Mathematical Biology

Phylogenetic network classes through the lens of expanding covers

Andrew Francis[1,2] · **Daniele Marchei[1,3]** · **Mike Steel[4]**

Abstract

It was recently shown that a large class of phylogenetic networks, the 'labellable' networks, is in bijection with the set of 'expanding' covers of finite sets. In this paper, we show how several prominent classes of phylogenetic networks can be characterised purely in terms of properties of their associated covers. These classes include the tree-based, tree-child, orchard, tree-sibling, and normal networks. In the opposite direction, we give an example of how a restriction on the set of expanding covers can define a new class of networks, which we call 'spinal' phylogenetic networks.

1 Introduction

Phylogenetic networks can provide more complete representations of evolutionary relationships among species than possible with a simple phylogenetic tree (Bapteste et al. 2013; Huson et al. 2010). Although a single tree can accurately show ancestral speciation events (splitting of lineages), it cannot display reticulate evolution (where the flow of genomic information follows the merging of ancestral lineages). Well-known reticulate processes in biology include hybridization, horizontal gene

---

✉ Andrew Francis
   a.francis@unsw.edu.au

1   Centre for Research in Mathematics and Data Science, Western Sydney University, Sydney, Australia

2   Present Address: School of Mathematics and Statistics, University of New South Wales, Sydney, Australia

3   Computer Science, University of Camerino, Camerino, Italy

4   Biomathematics Research Centre, University of Canterbury, Christchurch, New Zealand

transfer, recombination, and endosymbiosis, in both the recent and distant past. By contrast, rooted phylogenetic networks can explicitly and simultaneously display both speciation and reticulate evolution. As a result, the mathematical and algorithmic investigation of phylogenetic networks has become a highly active field over the last ∼15 years, and numerous classes of networks have been defined and studied (Kong et al. 2022).

In this paper, we show how a recently introduced correspondence for a large class of phylogenetic networks (the *labellable* networks (Francis and Steel 2023)) can be used to characterise a number of widely used other classes of network. Classes of network have been introduced for a variety of reasons, but usually in order to capture some feature that seems biologically important, or because they are mathematically convenient. Their definitions typically involve constraints on their structures as graphs. For instance, tree-child networks are those for which no vertex has only reticulations as its children, whereas tree-based networks are those that can be constructed from a base tree by adding additional edges between the tree edges.

The class of labellable networks contains many commonly studied classes. They have been shown to correspond to a set of covers of finite sets that satisfy a property called "expanding". We explore features of covers arising from networks, and characterise many of the familiar classes in terms of properties of their associated covers. It is to be hoped that encoding network properties in the properties of sets of sets will enable some new directions to be pursued in studying phylogenetic networks.

This paper aims to demonstrate how this encoding of labellable networks into covers may be of broad use in the classification of network classes. Different classes of networks are defined in different ways, and it can be difficult to present a clear hierarchy (there have been several visual attempts, for instance (Kong et al. 2022, Fig. 12) and (Francis and Steel 2023, Fig. 6). Being able to characterise different network classes by the properties of their covers gives a unified framework for defining networks, in the sense that one may add or remove axioms depending on the class of networks one wants to describe. In that sense, moving from one class to another may be just a matter of changing the axioms, providing a potentially useful lens for visualizing the relationships among classes.

We begin by defining what we mean by a phylogenetic network, recalling the key results linking labellable networks with expanding covers (from Francis and Steel (2023)), in Sect. 2. We give some general properties of covers arising from networks, before characterising the classes of tree-based labellable networks (Sect. 3), then tree-child networks (Sect. 4), normal networks (Sect. 5), tree-sibling networks (Sect. 6), and orchard networks (Sect. 7). These are some of the more widely seen classes, and they are amenable to being described in terms of covers. We also demonstrate how the language of covers can allow one to define new classes of network by changing the constraints on the covers: one small change to the constraints defines a new class we call 'spinal' networks, that have an interesting structure (Sect. 8). We finish by discussing some open questions and opportunities for further development.

2 Preliminaries

A *phylogenetic network* on n leaves is a directed acyclic graph with a single vertex of in-degree zero, called the root, and n vertices of in-degree 1 and out-degree zero, labelled by $[n] := \{1, \ldots, n\}$. Note that this includes the possibility of vertices that have in-degree and out-degree both equal to 1, or both strictly greater than 1; such vertices are called *degenerate*. If N has any degenerate vertices, it is said to be a *degenerate* network; otherwise, it is *non-degenerate*. In this paper, all networks are drawn with the root at the top and leaves at the bottom, and all edges are directed with orientation down the page.

If every vertex has in-degree and out-degree at most 2, then the network is said to be *binary*. If N is non-degenerate and binary, then all vertices other than the leaves and root have total degree 3.

Vertices in a network that have in-degree 1 are called *tree vertices*, and those with in-degree greater than 1 are called *reticulate vertices*, or *reticulations*. We will typically use k to denote the number of reticulations in a network, and m to denote the number of non-root vertices in total.

A *labellable* phylogenetic network is one whose vertices can be deterministically labelled according to an algorithm that generalises one for trees (the algorithm for trees is due to Erdős and Székely (1989)) (Francis and Steel 2023). Such networks are characterized topologically by the property that the map from non-leaf vertices to their sets of children is one-to-one (Francis and Steel 2023, Thm.3.3).

A *cover* of a finite set A is a set of non-empty subsets of A whose union is A. If the sets in a cover of A are pairwise disjoint, it is called a *partition* of A. The cardinality $|\mathcal{C}|$ of a cover $\mathcal{C}$ is the number of sets it contains. We use $||\mathcal{C}||$ to denote the number of distinct elements in the sets in $\mathcal{C}$, that is, $||\mathcal{C}|| := |\bigcup_{C_i \in \mathcal{C}} C_i|$.

Recall the definition from Francis and Steel (2023):

Definition 2.1 A cover $\mathcal{C}$ of $[m]$ is *expanding* if, for $n = m - |\mathcal{C}| + 1$, it satisfies:

(1) No element of $[n]$ appears more than once, and
(2) For $i = 1, \ldots, |\mathcal{C}|$, the cover contains at least i subsets of $[n + i - 1]$.

Theorem 2.2 (Francis and Steel 2023, Thm. 4.4) *The class of labellable phylogenetic networks is in bijection with the collection of expanding covers of finite sets.*

The map from a labellable phylogenetic network to its expanding cover takes each non-leaf vertex to the set of labels of its children. That is, sets in the cover are sets of labels of sibling vertices sharing a parent. The map from an expanding cover $\mathcal{C}$ to a labellable network is a constructive map that first establishes the number of leaves in the network via the following formula (Francis and Steel 2023, Lemma 4.1):

$$n = ||\mathcal{C}|| - |\mathcal{C}| + 1.$$

The construction of the network then begins with n isolated leaf vertices, and adds parent vertices to sets of vertices present in the growing network, and lexicographically minimal of those in $\mathcal{C}$. The expanding conditions ensure that there is always such a set, and that the map is well-defined. For examples of this construction the reader is referred to Francis and Steel (2023).

While the condition for a cover to be expanding may seem artificial, and it certainly restricts from the collection of all covers of a set, it can be seen as a natural extension of the notion of partitions. In particular, it turns out that all partitions are expanding covers.

Lemma 2.3 *Every set partition is an expanding cover.*

Proof Let π be a partition of $[m]$ with $\ell = |\pi|$ blocks, and set $n = m - \ell + 1$. Two conditions define an expanding cover. The first is that elements of $\{1, \ldots, n\}$ are not repeated in π, which is satisfied by virtue of π being a partition. The second is that for each $i = 1, \ldots, \ell$, π contains at least i subsets of $[n + i - 1]$, and we prove this by induction on i.

First, consider the base case $i = 1$. We need to show that there is at least one set in π that is a subset of $[n]$. There are $\ell = m - n + 1$ pairwise disjoint subsets of $[m]$ in π, and there are $m - n$ integers in $[m]$ that are not in $[n]$. Therefore, there must be at least one set in π that does not contain an element of $\{n + 1, \ldots, m\}$ and is thus a subset of $[n]$, as required.

Suppose that for $i = k$, π contains at least k subsets of $[n + k - 1]$. We would like to show that π contains at least $k + 1$ subsets of $[n + (k + 1) - 1] = [n + k]$. The proof proceeds in the same manner as the case of $i = 1$.

First remove k subsets of $[n + k - 1]$ from π, so that π has $\ell - k$ sets remaining. We need to show at least one remaining set is entirely contained within $[n + k]$. There are $m - (n + k)$ integers in π that are *not* in $[n + k]$, and $\ell - k = (m - n + 1) - k = m - (n + k) + 1$ sets are available. Therefore, at least one must not contain any element outside $[n + k]$, as required. □

Since all set partitions are expanding covers, we can ask what sort of networks have partitions as their covers. A partition has a single occurrence of each integer, which means that each vertex of the network (each label) has a single set of siblings. In other words, the network has no reticulations, and thus is a tree. This correspondence of trees with partitions allows trees with degenerate vertices (i.e., vertices with in-degree and out-degree 1). In this way, the correspondence for partitions is closer to the result of Erdős and Székely (1989) than the non-degenerate framework that has partitions in bijection with phylogenetic forests in Francis and Jarvis (2022).

