Bulletin of Mathematical Biology (2024) 86:10
https://doi.org/10.1007/s11538-023-01239-z

Society for Mathematical Biology

ORIGINAL ARTICLE

Generation of Orchard and Tree-Child Networks

Gabriel Cardona[1] · Gerard Ribas[2] · Joan Carles Pons[1]

Received: 29 July 2023 / Accepted: 22 November 2023 / Published online: 20 December 2023
© The Author(s) 2023

Abstract
Phylogenetic networks are an extension of phylogenetic trees that allow for the representation of reticulate evolution events. One of the classes of networks that has gained the attention of the scientific community over the last years is the class of orchard networks, that generalizes tree-child networks, one of the most studied classes of networks. In this paper we focus on the combinatorial and algorithmic problem of the generation of binary orchard networks, and also of binary tree-child networks. To this end, we use that these networks are defined as those that can be recovered by reversing a certain reduction process. Then, we show how to choose a "minimum" reduction process among all that can be applied to a network, and hence we get a unique representation of the network that, in fact, can be given in terms of sequences of pairs of integers, whose length is related to the number of leaves and reticulations of the network. Therefore, the generation of networks is reduced to the generation of such sequences of pairs. Our main result is a recursive method for the efficient generation of all minimum sequences, and hence of all orchard (or tree-child) networks with a given number of leaves and reticulations. An implementation in C of the algorithms described in this paper, along with some computational experiments, can be downloaded from the public repository https://github.com/gerardet46/OrchardGenerator. Using this implementation, we have computed the number of binary orchard networks with at most 6 leaves and 8 reticulations.

Keywords Phylogenetic network · Orchard · Tree-child · Generation

✉ Joan Carles Pons
joancarles.pons@uib.es

1 Department of Mathematics and Computer Science, University of the Balearic Islands, Ctra. Valldemossa, km. 7.5, 07122 Palma, Spain

2 Higher Polytechnic School, University of the Balearic Islands, Ctra. Valldemossa, km. 7.5, 07122 Palma, Spain

1 Introduction

For decades, phylogenetic trees have been the model used to represent the branching pattern for the evolution of a set of Operational Taxonomic Units (OTUs for short). From the 1980 s onward, it became evident that phylogenetic networks were a more accurate framework, with the potential to cover more complex evolutionary scenarios such as hybridizations, recombinations, or lateral gene transfers.

In the broadest sense, phylogenetic networks are directed acyclic graphs whose leaves are labelled by the organisms under study. This general definition, while allowing a wide range of biological processes to be considered, lacks mathematical tractability. For this reason, some other constraints must be considered, resulting in a wide variety of classes of phylogenetic networks (see Kong et al. 2022 for a recent review, or Steel 2016, Chapter 10). In this work we focus on the class of *orchard* networks (Erdős et al. 2019) (also called *cherry-picking* networks Janssen and Murakami 2021) and *tree-child* networks (Cardona et al. 2009), a subclass of the first and one of the most explored classes of networks.

Orchard networks are networks that can be reduced to a trivial network by iteratively identifying and reducing certain substructures (namely, cherries and reticulated cherries) involving two leaves. Orchard networks are one of those classes of networks with biological significance (according to Kong et al. 2022) because they can be viewed as a backbone tree with additional "horizontal" arcs (see van Iersel et al. 2022 for more details).

One of the relevant problems in the study of phylogenetic networks is that of their sequential generation; that is, obtaining a method to generate them in an efficient and unique way. Generation of phylogenetic networks is useful, for example, for testing the performance of methods in phylogenetics and for testing hypotheses about the evolutionary relationships among organisms by the comparison of different network topologies.

Up to our knowledge, there exist two previous works on the generation of orchard networks. First, the generator of LGT networks introduced in (Pons et al. 2019) can be adapted to generate orchard networks, simply by forgetting the distinction between principal and secondary arcs ending in reticulations. Second, in (Janssen and Liu 2021), the authors present an extension of the beta-splitting tree generator (Aldous 1996) that adds reticulations to generated trees in various ways; one of them, called n-type reticulations, produces orchard networks. It should be noted that none of these two methods generates the networks in an unique way, and in particular they cannot be used to attack the problem of its counting. The situation for tree-child networks is slightly better, since there are previous works on the enumeration (Fuchs et al. 2021; Pons and Batle 2021) and generation (Cardona et al. 2019; Cardona and Zhang 2020) of this kind of networks, but much less efficiently than the method given here (see Sect. 8); finally, there exist results on the asymptotic behavior of the number of tree-child networks (Fuchs et al. 2021).

In this paper, we shall focus on the problem of the effective and injective generation of binary orchard and tree-child phylogenetic networks; that is, no pair of generated networks will be the same (technically, isomorphic), and we can promptly get many networks with the number of leaves and reticulations that we want. Our method of generation is based on the construction of sequences of pairs of integers that encode orchard (and, in particular, tree-child) networks as introduced in (Janssen and Murakami 2021). However, there are different sequences that generate the same network, so that we choose among them a *minimum* one that uniquely represents it. Hence, our strategy to generate orchard (and tree-child) networks is based on the generation of those minimum sequences.

The paper is organized as follows. In Sect. 2 we give basic definitions used throughout the manuscript. In Sect. 3 we define orchard networks and how they can be reduced by means of reducible pairs sequences. In Sect. 4 we show that we can choose a minimum (in a sense to be defined) reducible pairs sequence in order to uniquely identify an orchard network up to isomorphism. Section 5 shows how the reduction of a pair can be reverted by means of augmentations, and in Sect. 6 it is used to describe how to recover an orchard network by reversing the whole reduction process, and how this process, together with the uniqueness of the minimum reducible pairs sequence, allows us to generate orchard networks injectively. In Sect. 7 we adapt our methods to generate tree-child networks, which constitute a relevant subclass of orchard networks. In Sect. 8 we present the implementation we have made of the methods contained in this paper and exhibit some computational experiments we have performed, including the computation of the number of binary orchard networks with up to 6 leaves and 8 reticulations. Finally, Sect. 9 contains the conclusions of the manuscript and some possible directions of future work.

2 Preliminaries

Throughout the paper, for any positive integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$.

The graphs $N = (V, A)$ we shall work with are directed and acyclic. Given two nodes $u, v \in V$, if there is an arc with tail u and head v (or from u to v), we denote it as uv. In that case, u is a *parent* of v and v is a *child* of u.

