LABELING AND FOLDING MULTI-LABELED TREES

VINCENT MOULTON AND ANDREAS SPILLNER

Abstract. In 1989 Erdős and Székely showed that there is a bijection between (i) the set of rooted trees with $n+1$ vertices whose leaves are bijectively labeled with the elements of $[\ell] = \{1, 2, \ldots, \ell\}$ for some $\ell \leq n$, and (ii) the set of partitions of $[n] = \{1, 2, \ldots, n\}$. They established this via a labeling algorithm based on the anti-lexicographic ordering of non-empty subsets of $[n]$ which extends the labeling of the leaves of a given tree to a labeling of all of the vertices of that tree. In this paper, we generalize their approach by developing a labeling algorithm for *multi-labeled trees*, that is, rooted trees whose leaves are labeled by positive integers but in which distinct leaves may have the same label. In particular, we show that certain orderings of the set of all finite, non-empty multisets of positive integers can be used to characterize partitions of a multiset that arise from labelings of multi-labeled trees. As an application, we show that the recently introduced class of labelable phylogenetic networks is precisely the class of phylogenetic networks that are stable relative to the so-called folding process on multi-labeled trees. We also give a bijection between the labelable phylogenetic networks with leaf-set $[n]$ and certain partitions of multisets.

1. Introduction

A *semi-labeled tree* is a rooted tree in which the leaves are bijectively labeled with the elements of $[\ell] = \{1, 2, \ldots, \ell\}$, for ℓ some positive integer (see e.g. Figure 1(a)). In the ground-breaking work [7], Erdős and Székely established a bijection between the set of partitions of $[n] = \{1, 2, \ldots, n\}$ and the set of semi-labeled trees with $n+1$ vertices. Their proof relies on (i) giving an algorithm for labeling the interior vertices of a semi-labeled tree that extends the leaf labeling and (ii) using the simple observation that, given a rooted tree T with $n+1$ vertices in which all vertices other than the root are bijectively labeled with the elements of $[n]$, a partition of $[n]$ is obtained by taking, for each non-leaf vertex v of T, the subset of $[n]$ labeling the children of v (see e.g. Figure 1(b)). A similar approach yields the bijection between matchings and binary rooted *phylogenetic trees* (semi-labeled trees in which every interior vertex has two children) described in [5], and is also used to devise phylogenetic tree encodings [2, 16, 17]. These types of results have subsequently been used to, for example, navigate tree space [16], and place a semi-group structure on the set of binary rooted phylogenetic trees [8].

(V. Moulton) School of Computing Sciences, University of East Anglia, UK
(A. Spillner) Merseburg University of Applied Sciences, Germany
E-mail addresses: v.moulton@uea.ac.uk, andreas.spillner@hs-merseburg.de.
Date: October 29, 2025.
Key words and phrases. multi-labeled tree, labeling algorithm, partitions of multisets.

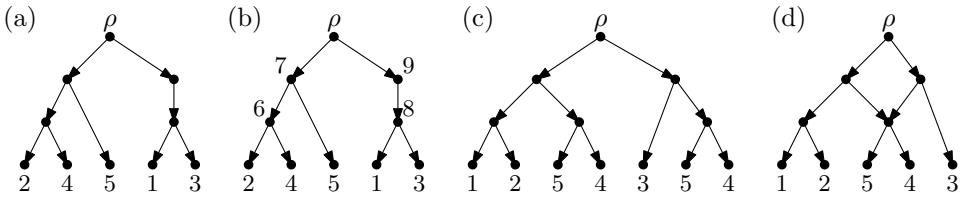

FIGURE 1. (a) A semi-labeled tree with $\ell = 5$ and $n = 9$. (b) The semi-labeled tree in (a) with vertices bijectively labeled with the elements of [9]. Taking for each non-leaf vertex v the subset of [9] labeling the children of v, yields the partition $\{\{1,3\},\{2,4\},\{5,6\},\{8\},\{7,9\}\}$ of [9]. (c) A multi-labeled tree. (d) The phylogenetic network obtained by folding the multi-labeled tree in (c).

In this paper, we investigate to what extent the results in [7] can be extended to rooted trees (and networks) whose leaves are labeled by positive integers but in which distinct leaves may have the same label (see e.g. Figure 1(c)). Such trees arise in phylogenetics under the name of *multi-labeled trees* (see e.g. [11]). We shall show that such trees are in bijection with certain partitions of multisets. As with Erdős and Székely's result, the key to establishing this bijection is to develop an algorithm that extends the labeling of the leaves of a multi-labeled tree to a labeling of all vertices of the tree in such a way that the tree and its labeling can be uniquely recovered from the resulting multiset partition. Our labeling algorithm is parameterized by *any* given ordering $\preceq$ of the set of all finite, non-empty multisets of positive integers. Moreover, choosing $\preceq$ to be the *anti-lexicographic ordering*, produces the same output as the algorithm used in [7] when applied to a semi-labeled tree.

We shall also describe an application of our multi-labeled tree results to *phylogenetic networks*, graphs that are used in phylogenetics to represent complex evolutionary relationships (see e.g. [18]). For our purposes, phylogenetic networks are essentially directed acyclic graphs whose leaves are bijectively labeled by the elements of $[\ell]$ for some $\ell \geq 1$ (cf. Section 6 for a full definition). Recently the class of *labelable* phylogenetic networks was introduced in [10]. These networks are defined by the property that a certain variant of the labeling algorithm from [7] extends the labeling of the leaves in the network to a labeling of all vertices so that distinct vertices have distinct labels. A family of label subsets can then still be obtained from the children of internal vertices of the network as we did for the tree in Figure 1(b), but the subsets need no longer be disjoint, and form a so-called *expanding cover* rather than a partition.

Based on our labeling algorithm, we establish an alternative characterization of labelable phylogenetic networks based on a process used to create phylogenetic networks called *folding* [11]. This process identifies and overlays isomorphic subtrees in a multi-labeled tree in a systematic way so as to produce a phylogenetic network in which distinct leaves have distinct labels (see e.g. Figure 1(d)). The reverse process, called *unfolding*, turns a phylogenetic network into a multi-labeled tree. Note that unfolding a phylogenetic network and then folding the resulting multi-labeled tree may yield a phylogenetic network that is not isomorphic to the original network; in case it is isomorphic the network is called

stable [12]. Here we prove that the labelable phylogenetic networks are precisely the stable networks.

We also consider the problem of characterizing when a partition of a multiset corresponds to a multi-labeled tree relative to some ordering $\preceq$ of the set of all finite, non-empty multisets of positive integers. Interestingly, this appears to be a difficult problem for an arbitrary ordering $\preceq$. In [7] this issue is overcome by selecting some specific ordering. Here we show that it is possible to characterize when a partition of a multiset corresponds to a multi-labeled tree for a more general class of orderings that contains the anti-lexicographic ordering. In this way, we are then able to characterize when a partition of a multiset corresponds to certain subclasses of stable phylogenetic networks. Note that in [9] various classes of phylogenetic networks are characterized by properties of their corresponding expanding covers. Our approach therefore provides a complementary way to characterize certain phylogenetic network classes.

The rest of this paper is structured as follows. In Section 2 we first provide formal definitions of some of the concepts informally described in this introduction and then present our labeling algorithm (Algorithm 1). Then, in Section 3, we show how multi-labeled trees are uniquely determined by their associated partitions of multisets (Theorem 3.3). In Section 4 we first discuss the role of the ordering $\preceq$ used in Algorithm 1 leading to a description (Lemma 4.1) of those orderings that allow to characterize the partitions of multisets that correspond to multi-labeled trees (Theorem 4.3). In Section 5 we establish the equivalence of labelable and stable networks (Theorem 5.3). Based on this, in Section 6, we illustrate how some classes of phylogenetic networks can be characterized by properties of partitions of multisets. We conclude with some remarks on potential future directions and applications of our results.