The lexicographic order on sets (given by $A \prec B$ if $A \subset B$ or $\min(B \setminus A) < \min(B \setminus A)$) that helps determine the labelling sequence is not always the ordering of sets used to label the internal vertices of the network; that sequence is given by the *labelling order*, which is defined as follows (Francis and Steel 2023, Sect. 4):

Definition 2.4 The *labelling order* for an expanding cover $\mathcal{C}$ is determined by the following procedure.

(1) For $i = 1, \ldots, |\mathcal{C}|$,

    (a) Set C_i to be the minimal set in $(\mathcal{C}, \prec)$ contained in $[n + i - 1]$; and
    (b) Redefine $\mathcal{C} = \mathcal{C} \setminus \{C_i\}$.

(2) Output the sequence $C_1, \ldots, C_{|\mathcal{C}|}$.

Fig. 1 A labellable phylogenetic network N with cover
1 | 2 | 3 | 4, 5 | 6, 8 | 6, 7 | 7, 8 | 11, 12 | 9, 13 | 10, 13 | 14, 15
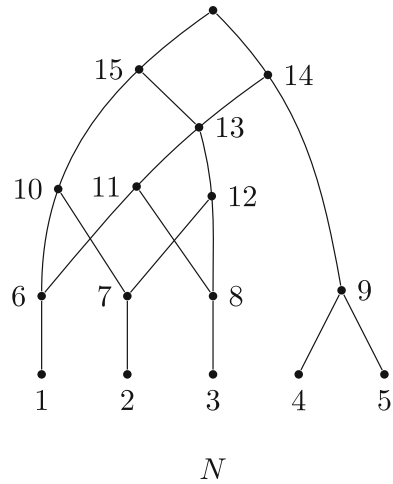
This order is necessary to establish conditions on a cover that give non-degenerate networks, for instance, and we will use it later in the present paper to describe *normal networks* (in Sect. 5) and *orchard networks* (Sect. 7).

Given a cover in labelling order, we can label every subset in position $1 \leq i < |\mathcal{C}|$ by $i + n$, whereas the last subset is labelled ρ for the *root*. In this way, the label for each subset corresponds to the label of its parent in the corresponding labellable network.

For example, the labelling order of the cover for the network shown in Fig. 1 is

$$1 \mid 2 \mid 3 \mid 4, 5 \mid 6, 7 \mid 6, 8 \mid 7, 8 \mid 11, 12 \mid 9, 13 \mid 10, 13 \mid 14, 15.$$

The first set gives rise to the vertex label $n + 1 = 6$, the second gives rise to 7, and so on. We can represent this more explicitly as follows, adding ρ to denote the root:

$$\{1\}_6, \{2\}_7, \{3\}_8, \{4, 5\}_9, \{6, 7\}_{10}, \{6, 8\}_{11}, \{7, 8\}_{12}, \{11, 12\}_{13}, \{9, 13\}_{14},$$
$$\{10, 13\}_{15}, \{14, 15\}_{\rho}.$$

2.1 Features of vertices in networks and their covers' properties

Many features of vertices in networks have direct translations into the language of covers, and we present some of them in Table 1. The first two lines of the table are clear: non-root vertices on a network are labelled by the labelling algorithm and those labels appear as integers in $[m]$, and the leaves are labelled by integers in $[n]$. The other lines of the table can be justified as follows.

A tree vertex in a network is a vertex with in-degree 1, which means it has only one parent and, therefore, is in only one set of sibling vertices. This set of sibling vertices could have any size greater than or equal to one, but it is only a single set. A reticulation vertex, on the other hand, has strictly more than one parent, and thus has two or more sets of siblings. No two vertices in a labellable network have the same set of children (Francis and Steel 2023, Thm 3.3), so the label of a reticulation vertex

Table 1 A translation of features of vertices in a labellable network with n leaves and m non-root vertices into features of the corresponding expanding cover

Network	Cover
Non-root vertex	An integer in $[m]$
Leaf	An integer in $[n]$
Tree vertex	An integer contained in just one subset
Reticulation vertex	An integer contained in more than one subset
In-degree of x	The number of subsets that contain x
Out-degree of x	Size of the subset with label x in the labelling order
Parents of x	All the subsets that contain x
Siblings of x	All the other integers contained in the subsets that contain x
Children of x	The subset with label x in the labelling order

will appear in at least two sets in the cover. The other translations in Table 1 follow immediately.

Throughout this paper, we will add additional translations to the table, with a summary table given in the Discussion.

3 Tree-based networks

A phylogenetic network is *tree-based* if it has a spanning tree whose leaves are those of the network (Francis and Steel 2015). Such a spanning tree is called a *base tree* for the network. Typically, a tree-based network can have many base trees. A similar notion that we will discuss is that of a *support tree* for a network. A support tree is a base tree but with additional degree 2 vertices where additional arcs are joined to complete the network. That is, the set of vertices in the support tree and the network are identical.

Unlike the other classes that we consider in the coming sections, not all tree-based networks are labellable, but neither are all labellable networks tree-based (Francis and Steel 2023). There is thus a non-trivial intersection of the two classes, and this intersection contains many other classes, including orchard, tree-child, and normal networks (Francis and Steel 2023). In the binary case, the tree-based networks that are labellable can be characterised in terms of their structural properties, as those for which no two reticulate vertices have the same sets of parents (Francis and Steel 2023, Thm. 6.3). In this section, we provide a new characterisation of the tree-based labellable networks in terms of their covers, and the existence of an "embedded" partition, in Theorem 3.2.

We say that a partition π *embeds* in $\mathcal{C}$ if there is a one-to-one map from π to $\mathcal{C}$ that maps each set A in π to a set A' in $\mathcal{C}$ so that $A \subseteq A'$. A partition π *fully embeds* in a cover $\mathcal{C}$ if π embeds in $\mathcal{C}$ and $|\pi| = |\mathcal{C}|$.

Recall from Sect. 2 that every partition of $[m]$ is an expanding cover. It is straightforward to see that every expanding cover has a partition that embeds into it, as follows.

Lemma 3.1 *For every expanding cover $\mathcal{C}$ of $[m]$, there is a partition of $[m]$ that embeds into $\mathcal{C}$.*

Proof If all repeats of integers in $\mathcal{C}$ are deleted, so that there is one occurrence of each integer, then the result is a partition of $[m]$.   □

Any partition obtained in this way will be expanding, according to Lemma 2.3. Note, however, that each such partition may not have the same number of sets as the cover, and therefore may be expanding for a different value of n.

The notion of embedding a partition into a cover turns out to help characterise tree-based networks.

Theorem 3.2 *An expanding cover $\mathcal{C}$ of $[m]$ corresponds to a tree-based network if and only if there is a partition π of $[m]$ that fully embeds in $\mathcal{C}$.*

Proof Suppose N is a tree-based network with expanding cover $\mathcal{C}$ of $[m]$. We will show that $\mathcal{C}$ has an embedded partition with length $|\mathcal{C}|$.

Label the vertices of N according to the labelling algorithm. This labelling gives rise to the expanding cover whose sets are the children of non-leaf vertices in N. Choose a support tree T for N, keeping the labels of the vertices from N. The labels of vertices in T are thus precisely $[m]$. Note that all vertices of N are present in T, but that each non-root vertex in T has in-degree 1. The set of children of each vertex in T is a subset of the set of children for the corresponding vertex in N.

Construct the cover for T using the inherited labelling of vertices, forming sets of labels of vertices that are the children of the same non-leaf vertex. Each set thus formed is a subset of one of the sets in the cover for N, because the children of vertex i in N are a subset of the children of vertex i in T. Each set is non-empty because the only leaves in the base tree are those of N. The cover for T contains no repeated integers because T is a tree and there are no vertices with in-degree greater than 1. Thus, the cover for T with the labelling inherited from N is a partition of $[m]$ of length $|\mathcal{C}|$, as desired.

Note that the labels on the vertices in T are those inherited from N. They are not the same as the labels that would be put on vertices by the labelling algorithm applied to T. Thus the partition obtained from T is not the same as the partition that would be obtained by labelling T directly.

For the reverse direction, suppose that the expanding cover $\mathcal{C}$ has a partition π that embeds into $\mathcal{C}$, and has length $|\mathcal{C}|$. We will show that the corresponding network is tree-based.

Let N be the network constructed by using $\mathcal{C}$. The partition π embeds in $\mathcal{C}$, so there is a one-to-one map from π to $\mathcal{C}$ that maps each set A in π to a set A' in $\mathcal{C}$ such that $A \subseteq A'$. The sets in $\mathcal{C}$ correspond to vertices in N and give the set of children of each vertex. For each non-leaf vertex in N, $A' \in \mathcal{C}$ labels its children, and there is a corresponding set $A \in \pi$ that is its pre-image in the embedding of π into $\mathcal{C}$, with $A \subseteq A'$.

For the non-leaf vertex in N with children A', delete the edges in N between it and the vertices labelled by $A' \setminus A$, and repeat this for each non-leaf vertex in N. The resulting network now has vertices whose children are labelled by the sets in π. We

claim that this resulting network $\hat{N}$ is a support tree for N. We need to show that $\hat{N}$ is a spanning tree whose leaves are those of N.