Given a node $u \in V$, $\operatorname{indeg} u$ (resp. $\operatorname{outdeg} u$) denotes the number of arcs whose head (resp. tail) is u. We say that u is *elementary* if $\operatorname{indeg} u = \operatorname{outdeg} u = 1$, and its *suppression* consists in removing it (together with its incident arcs) and connecting its single parent to its single child.

Given a set X of *taxa*, a (rooted binary) *phylogenetic network*, or simply a *network*, on X, is a directed acyclic graph (V, A) without parallel arcs such that any node $u \in V$ is either:

(i) a *root*, with $\operatorname{indeg} u = 0$, $\operatorname{outdeg} u = 1$ (and there can only be one root), or
(ii) a *leaf*, with $\operatorname{indeg} u = 1$, $\operatorname{outdeg} u = 0$, or
(iii) a *tree node*, with $\operatorname{indeg} u = 1$, $\operatorname{outdeg} u = 2$, or
(iv) a *reticulation*, with $\operatorname{indeg} u = 2$, $\operatorname{outdeg} u = 1$,

together with a fixed bijection between X and the set of leaves.

We shall hereafter identify the set X of taxa and the set of leaves, and we shall always assume that X is formed by positive integers, and hence $X \subseteq [n]$ for some n.

Two networks N and N' are *isomorphic*, in symbols $N \cong N'$, if there exists a bijection ϕ between the respective set of nodes that reflects and preserves the arcs (that is, uv is an arc in N if, and only if, $\phi(u)\phi(v)$ is an arc in N'), which is the identity on the leaves (that is, if l is a leaf, $\phi(l) = l$). Hereafter, we shall simply say that two networks are equal if they are isomorphic.

In case that $X = \{l\}$, for some $l \in [n]$, we define the *trivial* network on $\{l\}$, and denote it by I_l, as the network that has two nodes, the root and the leaf l, connected by an arc.

3 Orchard Networks

Let $N = (V, A)$ be a network on $X \subseteq [n]$ and let $(i, j) \in X \times X$ with $i \neq j$. Also, denote by p_i, p_j the parents of the leaves i and j in N, respectively. We call (i, j) a *cherry* if $p_i = p_j$, and we call it a *reticulated-cherry* if p_i is a reticulation, p_j is a tree node, and p_j is one of the parents of p_i. In either case (i, j) is a cherry or a reticulated-cherry, we say that (i, j) is a *reducible pair* in N. In order to identify which kind of reducible pair is (i, j) in N, we will define its *character* as $\chi_N(i, j) = \mathsf{C}$ if it is a cherry and $\chi_N(i, j) = \mathsf{R}$ if it is a reticulated-cherry. Notice that the conditions of being a cherry and a reticulated-cherry are clearly incompatible, which implies that χ_N is well defined. If the network is clear from the context, we will simply write $\chi(i, j)$. To ease notations, if a pair (i, j) has character $\chi = \chi_N(i, j)$, we shall write the *annotated pair* as $(i, j)^\chi$.

Given a network N, we shall denote by $\mathrm{RP}(N)$ the set of reducible pairs of N, by χ_N the mapping $\mathrm{RP}(N) \to \{\mathsf{C}, \mathsf{R}\}$ that gives the character of the reducible pairs, and by $\mathrm{ARP}(N)$ the set of annotated reducible pairs of N.

If $(i, j) \in \mathrm{RP}(N)$, the *reduction* of (i, j) in N, denoted by $N^{(i,j)}$, is the result of:

- If $\chi(i, j) = \mathsf{C}$, then remove the leaf i (and its incoming arc) and suppress p_i, which is now an elementary node.
- If $\chi(i, j) = \mathsf{R}$, then remove the arc $p_j p_i$ and then suppress p_i and p_j, which are now elementary nodes.

Given a sequence of pairs of integers $S = (s_1, \dots, s_k)$ which, for brevity, we will write as $S = s_1 \dots s_k$, with $s_t = (i_t, j_t)$ and $i_t, j_t \in [n]$, of length $k \geq 1$, we say that S is *a reducible pairs sequence* in N if:

- s_1 is reducible in N.
- For every $t \in \{2, \dots, k\}$, s_t is reducible in $(\dots(N^{s_1})^{s_2}\dots)^{s_{t-1}}$.

In such a case, we shall define the *reduction* of N with respect to S as $(\dots(N^{s_1})^{s_2}\dots)^{s_k}$ and it will be denoted by N^S.

Moreover, we say that S is *complete* if $N^S = I_l$ for some $l \in X$ and, in case one such complete sequence exists, we call N an *orchard network* (Erdős et al. 2019; Janssen and Murakami 2021). We shall also consider the trivial networks I_l as orchard networks, corresponding to the case when the sequence S is empty. Notice that trivial networks are the only orchard networks that have a single leaf.

The fundamental result that allows one to classify orchard networks using complete reducible pairs sequences is the following, which is adapted from (Janssen and Murakami 2021, Corollary 1).

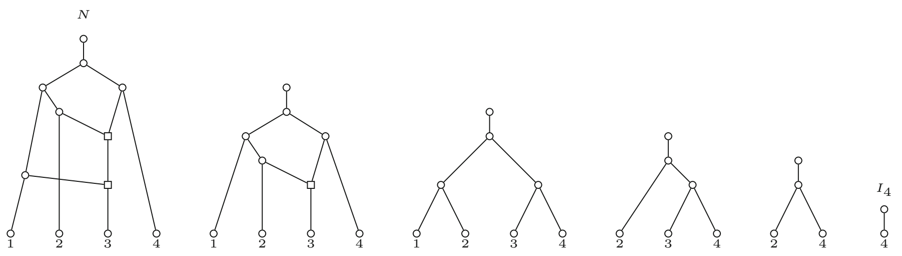

Fig. 1 An orchard network N and the set of intermediate networks obtained by (cherry or reticulated-cherry) reductions until reaching $N^S = I_4$, the reduction of N with respect to the complete sequence $S = (3, 1)(3, 2)(1, 2)(3, 4)(2, 4)$. For instance, the second network is $N^{(3,1)}$, the result of the reduction of (the reticulated-cherry) $(3, 1)$ in N

Theorem 1 *Let S be a complete reducible pairs sequence for two orchard networks N and N'. Then, $N \cong N'$.*

Notice, however, that the complete reducible pairs sequence for an orchard network is not unique. For instance, Fig. 1 shows an orchard network N together with the networks that are obtained by application of the reductions in the sequence $S = (3, 1)(3, 2)(1, 2)(3, 4)(2, 4)$, but it is easy to check that $S' = (3, 1)(3, 4)(2, 3)(1, 3)(3, 4)$ is another complete reducible pairs sequence for N.

