogenetic network exhibited by $\mathcal{N}_B$ on A. □

The last lemma of this section uses each of Lemmas 2.3–2.6 in its proof.

Lemma 2.7. *Let $\mathcal{N}$ be a phylogenetic network on X, where $|X| \geq 3$, and let (a, b) be a reticulated cherry of $\mathcal{N}$. Let p_b denote the parent of b, and let $A \subseteq X$ such that $A = \{b, x, y\}$. Then p_b is a vertex of $\mathcal{N}_A$ if and only if p_b is a vertex of at least one of $\mathcal{N}_{\{b,x\}}$ and $\mathcal{N}_{\{b,y\}}$.*

Proof. First suppose that p_b is a vertex of $\mathcal{N}_A$. Let v be a tree vertex (possibly the root) of $\mathcal{N}_A$ with the property that there is a path P from v to p_b such that every vertex on this path (except v itself) is a reticulation. Note that such a vertex can be found by starting at p_b and moving along reticulation arcs towards the root of $\mathcal{N}_A$. If neither x nor y is a descendant of v, then, by Lemmas 2.5 and 2.6, p_b is a vertex of both $\mathcal{N}_{\{b,x\}}$ and $\mathcal{N}_{\{b,y\}}$. Therefore, without loss of generality, we may assume x is descendant of v in $\mathcal{N}_A$. Let w denote the first reticulation along P, and note that w could be p_b. Since the only leaf descendant of w is b, it follows by Lemma 2.3 that v is a vertex of $\mathcal{N}_{\{b,x\}}$. By repeated applications of Lemma 2.4 to the reticulations along P, we deduce that p_b is also a vertex of $\mathcal{N}_{\{b,x\}}$.

Now suppose that p_b is a vertex of $\mathcal{N}_{\{b,z\}}$, where $z \in \{x, y\}$. By Lemma 2.6, the phylogenetic network $\mathcal{N}_{\{b,z\}}$ is the phylogenetic network exhibited by $\mathcal{N}_A$ on $\{b, z\}$, and so p_b is a vertex of $\mathcal{N}_A$. □

3. Cherry and Reticulated-Cherry Lemmas

The lemmas in this section are more aligned with orchard networks. We begin by showing that orchard networks are recoverable.

Lemma 3.1. *Let $\mathcal{N}$ be an orchard network on X. Then $\mathcal{N}$ is recoverable.*

Proof. Let ρ denote the root of $\mathcal{N}$. The proof is by induction on the sum of the number $n = |X|$ of leaves and the number r of reticulations of $\mathcal{N}$. If $n+r = 1$, then $\mathcal{N}$ has exactly one leaf and no reticulations. Thus $\mathcal{N}$ consists of the single vertex in X, and so the lemma holds. If $n + r = 2$, then, as $\mathcal{N}$ is orchard, $\mathcal{N}$ consists of two leaves adjoined to ρ. Again, the lemma holds.

Now suppose that $n + r \geq 3$, and that every orchard network in which the sum of the number of leaves and the number of reticulations is at most $n + r - 1$ is recoverable. Let $\mathcal{N}'$ be a phylogenetic network on X' that is obtained from $\mathcal{N}$ by picking either a cherry $\{a, b\}$ or a reticulated cherry (a, b). Note that the roots of $\mathcal{N}$ and $\mathcal{N}'$ coincide as $\mathcal{N}$ does not consist of two leaves adjoined to the root. By Lemma 1.1, $\mathcal{N}'$ is orchard. Therefore, as the sum of the number of leaves and number of reticulations of $\mathcal{N}'$ is $n + r - 1$, it follows by the induction assumption that $\mathcal{N}'$ is recoverable. That is, the root of $\mathcal{N}'$ is the unique stable ancestor of X' in $\mathcal{N}'$. As the roots of $\mathcal{N}$ and $\mathcal{N}'$ coincide, up to traversing p_a and p_b (the parents of a and b, respectively, in $\mathcal{N}$), every path in $\mathcal{N}'$ from the root to a leaf x in X' is also a path in $\mathcal{N}$ from ρ to x. It follows that ρ is the unique stable ancestor of X in $\mathcal{N}$, and so $\mathcal{N}$ is recoverable. $\square$

Lemma 3.2. *Let $\mathcal{N}$ be a (arbitrary) recoverable phylogenetic network, and suppose that $\mathcal{N}'$ is obtained from $\mathcal{N}$ by picking either a cherry or a reticulated cherry. Then $\mathcal{N}'$ is recoverable.*

Proof. Let X' denote the leaf set of $\mathcal{N}'$, and let $\{a, b\}$ or (a, b) be the cherry or reticulated cherry of $\mathcal{N}$ that is picked to obtain $\mathcal{N}'$. Observe that we may assume the roots of $\mathcal{N}$ and $\mathcal{N}'$ coincide; otherwise, $\mathcal{N}'$ consists of a single vertex and the lemma holds. Suppose that $\mathcal{N}'$ is not recoverable. Then there is a non-root vertex v' of $\mathcal{N}'$ that is a stable ancestor of X'. Consider v' in $\mathcal{N}$. Since $\mathcal{N}$ is recoverable, there must be a path P from the root of $\mathcal{N}$ to a leaf that does not traverse v'. As $\mathcal{N}'$ is obtained from $\mathcal{N}$ by picking either $\{a, b\}$ or (a, b), this path P must end at b. It follows that $\mathcal{N}'$ is obtained from $\mathcal{N}$ by picking (a, b) and P traverses (p_a, p_b). But this implies there is a path in $\mathcal{N}'$ from the root of $\mathcal{N}'$ to a that does not traverse v', contradicting that v' is a stable ancestor of X'. Hence $\mathcal{N}'$ is recoverable. $\square$

Let $\mathcal{N}$ be a phylogenetic network on X, and let $\{a, b\} \subseteq X$. If $\{a, b\}$ is a cherry of $\mathcal{N}$, we refer to p_a as the *tree vertex of* $\{a, b\}$, while if (a, b) is a reticulated cherry, we refer to p_a as the *tree vertex of* (a, b).