First, $\hat{N}$ contains all vertices of N, since only edges were removed. Second, it is a tree, since no label is repeated in π by virtue of it being a partition, and therefore no vertex has more than one parent. Third, each vertex v that is not a leaf of N has at least one child, since v has a non-empty set of children whose labels are a set in π (the length of π is $|\mathcal{C}|$), and thus the only leaves of $\hat{N}$ are those of N.

Thus, $\hat{N}$ is a support tree for N, and so N is tree-based, as required.   □

This result gives an alternative way to characterise support trees for a tree-based network, as follows.

Corollary 3.3 *The set of support trees for a tree-based network N is in bijection with the set of full embeddings of partitions in the expanding cover for N.*

Proof As seen in the proof of Theorem 3.2, each support tree for N gives rise to a full embedding of a partition in the cover for N. Conversely, every full embedding of a partition into the cover for N constitutes a choice of parent for each reticulation vertex (any element that appears more than once in the cover), and thus gives a support tree for N.   □

Note that it is possible for a particular partition to embed in more than one way into a cover, and that each such embedding gives a different support tree for the network.

Example 3.4 Figure 1 shows a network with cover $\mathcal{C} = 1 \mid 2 \mid 3 \mid 4, 5 \mid 6, 8 \mid 6, 7 \mid 7, 8 \mid 11, 12 \mid 9, 13 \mid 10, 13 \mid 14, 15$. The embeddings of partitions into $\mathcal{C}$ can be enumerated as follows. First, consider the elements that appear exactly once in $\mathcal{C}$: 1, 2, 3, 4, 5, 9, 10, 11, 12, 14, 15. These must appear in the partition where they are in the cover (one appearance means only one possibility), so any embedded partition into $\mathcal{C}$ has form

$$1 \mid 2 \mid 3 \mid 4, 5 \mid \_, \_ \mid \_, \_ \mid \_, \_ \mid 11, 12 \mid 9, \_ \mid 10, \_ \mid 14, 15.$$

Consider then the integer 6, which, in the partition, must be either embedded into the set $\{6, 7\}$ or $\{6, 8\}$. If the former, then 8 must embed into the latter; otherwise, the partition would not be a full embedding (we cannot allow empty sets), which forces 7 to embed into the set $\{7, 8\}$. In short, the three sets $6, 8 \mid 6, 7 \mid 7, 8$ can only have embedded either $6 \mid 7 \mid 8$ or $8 \mid 6 \mid 7$. These amount to the same partition but two distinct embeddings that give different support trees because they correspond to different choices of child for each vertex. The other choice for embedding a partition involves the placement of 13, which can either be with 9 or 10.

Thus, there are four full embeddings of partitions π_i into $\mathcal{C}$, as follows:

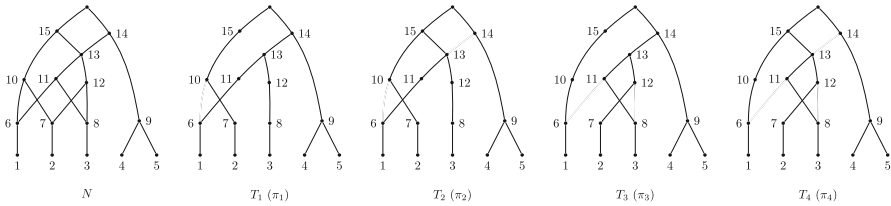| $\mathcal{C}$: | 1 | 2 | 3 | 4,5 | 6,8 | 6,7 | 7,8 | 11,12 | 9,13 | 10,13 | 14,15 |
π_1 :	1	2	3	4,5	6	7	8	11,12	9,13	10	14,15
π_2 :	1	2	3	4,5	6	7	8	11,12	9	10,13	14,15
π_3 :	1	2	3	4,5	8	6	7	11,12	9,13	10	14,15
π_4 :	1	2	3	4,5	8	6	7	11,12	9	10,13	14,15

Fig. 2 The tree-based labellable network N and the four support trees given by the four embeddings $\pi_1, \ldots, \pi_4$, as described in Example 3.4

Table 2 Translation of concepts arising in tree-based labellable networks	Network	Cover
	Spanning tree	A partition embedded in $\mathcal{C}$
	Support tree	A full embedding of a partition in $\mathcal{C}$

The support trees corresponding to these embeddings of partitions are shown in Fig. 2.

3.1 Support trees for a binary tree-based network

Support trees for binary tree-based networks have been counted in earlier work (Pons et al. 2019; Hayamizu 2021), building on an upper bound from Jetten (2015). Covers provide an alternative and clear approach that replicates these results.

For instance (and without giving details of all the components of the statement):

Theorem 3.5 (Pons et al. 2019, Thm. 8) *For a binary tree-based network N, the number of support trees is:*

$$2^c \times \prod_{P \in \pi(\mathcal{J}_N)} \frac{1}{2}(v(P) + 1),$$

where

- *$\mathcal{J}_N$ is a bipartite graph derived from N with parts given by the set of vertices with a reticulate child, and reticulations without a reticulate parent,*
- *c is the number of cycle components in $\mathcal{J}_N$,*
- *$\pi(\mathcal{J}_N)$ is the set of path components in $\mathcal{J}_N$ without an omnian terminal vertex (an omnian vertex in a network is one whose children are all reticulations), and*
- *$v(P)$ is the number of vertices in the path component P.*

This is an explicit formula based on features of the network, using a representation of key features in the bipartite graph $\mathcal{J}_N$ in particular.

It was subsequently demonstrated that this formula relied on two key structural elements of the network: the number of "crowns" and the lengths of each "M-fence" (Hayamizu 2021, Sect. 5.3). These are types of "zig-zag trails", which are undirected paths of vertices in the network that alternate between tree and reticulation vertices (Zhang 2016). A maximal length zig-zag trail is called a *crown* if it forms a cycle, and is called an *M-fence* if the ends of the path are tree vertices. Crowns and fences arise naturally when looking at the problem through the lens of covers. We are able to obtain, by using covers, a formula that is analogous to that of Theorem 3.5, as follows.

Suppose N is a binary tree-based network. We allow degenerate vertices with in-degree 2 as well as out-degree 2. The cover $\mathcal{C}$ for N then consists of sets of size 1 or 2, and each integer appearing in $\mathcal{C}$ appears either once, if it is a tree vertex (in-degree 1), or twice if it is a reticulation (in-degree 2).

We will now describe an algorithm for obtaining an embedded partition (support tree) from $\mathcal{C}$, and this will allow us to count the number of such support trees.

The sets in $\mathcal{C}$ fall into exactly five categories:

(1) Singletons containing integers appearing once in $\mathcal{C}$,
(2) Singletons containing integers appearing twice in $\mathcal{C}$,
(3) Pairs containing integers each appearing once in $\mathcal{C}$,
(4) Pairs containing integers each appearing twice in $\mathcal{C}$, and
(5) Pairs containing one integer appearing once and the other appearing twice in $\mathcal{C}$.

Sets that contain elements that appear only once in $\mathcal{C}$ must be fully retained in any embedded partition. Thus sets from categories ((1)) and ((3)) must be in the embedded partition, and there is no choice.

Because the partition embeds into $\mathcal{C}$, a set containing a singleton $\{a\}$ in $\mathcal{C}$ must also appear in the embedded partition. Therefore, if $\{a\}$ is in category ((2)), none of the other occurrences of a in other sets in $\mathcal{C}$ can appear in the partition, and we delete them from the sets in the cover. This will create new sets of size 1, and possibly of category ((2)). We repeat this process until all sets in category ((2)) are gone, creating a new cover we denote $\mathcal{C}_1$. Note that $\mathcal{C}_1$ is uniquely determined from $\mathcal{C}$ and embeds into it. Note also that $\mathcal{C}_1$ does not contain any sets in category ((2)) above.

This leaves sets from categories ((4)) and ((5)) to deal with. These sets are connected. If a set is in category ((5)), then one of its elements appears elsewhere, and it can only be in a set from category ((5)) or ((4)). We can thus form sequences of such sets in $\mathcal{C}_1$ by connecting a set from category ((5)) with a sequence of sets from category ((4)) and ending with another set from category ((5)). These sequences are uniquely determined by $\mathcal{C}_1$, and every set from category ((5)) is in precisely one sequence of this form. For example, such sequences are of the form

$$a_0, a_1 \mid a_1, a_2 \mid \cdots \mid a_{t-1}, a_t \mid a_t, a_{t+1}, \tag{1}$$

where a_0 and a_{t+1} do not appear elsewhere in $\mathcal{C}_1$ (note that t could be 1). We call such sequences *fences* (they correspond to the *M*-fences defined above). The notions of crowns and fences for covers are summarized in Table 3.

Let $\mathcal{F}$ denote the set of fences in N. For each fence f, let $r(f)$ denote the number of repeated integers in f, which we call its length. The fence in Eq. (1) has length $r(f) = t$.

A set from category ((4)) may be in a sequence such as the one above, or in a sequence of at least three sets from the same category:

$$a_0, a_1 \mid a_1, a_2 \mid \cdots \mid a_{t-1}, a_t \mid a_t, a_0, \tag{2}$$

where $t \geq 2$. These correspond precisely to the 'crowns' of Hayamizu (2021).

For either fences or crowns, we can count the number of selections of unique elements as follows.