4 Minimum Reducible Pairs Sequences

As observed before, there may exist different complete reducible pairs sequences for a given orchard network. Our goal in this section is to define a unique representative among all sequences giving the same network.

Let $(i, j), (i', j')$ be two pairs of different integers. We say that $(i, j) \leq (i', j')$ if $i < i'$ or $i = i'$ and $j \leq j'$. If $(i, j) \leq (i', j')$ and $(i, j) \neq (i', j')$, we simply write $(i, j) < (i', j')$.

Given two sequences of pairs of integers of the same length, $S = s_1 \dots s_k$ and $S' = s'_1 \dots s'_k$, we say that $S < S'$ if, for some $l \in [k]$ we have that $s_1 = s'_1, \dots, s_{l-1} = s'_{l-1}$ and $s_l < s'_l$.

It is easy to check that the relations just defined are total orders (on pairs and sequences of pairs of fixed length, respectively).

Given a non trivial orchard network N, consider the set $\mathrm{RP}(N)$ of reducible pairs of N. We define the *minimum reducible pair* of N, $\mathrm{MRP}(N)$, as the minimum (with respect to the ordering just defined) pair in $\mathrm{RP}(N)$. Also, we denote by $\mathrm{CRS}(N)$ the set of complete reducible pairs sequences of N, and we define the *minimum complete reducible pairs sequence* of N, $\mathrm{MCRS}(N)$, as the minimum (with respect to the ordering just defined) of $\mathrm{CRS}(N)$.

Following the example of the two complete reducible pairs sequences $S = (3, 1)(3, 2)(1, 2)(3, 4)(2, 4)$ and $S' = (3, 1)(3, 4)(2, 3)(1, 3)(3, 4)$ for the orchard network N depicted in Fig. 1, notice that since $(3, 2) < (3, 4)$ then $S < S'$. In fact, it can be checked that $\mathrm{MCRS}(N) = S$.

Notice that all the complete reducible pairs sequences of a given orchard network have the same length, since this length is equal to $|X| + r - 1$, where r is the number of reticulations of N. We show that the two minimums just defined are related.

Proposition 2 *Let N be a non trivial orchard network. Then, the first pair in* $\mathrm{MCRS}(N)$ *is* $\mathrm{MRP}(N)$.

Proof Let s be the first pair in $S = \mathrm{MCRS}(N)$ and $s' = \mathrm{MRP}(N)$. Obviously, $s \in \mathrm{RP}(N)$ and, from the definition of $\mathrm{MRP}(N)$, it follows that $s' \leq s$. Due to (Erdős et al. 2019, Proposition 4.1), the sequence with the single pair s' can be extended to give a complete sequence $S' \in \mathrm{CRS}(N)$. Since the minimum complete sequence is S, we have that $S \leq S'$, and hence $s \leq s'$. Therefore, $s = s'$ and the result is proved. □

We define $\mathcal{S}(X, r)$ as the set whose elements are the sequences $\mathrm{MCRS}(N)$ for every orchard network N over X with exactly r reticulations.

Proposition 3 *There is a bijection between $\mathcal{S}(X, r)$ and the set of orchard networks over X with exactly r reticulations.*

Proof The result follows from Theorem 1 and the unicity of $\mathrm{MCRS}(N)$. □

5 Augmentation of Networks

In this section, we present an augmentation construction, which is the inverse of the reduction defined before, and show how we can determine the ARP of the obtained network from that of the original network.

Throughout this section we consider that N is a network on $X \subseteq [n]$ and $(i, j) \in [n] \times [n]$ is a pair of integers with $i \neq j$ and $j \in X$.

We define the *augmentation* of (i, j) in N, denoted by ${}^{(i,j)}N$, as the result of:

- if $i \notin X$, create a new (leaf) node i, subdivide the arc ending in j creating an elementary node p_j, and add the arc $p_j i$.
- if $i \in X$, subdivide both arcs ending in i and j creating elementary nodes p_i and p_j, and add an arc $p_j p_i$.

Similarly as in the reduction case, we shall define the *augmentation* of an orchard network N (which could be a trivial network I_l) with respect to a sequence $S = s_1 \dots s_k$ as ${}^{s_1}(\dots({}^{s_{k-1}}({}^{s_k}N)))$ and it will be denoted by ${}^{S}N$.

Notice that (i, j) is a cherry in ${}^{(i,j)}N$, in symbols $\chi(i, j) = \mathsf{C}$, when $i \notin X$, and (i, j) is a reticulated cherry in ${}^{(i,j)}N$, in symbols $\chi(i, j) = \mathsf{R}$, when $i \in X$. Then, the augmenting operation *is* the inverse of the reduction operation, in the sense that ${}^{(i,j)}\left(N^{(i,j)}\right) \cong N$. This leads to present an alternative definition for orchard networks as those that can be obtained by an augmentation of a trivial network I_l.

Note also that if $N = {}^{S}I_l$ (for some $l \in [n]$), then necessarily the last pair in S must be (i, l) (for some $i \in [n]$). Hence, l is determined by S and can be omitted from I_l. Therefore, from now on we will simply write $N = {}^{S}I$.

We describe now how one can compute $\mathrm{ARP}({}^{(i,j)}N)$ from $\mathrm{ARP}(N)$. That is, we show how the cherries and reticulated cherries of $N' = {}^{(i,j)}N$ can be found from the knowledge of those of N. Some remarks are due.