Lemma 3.3. *Let $\mathcal{N}$ be a phylogenetic network on X, where $|X| \geq 3$, and let $\{a, b\} \subseteq X$. Then*

(i) $\{a, b\}$ *is a cherry of* $\mathcal{N}$ *if and only if, for all* A *with* $\{a, b\} \subseteq A \subseteq X$ *and* $|A| = 3$*, we have that* $\{a, b\}$ *is a cherry of* $\mathcal{N}_A$*, and*

(ii) (a, b) *is a reticulated cherry of* $\mathcal{N}$ *if and only if, for all* A *with* $\{a, b\} \subseteq A \subseteq X$ *and* $|A| = 3$*, we have that* (a, b) *is a reticulated cherry of* $\mathcal{N}_A$*.*

Proof. We will prove (ii). The proof of (i) closely follows the proof of (ii) and is omitted. Let $A \subseteq X$ such that $\{a, b\} \subseteq A$ and $|A| = 3$. If (a, b) is a reticulated cherry of $\mathcal{N}$, then p_a satisfies the conditions of Lemma 2.5. Thus a, b, p_a, and p_b are all vertices of $\mathcal{N}_A$, and so (a, b) is a reticulated cherry of $\mathcal{N}_A$.

For the converse of (ii), suppose that (a, b) is not a reticulated cherry of $\mathcal{N}$. We will show that there is a trinet exhibited by $\mathcal{N}$ whose leaf set contains a and b, but (a, b) is not a reticulated cherry of this trinet. If there is no trinet exhibited by $\mathcal{N}$ in which (a, b) is a reticulated cherry, then the desired outcome holds. So we may assume that there exists a trinet $\mathcal{N}_A$ exhibited by $\mathcal{N}$ in which (a, b) is a reticulated cherry. Let u be the tree vertex of (a, b) of $\mathcal{N}_A$. In $\mathcal{N}$, the vertex u is a tree vertex of which a is a descendant. Since u is stable ancestor of a in $\mathcal{N}_A$, it follows by Lemma 2.2 that every path from the root of $\mathcal{N}$ to a traverses u. Thus, if (a, b) is a reticulated cherry of another trinet exhibited by $\mathcal{N}$ and u' is the tree vertex of (a, b) of this trinet, then u is either an ancestor or a descendant of u' in $\mathcal{N}$. It now follows that there is a path P in $\mathcal{N}$ from the root of $\mathcal{N}$ to a containing every vertex that is the tree vertex of (a, b) of a trinet exhibited by $\mathcal{N}$ in which (a, b) is a reticulated cherry.

Let v denote the last such tree vertex along P. If a and b are the only leaf descendants of v in $\mathcal{N}$, then, by Lemma 2.5, for any choice of A containing a and b, all descendant vertices of v in $\mathcal{N}$ are vertices of the trinet exhibited by $\mathcal{N}$ on A. But (a, b) is not a reticulated cherry of $\mathcal{N}$, so v is not the tree vertex of any trinet exhibited by $\mathcal{N}$ in which (a, b) is a reticulated cherry, a contradiction. Therefore, in $\mathcal{N}$, the vertex v has a leaf descendant, say ℓ, other than a and b. Consider the trinet exhibited by $\mathcal{N}$ on $\{a, b, \ell\}$. If v is not a vertex of the path graph of $\mathcal{N}$ on $\{a, b, \ell\}$, then $\mathrm{lsa}(\{a, b, \ell\})$ is a descendant of v in $\mathcal{N}$, and so, by the choice of v, the ordered pair (a, b) is not a reticulated cherry of $\mathcal{N}_{\{a,b,\ell\}}$, and we have the desired outcome. Thus we may assume that v is a vertex of the path graph of $\mathcal{N}$ on $\{a, b, \ell\}$. If v is a vertex of $\mathcal{N}_{\{a,b,\ell\}}$, then a, b, and ℓ are descendants of v in $\mathcal{N}_{\{a,b,\ell\}}$. Therefore, if (a, b) is a reticulated cherry of $\mathcal{N}_{\{a,b,\ell\}}$, then v is not its tree vertex. But every other possible such tree vertex is an ancestor of v in $\mathcal{N}$. Hence, (a, b) is not a reticulated cherry of $\mathcal{N}_{\{a,b,\ell\}}$. The final case to consider is when v is suppressed in the process of obtaining $\mathcal{N}_{\{a,b,\ell\}}$ from the path graph of $\mathcal{N}$ on $\{a, b, \ell\}$. Then the unique child of v in this step of the process or a descendant of this child is a vertex of $\mathcal{N}_{\{a,b,\ell\}}$ and has a, b, and ℓ as descendants. But every vertex which is a tree vertex of (a, b) in some trinet exhibited by $\mathcal{N}$ is an ancestor of this descendant of v, so (a,b) is not a reticulated cherry of $\mathcal{N}_{\{a,b,\ell\}}$. This completes the proof of the converse of (ii), and thus the lemma. $\square$

Lemma 3.4. *Let $\mathcal{N}$ be a phylogenetic network on X, and let $\{a,b\}$ be a cherry of $\mathcal{N}$. Let $\mathcal{N}'$ be the phylogenetic network obtained from $\mathcal{N}$ by reducing b, and suppose that $A \subseteq X - \{b\}$. Then $\mathcal{N}_A = \mathcal{N}'_A$.*