In the case of fences of length t (Eq. (1)), the number of choices is simply $t + 1$, since there are $t + 2$ elements to go into $t + 1$ non-empty sets, so one has two elements and the rest have one element. There are $t + 1$ choices for the set with two elements. For example, with the fence $a, b \mid b, c \mid c, d \mid d, e$, we have $t = 3$ and the choices are:

$$a, b \mid c \mid d \mid e$$
$$a \mid b, c \mid d \mid e$$
$$a \mid b \mid c, d \mid e$$
$$a \mid b \mid c \mid d, e.$$

In the case of a crown, as in Eq. (2), there is only one embedded partition. We have the same number of elements as we have non-empty sets, and so there is only one option for selecting unique elements. Each element forms a singleton. For example, in the crown $a, b \mid b, c \mid c, d \mid d, a$, we have only $a \mid b \mid c \mid d$. However, although there is only one embedded partition, that partition has exactly two distinct embeddings. We could have:

$$a \mapsto \{a, b\}, \ b \mapsto \{b, c\}, \ c \mapsto \{c, d\} \text{ and } d \mapsto \{d, a\}, \text{ or}$$
$$b \mapsto \{a, b\}, \ c \mapsto \{b, c\}, \ d \mapsto \{c, d\} \text{ and } a \mapsto \{d, a\}.$$

Therefore, we have shown the following result, which is equivalent to Theorem 3.5:

Theorem 3.6 *Let N be a binary tree-based network with cover $\mathcal{C}$. The number of embedded partitions in $\mathcal{C}$, and therefore the number of support trees for N, is*

$$2^c \times \prod_{f \in \mathcal{F}} (r(f) + 1)$$

if $\mathcal{F}$ is non-empty, and is 2^c if $\mathcal{F} = \emptyset$, where c is the number of crowns in $\mathcal{C}$.

Note that the number of crowns, c, is the same as the number of components referred to in Theorem 3.5.

Given a cover $\mathcal{C}$, we can compute the number of crowns and the lengths of fences, and thus the number of embedded partitions, by using Algorithm 1, which uses the definition of 'acquaints'.

Definition 3.7 Set $x \sim y$ if $x = y$ or x, y are siblings, and consider the transitive closure of $\sim$, which is an equivalence relation on the set of vertices of the network. Two vertices in an equivalence relation are said to be *acquaints* of each other.

Acquaints can be defined self-referentially by saying that an *acquaint* of a vertex x is a sibling of x or is a sibling of an acquaint of x. Fences and crowns can be described in terms of acquaints, as follows.

Theorem 3.8 *Let N be a binary tree-based network with cover $\mathcal{C}$. Then*

(1) N has a fence if and only if there exists a set of acquaints in which exactly two vertices that appear uniquely in $\mathcal{C}$ have one sibling.
(2) N has a crown if and only if there exists a set of acquaints in which no vertex has one sibling.

Proof (1) For the forward direction, suppose that we have a fence like that in Table 3. The integers in the set $\{a_0, a_1, a_2, \ldots, a_{t-1}, a_t, a_{t+1}\}$ are acquaints, a_0 and a_{t+1} have only one sibling (a_1 and a_t respectively), and they appear uniquely by assumption.

Conversely, assume there is a set of acquaints in which exactly two vertices (say a_i and a_j) that appear uniquely in $\mathcal{C}$ have one sibling. Since we assume that the network is binary, a_i and a_j appear in only one subset, but they can not be in the same one; otherwise, they would not be acquainted with the other vertices.

It is also the case that every other vertex will appear in exactly two subsets; otherwise, it would imply an in-degree greater than 2, which is not allowed in a binary network.

Therefore, we have a set of a type described in Table 3, and the network has a fence.

(2) For the forward direction, suppose we have a crown (as indicated in Table 3). The integers in the set $\{a_0, a_1, a_2, \ldots, a_{t-1}, a_t\}$ are acquaints, and none of them has exactly one sibling.

Conversely, assume there is a set of acquaints in which no vertex has one sibling. Since we assume the network is binary, every vertex will appear in exactly two subsets; otherwise, it would imply an in-degree greater than 2, which is not allowed in a binary network. On the other hand, if a vertex appeared in exactly one subset, this would imply that it had only one sibling, which violates the assumption.

Therefore, we have a set of a type described in Table 3, and the network has a crown.   □

According to the theorem above, we can use Algorithm 1 to count the number of embedded partitions by enumerating the acquaints of all integers that are inside a set of size 2, because, in the definitions of crown and fences (Table 3), they do not contain sets of any other sizes.

Example 3.9 We saw in Example 3.4 that the cover for the binary tree-based network in Fig. 1 has four embedded partitions, and hence the network has four support trees

Algorithm 1 Count the number of support trees in a binary tree-based labellable network.

---

procedure TRAVERSEACQUAINTS(τ, i, A)
    add i to A
    mark i in τ as visited
    for $s \in \tau_i$ such that s is not marked as visited **do**
        TRAVERSEACQUAINTS(τ, s, A)
    end for
end procedure
procedure COUNTSUPPORTTREESBINARYNETWORKS($\mathcal{C}$)
    $\sigma_i \leftarrow$ number of times integer i appears in $\mathcal{C}$
    $\mathcal{C}^{=2} \leftarrow$ all subsets of $\mathcal{C}$ of size 2
    $I \leftarrow$ integers appearing in $\mathcal{C}^{=2}$
    $\tau \leftarrow$ table in which index $i \in I$ contains all siblings of i ▷ See Table 1
    $c \leftarrow 0$ ▷ Number of crowns
    $f \leftarrow 1$ ▷ Number of fences
    while τ not all i are marked as visited **do**
        select i not marked as visited in τ
        $A \leftarrow \emptyset$
        TRAVERSEACQUAINTS(τ, i, A)   ▷ A will contain the acquaints of i
        $A_1 \leftarrow$ set of integers i in A that have one sibling in τ_i
        if $|A_1| = 0$ **then**
            $c \leftarrow c + 1$ ▷ Theorem 3.8
        end if
        if $|A_1| = 2$ **then**
            $a, b \in A_1$
            if $\sigma_a = 1$ and $\sigma_b = 1$ **then**
                $f \leftarrow f \times (|A| - 1)$ ▷ Theorem 3.8. By Theorem 3.6, we have to add 1 to the length
            end if
        end if
    end while
    return $2^c \times f$
end procedure

Table 3 Translation of concepts arising from counting support trees for binary tree-based labellable networks

Network	Cover
Crown	Collection of sets $a_0, a_1 \mid a_1, a_2 \mid \cdots \mid a_{t-1}, a_t \mid a_t, a_0$
Fence	Collection of sets $a_0, a_1 \mid a_1, a_2 \mid \cdots \mid a_{t-1}, a_t \mid a_t, a_{t+1}$ with $a_0 \neq a_{t+1}$ both appearing uniquely

(shown in Fig. 2). These can be counted using Theorem 3.6 as follows. The cover $\mathcal{C} = 1 \mid 2 \mid 3 \mid 4, 5 \mid 6, 8 \mid 6, 7 \mid 7, 8 \mid 11, 12 \mid 9, 13 \mid 10, 13 \mid 14, 15$ has one crown, namely $6, 8 \mid 6, 7 \mid 7, 8$, and one fence $9, 13 \mid 10, 13$, which has length 1 (a single reticulation). Hence, the number of support trees is $2^1 \times (1 + 1) = 4$, as expected.

4 Tree-child networks

Tree-child networks are phylogenetic networks for which every vertex has a child that is a tree vertex (Cardona et al. 2008). They satisfy a number of important properties. For instance, they have the property that every vertex is *visible*. This is a property that we describe in Sect. 4.1, but first, tree-child networks turn out to have a very natural description in terms of covers, as follows.

Theorem 4.1 *Tree-child networks are in bijection with expanding covers for which each set contains an integer that appears exactly once in the cover.*

Proof The proof relies on the fact that the integers that appear precisely once in a cover are exactly the tree vertices.

Let N be a tree-child network with expanding cover $\mathcal{C}$. Each non-leaf vertex v in N corresponds to a specific set C_v in $\mathcal{C}$, whose elements label the children of v in N. Because N is a tree-child network, each such vertex v has at least one child that is a tree vertex. The labels of the tree vertices appear precisely once in the cover, so the set C_v contains at least one element that appears precisely once in the cover. This holds for every non-leaf vertex, and so for every set in $\mathcal{C}$, which establishes the forward direction.

The reverse direction is also straightforward. Suppose that every set in an expanding cover $\mathcal{C}$ has an element that appears precisely once in $\mathcal{C}$. Since each set in the cover is the set of labels of the children of a non-leaf vertex, this implies that every non-leaf vertex has at least one child whose label appears once in the cover. In other words, it is a tree vertex. Thus, the network corresponding to $\mathcal{C}$ is a tree-child network.   □

4.1 Visible vertices

An important property of tree-child networks is that all of their vertices are *visible* (Cardona et al. 2008, Lemma 2). A vertex v in a network is visible if there is a leaf x for which every path from the root to x passes through v. In this section, we show how visibility can be interpreted by using covers, beginning with the definition of the *backtrack* of a label in a cover.

Definition 4.2 Let $\mathcal{C}$ be an expanding cover in labelling order and let x be an element of $[m]$. Then a *backtrack for* x is a sequence of sets $S_1, \ldots, S_t$ in $\mathcal{C}$ for which the label of a set containing x is in S_1, and the label of S_i is an element of S_{i+1} for each $i = 1, \ldots, t-1$. This corresponds to the output of Algorithm 2. Let $B_{\mathcal{C}}(x)$ denote the set of all backtracks of x in $\mathcal{C}$.