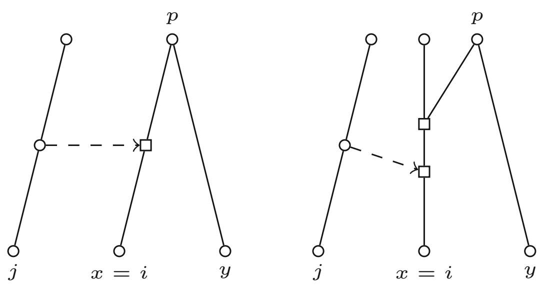

Fig. 2 Support picture for the case $x = i$ and $y \neq j$. In the left, (x, y) is a cherry in N. In the right, (x, y) is a reticulated cherry in N. The dashed arrow indicates the added arc to transform N into N' by the augmentation operation. Arcs whose tips are not explicitly drawn go from top to bottom

1. It is clear that the augmentation is a local operation; more precisely, a cherry (resp. reticulated cherry) in N that is disjoint from (i, j) keeps being a cherry (resp. reticulated cherry) in N'.
2. One only needs to check if the augmentation operation makes that some reducible pair disappears or changes its character (passes from cherry to reticulated cherry or viceversa), and if some new reducible pair appears. As for this last case, notice that the only reducible pair that can appear is (i, j).

Hence, we shall take any pair (x, y) and decide if it is a reducible pair in N' (that is, whether or not $(x, y) \in \mathrm{RP}(N')$) and, in such a case, if either $(x, y)^{\mathsf{C}} \in \mathrm{ARP}(N')$ or $(x, y)^{\mathsf{R}} \in \mathrm{ARP}(N')$ (equivalently, the value of $\chi_{N'}(x, y)$):

- Case $\{x, y\} = \{i, j\}$:
  - Case $i \notin X$: Both (i, j) and (j, i) are cherries in N' and hence $(i, j)^{\mathsf{C}}, (j, i)^{\mathsf{C}} \in \mathrm{ARP}(N')$.
  - Case $i \in X$: Now (i, j) is a reticulated cherry and hence $(i, j)^{\mathsf{R}} \in \mathrm{ARP}(N')$, but $(j, i) \notin \mathrm{RP}(N')$.

Note that from now on we can restrict ourselves to pairs (x, y) in $\mathrm{RP}(N)$, since no other new pairs can appear.

- Case $\{x, y\} \cap \{i, j\} = \emptyset$: From the local character of augmentation, $(x, y) \in \mathrm{RP}(N')$ and $\chi_{N'}(x, y) = \chi_N(x, y)$.
- Case $x = i, y \neq j$ (see Fig. 2): If (x, y) is a cherry in N, say that p is their common parent, then in N' the arc $pi = px$ is split introducing a node which will be a reticulation; hence, (x, y) is a reticulated cherry in N'. If (x, y) is a reticulated cherry in N, then the parent of y will no longer be a grandparent of $x = i$ in N' (since the arc leading to i is split in two). In brief, $(x, y)^{\mathsf{R}} \in \mathrm{ARP}(N')$ if $(x, y)^{\mathsf{C}} \in \mathrm{ARP}(N)$, and $(x, y) \notin \mathrm{RP}(N')$ otherwise.
- Case $x \neq j, y = i$ (see Fig. 3): The same argument as in the previous case gives that if (x, y) is a cherry in N, then (y, x) (notice the transposition) is a reticulated cherry in N' (and hence $(x, y) \notin \mathrm{RP}(N')$). Note that, if (x, y) is a cherry of N, so is (y, x), and hence the fact that $(y, x)^{\mathsf{R}}$ belongs to $\mathrm{ARP}(N')$ will be covered by the application to the previous case applied to (y, x). As before, if (x, y) is a reticulated cherry of N, then it is no longer reducible in N'. Therefore, in either case we have that $(x, y) \notin \mathrm{RP}(N')$.

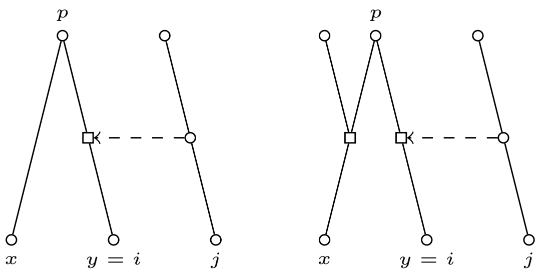

Fig. 3 Support picture for the case $x \neq j$ and $y = i$

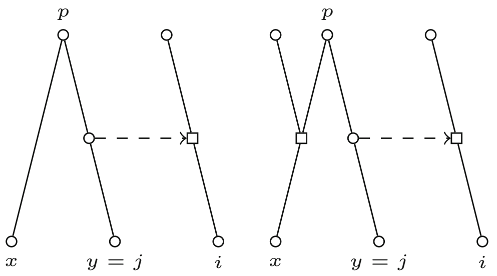

Fig. 4 Support picture for the case $x \neq i$ and $y = j$

- Case $x \neq i, y = j$ (see Fig. 4): Let $py = pj$ be the arc leading to $y = j$; this arc is split in N' by introducing a node that will be a tree node; this implies that (x, y) will no longer be reducible in N' and hence $(x, y) \notin \mathrm{RP}(N')$.
- Case $x = j, y \neq i$: The same argument as in the previous case, taking now the arc leading to $x = j$ implies that $(x, y) \notin \mathrm{RP}(N')$.

We can summarize these computations in the following result.

Theorem 4 *Let N be an orchard network on $X \subseteq [n]$, and $(i, j) \in [n] \times [n]$ a pair with $i \neq j$ and $j \in X$. Consider the set of pairs $(x, y) \in [n] \times [n]$ such that one of the following conditions hold:*

1. *$(x, y) \in \mathrm{RP}(N)$, and $\{x, y\} \cap \{i, j\} = \emptyset$,*
2. *$(x, y) \in \mathrm{RP}(N)$, $i = x$, $j \neq y$, and $\chi(x, y) = C$,*
3. *$(x, y) = (i, j)$,*
4. *$(x, y) = (j, i)$, and $i \notin X$.*

Annotate these pairs with the character $\chi'(x, y)$ given, in each case, by:

1. *$\chi'(x, y) = \chi(x, y)$,*
2. *$\chi'(x, y) = R$,*
3. *$\chi'(x, y) = C$ if $i \notin X$, and $\chi'(x, y) = R$ if $i \in X$,*
4. *$\chi'(x, y) = C$.*

Denote by ${}^{(i,j)}\mathrm{ARP}(N)$ the set of annotated pairs that is obtained by application of the procedure above. Then, ${}^{(i,j)}\mathrm{ARP}(N) = \mathrm{ARP}({}^{(i,j)}N)$.

As a result of the last theorem, in order to compute the set of annotated reducible pairs of the augmentation of a network, it is enough to traverse the annotated reducible pairs of the network.