Proof. First observe that $\mathrm{lsa}(A)$ of $\mathcal{N}$ is also $\mathrm{lsa}(A)$ of $\mathcal{N}'$. Clearly the lemma holds if $|A| = 1$, so we may assume that $|A| \geq 2$. Let G_A be the path graph of $\mathcal{N}$ on A, and let G'_A be the path graph of $\mathcal{N}'$ on A. Let $\ell \in A$, where $\ell \neq a$. Then every path in $\mathcal{N}$ from $\mathrm{lsa}(A)$ to ℓ is a path in $\mathcal{N}'$ from $\mathrm{lsa}(A)$ to ℓ. Therefore, if $a \notin A$, the path graph G'_A is identical to G_A, and so $\mathcal{N}_A = \mathcal{N}'_A$. On the other hand, if $a \in A$, then every path in $\mathcal{N}$ from $\mathrm{lsa}(A)$ to a traverses p_a, and so suppressing p_a in such a path produces a path in $\mathcal{N}'$ from $\mathrm{lsa}(A)$ to a. Moreover, all paths in $\mathcal{N}'$ from $\mathrm{lsa}(A)$ to a can be obtained in this way. Thus, in G_A, the vertex p_a has in-degree one and out-degree one, and so G'_A is obtained from G_A by suppressing p_a. It follows that $\mathcal{N}_A = \mathcal{N}'_A$. $\square$

Lemma 3.5. *Let $\mathcal{N}$ be a phylogenetic network on X, and let (a,b) be a reticulated cherry of $\mathcal{N}$. Let p_a and p_b denote the parents of a and b, respectively, in $\mathcal{N}$. Let $\mathcal{N}'$ be the phylogenetic network obtained from $\mathcal{N}$ by cutting (a,b), and suppose that $A \subseteq X$. Then each of the following holds:*

(i) *If $b \notin A$, then $\mathcal{N}_A = \mathcal{N}'_A$.*
(ii) *If $a, b \in A$, then $\mathcal{N}'_A$ is obtained from $\mathcal{N}_A$ by deleting (p_a, p_b) and suppressing p_a and p_b.*
(iii) *If $a \notin A$, $b \in A$, and p_b is not a vertex of $\mathcal{N}_A$, then $\mathcal{N}_A = \mathcal{N}'_A$.*
(iv) *If $a \notin A$, $b \in A$, and p_b is a vertex of $\mathcal{N}_A$, then $\mathcal{N}'_A$ is obtained from $\mathcal{N}_A$ by*
    (I) *deleting the arc (u, p_b), where u is a vertex such that there is a path in $\mathcal{N}$ from u to p_b traversing p_a, and every non-terminal vertex along this path is not a vertex of $\mathcal{N}_A$,*
    (II) *repeatedly deleting non-leaf vertices of out-degree zero until there are no such vertices, and*
    (III) *taking the full simplification of the resulting directed graph.*

Proof. Let $A \subseteq X$. Let G_A be the path graph of $\mathcal{N}$ on A, and let G'_A be the path graph of $\mathcal{N}'$ on A. If $a, b \notin A$, then $G_A = G'_A$, so $\mathcal{N}_A = \mathcal{N}'_A$. If $a \in A$ and $b \notin A$, then, up to suppressing p_a, we have $G_A = G'_A$. Thus $\mathcal{N}_A = \mathcal{N}'_A$. Therefore (i) holds and so, for the remainder of the proof, we may assume that $b \in A$, in which case (p_a, p_b) is an arc of G_A.

Let $G^0_A = G_A$, and let H^0_A be the directed graph obtained from G_A by deleting (p_a, p_b). Note that if $a \in A$, then G'_A can be obtained from H^0_A

by suppressing p_a and p_b. Furthermore, if $a \notin A$, then G'_A can be obtained from H^0_A by suppressing p_b and repeatedly deleting non-leaf vertices with out-degree zero.

Suppose that u_0 is a vertex of G^0_A with in-degree one and out-degree one, but $u_0 \neq p_a$. Note that $u_0 \neq p_b$. In constructing H^0_A from G^0_A, the only vertices whose degrees changed were p_a and p_b. Therefore, u_0 also has in-degree one and out-degree one in H^0_A. Construct G^1_A and H^1_A from G^0_A and H^0_A, respectively, by suppressing u_0 and deleting exactly one arc of any resulting pair of parallel arcs. Observe that if an arc in parallel is deleted, then it is not incident with p_a or p_b. Furthermore, H^1_A can be obtained from G^1_A by deleting (p_a, p_b), and that $\mathcal{N}'_A$ can be obtained from H^1_A by repeatedly deleting any non-leaf vertices with out-degree zero until there are no such vertices, and then taking the full simplification of the resulting directed graph.

Now iteratively repeat this process. That is, for $i \geq 1$, suppose that u_i is a vertex of G^i_A with in-degree one and out-degree one, but $u_i \neq p_a$. Construct G^{i+1}_A and H^{i+1}_A from G^i_A and H^i_A, respectively, by suppressing u_i and deleting exactly one arc of any resulting pair of parallel arcs. In general, H^i_A can be obtained from G^i_A by deleting (p_a, p_b), and $\mathcal{N}'_A$ can be obtained from H^i_A by repeatedly deleting any non-leaf vertices with out-degree zero until there are no such vertices, and then taking the full simplification of the resulting directed graph. Eventually, after, say k, iterations, we construct G^k_A and H^k_A where, except possibility p_a, there is no vertex of G^k_A with in-degree one and out-degree one.