We can characterise visibility in a network by using the backtracking algorithm. Given $x \in [m]$ and a backtrack β for x, we define $L(\beta) = \{\text{label of } s \mid s \in \beta\}$. In this way, $L(\beta)$ contains the vertices of a path from x to ρ (the root), $\bigcup_{\beta \in B_{\mathcal{C}}(x)} L(\beta)$ is the set of all vertices that *can be* visited with a path from x to ρ, and $\bigcap_{\beta \in B_{\mathcal{C}}(x)} L(\beta)$ is the set of all vertices that *must be* visited on a path from x to ρ.

Algorithm 2 Backtracking algorithm

Require: Expanding cover $\mathcal{C}$ in labelling order
  procedure BACKTRACK($\mathcal{C}, x$)
    seq ← [ ]
    s ← a subset of $\mathcal{C}$ containing x
    while label of s is not ρ **do**
      s ← a subset of $\mathcal{C}$ that contains the label of s as an element
      add s to seq
    end while
    return seq
  end procedure

Table 4 Translation of the concepts arising in tree-child networks

Network	Cover
Path from node x to the root	A backtrack for x
Visible vertex x	There is a $y \in [n]$ such that $x \in \bigcap_{\beta \in B_{\mathcal{C}}(y)} L(\beta)$

Theorem 4.3 *Given a cover $\mathcal{C}$ in labelling order and $x \in [m]$, x is a visible vertex in the corresponding network if and only if there exists $y \in [n]$ such that $x \in \bigcap_{\beta \in B_{\mathcal{C}}(y)} L(\beta)$.*

Proof For the forward direction, assume that a vertex x of a network is visible. By definition, there exists a leaf y (in other words, $y \in [n]$) such that all paths from the root to y pass through x. Since $\bigcap_{\beta} L(\beta)$ is the set of all vertices we *have to* visit from y to ρ, x must be in this intersection.

For the backward direction, let $y \in [n]$ and $x \in \bigcap_{\beta \in B_{\mathcal{C}}(y)} L(\beta)$. Then it means that all paths from y to ρ contain x. Therefore x is visible in the corresponding network. □

Since all $x \in \bigcap_{\beta \in B_C(x)} L(\beta)$ are visible vertices and vice versa, we obtain the following corollary.

Corollary 4.4 *Given a cover $\mathcal{C}$ in labelling order, then all $x \in \bigcup_{y \in [n]} \bigcap_{\beta \in B_{\mathcal{C}}(y)} L(\beta)$ are visible vertices in the corresponding network and vice versa.*

4.2 Support trees for tree-child networks

Theorem 4.1 allows us to provide an alternative proof of a result about support trees in tree-child networks, as follows.

Corollary 4.5 *(Francis and Moulton 2018, Theorem 3.3) A binary tree-child network with k reticulations has 2^k support trees.*

Proof Since each set in the cover for a tree-child network has a uniquely appearing element, there are no sets containing only reticulations (i.e. no singletons with elements that appear elsewhere, and no pairs in which both elements are repeated). Using the categories above, all sets in such a cover are from Categories ((1)), ((3)), or ((5)).

As a consequence, there are no crowns, which require sets with two reticulations, and each fence can only have length 1, being of the form a, $b \mid b$, c, and containing only one reticulation (b in this case). Furthermore, each repeated integer in the cover (i.e., each reticulation) is in a fence, since it must be part of a pair with a uniquely appearing element (a tree vertex). Thus, the number of fences is the number of reticulations, and each fence has length 1.

Therefore, by Theorem 3.6, there are 2^k support trees.   □

Corollary 4.5 also follows immediately by combining both parts of the following result.

Theorem 4.6 *(i) The number of spanning trees in a phylogenetic network is the product of all the in-degrees of the reticulation vertices.*
(ii) A network is a tree-child network if and only if every spanning tree is also a support tree.

Proof Part (i): A reticulation vertex x is an integer contained in $k > 1$ subsets of $\mathcal{C}$ (Table 1) and a spanning tree is an embedded partition (Table 2). Thus, to obtain an embedded partition from a cover, we have to remove $k - 1$ instances of x from $\mathcal{C}$. This can be done in $\binom{k}{k-1} = k$ different ways, and each choice is independent of the others. Since k is also the in-degree for vertex x, it follows than the number of embedded partitions (spanning trees) is $\prod_x$ *in-degree*(x), where x is a reticulation vertex. If x is a tree vertex, then in-degree$(x) = 1$ and, therefore, it does not contribute to the product.

Part (ii): By Theorem 4.1, every subset of a tree-child cover has at least one element that is not present in any other subset. This implies that every embedding partition must contain at least one element for each subset; hence, it has the same size as $|\mathcal{C}|$.

To show the forward direction, suppose that N is not a tree-child network. We will show that there must be a spanning tree for N that is not a support tree. If N is not tree-child, then it has at least one vertex that is not visible. Let v be a non-visible vertex that is maximally distant from the root, so that all vertices descended from v are visible. If we delete each arc out of v, then there is a path from the root to each vertex, so N has a spanning tree T. However, in this tree, T has v as a leaf. The tree T is therefore a spanning tree of N and not all its leaves are in X, so T is not a support tree.   □

5 Normal networks

Normal networks are a subclass of the tree-child networks, with the added constraint that they contain no "shortcuts" (Willson 2010). A *shortcut* is an edge (u, v) for which there is an alternative directed path from u to v in the network.

To capture this information in terms of covers, we need a way to record paths in that context. This motivated the definition of backtrack (Definition 4.2), which requires the

labelling order that was defined in Sect. 2. The backtrack algorithm identifies a path from the vertex labelled x back to the root, expressing the path in terms of a sequence of sets in the cover. The edges between the parent vertices that correspond with these sets defines the path.

Example 5.1 Recall the cover $\mathcal{C} = 1 \mid 2 \mid 3 \mid 4, 5 \mid 6, 8 \mid 6, 7 \mid 7, 8 \mid 11, 12 \mid 9, 13 \mid 10, 13 \mid 14, 15$ from Example 3.4 for the network in Fig. 1. This cover has the labelling order

$$\mathcal{C} = \{1\}_6, \{2\}_7, \{3\}_8, \{4, 5\}_9, \{6, 7\}_{10}, \{6, 8\}_{11}, \{7, 8\}_{12}, \{11, 12\}_{13},$$
$$\{9, 13\}_{14}, \{10, 13\}_{15}, \{14, 15\}_{\rho}.$$

A backtrack for $x = 3$ starts with a subset containing 8 (with the label of $\{3\}$ in the labelling order). There are two choices; suppose we pick $\{7, 8\}$. The label of $\{7, 8\}$ is 12, so now we must find a set containing 12. There is only one, so we add $\{11, 12\}$ to the backtrack sequence. $\{11, 12\}$ has label 13, so we look for a set containing 13 and choose one of the two options, say $\{10, 13\}$. This has label 15 in the order, so we look for a set containing 15. There is one, namely $\{14, 15\}$, and its label is ρ, which means we terminate the algorithm and output the backtrack sequence

$$\{7, 8\}, \{11, 12\}, \{10, 13\}, \{14, 15\}.$$

Note, each such backtrack defines a path from 3 to the root ρ; in this case, $3 \rightarrow 8 \rightarrow 12 \rightarrow 13 \rightarrow 15 \rightarrow \rho$.

Theorem 5.2 *Let N be a phylogenetic network with expanding cover $\mathcal{C}$, in labelling order. Then N has a shortcut if and only if there is a backtrack for an $x \in [m]$ that includes a subset containing x.*

Proof Suppose N has a shortcut. Then there is a vertex x with a non-trivial path from some vertex v to x, and there is also an edge (v, x). The existence of a non-trivial path from v to x means that the cover has a non-trivial backtrack from x, which includes the children of v as a set. However, x is also a child of v, so x is in a set in the backtrack.

Conversely, suppose that the cover contains a backtrack for x that includes a set S containing x. Let v be the label of the parent of S. Then x is a child of v, meaning there is an edge (v, x) in N. However, the backtrack provides a non-trivial path in N from v to x through S. That is, N contains a shortcut. □

Corollary 5.3 *Let $\mathcal{C}$ be a cover in labelling order for a tree-child network. Then $\mathcal{C}$ is a cover for a normal network if and only if, for all $x \in [m]$, no backtrack for x has a subset that contains x.*

Without loss of generality, in Theorem 5.2 and Corollary 5.3, we can assume that x is a reticulation vertex (i.e., a value in $[m]$ that is contained in more that one subset of $\mathcal{C}$), since, by definition, reticulations have in-degree greater than one and thus are the only vertices that can have shortcuts.

Using Theorem 5.2, we can construct an algorithm that removes all the shortcuts from a cover. This implies that, given a tree=child network, we can transform it to a normal network by removing all the shortcuts via Algorithm 3.