The next proposition shows that, given $S = \mathrm{MCRS}(N)$, it can be checked if $(i, j)S = \mathrm{MCRS}\left({}^{(i,j)}N\right)$ using only the information in $\mathrm{ARP}(N)$, without the need for knowing N itself.

Proposition 5 *Let* $S = \mathrm{MCRS}(N)$. *Then,* $(i, j)S = \mathrm{MCRS}\left({}^{(i,j)}N\right)$ *if and only if* $(i, j) = \mathrm{MRP}\left({}^{(i,j)}N\right)$.

Proof It is a direct consequence of Proposition 2. □

6 Augmentation Sequences and Generation of Orchard Networks

The goal of this section is to present an algorithm to generate the set of orchard networks over a set $[n]$ with exactly r reticulations. Thanks to Proposition 3, this is equivalent to computing $\mathcal{S}([n], r)$. Our strategy is to build these sequences starting with sequences of length one and, step by step, finding all possible pairs that can be prepended in order to get the sequences in $\mathcal{S}([n], r)$.

Let $S = s_1 \dots s_k$ be a sequence of pairs of integers, say $s_t = (i_t, j_t)$ where $t = 1, \dots, k$. We call the *support* of S the set $\mathrm{Supp}(S) = \{i_1, j_1, \dots, i_k, j_k\}$. For every $t = 1, \dots, k$, we denote by S_t the *suffix* $s_t \dots s_k$. We say that a sequence S as above is an *augmentation sequence* if for each $t = 1, \dots, k-1$, we have that $j_t \in \mathrm{Supp}(S_{t+1})$ and $i_t \neq j_t$. We remark that, although the formulation is not exactly the same, what we call augmentation sequences corresponds to *cherry-picking sequences* in (Janssen and Murakami 2021, Definition 6).

It is clear that given an augmentation sequence S, we can consider the orchard network $N = {}^{S}I$, and also that S will be a complete reducible pairs sequence for N. From now on, all properties that can be defined for networks (taxa, number of reticulations, …) will be defined for augmentation sequences by applying them on the network that the sequence generates. For instance, we can define $\mathrm{MRP}(S) := \mathrm{MRP}({}^{S}I)$ and $\mathrm{MCRS}(S) := \mathrm{MCRS}({}^{S}I)$. Note also that some of the properties can be found without having to construct the network itself. For instance, the number of reticulations of $S = (i_1, j_1) \dots (i_k, j_k)$, which by definition is the number of reticulations of ${}^{S}I$ can be found counting for how many indices $t = 1, \dots, k-1$ we have that $i_t \in \mathrm{Supp}(S_{t+1})$. Also, using Theorem 4 recursively, we can compute $\mathrm{ARP}(S)$.

We shall say that an augmentation sequence S is a *minimum* augmentation sequence if $S = \mathrm{MCRS}(N)$ for some network N. It is clear that it happens exactly when $S = \mathrm{MCRS}(S)$, and recall that $\mathrm{MCRS}(S) = \mathrm{MCRS}({}^{S}I)$. This provides an alternative definition for $\mathcal{S}(X, r)$ as the set of augmentation sequences that are stable under application of MCRS, with support X and with r reticulations.

These notations allow us to translate many properties that have been stated in terms of orchard networks into the language of sequences. For instance, Propositions 2 and 5 can be rewritten as follows.

Proposition 6 *Let S be a minimum augmentation sequence. Then:*

1. *The first pair in S is* $\mathrm{MRP}(S)$.
2. *Given a pair* (i, j) *with* $j \in \mathrm{Supp}(S)$, $(i, j)S$ *is a minimum augmentation sequence if, and only if,* $(i, j) = \mathrm{MRP}((i, j)S)$.

We give now two results that characterize the suffixes of minimum augmentation sequences and, in particular, show that the last pair in such a sequence has a well determined form.

Lemma 7 *Let $S = s_1 \dots s_k$ be a minimum augmentation sequence. Then, every suffix $S_t = s_t \dots s_k$ ($t = 1, \dots, k$) is a minimum augmentation sequence.*

Proof It is clear that $S_t \in \mathrm{CRS}(S_t)$. If there existed some $S'_t \in \mathrm{CRS}(S_t)$ with $S'_t < S_t$, then the concatenation $S' = s_1 \dots s_{t-1} S'_t$ would be strictly smaller than S and also $S' \in \mathrm{CRS}(S)$, against the minimality of S. □

Proposition 8 *Let (i, j) be the last pair in a minimum augmentation sequence $S \in \mathcal{S}([n], r)$. Then, $j = n$.*

Proof Write $\mathrm{MCRS}(N)$ as $S = (i_1, j_1) \dots (i_k, j_k)$, where $(i_k, j_k) = (i, j)$ and assume that $j \neq n$.

Suppose first that $i = n$ and $j < n$. From Lemma 7, (n, j) is a minimum complete reduction sequence (of the cherry ${}^{(n,j)}I$), but in this case (j, n) is also a complete reduction sequence, and $(j, n) < (n, j)$, leading to a contradiction.

Now, we can assume that $i, j < n$. Let $t < k$ be such that (i_t, j_t) is the last pair where one of its entries is n (it exists because $S \in \mathcal{S}([n], r)$, and hence $n \in \mathrm{Supp}(S)$). Now, $S_{t+1} = (i_{t+1}, j_{t+1}) \dots (i_k, j_k)$ is a minimum augmentation sequence thanks to Lemma 7. Since n does not belong to $\mathrm{Supp}(S_{t+1})$, but does belong to the support of $S_t = (i_t, j_t) S_{t+1}$, we have that $i_t = n$ and $j_t < n$. Moreover, (n, j_t) is a cherry in $N' = {}^{S_t}I$. Thanks again to Lemma 7, $S_t = (n, j_t) S_{t+1}$ is a minimum augmentation sequence and, thanks to Proposition 2, $(n, j_t) = \mathrm{MRP}(N')$. However, since (n, j_t) is a cherry of N', then (j_t, n) is also a cherry, and $(j_t, n) < (n, j_t)$, leading to a contradiction. □

For every $m \in \{1, \dots, n-1\}$ we define the set

$$\mathcal{S}_m([n], r) = \{\tilde{S} \mid \tilde{S} \in \mathcal{S}([n], r) \text{ and } \tilde{S} \text{ ends in } (m, n)\}.$$

A direct consequence of Proposition 8 is the following result, that states that the computation of $\mathcal{S}([n], r)$ is reduced to the computation of the subsets $\mathcal{S}_m([n], r)$.