If $a \in A$, then p_a does not have in-degree one and out-degree one in G^k_A, so G^k_A has no vertices of in-degree one and out-degree one (and thus, no pair of parallel arcs). Therefore $G^k_A = \mathcal{N}_A$. Thus, as H^k_A is obtained from G^k_A by deleting (p_a, p_b) and $a \in A$, it follows that $\mathcal{N}'_A$ is obtained from $\mathcal{N}_A$ by deleting (p_a, p_b) and suppressing p_a and p_b. Hence (ii) holds. Therefore we may now assume $a \notin A$.

Since $a \notin A$, the vertex p_a has in-degree one and out-degree one in G^k_A, and p_a has out-degree zero in H^k_A. Let p be the parent of p_a in G^k_A. Construct G^{k+1}_A from G^k_A by suppressing p_a, and construct H^{k+1}_A from H^k_A by deleting p_a. Observe that H^{k+1}_A can be obtained from G^{k+1}_A by deleting (p, p_b). If $G^{k+1}_A = \mathcal{N}_A$, then p_b is a vertex of G^{k+1}_A, and H^{k+1}_A can be obtained from $\mathcal{N}_A$ by deleting (p, p_b). Therefore, $\mathcal{N}'_A$ can be obtained from $\mathcal{N}_A$ by deleting (p, p_b), repeatedly deleting non-leaf vertices of out-degree zero until there are no such vertices, and then taking the full simplification of the resulting directed graph. Thus (iv) holds.

If $G_A^{k+1} \neq \mathcal{N}_A$, then G_A^{k+1} has either a vertex of in-degree one and out-degree one, or a pair of parallel arcs. By the construction of G_A^{k+1}, the only possibility is that p has a pair of outgoing parallel arcs to p_b. In this case, $\mathcal{N}_A$ is obtained from G_A^{k+1} by deleting one of these arcs to p_b and suppressing p and p_b. Since H_A^{k+1} is obtained from G_A^{k+1} by deleting (p, p_b), it follows that $\mathcal{N}_A$ is obtained from H_A^{k+1} by suppressing p and p_b. Hence $\mathcal{N}_A = \mathcal{N}'_A$, thereby establishing (iii) and completing the proof of the lemma. $\square$

4. Proof of Theorem 1.3

This section consists of the proof of Theorem 1.3. We begin by first establishing Theorem 1.3(i).

Proof of Theorem 1.3(i). Let $\mathcal{N}$ be an orchard network on X, where $|X| \geq 3$, and let $\mathcal{N}_0$ be a recoverable phylogenetic network on X such that, up to isomorphism, $Tn(\mathcal{N}_0) = Tn(\mathcal{N})$. The proof is by induction on the sum of the number n of leaves and the number r of reticulations of $\mathcal{N}$. If $r = 0$, then $\mathcal{N}$ is a phylogenetic tree and $Tn(\mathcal{N})$ consists of all rooted triples exhibited by $\mathcal{N}$. Thus, by [19, Theorem 6.4.1], Theorem 1.3(i) holds. Furthermore, if $n = 3$, then $\mathcal{N}$ exhibits exactly one trinet. By Lemma 3.1, orchard networks are recoverable, and so $\mathrm{lsa}(X)$ is the root of $\mathcal{N}$. Therefore this trinet is $\mathcal{N}$ itself. Since, up to isomorphism, $Tn(\mathcal{N}) = Tn(\mathcal{N}_0)$ and $\mathcal{N}_0$ is recoverable, it follows that $\mathcal{N} \cong \mathcal{N}_0$.

Now suppose that $n \geq 4$ and $r \geq 1$, so $n + r \geq 5$, and that the theorem holds for all orchard networks in which the sum of the number of leaves and the number of reticulations is at most $n + r - 1$. Since $\mathcal{N}$ is orchard, $\mathcal{N}$ has either a cherry, say $\{a, b\}$, or a reticulated cherry, say (a, b). Up to isomorphism, $Tn(\mathcal{N}) = Tn(\mathcal{N}_0)$ and so, by Lemma 3.3, either $\{a, b\}$ is a cherry or (a, b) is a reticulated cherry of $\mathcal{N}_0$, respectively. Let $\mathcal{N}'$ and $\mathcal{N}'_0$ be the phylogenetic networks obtained from $\mathcal{N}$ and $\mathcal{N}_0$, respectively, by reducing b or cutting (a, b). By Lemma 1.1, $\mathcal{N}'$ is orchard and, by Lemma 3.2, $\mathcal{N}'_0$ is recoverable.

First suppose that $\{a, b\}$ is a cherry of $\mathcal{N}$ and $\mathcal{N}_0$. By Lemma 3.4, $Tn(\mathcal{N}')$ and $Tn(\mathcal{N}'_0)$ are obtained from $Tn(\mathcal{N})$ and $Tn(\mathcal{N}_0)$, respectively, by excluding those trinets whose leaf set contains b. Therefore, up to isomorphism, as $Tn(\mathcal{N}) = Tn(\mathcal{N}_0)$, we have $Tn(\mathcal{N}') = Tn(\mathcal{N}'_0)$. Thus, by the induction assumption, $\mathcal{N}' \cong \mathcal{N}'_0$. Since $\{a, b\}$ is a cherry of $\mathcal{N}$ and $\mathcal{N}_0$, we deduce that $\mathcal{N} \cong \mathcal{N}_0$.