Algorithm 3 Remove all shortcuts from a cover

Require: Expanding cover $\mathcal{C}$ in labelling order
  procedure REMOVESHORTCUTS($\mathcal{C}$)
    Compute all backtracks for all reticulation vertices
    for backtrack β for reticulation vertex x **do**
      for $s \in \beta$ **do**
        if $x \in s$ **then**
          Remove x from s
        end if
      end for
    end for
  end procedure

Table 5 A translation of a shortcut into a feature of the corresponding expanding cover

Network	Cover
Shortcut to x	A backtrack of x that includes a set containing x

6 Tree-sibling networks

Tree-sibling networks are also amenable to a description in terms of covers.

Definition 6.1 (Cardona et al. 2008) A tree-sibling network is a network in which every reticulation vertex is a sibling of a tree vertex.

Theorem 6.2 *Tree-siblings networks are in bijection with those expanding covers for which every repeated integer lies in at least one set with an integer that appears only once.*

Proof The statement is a direct translation of the definition of tree-sibling into the language of covers, according to Table 1. Reticulation vertices are those that appear more than once in the cover, and vertices are siblings when they appear in the same set in the cover. □

We have already seen a characterisation of tree-child networks using covers in Theorem 4.1. Covers for tree-child networks are those for which every set has a uniquely appearing element. However, there is a close connection between tree-child and tree-sibling networks, which can be captured in a cover description for tree-child networks, as follows.

Theorem 6.3 *Tree-child networks are in bijection with expanding covers for which, for every repeated element k in $\mathcal{C}$, every subset containing k also contains an integer that appears only once.*

Proof We will prove that this statement is equivalent to Theorem 4.1.

For the forward direction, suppose that a cover satisfies the condition in Theorem 4.1. If every subset contains a uniquely occurring integer, then all subsets that contain a reticulation will do also do so.

For the backward direction, by assumption, every subset that contains a reticulation vertex has an integer that is not contained in another subset. This implies that all other subsets do not contain a reticulation vertex, and therefore, it contains a tree vertex that is not contained in any other subset (Table 1).   □

In other words, tree-child networks are networks in which every parent of a reticulation vertex has a tree-vertex as a child. Therefore, we recover the well-known fact that all tree-child networks are tree-sibling networks.

7 Orchard networks

Orchard networks are non-degenerate phylogenetic networks defined by the property that they can be reduced to a trivial network (a single vertex) by a series of cherry or reticulated cherry reductions (Erdős et al. 2019; Janssen and Murakami 2021; van Iersel et al. 2022). In the present paper, we will restrict our attention to *binary* orchard networks.

A *cherry* is a pair of leaves that are siblings; a *reticulated cherry* is a pair of leaves, one of which has a reticulate parent and the other is the sibling of that reticulate parent. Cherry reduction involves replacing the cherry with a single vertex. Reticulated cherry reduction involves deleting the arc between the parents of the two leaves and then suppressing degree-2 vertices. By a theorem of Erdős et al. (2019); Janssen and Murakami (2021), for orchard networks, the order in which these are performed is not important.

To translate this definition into covers, we need to first characterise cherries and reticulated cherries as they are manifested in covers, and then describe the action of such reductions in terms of the cover. The first of these requirements is routine; the second is not, as it requires us to augment the cover with its set of leaves. We will describe a test for orchard networks that reduces an expanding cover to a trivial cover but, along the way, passes through covers that are not expanding.

In covers, a cherry is given by a set consisting of two elements of $[n]$ (the leaves), whereas a reticulated cherry is given by a singleton subset of $[n]$ appearing in position j in the labelling order, and a pair $\{n + j, i\}$ where $i \in [n]$ (summarized in Table 7). An example is shown in Fig. 3.

7.1 The cherry reduction process via covers

The cherry reduction test for orchard networks can be defined efficiently using covers by keeping track of the changing set of leaf labels $\mathcal{L}$ within the algorithm, as follows. Identifying a cherry or reticulated cherry in a cover can be done using the translations given in Table 7. The process in Algorithm 4 chooses to reduce a cherry first, if there is one, as it involves fewer checks.

In general, the set $\mathcal{C}$ that is redefined during Algorithm 4 may not be an expanding cover, but these processes do nevertheless model the network cherry and reticulated cherry reduction steps, applied to a labellable network.
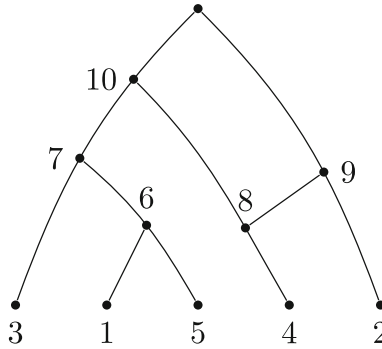
Fig. 3 A labellable network with a cherry and a reticulated cherry. The cover, in labelling order, is $(\{1, 5\}, \{3, 6\}, \{4\}, \{2, 8\}, \{7, 8\}, \{9, 10\})$. The cherry can be identified in the cover as a pair of integers that are a subset of the leafset [5]. In this cover, a cherry is $\{1, 5\}$. The reticulated cherry is identified in the cover as a pair of sets: one is a singleton subset of the leafset, in position j; the other is the pair $\{n + j, i\}$ with i in the leafset. In this cover, there is a reticulated cherry consisting of the singleton $\{4\}$ (contained in the leafset), which appears in position 3 in the labelling order, and the pair $\{2, 8\}$, noting that $8 = n + 3$ and 2 is in the leafset

---

Algorithm 4 Test whether the expanding cover $\mathcal{C}$ corresponds to an orchard network

---

Require: Expanding cover $\mathcal{C}$ in labelling order
  procedure IsOrchard($\mathcal{C}$)
    $n \leftarrow ||\mathcal{C}|| - |\mathcal{C}| + 1$
    $\mathcal{L} \leftarrow [n]$
    $reduced \leftarrow true$
    while $reduced = true$ **and** $|\mathcal{C}| > 0$ **do**
      $reduced \leftarrow false$
      if there is a set of form $\{a, b\}_j \in \mathcal{C}$ with $a, b \in \mathcal{L}$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \setminus \{a, b\}$ $\triangleright$ Cherry reduction
        $\mathcal{L} \leftarrow (\mathcal{L} \setminus \{a, b\}) \cup \{j\}$
        $reduced \leftarrow true$
      else
        if there is a set of form $\{a\}_j$ in $\mathcal{C}$ and a set of form $\{j, b\} \in \mathcal{C}$, with $a, b \in \mathcal{L}$ **then**
          $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\{a\}, \{j, b\}\}$ $\triangleright$ Reticulated cherry reduction
          $\mathcal{L} \leftarrow (\mathcal{L} \setminus \{a, b\}) \cup \{j, k\}$
          $reduced \leftarrow true$
        end if
      end if
    end while
    if $\mathcal{C} = \emptyset$ **then**
      return "$\mathcal{C}$ is orchard"
    else
      return "$\mathcal{C}$ is not orchard"
    end if
  end procedure

Theorem 7.1 *Algorithm 4 determines whether the network from the expanding cover $\mathcal{C}$ is orchard.*

Proof A network is orchard, by definition, if and only if it can be reduced to a trivial network by cherry or reticulated cherry reductions. According to a result of Erdős et al. (2019); Janssen and Murakami (2021), the order of such reductions is not important.
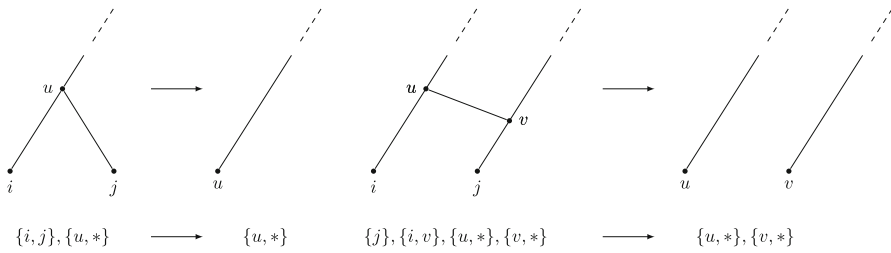
Fig. 4 Cherry (left) and reticulated cherry (right) reductions and their effects on the covers. Here, $*$ represents other sibling vertices, which could be an empty set

The procedures in Algorithm 4 exactly reflect the effect on the cover of these operations on the network, as can be seen in Fig. 4.   □

Example 7.2 The cherry reduction process in Algorithm 4, applied to the cover for the network in Fig. 3, proceeds as described in Table 6.

8 A new class of network detected through the lens of covers

We have used covers to describe several classes of phylogenetic networks. However, the encoding into covers also creates the opportunity to define new classes of networks that correspond to particular features of covers. Such classes might currently have little direct utility for application to phylogenetics, but they may have an indirect value in that algorithms and methods using covers may involve such classes in passing. We introduce one such class as an example of this opportunity.

Recall that the definition of an expanding cover has two criteria (Definition 2.1). The first is that elements of the leafset $[n]$ are not repeated, and the second ensures that the labelling algorithm is well-defined by requiring at least i subsets of $[n+i-1]$ to be in the cover.

If a cover contains *exactly* i subsets of $[n+i-1]$, it has a strong consequence for the network, as follows. We define a *spine* in a phylogenetic network to be a directed path from a leaf to the root that traverses all non-leaf vertices, and we call a network *spinal* if it has a spine.