2. Labelings and the labeling algorithm

In this section we present our labeling algorithm. We begin with some definitions. A *rooted network* $\mathcal{N} = (G, \rho)$ is a directed acyclic graph (DAG) G with a unique vertex ρ of in-degree 0. ρ is called the *root* of $\mathcal{N}$. A *leaf* of $\mathcal{N}$ is a vertex of out-degree 0 in G. We denote the set of leaves of $\mathcal{N}$ by $L(\mathcal{N})$. A vertex in G that is not a leaf is an *interior vertex* of $\mathcal{N}$. Let u be a vertex in a rooted network $\mathcal{N} = ((V, E), \rho)$. Then the set of all vertices v with $(u, v) \in E$ is denoted by C_u and the vertices in C_u are called the *children* of u. Note that a vertex in a rooted network is allowed to have in-degree and out-degree both equal to 1.

Let $\mathbb{N}^* = \mathbb{N} - \{0\}$ and, for any $n \in \mathbb{N}^*$, let $[n] = \{1, 2, \ldots, n\}$. A *leaf labeling* of a rooted network $\mathcal{N}$ is a map $\lambda : L(\mathcal{N}) \to \mathbb{N}^*$. Note that we don't require that λ is injective. A *leaf-labeled network* $(\mathcal{N}, \lambda)$ consists of a rooted network $\mathcal{N}$ and a leaf labeling λ of $\mathcal{N}$ (see Figure 2(a) for an example). Two leaf-labeled networks $((G = (V, E), \rho), \lambda)$ and $((G' = (V', E'), \rho'), \lambda')$ are *isomorphic* if there exists a DAG-isomorphism $f : V \to V'$ such that $\lambda(v) = \lambda'(f(v))$ for all $v \in L(G, \rho)$. A leaf-labeled network $(\mathcal{N}, \lambda)$ is called a *leaf-labeled tree* if all vertices in $\mathcal{N}$ have in-degree at most 1.

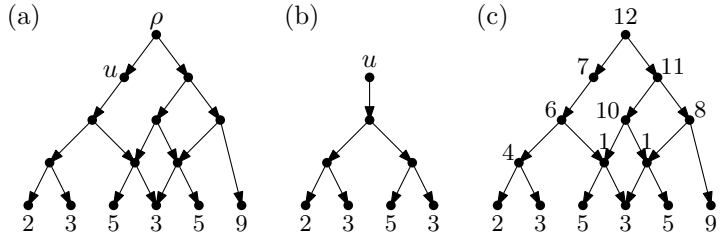

FIGURE 2. (a) A leaf-labeled network $(\mathcal{N}, \lambda)$. (b) The leaf-labeled network $(\mathcal{N}_u, \lambda_u)$ for the vertex u in (a). (c) The fully-labeled network produced by Algorithm 1 from the leaf-labeled network in (a) using as $\preceq$ the lexicographic ordering.

We remark that the definition of leaf-labeled networks may appear slightly more general than necessary. The motivation for this is twofold: We want it to include all the types of networks mentioned in the introduction (i.e. semi-labeled trees, multi-labeled trees and phylogenetic networks) and we want to avoid certain technical problems when considering induced subnetworks. To formally define the latter, let u be a vertex in a leaf-labeled network $(\mathcal{N} = (G = (V, E), \rho), \lambda)$. Then the leaf-labeled network $(\mathcal{N}_u, \lambda_u)$ consists of (see Figure 2(b) for an example)

- the rooted network $\mathcal{N}_u = (G_u, u)$ with root u that is obtained from the sub-DAG G_u of G induced by the set of those vertices $v \in V$ for which there exists a directed path from u to v in G, and
- the restriction λ_u of the map λ to the set of leaves of $\mathcal{N}_u$.

Given a leaf-labeled network $(\mathcal{N} = (G = (V, E), \rho), \lambda)$, we want to extend the leaf labeling λ to a map $\phi : V \to \mathbb{N}^*$. Such a map ϕ is called a *full labeling* of $\mathcal{N}$ and $(\mathcal{N}, \phi)$ is referred to as a *fully labeled network*. In addition, for an interior vertex u of a fully labeled network $(\mathcal{N}, \phi)$, let $F_{(u,\phi)} = \{\phi(v) : v \in C_u\}$ denote the multiset of labels assigned by ϕ to the children of u.

A particular full labeling is obtained by Algorithm 1. Let $\mathbb{M}$ denote the set of all finite, non-empty multisets whose elements are from the set $\mathbb{N}^*$. For a multiset $M \in \mathbb{M}$ and $x \in \mathbb{N}^*$, the multiplicity of x in M is denoted by $M(x)$, with $M(x) = 0$ indicating that x is not contained in M. Algorithm 1 is provided with an ordering $\preceq$ of the multisets in $\mathbb{M}$ (we write $M \prec M'$ for $M, M' \in \mathbb{M}$ if $M \preceq M'$ and $M \neq M'$). The full labeling produced by Algorithm 1 for a given leaf-labeled network $(\mathcal{N}, \lambda)$ depends on the choice of $\preceq$ and is denoted by $\phi_{(\lambda,\preceq)}$. In the literature, various orderings of the multisets in $\mathbb{M}$ have been considered (see e.g. [15]). This includes the *lexicographic ordering* defined by putting $M \preceq M'$ for $M, M' \in \mathbb{M}$ if either $M = M'$ or $M(x) < M'(x)$ for the smallest $x \in \mathbb{N}^*$ with $M(x) \neq M'(x)$. An example of the output of Algorithm 1 is shown in Figure 2(c).

Note that Algorithm 1 is related to a (reverse) *topological sort* of the vertices of a DAG (see e.g. [3, Sec. 20.4]). A DAG has, in general, many different orderings of its vertices that form a valid output of a topological sort. The ordering $\preceq$ on $\mathbb{M}$ used in Algorithm 1

Algorithm 1 Labeling the interior vertices of a leaf-labeled network $(\mathcal{N}, \lambda)$

procedure ComputeFullLabeling($\mathcal{N}$,λ)
  $\phi(v) \leftarrow \lambda(v)$ for all $v \in L(\mathcal{N})$ ▷ labels of leaves are given
  $\mathbb{L} \leftarrow \lambda(L(\mathcal{N}))$ ▷ set of labels currently used
  $U \leftarrow$ set of interior vertices of $\mathcal{N}$ ▷ set of currently unlabeled vertices
  while $U \neq \emptyset$ **do**
    $W \leftarrow \{u \in U$: all children of u have been labeled by $\phi\}$
    $W' \leftarrow \{w \in W : F_{(w,\phi)}$ is minimum with respect to $\preceq\}$
    $k \leftarrow \min(\mathbb{N}^* - \mathbb{L})$ ▷ select value used as next label
    $\phi(w) \leftarrow k$ for all $w \in W'$ ▷ label all vertices in W'
    $\mathbb{L} \leftarrow \mathbb{L} \cup \{k\}$ ▷ update set of labels currently used
    $U \leftarrow U - W'$ ▷ update set of unlabeled interior vertices
  end while
end procedure

essentially guides which of these orderings is chosen. In the following, we state some key properties of Algorithm 1.