Proposition 9 $\mathcal{S}([n], r) = \bigsqcup_{m=1}^{n-1} \mathcal{S}_m([n], r)$.

We know, from Proposition 8, the form of the last pair in a minimum augmentation sequence. It is also clear that any such pair (m, n) is a minimum augmentation sequence. Our next result shows how minimum augmentation sequences can be extended by prepending pairs of integers in order to generate other minimum augmentation sequences.

Proposition 10 *Let $S' = (i, j)S$ be an augmentation sequence. Then, S' is a minimum augmentation sequence if, and only if, S is a minimum augmentation sequence and $(i, j) = \mathrm{MRP}((i, j)S)$. In such a case, say that $S \in \mathcal{S}(X, r)$ and $S' \in \mathcal{S}(X', r')$. If $i \in \mathrm{Supp}(X)$, then $r' = r + 1$ and $X' = X$; otherwise, $r' = r$ and $X' = X \cup \{i\}$.*

Proof The non-trivial parts of the statement follow from Proposition 6. □

Using these results, is easy to give an algorithm that generates all the orchard networks over a set $[n]$ of taxa and with a given number r of reticulations. Indeed, it is enough to generate, for each positive integer $m < n$, the set $\mathcal{S}_m([n], r)$, and the latter can be generated using Algorithm 1.

Algorithm 1 Computation of $\mathcal{S}_m([n], r)$

```
1: Start with the sequence $S = (m, n)$, of length 1, with support $X = \{m, n\}$, and whose set of annotated
   reducible pairs is ARP = ARP$(S) = \{(m, n)^{\mathsf{C}}, (n, m)^{\mathsf{C}}\}$.
2: Recursively, given a sequence of pairs $S$, with support $X$, and given also the set ARP = ARP$(S)$, find
   all possible pairs $(i, j)$ such that $(i, j) = \mathrm{MRP}((i, j)S)$. For each such $(i, j)$, consider the extended
   sequence $S' = (i, j)S$ with support $X' = S \cup \{i\}$ and with set of annotated reducible pairs ARP$' = {}^{s}$ARP.
3: If $X = [n]$ and the length of the obtained sequence is $|S| = n + r - 1$, then yield the sequence $S$.
```

Theorem 11 *The set of sequences yielded by the Algorithm 1 is $\mathcal{S}([n], r)$.*

Proof Let S be a sequence yielded by the algorithm. The condition that S has support $[n]$ and has r reticulations is guaranteed by the condition in step 3 of the algorithm. The condition that S is a minimum augmentation sequence follows by applying recursively Proposition 10, thanks to the condition in step 2, and with the starting condition in step 1 being justified by Proposition 9.

Conversely, if $S = s_1 \dots s_k \in \mathcal{S}([n], r)$, then $s_k = (m, n)$ for some m (thanks to Proposition 9), and it will be considered in step 1. At each step, considering the suffix $S_t = s_t \cdots s_k$ in step 2, the pair s_{t-1} will fulfill the conditions (thanks to Proposition 10), and hence $S_{t-1} = s_{t-1}s_t \dots s_k$ will be considered in the next iteration. Finally, in step 3, the sequence S will be yielded. □

Some remarks are due:

1. The set ^{s}ARP in step 2 can be computed using Theorem 4, and it can be done in linear time with respect to the length of S. Also, if the pairs in ARP are stored increasingly ordered with respect to the lexicographic ordering, then the computation of ^{s}ARP can be performed so that ^{s}ARP keeps being ordered and, in particular, its minimum element can be found in constant time.
2. Another advantage of storing the pairs in ARP ordered is that, in order to determine if $(i, j) = \mathrm{MRP}((i, j)S)$, one does not need to compute the whole set ${}^{(i,j)}$ARP. Indeed, in the process of building ${}^{(i,j)}$ARP, at most three pairs in ARP can disappear, and hence one only needs to take the first four elements in ARP, decide which of them belong to ${}^{(i,j)}$ARP, and test if (i, j) is smaller than each of those.
3. Given a minimum augmentation sequence S, it is possible that it can not be extended to another minimum augmentation sequence $(i, j)S$ if we want to keep the number of reticulations. For instance, if we consider the sequence $S = (1, 2)(2, 4) \in \mathcal{S}_2(\{1, 2, 4\}, 0)$, the only possible extensions that keep the number of reticulations are obtained by prepending one of the pairs $(3, 1)$, $(3, 2)$ or $(3, 4)$; however, none of these sequences is minimum, as can be easily checked in each case.

4. The search of extensions can be pruned. For instance, if at a given stage, the sequence S has r reticulations, the only pairs (i, j) that have to be considered are those with $i \notin \mathrm{Supp}(S)$, since otherwise the number of reticulations would be greater than r.
5. Also in the case that we are adding a cherry (that is, when $i \notin \mathrm{Supp}(S)$) we can restrict ourselves to the case that $i < j$, since otherwise (j, i) would be a reducible pair in ${}^{(i,j)S}I$, and since $(j, i) < (i, j)$, it is impossible that $(i, j) = \mathrm{MRP}((i, j)S)$.
6. The algorithm can be easily modified, so that instead of generating all the sequences with exactly r reticulations, it generates all sequences with *at most* r reticulations.

7 Generation of Tree-Child Networks

A network is *tree-child* if every node that is not a leaf has a child that is a tree node (Cardona et al. 2009). For brevity, we shall simply say that each interior node has a *tree child*.

The same algorithm we have described to generate orchard networks can be adapted to generate all tree-child networks over $[n]$, by adding some conditions to ensure that the generated sequences correspond to tree-child networks.

First, we need to decide when the reductions and augmentations defined in the previous sections produce tree-child networks.

We start with the following result, adapted from (Bordewich and Semple 2016, Lemma 4.1), that states that reductions of tree-child networks are tree-child networks.

Lemma 12 *Let N be a tree-child network. Then, N is an orchard network and, if $(i, j) \in \mathrm{RP}(N)$, then $N^{(i,j)}$ is also a tree-child network.*

In order to decide whether or not an augmentation of a tree-child network is tree-child, we need to introduce new terminology. Let N be a network over $X \subseteq [n]$. Then, we define the *state* $\sigma_N(i)$ of $i \in [n]$ as follows:

- if $i \notin X$, then $\sigma_N(i) = \mathsf{N}$;
- otherwise, if the parent of i is a reticulation, then $\sigma_N(i) = \mathsf{P}$;
- otherwise, if the sibling of i is a reticulation, then $\sigma_N(i) = \mathsf{S}$;
- otherwise, $\sigma_N(i) = \mathsf{T}$.