Theorem 8.1 *A network is spinal if and only if its cover $\mathcal{C}$ has exactly i subsets of $[n+i-1]$, for each $i = 1, \ldots, |\mathcal{C}|$.*

Proof We prove the reverse direction first. Suppose that the cover $\mathcal{C}$ has exactly i subsets of $[n+i-1]$, for each $i = 1, \ldots, |\mathcal{C}|$, and consider its labelling order. The first set in the labelling order is the unique set that is contained in $[n]$, and its label is $n+1$. For each $i = 1, \ldots, |\mathcal{C}| - 1$, the ith set in the labelling order is contained in $[n+i-1]$, according to the expanding property, but is not contained in $[n+i-2]$, according to our assumption about $\mathcal{C}$. Therefore, it must contain the integer $n+i-1$.

Table 6 Cherry reduction algorithm acting on the cover in Example 7.2 via Algorithm 4, with the effects of reduction on the network shown at right (for illustration only)

Cherry reduction of the cover $\mathcal{C}$ in Example 7.2, following Algorithm 4.

$\mathcal{C} = \big\{\{1, 5\}_6, \{3, 6\}_7, \{4\}_8, \{2, 8\}_9, \{7, 8\}_{10}, \{9, 10\}_\rho\big\};$
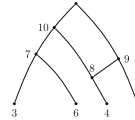
$\mathcal{L} = \{1, 2, 3, 4, 5\}.$

1    $\mathcal{C}$ contains the cherry $\{1, 5\}_6$ (and the reticulated cherry, $\{4\}_8, \{2, 8\}_9$). We reduce the cherry.

$$\begin{aligned} \mathcal{C}_1 \quad &= \mathcal{C} \setminus \{\{1, 5\}_6\} \\ &= \big\{\{3, 6\}_7, \{4\}_8, \{2, 8\}_9, \{7, 8\}_{10}, \{9, 10\}_\rho\big\} \\ \mathcal{L}_1 \quad &= (\mathcal{L} \setminus \{1, 5\}) \cup \{6\} \\ &= \{2, 3, 4, 6\} \end{aligned}$$

2    $\mathcal{C}_1$ contains cherry $\{3, 6\}_7$ (and the reticulated cherry $\{4\}_8, \{2, 8\}_9$). We reduce the cherry.

$$\begin{aligned} \mathcal{C}_2 \quad &= \mathcal{C}_1 \setminus \{\{3, 6\}_7\} \\ &= \big\{\{4\}_8, \{2, 8\}_9, \{7, 8\}_{10}, \{9, 10\}_\rho\big\}. \\ \mathcal{L}_2 \quad &= (\mathcal{L}_1 \setminus \{3, 6\}) \cup \{7\} \\ &= \{2, 4, 7\}. \end{aligned}$$
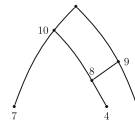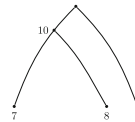
3    $\mathcal{C}_2$ contains no cherry, but contains the reticulated cherry $\{4\}_8, \{2, 8\}_9$, which we reduce.

$$\begin{aligned} \mathcal{C}_3 \quad &= \mathcal{C}_2 \setminus \{\{4\}_8, \{2, 8\}_9\} \\ &= \big\{\{7, 8\}_{10}, \{9, 10\}_\rho\big\}. \\ \mathcal{L}_3 &= (\mathcal{L}_2 \setminus \{2, 4\}) \cup \{8, 9\} \\ &= \{7, 8, 9\}. \end{aligned}$$
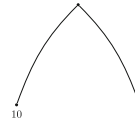
4    $\mathcal{C}_3$ contains the cherry $\{7, 8\}_{10}$, which we reduce.

$$\begin{aligned} \mathcal{C}_4 \quad &= \mathcal{C}_3 \setminus \{7, 8\}_{10} \\ &= \big\{\{9, 10\}_\rho\big\}. \\ \mathcal{L}_4 &= (\mathcal{L}_3 \setminus \{7, 8\}) \cup \{10\} \\ &= \{9, 10\}. \end{aligned}$$

5    $\mathcal{C}_4$ contains (only) the cherry $\{9, 10\}_\rho$, which we reduce.

$\mathcal{C}_5 = \mathcal{C}_4 \setminus \{9, 10\}_\rho = \emptyset$, which means the algorithm ends

---

Table 7 A translation of features that are relevant to orchard networks into features of the corresponding expanding cover

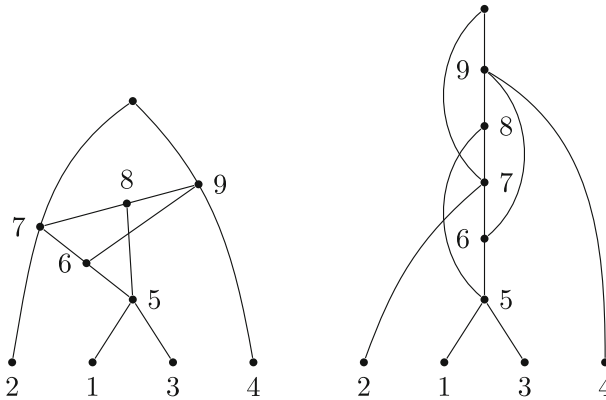| Network	Cover
Cherry	A set consisting of two elements of $[n]$
Reticulated cherry	A singleton subset of $[n]$ appearing in position j in the labelling order, and a pair $\{n + j, i\}$ where $i \in [n]$

Fig. 5 A spinal network with cover 1, 3 | 5 | 2, 6 | 5, 7 | 4, 6, 8 | 7, 9. Note that $n = 4$ and the cover has one set in [4], two in [5], three in [6], four in [7], five in [8], and six in [9]. There is a path from the elements of the set that is in [4], namely 1 and 3, to the root, and this path traverses every non-leaf vertex. Observe that this path is in labelling sequence. The spine is particularly clear when the network is drawn as shown on the right

Table 8 A translation of the feature of spinal networks into a feature of the corresponding expanding cover

Network	Cover
Spine	Exactly i subsets of $[n + i - 1]$, for each i

The ith set in the labelling order has label $n + i$, which means that $n + i$ is a parent of $n + i - 1$. Since this holds for each $i > 1$, this determines a path from a leaf (labelled by an element of the first set in the labelling order) through every vertex with label $n + 1$ to $n + |\mathcal{C}| - 1$, and the last set, containing $||\mathcal{C}|| = n + |\mathcal{C}| - 1$, has the root as parent. Thus the network is spinal.

We now prove the forward direction. Suppose that N is spinal with cover $\mathcal{C}$. Being spinal means that N has a path of length $|\mathcal{C}|$ from a leaf to the root. This means that there is a backtrack of a leaf that has length $|\mathcal{C}| - 1$. That is, a sequence of sets from the cover such that the label of one set (from the labelling order on $\mathcal{C}$) is an element of the next set in the backtrack sequence. Because the label of a set in the cover is strictly greater than all the elements of the set, the maximal elements of the sets in a backtrack are strictly increasing.

Now consider the backtrack arising from the spine (the path from a leaf to the root traversing all non-leaf vertices). The leaf at the base of the spine must be in a set contained in $[n]$; otherwise, there would be no path from it to the vertex labelled $n + 1$. Therefore, the first set in the backtrack contains $n + 1$ as its maximal element because that is the parent label for the set containing the initial leaf. The spine has $|\mathcal{C}| - 1$ vertices in it, including the initial leaf, because it includes all except $n - 1$ of the vertices in the network (the network has $||\mathcal{C}|| + 1 = |\mathcal{C}| + n$ vertices in total). Therefore, the backtrack for the initial leaf has $|\mathcal{C}| - 1$ sets. The maximal elements of these $|\mathcal{C}| - 1$ sets are strictly increasing, and run from $n + 1$ to $m = ||\mathcal{C}|| = |\mathcal{C}| + n - 1 = n + (|\mathcal{C}| - 1)$. This forces each set in the backtrack to have a distinct maximal element. Put together

Table 9 A translation of the features of a labellable network with n leaves and m non-root vertices, into the features of the corresponding expanding cover

Network N	Cover $\mathcal{C}$
Non-root vertex	An integer in $[m]$
Leaf	An integer in $[n]$
Tree vertex	An integer contained in just one subset
Reticulation vertex	An integer contained in more than one subset
In-degree of x	The number of subsets that contain x
Out-degree of x	Size of the subset with label x in the labelling order
Parents of x	All the subsets that contain x
Siblings of x	All the other integers contained in the subsets that contain x
Children of x	The subset with label x in the labelling order
Spanning tree	A partition embedded in $\mathcal{C}$
Support tree	A full embedding of a partition in $\mathcal{C}$
Crown	Collection of sets $a_0, a_1 \mid a_1, a_2 \mid \cdots \mid a_{t-1}, a_t \mid a_t, a_0$
Fence	Collection of sets $a_0, a_1 \mid a_1, a_2 \mid \cdots \mid a_t, a_{t+1}$, with $a_0 \neq a_{t+1}$ both unique
Path from x to ρ	A backtrack for x
Visible x	There is a $y \in [n]$ such that $x \in \bigcap_{\beta \in B_{\mathcal{C}}(y)} L(\beta)$
Shortcut to x	A backtrack of x that includes a set containing x
Cherry	A set consisting of two elements of $[n]$
Reticulated cherry	A subset $\{a\}_k$ of $[n]$, and a pair $\{k, b\}$ with $b \in [n]$
Spine	Exactly i subsets of $[n + i - 1]$, for each i

with the set containing the initial leaf, which is a subset of $[n]$, this means that there are exactly i subsets of $[n + i - 1]$, for each $i = 1, \ldots, |\mathcal{C}|$, as required.   □

In the light of Theorem 8.1, we say that a cover $\mathcal{C}$ is *spinal* if it contains exactly i subsets of $[n + i - 1]$, for each $i = 1, \ldots, |\mathcal{C}|$. An example of a spinal network is shown in Fig. 5. Spinal networks have some non-trivial intersections with other classes; for example, the spinal network $1 \mid 2 \mid 2, 3 \mid 3, 4$ is not a tree-child, tree-sibling, or orchard network. It can, however, be shown that the class of spinal networks lies within the intersection of the labellable and tree-based classes of networks.