Lemma 2.1. *Let $(\mathcal{T} = ((V, E), \rho), \lambda)$ be a leaf-labeled tree and $\phi = \phi_{(\lambda,\preceq)}$ be the full labeling of $\mathcal{T}$ produced by Algorithm 1. Then, for any two interior vertices $u, v \in V$, the leaf-labeled trees $(\mathcal{T}_u, \lambda_u)$ and $(\mathcal{T}_v, \lambda_v)$ are isomorphic if and only if $\phi(u) = \phi(v)$ (see Figure 3 for an example).*

*Proof:*Consider interior vertices $u, v \in V$. Since Algorithm 1 is deterministic, $(\mathcal{T}_u, \lambda_u)$ being isomorphic to $(\mathcal{T}_v, \lambda_v)$ immediately implies $\phi(u) = \phi(v)$.

Conversely, assume for a contradiction that there exist interior vertices $u, v \in V$ such that $\phi(u) = \phi(v)$ but $(\mathcal{T}_u, \lambda_u)$ is not isomorphic to $(\mathcal{T}_v, \lambda_v)$. Consider the first iteration of the while-loop of Algorithm 1 where such two vertices are assigned their labels. Since $\phi(u) = \phi(v)$ the multiset of labels of the children of u and v must coincide. But this implies, by the choice of the iteration, that there is a one-to-one correspondence between the leaf-labeled trees $(\mathcal{T}_w, \lambda_w)$, $w \in C_u$, and the leaf-labeled trees $(\mathcal{T}_{w'}, \lambda_{w'})$, $w' \in C_v$, such that the corresponding leaf-labeled trees are isomorphic. It follows that $(\mathcal{T}_u, \lambda_u)$ is isomorphic to $(\mathcal{T}_v, \lambda_v)$, a contradiction. ∎

The following can be regarded as being an analogue of [10, Thm. 3.3] and will form the basis for the definition of labelable networks (see Section 5).

Theorem 2.2. *Let $(\mathcal{N} = ((V, E), \rho), \lambda)$ be a leaf-labeled network. Then the following are equivalent:*

(a) *The full labeling $\phi = \phi_{(\lambda,\preceq)}$ produced by Algorithm 1 is injective.*
(b) *The leaf labeling λ is injective and $C_u = C_v$ implies $u = v$ for all $u, v \in V$.*

*Proof:*Clearly, (a) implies (b). So assume that (b) holds. Assume for a contradiction that there exist two distinct vertices $u, v \in V$ with $\phi(u) = \phi(v)$. Consider the first iteration of

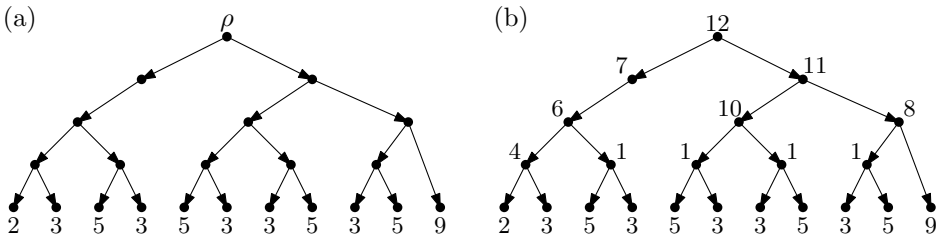

FIGURE 3. (a) A leaf-labeled tree $(\mathcal{T}, \lambda)$. (b) The full labeling $\phi = \phi_{(\lambda,\preceq)}$ of the leaf-labeled tree in (a) produced by Algorithm 1 using as $\preceq$ the lexicographic ordering. Interior vertices u, v with the same label have isomorphic leaf-labeled trees $(\mathcal{T}_u, \lambda_u)$ and $(\mathcal{T}_v, \lambda_v)$ (cf. Lemma 2.1).

the while-loop of Algorithm 1 where such two vertices are assigned their labels. By the choice of the iteration, the children of u and v, respectively, have pairwise distinct labels. Thus, since $C_u \neq C_v$, the set of labels of the children of u and v are different. But then u and v are not labeled in the same iteration, a contradiction. ∎

3. Partitions from leaf-labeled trees

In the last section we presented Algorithm 1 which, in particular, can be applied to any leaf-labeled tree $(\mathcal{T}, \lambda)$ to produce a full labeling of that tree. In this section, we first describe how a partition of a multiset can be obtained from this full labeling and then present an algorithm (Algorithm 2) that reconstructs the leaf-labeled tree $(\mathcal{T}, \lambda)$ from the so-generated partition.

Let $(\mathcal{T}, \lambda)$ be a leaf-labeled tree and $\phi = \phi_{(\lambda,\preceq)}$ be the full labeling of $\mathcal{T}$ produced by Algorithm 1. The multiset of labels of the leaves of $(\mathcal{T}, \lambda)$ is denoted by $M_{(\mathcal{T},\lambda)}$ and the multiset containing the labels of all of the vertices, except the root, of $\mathcal{T}$ is denoted by $P_{(\mathcal{T},\lambda,\preceq)}$. In addition, put

$$\Pi_{(\mathcal{T},\lambda,\preceq)} = \{F_{(u,\phi)} : u \text{ is an interior vertex of } \mathcal{T}\},$$

implying that $\Pi_{(\mathcal{T},\lambda,\preceq)}$ is a partition of $P_{(\mathcal{T},\lambda,\preceq)}$. As an example, consider the leaf-labeled tree $(\mathcal{T}, \lambda)$ in Figure 3(a) for which we obtain the multisets

$$\begin{aligned} M_{(\mathcal{T},\lambda)} &= \{2,3,3,3,3,3,5,5,5,5,9\},\\ P_{(\mathcal{T},\lambda,\preceq)} &= \{1,1,1,1,2,3,3,3,3,3,4,5,5,5,5,6,7,8,9,10,11\} \end{aligned}$$

of labels and the partition

$$\begin{aligned} \Pi_{(\mathcal{T},\lambda,\preceq)} = \{&\{2,3\},\{3,5\},\{3,5\},\{3,5\},\{3,5\},\{1,4\},\\ &\{1,1\},\{1,9\},\{6\},\{8,10\},\{7,11\}\}. \end{aligned}$$

We want to reconstruct the leaf-labeled tree $(\mathcal{T}, \lambda)$ from the partition $\Pi_{(\mathcal{T},\lambda,\preceq)}$. The following lemma will be our starting point.

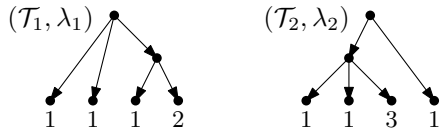

FIGURE 4. Two non-isomorphic leaf-labeled trees $(\mathcal{T}_1, \lambda_1)$ and $(\mathcal{T}_2, \lambda_2)$ with $\Pi_{(\mathcal{T}_1,\lambda_1,\preceq)} = \Pi_{(\mathcal{T}_2,\lambda_2,\preceq)} = \{\{1,2\},\{1,1,3\}\}$ (independently of the choice of $\preceq$ in Algorithm 1).

Lemma 3.1. *Let $(\mathcal{T}, \lambda)$ be a leaf-labeled tree. Then*

$$|P_{(\mathcal{T},\lambda,\preceq)}| + 1 - |\Pi_{(\mathcal{T},\lambda,\preceq)}|$$

equals the number of leaves of $\mathcal{T}$.

*Proof:*By definition, $|P_{(\mathcal{T},\lambda,\preceq)}|+1$ equals the number of all vertices of $\mathcal{T}$ and $|\Pi_{(\mathcal{T},\lambda,\preceq)}|$ equals the number of interior vertices of $\mathcal{T}$. ∎

So, we can extract some information about $(\mathcal{T}, \lambda)$ from $\Pi_{(\mathcal{T},\lambda,\preceq)}$, but, as can be seen in Figure 4, without any further assumptions, the partition $\Pi_{(\mathcal{T},\lambda,\preceq)}$ does not uniquely determine the leaf-labeled tree $(\mathcal{T}, \lambda)$.