Now suppose that (a, b) is a reticulated cherry of $\mathcal{N}$ and $\mathcal{N}_0$. We will use Lemma 3.5 to show that the trinets exhibited by $\mathcal{N}'$ can be determined from the trinets exhibited by $\mathcal{N}$. The same argument will also show that the trinets exhibited by $\mathcal{N}'_0$ can be determined from the trinets exhibited by $\mathcal{N}_0$ in the same way. Noting that the leaf set of $\mathcal{N}'$ is X, let $A \subseteq X$, where $|A| = 3$. If $b \notin A$ or $a, b \in A$, then we can construct $\mathcal{N}'_A$ from $\mathcal{N}_A$ as described by Lemma 3.5(i) and (ii), respectively. Thus we may assume that $b \in A$, but $a \notin A$. Say $A = \{b, x, y\}$, where $a \notin \{x, y\}$. Let p_a and p_b denote the parents of a and b, respectively, in $\mathcal{N}$. We next use the trinets exhibited by $\mathcal{N}$ to determine whether or not p_b is a vertex of $\mathcal{N}_A$.

Consider $\mathcal{N}_{\{a,b,x\}}$. By Lemma 3.3, (a, b) is a reticulated cherry of $\mathcal{N}_{\{a,b,x\}}$, and so p_b is a vertex of $\mathcal{N}_{\{a,b,x\}}$. By Lemma 2.6, the phylogenetic network exhibited by $\mathcal{N}$ on $\{b, x\}$ is also the phylogenetic network exhibited by $\mathcal{N}_{\{a,b,x\}}$ on $\{b, x\}$. Thus we can construct $\mathcal{N}_{\{b,x\}}$ from $\mathcal{N}_{\{a,b,x\}}$. In particular, we can decide whether or not p_b is a vertex of $\mathcal{N}_{\{b,x\}}$ from $\mathcal{N}_{\{a,b,x\}}$. Similarly, we can decide whether or not p_b is a vertex of $\mathcal{N}_{\{b,y\}}$ from $\mathcal{N}_{\{a,b,y\}}$. If p_b is a vertex of neither $\mathcal{N}_{\{b,x\}}$ nor $\mathcal{N}_{\{b,y\}}$, then, by Lemma 2.7, p_b is not a vertex of $\mathcal{N}_A$, and so, by Lemma 3.5(iii), $\mathcal{N}'_A = \mathcal{N}_A$. Therefore, we may assume there exists $z \in \{x, y\}$ such that p_b is a vertex of $\mathcal{N}_{\{b,z\}}$, in which case, by Lemma 2.7, p_b is a vertex of $\mathcal{N}_A$. Let p_1 and p_2 be the parents of p_b in $\mathcal{N}_A$. Recalling that $\mathcal{N}'$ is obtained from $\mathcal{N}$ by cutting (a, b), to construct $\mathcal{N}'_A$, we need to determine which of the arcs (p_1, p_b) and (p_2, p_b) to delete from $\mathcal{N}_A$.

Construct $\mathcal{N}_{\{b,z\}}$ from $\mathcal{N}_{\{a,b,z\}}$ in the usual way but with the following modification. Initially mark p_a in $\mathcal{N}_{\{a,b,z\}}$. When suppressing a marked vertex, mark its parent. The end result is $\mathcal{N}_{\{b,z\}}$ with one of the parents of p_b marked. The arc from the marked parent to p_b corresponds to a path in $\mathcal{N}_{\{a,b,z\}}$ from the marked parent to p_b through p_a, and thus the arc we want to delete. On the other hand, we can also construct $\mathcal{N}_{\{b,z\}}$ as the phylogenetic network exhibited by $\mathcal{N}_{A=\{b,x,y\}}$ on $\{b, z\}$. In doing this, mark the vertex p_1. If a marked vertex is suppressed, mark its parent. We again get $\mathcal{N}_{\{b,z\}}$ with a parent of p_b marked, and can compare our two marked parents. By Lemma 3.5(iv), if they are the same vertex, $\mathcal{N}'_A$ is constructed from $\mathcal{N}_A$ by deleting the arc (p_1, p_b), repeatedly deleting vertices of out-degree zero, and taking the full simplification of the resulting directed graph. Otherwise, by Lemma 3.5(iv) again, $\mathcal{N}'_A$ is constructed from $\mathcal{N}_A$ by deleting the arc (p_2, p_b), repeatedly deleting vertices of out-degree zero, and taking the full simplification of the resulting directed graph.

We conclude that the trinets exhibited by $\mathcal{N}'$ (resp. $\mathcal{N}'_0$) can be determined from the trinets exhibited by $\mathcal{N}$ (resp. $\mathcal{N}_0$). Since, up to isomorphism, $Tn(\mathcal{N}) = Tn(\mathcal{N}_0)$ and there is no difference in the way $Tn(\mathcal{N}')$ and $Tn(\mathcal{N}'_0)$ are determined from $Tn(\mathcal{N})$ and $Tn(\mathcal{N}_0)$, respectively, it follows that, up to isomorphism, $Tn(\mathcal{N}'_0) = Tn(\mathcal{N}')$. Therefore, by the induction assumption, $\mathcal{N}' \cong \mathcal{N}'_0$. To construct $\mathcal{N}$ and $\mathcal{N}_0$ from $\mathcal{N}'$ and $\mathcal{N}'_0$, respectively, we need to realise (a, b) as a reticulated cherry. The only way this can be done for $\mathcal{N}'$ (and similarly for $\mathcal{N}'_0$) is by subdividing the arcs into a and b with new vertices p_a and p_b, and then adding an arc from p_a to p_b. Hence $\mathcal{N} \cong \mathcal{N}_0$, and this completes the proof of Theorem 1.3(i). □

4.1. **Algorithm.** Let $\mathcal{N}$ be an orchard network on X, where $|X| \geq 3$. The inductive proof of Theorem 1.3 implies a recursive algorithm that takes X and $Tn(\mathcal{N})$ as its input and returns an orchard network $\mathcal{N}_0$ isomorphic to $\mathcal{N}$. Called Construct Orchard, we next describe this algorithm and give its running time. The correctness of Construct Orchard is essentially established in the proof of Theorem 1.3(i), and so it is omitted.