If the network is clear from the context we shall simply write $\sigma(i)$ instead of $\sigma_N(i)$. Then, $\sigma_N : [n] \to \{\mathsf{N}, \mathsf{P}, \mathsf{S}, \mathsf{T}\}$ is a mapping that gives the state of each $i \in [n]$ in N. We also define the *state of a network* N as $\sigma(N) = (\sigma_N(1), \ldots, \sigma_N(n))$. Finally, if S is an augmentation sequence, we shall denote $\sigma(S) = \sigma({}^{S}I)$.

The following result gives the conditions under which an augmentation produces a tree-child network.

Theorem 13 *Let N be an orchard network. Then, ${}^{(i,j)}N$ is a tree-child network if, and only if, N is tree-child and $\sigma_N(i) \in \{\mathsf{N}, \mathsf{T}\}$.*

Proof Let $N' = {}^{(i,j)}N$. From Lemma 12 we know that if N' is tree-child, N is also tree-child.

Now, suppose that $\sigma_N(i) = \mathsf{P}$. Then, i is a leaf in N and its parent p_i is a reticulation. When applying the augmentation (i, j), the arc $p_i i$ is split introducing a new node v that shall become a reticulation. Then, in N', the only child of p_i is v, which is a reticulation. Therefore, N' is not tree-child.

Similarly, suppose that $\sigma_N(i) = \mathsf{S}$. Then, i is a leaf in N, its parent p_i is a tree node and its sibling s_i is a reticulation. Again, in the process of applying the augmentation (i, j), the arc $p_i i$ is subdivided introducing a new reticulation v. Thus, the children of p_i in N' are v and s_i, both reticulations, so N' is not tree-child, against the hypothesis. Therefore, $\sigma_N(i) \notin \{\mathsf{P}, \mathsf{S}\}$, which is equivalent to $\sigma_N(i) \in \{\mathsf{N}, \mathsf{T}\}$.

Conversely, assume that N is tree-child. Due to the local nature of the augmentation processes, the condition that each node (other than a leaf) in N' has a tree child needs only to be tested for the nodes that are adjacent to the leaves involved in the augmentation.

First, assume that $\sigma_N(i) = \mathsf{N}$, and let p_j be the parent of j in N. The augmentation process creates a tree node v in N' with children i, j and parent p_j. Now, p_j keeps having a tree child (the node v), and the new internal node v has both children that are tree nodes (the leaves i and j). Hence, the condition of being tree-child is preserved.

Second, assume that $\sigma_N(i) = \mathsf{T}$, which implies that $i \in X$ and hence the augmentation process creates two elementary nodes: u (a tree node) between j and its parent p_j, and v (a reticulation) between i and its parent p_i. Also, since $\sigma_N(i) = \mathsf{T}$, we have that the sibling s_i of i in N (that is, the child of p_i in N different from i) is a tree node. In N', p_j has u as a tree child, p_i has s_i, u has j, and v has i. Hence, the condition of being tree-child is preserved. □

We describe now how to compute $\sigma({}^{(i,j)}N)$ from $\sigma(N)$. For simplicity, we write $N' = {}^{(i,j)}N$, $\sigma = \sigma_N$ and $\sigma' = \sigma_{N'}$, and we will restrict to the cases of interest that $\sigma(i) \in \{\mathsf{N}, \mathsf{T}\}$.

- Case $\sigma(i) = \mathsf{N}$. In this case, (i, j) is a cherry in N', and therefore $\sigma'(i) = \sigma'(j) = \mathsf{T}$. From the local behavior of the augmentation, for any other leaf l in N, its parent (and its sibling, in case it has one) remains the same. Therefore, we conclude that $\sigma'(i) = \sigma'(j) = \mathsf{T}$ and $\sigma'(l) = \sigma(l)$ for all $l \in [n]\setminus\{i, j\}$.
- Case $\sigma(i) = \mathsf{T}$. In this case, (i, j) is a reticulated cherry in N', hence $\sigma'(i) = \mathsf{P}$ and $\sigma'(j) = \mathsf{S}$. Thanks again to the local behaviour of the augmentation, the state of a leaf l in N' can only differ from its state in N if its parent or sibling change from being a tree node to a reticulation (or viceversa). Hence, only siblings of i and j have to be taken into consideration. If j was the sibling of another leaf l in N, then l would still have a sibling in N' that is a tree node (namely, the parent of j in N') and hence the state of l would not change. If i was the sibling of another leaf l in N, which can be written as $(i, l)^{\mathsf{C}} \in \mathrm{ARP}(N)$, then l would change from having a sibling that is a tree node (the leaf i) to having a sibling that is a reticulation (the parent of i in N'). Hence $\sigma(l) = \mathsf{T}$ but $\sigma'(l) = \mathsf{S}$.

We can summarize these computations in the following result.

Theorem 14 *Let N be a tree-child network over $X \subseteq [n]$ with state function σ. Let $j \in X$ and $i \in [n]$, $i \neq j$, with $\sigma(i) \in \{N, T\}$. Consider the function ${}^{(i,j)}\sigma : [n] \to \{N, P, S, T\}$ defined as follows:*

- *If* $\sigma(i) = \mathsf{N}$,
  - $^{(i,j)}\sigma(i) = {}^{(i,j)}\sigma(j) = \mathsf{T}$,
  - $^{(i,j)}\sigma(l) = \sigma(l)$ *for all* $l \in [n]\setminus\{i, j\}$.
- *If* $\sigma(i) = \mathsf{T}$,
  - $^{(i,j)}\sigma(i) = \mathsf{P}$,
  - $^{(i,j)}\sigma(j) = \mathsf{S}$,
  - $^{(i,j)}\sigma(l) = \begin{cases} \mathsf{S}, & \textit{if } (i,l)^{\mathsf{C}} \in \mathrm{ARP}(N), \\ \sigma(l), & \textit{otherwise} \end{cases}$ *for all* $l \in [n] \setminus \{i, j\}$.