A multiset $A \in \mathbb{M}$ is *gap-less* if $A(x) \geq 1$ for all $1 \leq x \leq \max(A)$. Let $\preceq$ be an ordering of the multisets in $\mathbb{M}$. A partition Π of a multiset $P \in \mathbb{M}$ is $\preceq$-*tree-generated* if there exists a leaf-labeled tree $(\mathcal{T}, \lambda)$ with $M_{(\mathcal{T},\lambda)}$ gap-less, $P = P_{(\mathcal{T},\lambda,\preceq)}$ and $\Pi = \Pi_{(\mathcal{T},\lambda,\preceq)}$. Moreover, for a partition Π of $P \in \mathbb{M}$, let $n_{(P,\Pi)}$ denote the element of P at position $|P|+1-|\Pi|$ when P is sorted non-decreasingly. As an example, consider the leaf-labeled tree $(\mathcal{T}, \lambda)$ in Figure 4(a). We have $M_{(\mathcal{T},\lambda)} = \{1,1,1,2\}$, which is gap-less, and $P = P_{(\mathcal{T},\lambda,\preceq)} = \{1,1,1,2,3\}$, with elements written in non-decreasing order. Hence, the partition

$$\Pi = \Pi_{(\mathcal{T},\lambda,\preceq)} = \{\{1,2\},\{1,1,3\}\}$$

of P is $\preceq$-tree-generated and $n_{(P,\Pi)} = 2$.

Lemma 3.2. *Let Π be a $\preceq$-tree-generated partition of $P \in \mathbb{M}$ and $(\mathcal{T}, \lambda)$ be a leaf-labeled tree with $P = P_{(\mathcal{T},\lambda,\preceq)}$ and $\Pi = \Pi_{(\mathcal{T},\lambda,\preceq)}$. Then*

$$M_{(\mathcal{T},\lambda)}(x) = \begin{cases} P(x) & \text{if } 1 \leq x \leq n_{(P,\Pi)} \\ 0 & \text{if } x > n_{(P,\Pi)}. \end{cases}$$

*Proof:*By Lemma 3.1, the number of leaves of $(\mathcal{T}, \lambda)$ equals $|P| + 1 - |\Pi|$. Since $M_{(\mathcal{T},\lambda)}$ is gap-less, the labels assigned by Algorithm 1 to interior vertices of $(\mathcal{T}, \lambda)$ are strictly larger than any element of $M_{(\mathcal{T},\lambda)}$. Thus, extracting the first $|P| + 1 - |\Pi|$ elements from the non-decreasingly sorted multiset P yields the elements of $M_{(\mathcal{T},\lambda)}$, with $n_{(P,\Pi)}$ being the largest element of $M_{(\mathcal{T},\lambda)}$. ∎

We are now ready to present Algorithm 2 which reconstructs a leaf-labeled tree from a $\preceq$-tree-generated partition.

Algorithm 2 Compute a leaf-labeled tree from a $\preceq$-tree-generated partition Π of a multiset $P \in \mathbb{M}$

```
1: procedure ComputeLeafLabeledTree(P, Π)
2:     n ← n_(P,Π)                                   ▷ largest label of a leaf
3:     M ← {x ∈ P : x ≤ n}                           ▷ multiset of leaf labels
4:     V ← set of |M| isolated vertices bijectively labeled by elements of M
5:     E ← ∅                                         ▷ initialization of arc set
6:     A ← M          ▷ multiset of labels of vertices currently having in-degree 0
7:     while Π ≠ ∅ do
8:         Π' ← minimal elements with respect to ⪯ in {F ∈ Π : F ⊆ A}
9:         n ← n + 1                                 ▷ next value used as a label
10:        for all F ∈ Π' do
11:            add a new vertex u to V and label u by n
12:            A(n) ← A(n) + 1
13:            for all x ∈ F do
14:                select v ∈ V with in-degree 0 and labeled by x
15:                add arc (u, v) to E
16:                A(x) ← A(x) − 1
17:            end for
18:        end for
19:        Π ← Π − Π'
20:    end while
21:    remove all labels of interior vertices        ▷ to obtain a leaf-labeled tree
22: end procedure
```

Theorem 3.3. *Let Π be a $\preceq$-tree-generated partition of a multiset $P \in \mathbb{M}$ and $(\mathcal{T}, \lambda)$ a leaf-labeled tree with $P = P_{(\mathcal{T},\lambda,\preceq)}$ and $\Pi = \Pi_{(\mathcal{T},\lambda,\preceq)}$. Then, on input P and Π, Algorithm 2 computes a leaf-labeled tree that is isomorphic to $(\mathcal{T}, \lambda)$.*

*Proof:*By Lemma 3.2 the set M of leaf labels of $(\mathcal{T}, \lambda)$ is uniquely determined by P and Π. Moreover, since M is gap-less, the set P is also gap-less and interior vertices of $\mathcal{T}$ are labeled by Algorithm 1 using the values $n_{(P,\Pi)} + 1, \ldots, \max(P)$.

We show that the following invariant holds for the while-loop in Algorithm 2: The current DAG (V, E) can be isomorphically embedded into $\mathcal{T}$ by the injective map f that assigns to each vertex v currently in V a vertex of $\mathcal{T}$ with the same label as v. This invariant clearly holds before the first iteration of the while-loop in Algorithm 2.

Consider an iteration of the while loop in Algorithm 2 and assume that the invariant holds before it. Then the multisets $F \in \Pi'$ selected in Line 8 of Algorithm 2 must coincide with the multisets $F_{(w,\phi)}$ considered in Line 7 of Algorithm 1. This ensures that the construction within the for-loop in Line 10 of Algorithm 2 can be performed and that the injective map f can be extended to the vertices u added in Line 11 of Algorithm 2 by assigning to them the vertices in W' considered in Line 7 of Algorithm 1.

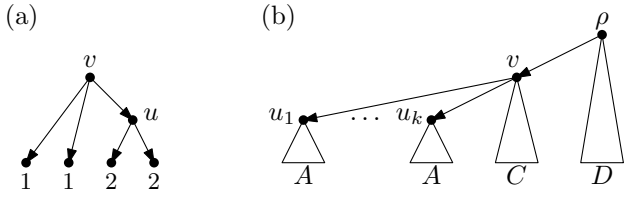

FIGURE 5. (a) A leaf-labeled tree illustrating that the lexicographic ordering of $\mathbb{M}$ is not labeling-consistent. (b) The structure of the leaf-labeled tree used in the proof of Lemma 4.1. Each triangle represents a set of leaves that are bijectively labeled by the elements of the multiset below the triangle.

It follows that the invariant also holds after the last iteration of the while loop in Algorithm 2, implying that the output of Algorithm 2 is a leaf-labeled tree that is isomorphic to $(\mathcal{T}, \lambda)$. ∎

Remark 3.4. Assuming that the outcome of the comparison of $A, B \in \mathbb{M}$ with respect to $\preceq$ can be computed in polynomial time, Algorithm 2 runs in polynomial time. Under the same assumption, Algorithm 2 can be adapted to decide in polynomial time (for a fixed ordering $\preceq$ of $\mathbb{M}$) if a given partition Π of $P \in \mathbb{M}$ is $\preceq$-tree-generated.

4. Labeling-consistent orderings of $\mathbb{M}$

Ideally, we would now like to characterize $\preceq$-tree-generated partitions. However, this appears to be difficult without placing some restriction on $\preceq$. The main obstacle is that sorting the elements in a $\preceq$-tree-generated partition of some $P \in \mathbb{M}$ non-decreasingly by $\preceq$ does not necessarily correspond to the order in which the elements of the partition are processed by Algorithm 2 (cf. Line 8). In [10], where a similar challenge arises, this is overcome by selecting, for each network, a suitable fixed order. In this section, we show that by placing a relatively mild assumption on the ordering $\preceq$ (labeling-consistency) we can characterize $\preceq$-tree-generated partitions for these orderings.