1. If $Tn(\mathcal{N})$ consists of a single trinet, and so $|X| = 3$, then return this trinet.
2. Else $Tn(\mathcal{N})$ contains at least two trinets, and so $|X| \geq 4$. Find elements $a, b \in X$ such that either $\{a, b\}$ is a cherry of every trinet in $Tn(\mathcal{N})$ whose leaf set contains both a and b, or (a, b) is a reticulated cherry of every trinet in $Tn(\mathcal{N})$ whose leaf set contains both a and b.
3. If $\{a, b\}$ is a cherry of every trinet in $Tn(\mathcal{N})$ whose leaf set contains both a and b, do the following:
   3.1 Let $Tn'(\mathcal{N})$ denote the set of trinets obtained from $Tn(\mathcal{N})$ by removing every trinet whose leaf set contains b.
   3.2 Apply Construct Orchard to input $X' = X - \{b\}$ and $Tn'(\mathcal{N})$, and construct $\mathcal{N}_0$ from the returned orchard network $\mathcal{N}'_0$ by subdividing the arc directed into a with a new vertex p_a, and adjoining a new leaf b to p_a via a new arc (p_a, b).
   3.3 Return $\mathcal{N}_0$.
4. Else (a, b) is a reticulated cherry of every trinet in $Tn(\mathcal{N})$ whose leaf set contains both a and b.
   4.1 Let $Tn'(\mathcal{N})$ denote the set of trinets obtained from $Tn(\mathcal{N})$ by replacing each trinet $\mathcal{N}_A \in Tn(\mathcal{N})$ in which $b \in A$ with the trinet $\mathcal{N}'_A$ constructed as follows:
      4.1.1 If $a \in A$, construct $\mathcal{N}'_A$ from $\mathcal{N}_A$ by deleting the reticulation arc of (a, b) and suppressing the two resulting vertices of in-degree one and out-degree one.
      4.1.2 Else $A = \{b, x, y\}$ for some distinct $x, y \in X - \{a, b\}$. Set p_x (resp. p_y) to be the parent of a in $\mathcal{N}_{\{a,b,x\}}$ (resp. $\mathcal{N}_{\{a,b,y\}}$), and set p'_x (resp. p'_y) to be the parent of b in $\mathcal{N}_{\{a,b,x\}}$ (resp. $\mathcal{N}_{\{a,b,y\}}$). Create a new directed graph G_x (resp. G_y) from $\mathcal{N}_{\{a,b,x\}}$ (resp. $\mathcal{N}_{\{a,b,y\}}$) by deleting a and taking the full simplification. Each time p_x (resp. p_y) is suppressed during this process, set p_x (resp. p_y) to be the parent of the suppressed vertex instead.
         4.1.2.1 If neither p'_x nor p'_y is a vertex of G_x and G_y, respectively, then choose $\mathcal{N}'_A$ to be $\mathcal{N}_A$.
         4.1.2.2 Else there is an element $z \in \{x, y\}$ such that p'_z is a vertex of G_z. Let $\{z, z'\} = \{x, y\}$. Denote the parent

of b in $\mathcal{N}_A$ by p_b, and let p_1 and p_2 be the parents of p_b in $\mathcal{N}_A$. Create a new directed graph G'_z from $\mathcal{N}_A$ by deleting every vertex of $\mathcal{N}_A$ whose only leaf descendant is z' and taking the full simplification. Each time p_1 is suppressed during this process, set p_1 to be the parent of the suppressed vertex instead.

4.1.2.3 Compare p_z and p_1 in the isomorphic directed graphs G_z and G'_z. If p_z and p_1 are the same vertex, then construct $\mathcal{N}'_A$ from $\mathcal{N}_A$ by deleting (p_1, p_b), repeatedly deleting vertices of out-degree zero, and then taking the full simplification. Else construct $\mathcal{N}'_A$ from $\mathcal{N}_A$ by deleting (p_2, p_b), repeatedly deleting vertices of out-degree zero, and then taking the full simplification.

4.2 Apply Construct Orchard to input X and $Tn'(\mathcal{N})$, and construct $\mathcal{N}_0$ from $\mathcal{N}'_0$ by subdividing the arcs directed into a and b with new vertices p_a and p_b, respectively, and adjoining p_a and p_b via a new arc (p_a, p_b).

4.3 Return $\mathcal{N}_0$.

We now consider the running time of Construct Orchard.

Proof of Theorem 1.3(ii). The algorithm takes as input a set X and the set $Tn(\mathcal{N})$ of trinets of an orchard network $\mathcal{N}$ on X. If $Tn(\mathcal{N})$ consists of a single trinet, then Construct Orchard runs in constant time. If $Tn(\mathcal{N})$ contains at least two trinets, and so $|X| \geq 4$, the algorithm begins by finding a 2-element subset $\{a, b\}$ of X such that either $\{a, b\}$ is a cherry of every trinet in $Tn(\mathcal{N})$ whose leaf set contains both a and b, or (a, b) is a reticulated cherry of every trinet in $Tn(\mathcal{N})$ whose leaf set contains both a and b. There at most $|X|^2$ choices for a 2-element subset of X. Since there are $O(|X|^3)$ trinets and deciding if $\{a, b\}$ is a cherry, or (a, b) or (b, a) is a reticulated cherry of a trinet takes constant time, the running time of Step 2 of Construct Orchard takes $O(|X|^5)$ time. Once such a 2-element subset is found, the algorithm constructs a new set $Tn'(\mathcal{N})$ of trinets from $Tn(\mathcal{N})$. In the worst possible instance, the longest running part of this process is when, (a, b) say, is a reticulated cherry of every trinet of $Tn(\mathcal{N})$ whose leaf set contains both a and b, and Step 4.1 is invoked.