9 Discussion

Sometimes a relatively small shift in perspective can open up new possibilities in surprising ways. What seems like a fairly straightforward idea in a paper by Diaconis and Holmes (the idea that rooted binary phylogenetic trees correspond to perfect matchings (Diaconis and Holmes 1998)), itself building on an elegant but simple way to label internal vertices (Erdős and Székely 1989), was loosened slightly to yield a correspondence between phylogenetic forests and all partitions of finite sets, as well as a raft of interesting questions in semigroup theory (Francis and Jarvis 2022). This subtle twist of an idea, like something from a Philip Pullman novel (Pullman 2015),
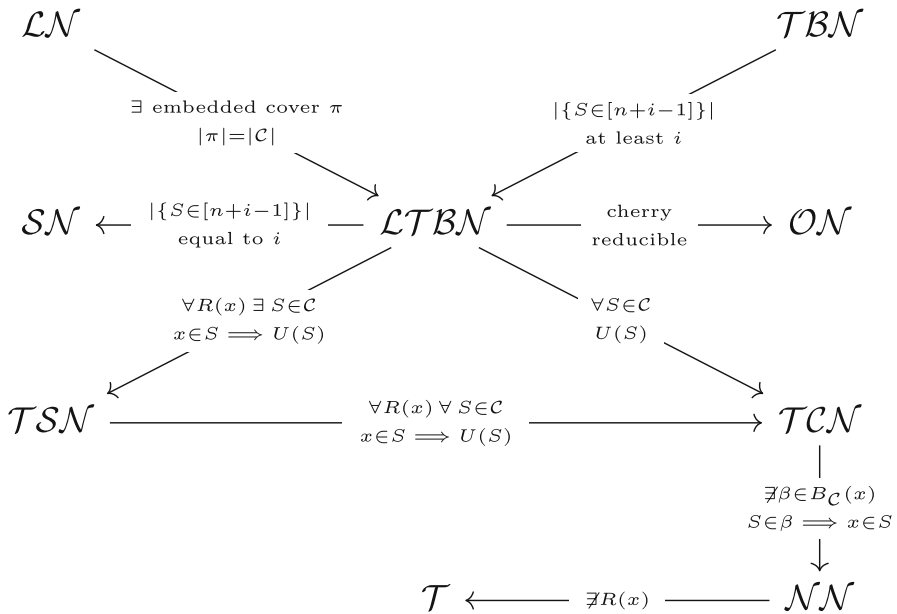
$\mathcal{LN}$ $\mathcal{TBN}$

$\exists$ embedded cover π
$|\pi|=|\mathcal{C}|$

$|\{S\in[n+i-1]\}|$
at least i

$\mathcal{SN}\ \leftarrow\ \begin{array}{c}|\{S\in[n+i-1]\}|\\ \text{equal to } i\end{array}\ -\ \mathcal{LTBN}\ -\ \begin{array}{c}\text{cherry}\\\text{reducible}\end{array}\ \longrightarrow\ \mathcal{ON}$

$\forall R(x)\,\exists\,S\in\mathcal{C}$
$x\in S \implies U(S)$

$\forall S\in\mathcal{C}$
$U(S)$

$\mathcal{TSN}\ -\!\!\!\begin{array}{c}\forall R(x)\,\forall\,S\in\mathcal{C}\\ x\in S \implies U(S)\end{array}\!\!\!\longrightarrow\ \mathcal{TCN}$

$\nexists\beta\in B_{\mathcal{C}}(x)$
$S\in\beta \implies x\in S$

$\mathcal{T}\ \longleftarrow\ \nexists R(x)\ -\ \mathcal{NN}$

Fig. 6 A diagram showing the hierarchy of networks. The nodes are classes of networks, the arrows represent inclusion and the labels indicate which axiom we add to obtain that class. $R(x)$ means "x is repeated in more than one subset" and $U(S)$ means "S contains an unique integer". The class labels are (from top to bottom): $\mathcal{LN}$ (labellable networks), $\mathcal{TBN}$ (tree-based networks), $\mathcal{LTBN}$ (labellable tree-based networks), $\mathcal{SN}$ (spinal networks), $\mathcal{ON}$ (orchard networks), $\mathcal{TSN}$ (tree-sibling networks), $\mathcal{TCN}$ (tree-child networks), $\mathcal{NN}$ (normal networks), and $\mathcal{T}$ (trees)

seems to have opened up further opportunities that, with a further gentle twist, have opened a new canvas on which to draw phylogenetic networks (Francis et al. 2023). Capturing the features that define different network classes on this canvas provided the underlying motivation for this paper.

Many core features discussed in the context of networks, such as reticulations, paths, cherries, siblings, and so on, have been translated into the language of covers; a summary is given in Table 9. These translations of features have been necessary for characterising several important classes of phylogenetic network in the language of covers. This includes some of the most prominent classes, including normal, tree-child, tree-sibling, orchard, and tree-based networks (relationships among the classes, determined by properties of their covers, are represented in Fig. 6). However there are many classes, each of which is important for its own reasons, and this list is not complete. Some classes that have been omitted in the present paper might be difficult to define with covers (for instance, level-k networks or HGT networks), whereas others might just be a matter of following through with the first steps we have taken here (for example, reticulation-visible networks, and non-binary orchard networks).

Defining a language is not the goal, however, despite it being a necessary step. The goal is to be able to efficiently work with phylogenetic networks—computationally, algorithmically, and mathematically—in order to establish robust methods of inference for networks that will eventually be of practical use for biological researchers. To that

end, encoding various classes of phylogenetic networks in terms of expanding covers provides an opportunity to make computation more effective and allow their structure to be seen more clearly.

Data availability Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

References

Bapteste E, van Iersel L, Janke S, Kelchner A, Kelk S, McInerney D, Morrison J, Nakhleh L, Steel M, Stougie L, Whitfield J (2013) Networks: expanding evolutionary thinking. Trends Genet 29:439–441

Cardona G, Llabrés M, Rosselló F, Valiente G (2008) A distance metric for a class of tree-sibling phylogenetic networks. Bioinformatics 24(13):1481–1488

Cardona G, Rosselló F, Valiente G (2008) Comparison of tree-child phylogenetic networks. IEEE/ACM Trans Comput Biol Bioinf 6(4):552–569

Diaconis PW, Holmes SP (1998) Matchings and phylogenetic trees. Proc Natl Acad Sci 95(25):14600–14602

Erdős PL, Székely LA (1989) Applications of antilexicographic order. I. An enumerative theory of trees. Adv Appl Math 10(4):488–496

Erdős PL, Semple C, Steel M (2019) A class of phylogenetic networks reconstructable from ancestral profiles. Math Biosci 313:33–40

Francis A, Jarvis PD (2022) Brauer and partition diagram models for phylogenetic trees and forests. Proc R Soc A 478(2262):20220044

Francis A, Moulton V (2018) Identifiability of tree-child phylogenetic networks under a probabilistic recombination-mutation model of evolution. J Theor Biol 446:160–167

Francis A, Steel M (2015) Which phylogenetic networks are merely trees with additional arcs? Syst Biol 64(5):768–777

Francis A, Steel M (2023) Labellable phylogenetic networks. Bull Math Biol 85(6):46

Francis A, Huber KT, Moulton V, Wu T (2023) Encoding and ordering *X*-cactuses. Adv Appl Math 142:102414

Hayamizu M (2021) A structure theorem for rooted binary phylogenetic networks and its implications for tree-based networks. SIAM J Discret Math 35(4):2490–2516

Huson DH, Rupp R, Scornavacca C (2010) Phylogenetic networks: concepts, algorithms and applications. Cambridge University Press, Cambridge

Janssen R, Murakami Y (2021) On cherry-picking and network containment. Theoret Comput Sci 856:121–150

Jetten L (2015) Characterising tree-based phylogenetic networks (Karakterisatie van fylogenetische netwerken die een boom als basis hebben). PhD thesis, Delft University of Technology

Kong S, Pons JC, Kubatko L, Wicke K (2022) Classes of explicit phylogenetic networks and their biological and mathematical significance. J Math Biol 84(6):1–44

Pons JC, Semple C, Steel M (2019) Tree-based networks: characterisations, metrics, and support trees. J Math Biol 78:899–918

Pullman P (2015) The subtle knife. Random House, New York

van Iersel L, Janssen R, Jones M, Murakami Y (2022) Orchard networks are trees with additional horizontal arcs. Bull Math Biol 84(8):1–21

Willson SJ (2010) Properties of normal phylogenetic networks. Bull Math Biol 72:340–358

Zhang L (2016) On tree-based phylogenetic networks. J Comput Biol 23(7):553–565