An ordering $\preceq$ of $\mathbb{M}$ is *labeling-consistent* if, for all leaf-labeled trees $(\mathcal{T} = ((V, E), \rho), \lambda)$ with $M_{(\mathcal{T},\lambda)}$ gap-less and all interior vertices $u, v \in V$,

$$\phi(u) \leq \phi(v) \text{ implies } F_{(u,\phi)} \preceq F_{(v,\phi)}, \tag{1}$$

where $\phi = \phi_{(\lambda,\preceq)}$. Note that the lexicographic ordering $\preceq$ of $\mathbb{M}$ defined in Section 2 is not labeling-consistent, as can be seen from the leaf-labeled tree $(\mathcal{T}, \lambda)$ in Figure 5(a) for which we have $\phi(u) = 3 \leq 4 = \phi(v)$ but $F_{(u,\phi)} = \{2, 2\} \not\preceq \{1, 1, 3\} = F_{(v,\phi)}$. An example of an ordering of $\mathbb{M}$ that is labeling-consistent (see Corollary 4.2 below) is the *anti-lexicographic ordering* defined by putting $M \preceq M'$ for $M, M' \in \mathbb{M}$ if either $M = M'$ or $M(x) < M'(x)$ for the largest $x \in \mathbb{N}^*$ with $M(x) \neq M'(x)$. This ordering is also considered in [7] (for finite non-empty subsets of $\mathbb{N}^*$). The following lemma characterizes the labeling-consistent orderings of $\mathbb{M}$.

Lemma 4.1. *An ordering $\preceq$ of $\mathbb{M}$ is labeling-consistent if and only if*

$$\max(A) < \max(B) \text{ implies } A \prec B \tag{2}$$

for all $A, B \in \mathbb{M}$.

*Proof:*First assume that $\preceq$ satisfies (2). We need to show that $\preceq$ is labeling-consistent. Let $(\mathcal{T}, \lambda)$ be a leaf-labeled tree with $M_{(\mathcal{T},\lambda)}$ gap-less and u, v be interior vertices of $\mathcal{T}$. Put $\phi = \phi_{(\lambda,\preceq)}$ and assume without loss of generality $\phi(u) \leq \phi(v)$. We consider two cases.

Case 1: In the iteration of the while-loop in Algorithm 1 where u is assigned its label $\phi(u)$, all children of v have been labeled already. Then, by the construction of the set W' in Line 7 of Algorithm 1, we must have $F_{(u,\phi)} \preceq F_{(v,\phi)}$, as required.

Case 2: In the iteration of the while-loop in Algorithm 1 where u is assigned its label $\phi(u)$, there exists a child w of v that has not been assigned a label yet. Since $M_{(\mathcal{T},\lambda)}$ is gap-less, we have $\max(F_{(u,\phi)}) < \phi(u) \leq \phi(w) \leq \max(F_{(v,\phi)})$. Thus, by the assumption that $\preceq$ satisfies (2), $F_{(u,\phi)} \prec F_{(v,\phi)}$, as required.

Next assume that there exist $A, B \in \mathbb{M}$ with $\max(A) < \max(B)$ and $B \prec A$. We need to show that $\preceq$ is not labeling-consistent. Put $m = \max(B)$ and $k = B(m)$. Consider the multisets $C, D \in \mathbb{M}$ with

$$C(x) = \begin{cases} B(x) & \text{if } x < m \\ 0 & \text{if } x \geq m \end{cases} \quad \text{and} \quad D(x) = \begin{cases} 1 & \text{if } x < m \\ 0 & \text{if } x \geq m \end{cases}$$

We consider the leaf-labeled tree $(\mathcal{T}, \lambda)$ whose structure is depicted in Figure 5(b). $\mathcal{T}$ has $k \cdot |A| + |C| + |D|$ leaves. More specifically, each of the interior vertices u_i, $1 \leq i \leq k$, has $|A|$ children that are all leaves and bijectively labeled by the elements of A. The interior vertex v has $k + |C|$ children, among them $|C|$ leaves that are bijectively labeled by the elements of C. Finally, the root of $\mathcal{T}$ has $1 + |D|$ children, among them $|D|$ leaves that are bijectively labeled by the elements of D. Then, by the definition of D, $M_{(\mathcal{T},\lambda)}$ is gap-less.

Put $\phi = \phi_{(\lambda,\preceq)}$. Then we have $\phi(u_i) = m$ for $1 \leq i \leq k$, $\phi(v) = m + 1$ and $\phi(\rho) = m + 2$. Thus, putting $u = u_1$, we have $\phi(u) < \phi(v)$ but $A = F_{(u,\phi)} \not\prec F_{(v,\phi)} = B$, implying that $\preceq$ is not labeling-consistent, as required. ∎

Corollary 4.2. *The anti-lexicographic ordering of $\mathbb{M}$ is labeling-consistent.*

*Proof:*It follows immediately from the definition of the anti-lexicographic ordering that (2) holds. ∎

For a partition Π of $P \in \mathbb{M}$ and $F \in \Pi$, the multiplicity of F in Π is denoted by $\Pi(F)$. Moreover, $\underline{\Pi}$ denotes the set containing a single copy of each $F \in \Pi$.

Theorem 4.3. *Let $\preceq$ be a labeling-consistent ordering of $\mathbb{M}$, Π be a partition of $P \in \mathbb{M}$ and $F_1, \ldots, F_k$ be the elements of $\underline{\Pi}$ sorted increasingly by $\preceq$. Then Π is $\preceq$-tree-generated if and only if*

(i) *P is gap-less,*
(ii) $|\underline{\Pi}| = \max(P) - n_{(P,\Pi)} = k$,

(iii) $P(i + n_{(P,\Pi)}) = \Pi(F_i)$ *for all* $1 \leq i \leq k$, *and*
(iv) $\max(F_i) \leq n_{(P,\Pi)} + i - 1$ *for all* $1 \leq i \leq k$.

*Proof:*First assume that Π is $\preceq$-tree-generated. Let $(\mathcal{T}, \lambda)$ be a leaf-labeled tree with $P = P_{(\mathcal{T},\lambda,\preceq)}$ and $\Pi = \Pi_{(\mathcal{T},\lambda,\preceq)}$. Since Π is $\preceq$-tree-generated, $M_{(\mathcal{T},\lambda)}$ is gap-less, implying (i). (ii) follows from Lemma 3.2. (iii) follows from the definition of $\Pi_{(\mathcal{T},\lambda,\preceq)}$ and the assumption that $\preceq$ as labeling-consistent. (iv) follows from the fact that, since $M_{(\mathcal{T},\lambda)}$ is gap-less, before the i-th iteration of the while-loop in Algorithm 1 the maximum label used so far is $n_{(P,\Pi)} + i - 1$.

Next assume that Π satisfies Properties (i)-(iv). We show that Algorithm 2 on input P and Π produces a leaf-labeled tree $(\mathcal{T}, \lambda)$ with $P = P_{(\mathcal{T},\lambda,\preceq)}$ and $\Pi = \Pi_{(\mathcal{T},\lambda,\preceq)}$. Properties (i)-(iii) ensure that the set M constructed in Line 3 of Algorithm 2 can be used as the multiset of leaf-labels of $(\mathcal{T}, \lambda)$.