Let V denote the vertex set of $\mathcal{N}$. Now, $Tn'(\mathcal{N})$ is obtained from $Tn(\mathcal{N})$ by modifying the trinets $\mathcal{N}_A$ of $Tn(\mathcal{N})$ whose leaf set contains b. Thus there are at most $|X|^2$ such trinets to consider. In terms of running time, the longest part of Step 4.1 is when $A = \{b, x, y\}$, where $a \not\in \{x, y\}$, and Step 4.1.2 is invoked. The directed graphs G_x and G_y take $O(|V|^2)$ time to construct from $\mathcal{N}_{\{a,b,x\}}$ and $\mathcal{N}_{\{a,b,y\}}$, respectively. After that, Step 4.1.2.1 takes constant time. If Step 4.1.2.2 is called, determining z takes constant time and constructing G'_z from $\mathcal{N}_A$, where $z \in \{x, y\}$, takes $O(|V|^2)$ time. In Step 4.1.2.3, the directed graphs G_z and G'_z are compared to decide whether p_z and p_1 are the same vertex. This comparison takes $O(|V|^2)$ time and, regardless of the decision, the resulting construction of $\mathcal{N}'_A$ takes $O(|V|^2)$ time. Hence the running time to complete Step 4.1 is $O(|X|^2|V|^2)$.

With Step 4.1 completed, Steps 4.2 and 4.3 each take constant time. It follows that each iteration takes $O(|X|^5 + |X|^2|V|^2)$ time. When recursing, the input to the recursive call is either a set $X' = X - \{b\}$ and a set $Tn'(\mathcal{N})$ of trinets of an orchard network on $|X| - 1$ leaves and r reticulations, or a set X and a set $Tn'(\mathcal{N})$ of trinets of an orchard network on $|X|$ leaves and $r - 1$ reticulations, where r is the number of reticulations of $\mathcal{N}$. Therefore the total number of iterations is $O(|X| + r)$. Since $|V| = 2(|X| + r) - 1$ [18], it follows that the total number of iterations is $O(|V|)$. Hence Construct Orchard completes in

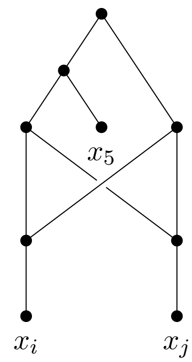
$$O(|V|(|X|^5 + |X|^2|V|^2))$$

time, that is, in $O(|V|^6)$ time as $|X| \leq |V|$. This completes the proof of Theorem 1.3(ii). □

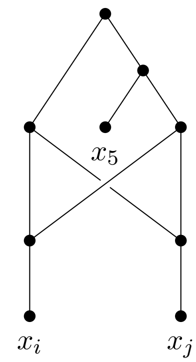
5. An Example

In this section, we show that the largest value of k such that all recoverable level-k phylogenetic networks $\mathcal{N}$ are encoded by $Tn(\mathcal{N})$ is at most 3. Consider the two level-4 phylogenetic networks $\mathcal{N}_1$ and $\mathcal{N}_2$ on $\{x_1, x_2, x_3, x_4, x_5\}$ shown in Fig. 3. Both $\mathcal{N}_1$ and $\mathcal{N}_2$ are recoverable, but $\mathcal{N}_1$ is not isomorphic to $\mathcal{N}_2$ as the vertex which is an ancestor of x_1 and x_2, and no other leaves, is a descendant of the parent of x_5 in $\mathcal{N}_1$, but is not a descendant of the parent of x_5 in $\mathcal{N}_2$.

Consider the path graphs of $\mathcal{N}_1$ and $\mathcal{N}_2$ on $\{x_1, x_2, x_3, x_4\}$. In each of these graphs, the parent of x_5 has in-degree one and out-degree one, and so this vertex will be suppressed in every trinet of $Tn(\mathcal{N}_1)$ and $Tn(\mathcal{N}_2)$ not containing x_5. Since the two graphs obtained after suppressing the parent of x_5 from the path graphs of $\mathcal{N}_1$ and $\mathcal{N}_2$ on $\{x_1, x_2, x_3, x_4\}$ are isomorphic, it follows that any trinet of $Tn(\mathcal{N}_1)$ and $Tn(\mathcal{N}_2)$ on the same leaf set not containing x_5 are isomorphic. Furthermore, every other trinet of $\mathcal{N}_1$ and $\mathcal{N}_2$ is isomorphic to the trinet shown in Fig. 4(i) and (ii), respectively, where $\{i, j\} \subseteq \{1, 2, 3, 4\}$. Since the trinets in this figure are isomorphic, it follows that $\mathcal{N}_1$ is not encoded by $Tn(\mathcal{N}_1)$.



(i) Trinet exhibited by $\mathcal{N}_1$ on $\{x_i, x_j, x_5\}$ for all $\{i, j\} \subseteq \{1, 2, 3, 4\}$.



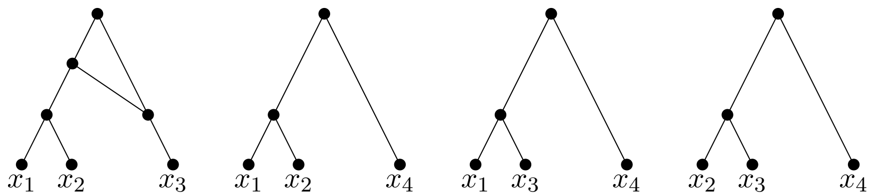
(ii) Trinet exhibited by $\mathcal{N}_2$ on $\{x_i, x_j, x_5\}$ for all $\{i, j\} \subseteq \{1, 2, 3, 4\}$.