To finish the proof, it suffices to show that the following invariant holds for the while-loop in Algorithm 2: The multiset $\{F \in \Pi : F \subseteq A\}$ is not empty. This invariant clearly holds before the first iteration of the while-loop in Algorithm 2 since, by (iv), $\max(F_1) \leq n_{(P,\Pi)} = \max(M)$.

Consider the i-th iteration, $1 \leq i \leq k$, of the while loop in Algorithm 2 and assume that the invariant holds before it. Then $\Pi(F_i)$ new vertices are added in the for-loop in Line 10 of Algorithm 2. All these new vertices are assigned $n_{(P,\Pi)} + i$ as their label. Thus, by (iv) and since Π is a partition of P, we must have $F_{i+1} \subseteq A$ and, therefore, the invariant holds also after the i-th iteration of the while loop in Algorithm 2. ∎

5. Folding and unfolding

We now shift our focus from leaf-labeled trees to more general leaf-labeled networks. We call a leaf-labeled network $(\mathcal{N}, \lambda)$ *labelable* if the full labeling $\phi = \phi_{(\lambda,\preceq)}$ of the network produced by Algorithm 1 is injective. Note that in view of Theorem 2.2, the choice of the ordering $\preceq$ has no impact on whether or not a leaf-labeled network is labelable. In this section, we present a characterization of labelable networks (Theorem 5.3), which we shall use in the next section to give a new characterization of labelable phylogenetic networks.

Let $(\mathcal{T} = ((V, E), \rho), \lambda)$ be a leaf-labeled tree and $\phi = \phi_{(\lambda,\preceq)}$ be the full labeling of $\mathcal{T}$ produced by Algorithm 1. Then $\mathfrak{F}(\mathcal{T}, \phi)$ denotes the fully labeled network $(((V', E'), \rho'), \phi')$ obtained as follows:

- $V' = \phi(V)$.
- $E' = \{(\phi(u), \phi(v)) : (u, v) \in E\}$.
- $\rho' = \phi(\rho)$.
- $\phi' : V' \to \mathbb{N}^*$ with $\phi'(k) = k$.

The fully labeled network $\mathfrak{F}(\mathcal{T}, \phi)$ is called the *folding* of $(\mathcal{T}, \phi)$ (see Figure 6 for an example). Note that in [12] the folding of a leaf-labeled tree $(\mathcal{T} = ((V, E), \rho), \lambda)$ is described in terms of isomorphisms between the leaf-labeled trees $(\mathcal{T}_u, \lambda_u)$, $u \in V$. In view of Lemma 2.1, this is equivalent to the definition above.

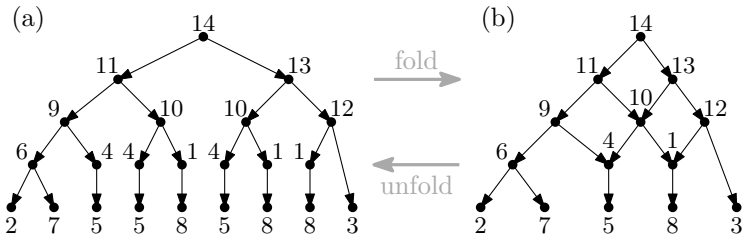

FIGURE 6. Illustration of folding and unfolding. (a) A fully labeled tree $(\mathcal{T}, \phi)$. The full labeling ϕ is obtained by applying Algorithm 1 using as $\preceq$ the lexicographic ordering. (b) The fully labeled network $(\mathcal{N}', \phi') = \mathfrak{F}(\mathcal{T}, \phi)$ obtained by folding $(\mathcal{T}, \phi)$. Unfolding $(\mathcal{N}', \phi')$ reproduces $(\mathcal{T}, \phi)$.

Next we describe the reverse process. Let $(\mathcal{N} = ((V, E), \rho), \phi)$ be a fully labeled network. Then $\mathfrak{U}(\mathcal{N}, \phi)$ denotes the fully labeled tree $(((V', E'), \rho'), \phi')$ obtained as follows:

- V' consists of the set of all directed paths in $\mathcal{N}$ from ρ to some $v \in V$.
- (π, π') is an arc in E' if the directed path π' arises from the directed path π by extending π along an arc in E.
- ρ' is the directed path in $\mathcal{N}$ consisting of the single vertex ρ.
- $\phi' : V' \to \mathbb{N}^*$ with $\phi'(\pi) = \phi(u)$, where u is the last vertex in the directed path π.

The fully labeled tree $\mathfrak{U}(\mathcal{N}, \phi)$ is called the *unfolding* of $(\mathcal{N}, \phi)$ (see again Figure 6 for an example).

The next lemma states more precisely the idea that the process of unfolding a fully-labeled network and the application of Algorithm 1 are compatible.

Lemma 5.1. *Let $(\mathcal{N} = ((V, E), \rho), \lambda)$ be a leaf-labeled network, $\phi = \phi_{(\lambda, \preceq)}$ be the full labeling of $\mathcal{N}$ produced by Algorithm 1, $(\mathcal{T}', \phi') = \mathfrak{U}(\mathcal{N}, \phi)$ and $\lambda' : L(\mathcal{T}') \to \mathbb{N}^*$ be the leaf labeling of $\mathcal{T}'$ obtained by putting $\lambda'(v') = \phi'(v')$ for all $v' \in L(\mathcal{T}')$. Then the full labeling $\phi'' = \phi_{(\lambda', \preceq)}$ of $\mathcal{T}'$ produced by Algorithm 1 coincides with ϕ'.*

*Proof:*The proof is by induction over the iterations of the while loop in Algorithm 1. The base case involves the leaves of $\mathcal{T}'$ and clearly holds.

If an interior vertex u' of $\mathcal{T}'$ is labeled by ϕ'' in an iteration of the while loop, all children of u' have been labeled by ϕ'' in some earlier iteration. Hence, by induction, we have $\phi''(w') = \phi'(w')$ for all children w' of u'. By construction of $(\mathcal{T}', \phi')$, u' corresponds to a directed path π from ρ to some vertex u in $\mathcal{N}$. Moreover, the children of u' in $\mathcal{T}'$ correspond to the extensions of π by an arc from u to a child of u in $\mathcal{N}$. Hence, the multiset of labels of the children of u' in $\mathcal{T}'$ coincides with the multiset of labels of the children of u in $\mathcal{N}$. But this implies that $\phi(u) = \phi'(u') = \phi''(u')$, as required. ∎

Remark 5.2. From the proof of Lemma 5.1 we see that Algorithm 1, when applied to $(\mathcal{N}, \lambda)$ and to $(\mathcal{T}', \lambda')$, respectively, labels, in each iteration, vertices whose children have the same multiset of labels.

A leaf-labeled network $(\mathcal{N}, \lambda)$ is called *stable* if the fully labeled network $(\mathcal{N}, \phi_{(\lambda,\preceq)})$ is isomorphic to $\mathfrak{F}(\mathfrak{U}(\mathcal{N}, \phi_{(\lambda,\preceq)}))$ [12]. As can be seen in Figure 6, stable networks exist. In contrast, the leaf-labeled network in Figure 2(a) is not stable, since unfolding it yields the fully labeled tree in Figure 3(b) but folding this tree does not reproduce the fully labeled network in Figure 2(b).

We now present the aforementioned characterization of labelable networks.

Theorem 5.3. *A leaf-labeled network $(\mathcal{N}, \lambda)$ is labelable if and only if it is stable.*

*Proof:*Assume that $(\mathcal{N}, \lambda)$ is stable. Then the full labeling $\phi_{(\lambda,\preceq)}$ of $\mathcal{N}$ produced by Algorithm 1 must be injective since the full labeling of $\mathfrak{F}(\mathfrak{U}(\mathcal{N}, \phi_{(\lambda,\preceq)}))$ is injective by the definition of the folding. Hence, $(\mathcal{N}, \lambda)$ is labelable.

Next assume that $(\mathcal{N}, \lambda)$ is labelable. Then the full labeling $\phi = \phi_{(\lambda,\preceq)}$ of $\mathcal{N}$ produced by Algorithm 1 is injective by definition. Then the required isomorphism f between $(\mathcal{N}, \lambda)$ and $\mathfrak{F}(\mathfrak{U}(\mathcal{N}, \phi_{(\lambda,\preceq)}))$ is obtained by putting $f(u) = \phi(u)$ for all vertices u of $\mathcal{N}$. ∎

6. Classes of phylogenetic networks through an expanding partition lens

A leaf-labeled network $(\mathcal{N}, \lambda)$ is a *phylogenetic network* if all leaves of $\mathcal{N}$ have in-degree 1 and λ is a bijection between $L(\mathcal{N})$ and $[n]$ for $n = |L(\mathcal{N})|$ [10]. We also call such a leaf-labeled network a *phylogenetic network on* $[n]$ for short. In [10] a bijection is established between the set of labelable phylogenetic networks on $[n]$ and expanding covers (defined below). In this section, via foldings, we extend this bijective correspondence to include stable phylogenetic networks and a certain class of partitions of multisets.

We first recall a key definition from [10]. An *expanding cover* is a collection $\mathcal{C}$ of non-empty subsets of $[m]$ for some $m \in \mathbb{N}^*$ such that

(EC1) $\bigcup_{C\in\mathcal{C}} C = [m]$,
(EC2) $n_{\mathcal{C}} = m - |\mathcal{C}| + 1 \geq 1$,
(EC3) For all $x \in [n_{\mathcal{C}}]$ there exists a unique $C \in \mathcal{C}$ with $x \in C$, and
(EC4) For all $1 \leq i \leq |\mathcal{C}|$ there exist at least i sets $C \in \mathcal{C}$ with $C \subseteq [n_{\mathcal{C}} + i - 1]$.

Now, let $\preceq$ be an ordering of $\mathbb{M}$. A $\preceq$-*expanding partition* is a $\preceq$-tree-generated partition Π of some $P \in \mathbb{M}^*$ such that

(EP1) for all $F \in \Pi$ and all $x \in F$ we have $F(x) = 1$, and
(EP2) For all $F, F' \in \Pi$ with $F \neq F'$ we have $F \cap F' \cap [n_{(P,\Pi)}] = \emptyset$.

Since P is determined by Π, we put $n_\Pi = n_{(P,\Pi)}$ for such a partition. We now extend the correspondence given in [10, Theorem 4.4].

Theorem 6.1. *Let $n \in \mathbb{N}^*$ and $\preceq$ an ordering of $\mathbb{M}$. The following sets are in bijective correspondence:*

(i) *The set of labelable phylogenetic networks on $[n]$.*
(ii) *The set of stable phylogenetic networks on $[n]$.*
(iii) *The set of expanding covers $\mathcal{C}$ with $n_{\mathcal{C}} = n$.*
(iv) *The set of $\preceq$-expanding partitions Π with $n_\Pi = n$.*

*Proof:*The bijective correspondence between the sets in (i) and (ii) follows from Theorem 5.3.

The bijective correspondence between the sets in (i) and (iii) is established in [10, Thm. 4.4].

To establish the bijective correspondence between the sets in (ii) and (iv), we consider the map p that associates to each stable phylogenetic network $(\mathcal{N}', \lambda')$ on $[n]$ the partition $\Pi_{(\mathcal{T},\lambda,\preceq)}$ of $P_{(\mathcal{T},\lambda,\preceq)}$, where $(\mathcal{T}, \lambda)$ is the leaf-labeled tree obtained by unfolding the fully labeled network $(\mathcal{N}', \phi_{(\lambda',\preceq)})$ and then restricting the labeling of the fully labeled tree $\mathfrak{U}(\mathcal{N}', \phi_{(\lambda',\preceq)})$ to its leaves. By Lemma 5.1, the fully labeled tree $(\mathcal{T}, \phi_{(\lambda,\preceq)})$ coincides with $\mathfrak{U}(\mathcal{N}', \phi_{(\lambda',\preceq)})$. Moreover, since $(\mathcal{N}', \lambda')$ is stable and, therefore, labelable, the full labeling $\phi_{(\lambda',\preceq)}$ of $\mathcal{N}'$ is, by definition, injective. But this implies, by the definition of $\mathfrak{U}(\mathcal{N}', \phi_{(\lambda',\preceq)})$, that $\Pi_{(\mathcal{T},\lambda,\preceq)}$ satisfies (EP1). In addition, since $(\mathcal{N}', \lambda')$ is a phylogenetic network on $[n]$, $n_{\Pi_{(\mathcal{T},\lambda,\preceq)}} = n$ and $\Pi_{(\mathcal{T},\lambda,\preceq)}$ satisfies (EP2). Thus, p maps every stable phylogenetic network on $[n]$ to a $\preceq$-expanding partition Π with $n_\Pi = n$.

It remains to show that the map p is bijective. It follows from the definition of stability of a leaf-labeled network together with Theorem 3.3 that p is injective. To show that p is also surjective, consider a $\preceq$-expanding partition Π of $P \in \mathbb{M}$ with $n_\Pi = n$. Let $(\mathcal{T}, \lambda)$ be the leaf-labeled tree produced by Algorithm 2 for Π and P. In addition, let $(\mathcal{N}', \lambda')$ be the leaf-labeled network obtained by folding the fully labeled tree $(\mathcal{T}, \phi_{(\lambda,\preceq)})$ and then restricting the labeling of the fully labeled network $\mathfrak{F}(\mathcal{T}, \phi_{(\lambda,\preceq)})$ to its leaves. Since $n_\Pi = n$, λ' is a bijection between n and the set of leaves of $\mathcal{N}'$. Moreover, since Π satisfies (EP2), all leaves of $\mathcal{N}'$ have in-degree 1. Hence, $(\mathcal{N}', \lambda')$ is a phylogenetic network on $[n]$.

We claim that $\mathfrak{U}(\mathcal{N}', \phi_{(\lambda',\preceq)})$ is isomorphic to $(\mathcal{T}, \phi_{(\lambda,\preceq)})$. Note that this claim immediately implies that $(\mathcal{N}', \lambda')$ is stable and $p(\mathcal{N}', \lambda') = \Pi$, as required. Put $\phi = \phi_{(\lambda,\preceq)}$ and let ρ denote the root of $\mathcal{T}$. To establish the claim, it suffices to show, in view of the definition of the unfolding of a leaf-labeled network, that, for any two directed paths $\rho = v_1, \ldots, v_k$ and $\rho = u_1, \ldots, u_\ell$ in $(\mathcal{T}, \phi_{(\lambda,\preceq)})$ with $v_k \neq u_\ell$, the sequences $\phi(v_1), \ldots, \phi(v_k)$ and $\phi(u_1), \ldots, \phi(u_\ell)$ of vertex labels differ. Assume for a contradiction that there exist two such paths with the same sequence of vertex labels. Then we have $k = \ell$ and, since $v_1 = \rho = u_1$ and $v_k \neq u_k$, there exists a unique $1 \leq i < k$ with $v_i = u_i$ and $v_{i+1} \neq u_{i+1}$. By our assumption we have $\phi(v_{i+1}) = \phi(u_{i+1})$, implying that the multiplicity of the label $\phi(v_{i+1})$ in $F_{(v_i,\phi)}$ is at least 2. This is a contradiction to the fact that, by construction, $F_{(v_i,\phi)} \in \Pi$ and Π satisfies (EP1). ∎

Remark 6.2. The expanding cover associated with a phylogenetic network arises in a similar way as a $\preceq$-tree-generated partition from a leaf-labeled tree (see [10] for details). In particular, it follows from the proof of Theorem 6.1 that the expanding cover $\mathcal{C}$ that corresponds to an expanding partition Π coincides with $\underline{\Pi}$.