FIGURE 4. The phylogenetic networks $\mathcal{N}_1$ and $\mathcal{N}_2$ as shown in Fig. 3 exhibit, up to isomorphism, the same trinets on $\{x_i, x_j, x_5\}$, where $\{i, j\} \subseteq \{1, 2, 3, 4\}$.

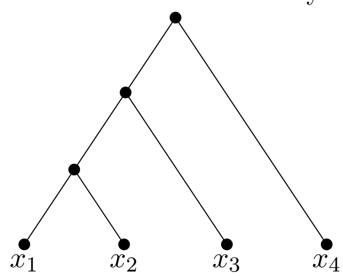
6. DISCUSSION

For a non-negative integer k, a phylogenetic network $\mathcal{N}$ is *level-k* if each biconnected component of $\mathcal{N}$ has at most k reticulations. It is shown in [13] that all recoverable (binary) level-2 networks are encoded by their sets of trinets. The authors comment that the approach taken to establish this uniqueness result does not extend to level-k networks, where $k \geq 4$. Curiously, the counterexample consists of two level-4 networks. This raises the question of whether a recoverable level-3 network is encoded by the set of trinets it exhibits.

The algorithm, CONSTRUCT ORCHARD, described in Section 4.1 takes as input the set of trinets of an orchard network, and outputs, up to isomorphism, the unique orchard network which exhibits the trinets in the input. However, the algorithm does not extend to decide whether an arbitrary inputted set of trinets is exhibited by an orchard network. For example, consider the set of trinets shown in Fig. 5(i). In this example, $X = \{x_1, x_2, x_3, x_4\}$ and we have exactly one trinet for each subset of X of size three. Applying CONSTRUCT ORCHARD to this set, we initially identify x_1 and x_2 as the leaves of a cherry, and discard every trinet containing x_2. After this, there is only one trinet remaining, so the leaf x_2 is reattached to this trinet to output the phylogenetic network shown in Fig. 5(ii). However, this network does not contain any reticulations, and so it does not exhibit the trinet on $\{x_1, x_2, x_3\}$. It remains an open problem to find a polynomial-time algorithm which, given a set of trinets, can determine whether or not there is an orchard network that exhibits those trinets.



(i) A set of trinets which are not exhibited by an orchard network.



(ii) The network produced when Construct Orchard is applied to the set of trinets in (i).

Figure 5. Applying Construct Orchard to the set of trinets shown in (i) outputs a phylogenetic network which does not exhibit all of the trinets.

Acknowledgments

We thank the referees for their comments.

References

[1] Aho AV, Sagiv Y, Szymanski TG, Ullman JD (1981) Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. SIAM Journal on Computing 10:405–421

[2] van Bemmelen J (2020) Reconstructing tree-child networks from their exhibited trinets. MSc thesis, Vrije Universiteit Amsterdam

[3] Bininda-Emonds ORP (2004) The evolution of supertrees. Trends in Ecology and Evolution 19:315–322

[4] Cardona G, Rosselló F, Valiente G (2009) Comparison of tree-child phylogenetic networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics 6:552–569

[5] Erdős PL, Semple C, Steel M (2019) A class of phylogenetic networks reconstructable from ancestral profiles. Mathematical Biosciences 313:33–40

[6] Felsenstein J (2004) Inferring Phylogenies, Sinauer Associates, Sunderland, MA

[7] Gambette P, Huber KT (2012) On encodings of phylogenetic networks of bounded level. Journal of Mathematical Biology 65:157–180

[8] Huber KT, Moulton V (2013) Encoding and constructing 1-nested phylogenetic networks with trinets. Algorithmica 66:714–738

[9] Huber KT, van Iersel L, Moulton V, Wu T (2015) How much information is needed to infer reticulate evolutionary histories? Systematic Biology 64:102–111

[10] Huber KT, van Iersel L, Moulton V, Scornavacca C, Wu T (2017) Reconstructing phylogenetic level-1 networks from nondense binet and trinet sets. Algorithmica 77:173–200

[11] Huson DH, Rupp R, Scornavacca C (2010) Phylogenetic Networks: Concepts, Algorithms and Applications, Cambridge University Press, London

[12] van Iersel L, Kelk S (2011) Constructing the simplest possible phylogenetic network from triplets. Algorithmica 60:207–235

[13] van Iersel L, Moulton M (2014) Trinets encode tree-child and level-2 phylogenetic networks. Journal of Mathematical Biology 68:1707–1729

[14] Janssen R, Murakami Y (2020) On cherry picking and network containment. arXiv:1812.08065v2

[15] Jansson J, Nguyen NB, Sung WK (2006) Algorithms for combining rooted triplets into a galled phylogenetic network. SIAM Journal on Computing 35:1098–1121

[16] Jansson J, Sung WK (2006) Inferring a level-1 phylogenetic network from a dense set of rooted triplets. Theoretical Computer Science 363:60–68

[17] Linz S, Semple C (2020) Caterpillars on three and four leaves are sufficient to reconstruct binary normal networks. Journal of Mathematical Biology 81:961–980

[18] McDiarmid C, Semple C, Welsh D (2015) Counting phylogenetic networks. Annals of Combinatorics 19:205–224

[19] Semple C, Steel M (2003) Phylogenetics, Oxford University Press, New York

[20] Willson SJ (2010) Properties of normal phylogenetic networks. Bulletin of Mathematical Biology 72:340–358

[21] Willson SJ (2011) Regular networks can be uniquely constructed from their trees. IEEE/ACM Transactions on Computational Biology and Bioinformatics 8:785–796

School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand

Email address: `charles.semple@canterbury.ac.nz`

School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand

Email address: `gerry.toft@pg.canterbury.ac.nz`