In [9, 10] some classes of phylogenetic networks are characterized in terms of their associated expanding covers. In view of Remark 6.2, these can be translated quite easily into characterizations in terms of their associated expanding partitions. We conclude this section by illustrating this process for two classes of phylogenetic networks.

A phylogenetic network is *non-degenerate* (again as defined in [10]) if it has no vertex with both in-degree and out-degree equal to 1 and also no vertex with both in-degree and out-degree strictly larger than 1.

Corollary 6.3. *Let $n \in \mathbb{N}^*$ and $\preceq$ a labeling-consistent ordering of $\mathbb{M}$. The following sets are in bijective correspondence:*

(i) *The set of non-degenerate labelable phylogenetic networks on $[n]$.*
(ii) *The set of $\preceq$-expanding partitions Π with $n_\Pi = n$ satisfying, for all $1 \leq i < k = |\underline{\Pi}|$, that*
   - *$|F_i| > 1$ implies that are are no $F, F' \in \Pi$ with $F \neq F'$, $F(n+i) \geq 1$ and $F'(n+i) \geq 1$, and*
   - *$|F_i| = 1$ implies that there are $F, F' \in \Pi$ with $F \neq F'$, $F(n+i) \geq 1$ and $F'(n+i) \geq 1$,*

   where $F_1, \ldots, F_k$ are the elements of $\underline{\Pi}$ sorted increasingly by $\preceq$.

*Proof:*This is a translation of [10, Thm. 5.1]. ∎

A phylogenetic network is a *tree-child network* if every interior vertex has a child with in-degree 1. It is shown in [10, Thm. 6.4] that tree-child networks are labelable.

Corollary 6.4. *Let $n \in \mathbb{N}^*$ and $\preceq$ an ordering of $\mathbb{M}$. The following sets are in bijective correspondence:*

(i) *The set of tree-child networks on $[n]$.*
(ii) *The set of $\preceq$-expanding partitions Π with $n_\Pi = n$ and the property that, for all $F \in \Pi$, there exists $x \in F$ such that $F'(x) = 0$ for all $F' \in \Pi$ with $F' \neq F$.*

*Proof:*This is a translation of [9, Thm. 4.1]. ∎

7. Concluding remarks

We conclude by suggesting some possible future directions. In the last section we characterized when $\preceq$-expanding partitions correspond to tree-child networks. It could be interesting to find similar characterizations for other classes of networks such as tree-based networks, and tree-sibling networks (see e.g. [13] for definitions of these and other classes of networks). Note that this question has been studied in the context of expanding covers in [9] (see e.g. Theorems 3.2 and 6.2 for tree-based and tree-sibling networks, respectively). More generally, it could also be interesting to study network classes that are not labelable from the labeling perspective. In particular, any DAG can be topologically sorted, and so we could ask under what assumptions is it possible to recover *any* leaf-labeled network from data produced by a sorting/labeling of its vertices?

In a different direction, in [8] tree labelings were used to produce a semi-group structure on the set of binary phylogenetic trees via the multiplication of Brauer diagrams. In that paper, it is also asked if it is possible to represent phylogenetic networks within a diagram semigroup framework (see p. 23). It would be interesting to investigate if our results could

be used to help answer this question. Moreover, in a similar vein, in [6, 8] bijections are given for rooted semi-labeled forests, and it might be worthwhile to see if our results can be developed in the multi-labeled setting. This could be interesting since it might help shed light on counting multi-labeled trees/forests where relatively little is known (see [4] for some example results).

Finally, encoding phylogenetic trees using labelings has recently become highly topical in phylogenetics since such labelings can be used to encode trees in terms of vectors. These so-called *vector encodings* are highly amenable to processing with machine learning algorithms which can be useful when dealing with massive collections of trees (see e.g. [2, 16, 17]). It could be useful to adapt the approach we detail here to derive similar encodings for stable phylogenetic networks to, for example, help with their fast comparison or in the computation of consensus networks. It would also be interesting to understand how comparison of vector encodings for (subclasses of) stable networks would compare with other phylogenetic network metrics (see e.g. [1]), a topic which has recently been studied for phylogenetic trees in [14].

Acknowledgment: VM would like to thank the Institute for Computational and Experimental Research in Mathematics in Providence, RI, where – during the semester program on "Theory, Methods, and Applications of Quantitative Phylogenomics" – he did some of the research presented in this paper. The authors also thank Katharina Huber for pointing out the connection mentioned above between labelings and vector encodings of phylogenetic trees.

References

[1] G. Cardona, F. Rosselló, and G. Valiente. Comparison of tree-child phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(4):552–569, 2008.

[2] C. Chauve, C. Colijn, and L. Zhang. A vector representation for phylogenetic trees. *Philosophical Transactions B*, 380:20240226, 2025.

[3] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2022.

[4] É. Czabarka, P.L. Erdős, V. Johnson, and V. Moulton. Generating functions for multi-labeled trees. *Discrete Applied Mathematics*, 161(1-2):107–117, 2013.

[5] P. Diaconis and S. Holmes. Matchings and phylogenetic trees. *Proceedings of the National Academy of Sciences*, 95(25):14600–14602, 1998.

[6] P.L. Erdős. A new bijection on rooted forests. *Discrete Mathematics*, 111(1-3):179–188, 1993.

[7] P.L. Erdős and L. Székely. Applications of antilexicographic order. I. An enumerative theory of trees. *Advances in Applied Mathematics*, 10(4):488–496, 1989.

[8] A. Francis and P. Jarvis. Brauer and partition diagram models for phylogenetic trees and forests. *Proceedings of the Royal Society A*, 478(2262):20220044, 2022.

[9] A. Francis, D. Marchei, and M. Steel. Phylogenetic network classes through the lens of expanding covers. *Journal of Mathematical Biology*, 88(5):58, 2024.

[10] A. Francis and M. Steel. Labellable phylogenetic networks. *Bulletin of Mathematical Biology*, 85(6):46, 2023.

[11] K. Huber and V. Moulton. Phylogenetic networks from multi-labelled trees. *Journal of Mathematical Biology*, 52:613–632, 2006.

[12] K. Huber, V. Moulton, M. Steel, and T. Wu. Folding and unfolding phylogenetic trees and networks. *Journal of Mathematical Biology*, 73:1761–1780, 2016.
[13] S. Kong, J. Pons, L. Kubatko, and K. Wicke. Classes of explicit phylogenetic networks and their biological and mathematical significance. *Journal of Mathematical Biology*, 84(6):47, 2022.
[14] S. Linz, K. St. John, C. Semple, and K. Wicke. Order-dependent dissimilarity measures on phylogenetic trees. *arXiv preprint arXiv:2507.11254*, 2025.
[15] U. Martin. A geometrical approach to multiset orderings. *Theoretical Computer Science*, 67(1):37–54, 1989.
[16] M. Penn, N. Scheidwasser, M. Khurana, D. Duchêne, C. Donnelly, and S. Bhatt. Phylo2Vec: a vector representation for binary trees. *Systematic Biology*, 74(2):250–266, 2025.
[17] H. Richman, C. Zhang, and F. Matsen IV. Vector encoding of phylogenetic trees by ordered leaf attachment. *arXiv preprint arXiv:2503.10169*, 2025.
[18] M. Steel. *Phylogeny: Discrete and random processes in evolution*. SIAM, 2016.