Then $^{(i,j)}N$ *is a tree-child network over* $X \cup \{i\}$ *with state function* $^{(i,j)}\sigma$.

We denote by $\mathcal{T}(X, r)$ the subset of $\mathcal{S}(X, r)$ formed by sequences S such that ^{S}I is a tree-child network. Thanks to Lemma 12, the set $\mathcal{T}(X, r)$ is in bijection with the set of tree-child networks over the set of taxa X and with r reticulations. Also, notice that $\mathcal{T}([n], r) = \bigsqcup_{m=1}^{n-1} \mathcal{T}_m([n], r)$, where $\mathcal{T}_m([n], r)$ denotes, for every $m \in \{1, \dots, n-1\}$, the subset of $\mathcal{S}_m([n], r)$ formed by sequences S such that ^{S}I is tree-child. Finally, notice that in the case of tree-child networks, the number of reticulations r is bounded by $n - 1$ (Cardona et al. 2009, Proposition 1).

Then, we can modify the algorithm that generates all orchard networks over $[n]$ with r reticulations to generate all tree-child networks over $[n]$ with r reticulations, provided that $r < n$.

Indeed, for every positive integer $m < n$, we can generate the sets $\mathcal{T}_m([n], r)$ using Algorithm 2.

Algorithm 2 Computation of $\mathcal{T}_m([n], r)$

```
1: Start with the sequence $S = (m, n)$, of length 1, with support $X = \{m, n\}$, with set of annotated reducible
   pairs ARP = ARP($S$) = $\{(m, n)^{\mathsf{C}}, (n, m)^{\mathsf{C}}\}$ and with state $\sigma = \sigma(S)$ whose entries are all N except the
   $m$-th and $n$-th entry which are T.
2: Recursively, given a sequence of pairs $S$ (and assuming that $^{S}I$ is tree-child), with support $X$, and given
   also the set ARP = ARP($S$) and the state $\sigma = \sigma(S)$, find all possible pairs $(i, j)$ such that $\sigma(i) \in \{\mathsf{N}, \mathsf{T}\}$
   and $(i, j) = \mathrm{MRP}((i, j)S)$. For each such $(i, j)$, consider the extended sequence $S' = (i, j)S$ with
   support $X' = S \cup \{i\}$, with set of annotated reducible pairs $\mathrm{ARP}' = {}^{s}\mathrm{ARP}$ and with state $\sigma' = {}^{s}\sigma$.
3: If $X = [n]$ and the length of the obtained sequence is $|S| = n + r - 1$, then yield the sequence $S$.
```

Theorem 15 *The set of sequences yielded by Algorithm 2 is* $\mathcal{T}([n], r)$.

Proof The result follows using the same reasoning as in Theorem 11, using now Theorem 13 to ensure that the yielded networks are tree-child. □

Some remarks follow:

1. The state function σ' in step 2. can be computed using Theorem 14, and notice that the information in ARP is also needed.
2. As in the case of orchard networks, the algorithm can be adapted to yield all tree-child networks over $[n]$ with at most r reticulations. In particular, since tree-child networks over $[n]$ have at most $n - 1$ reticulations (Cardona et al. 2009, Proposition 1), we can generate all of them.

3. The algorithm given for generating tree-child networks can be adapted to generate all stack-free (Semple and Simpson 2018) orchard networks, simply checking if $\sigma(i) \neq \mathsf{P}$ instead of checking if $\sigma(i) \in \{\mathsf{N}, \mathsf{T}\}$.

8 Computational Experiments

The algorithms to generate orchard and tree-child networks described in this paper have been implemented in C. Source files, documentation and examples are available in the repository https://github.com/gerardet46/OrchardGenerator. Notice that the output of the implementation are complete reducible pairs sequences, given as strings, and that they can be used as input to build and manipulate networks using the Python package PhyloNetworks (Cardona 2023).

There are some interesting details to comment. First, as we said, the set ARP is kept ordered, and the cherries (i, j) with $i > j$ are ignored. Taking this into account, notice that if N is an orchard network on $X \subseteq [n]$, it holds that $|\operatorname{ARP}(N)| \leq \frac{2}{3}|X| \leq \frac{2}{3}n$ (and there is always an orchard network such that the equality holds). Therefore, the set ARP can be implemented as a static array, which is much faster than a dynamic one. Also, given a sequence $S = s_1 \dots s_k$, we store the set $\{\operatorname{ARP}(s_k), \operatorname{ARP}(s_{k-1}s_k), \dots, \operatorname{ARP}(S)\}$ for faster access when trying different candidate extensions.

Notice also that the only data needed to store the networks is X, S and ARP (and σ for tree-child networks), but there is no need to store the network N itself. Also, the operations involved in the algorithm are very simple, so they could be easily implemented in C, optimizing the performance.

We have also implemented a random orchard network generator that follows the same lines of the algorithm to generate all the networks, but choosing a random pair at each step in order to produce a sequence, instead of trying all the candidates. Notice however that this generator does not generate networks uniformly. Indeed, even at the first step, the number of MCRS ending in $(n-1, n)$ is greater than the number of those ending in $(1, n)$.

Finally, the algorithm can be parallelized, considering a partition of suffixes and creating a process for each subset of suffixes, which generates all sequences ending in a suffix from the corresponding subset.

Using this implementation, we have computed the number of orchard networks for small number of leaves and reticulations, shown in Table 1. As for the generation of tree-child networks, it is worth to mention the speed of the computation compared to previously implemented methods. Indeed, Table 2 shows the time of execution for the generation of all tree-child networks with $n = 5$ leaves using the implementations of the results in this paper compared to those in (Cardona et al. 2019) and (Cardona and Zhang 2020).

Table 1 Number of orchard networks with n leaves and r reticulations, for $2 \le n \le 6$ and $0 \le r \le 8$, together with the total time used to compute these numbers, for each value of n

$r\backslash n$	2	3	4	5	6
0	1	3	15	105	945
1	2	21	228	2 805	39 330
2	4	132	2 832	57 150	1 185 300
3	8	804	32 880	1 054 200	31 481 280
4	16	4 848	370 320	18 520 320	783 492 840
5	32	29 136	4 107 648	316 583 280	18 766 151 280
6	64	174 912	45 197 952	5 323 207 200	438 647 126 400
7	128	1 049 664	495 183 360	88 589 126 400	10 087 314 094 080
8	256	6 298 368	5 412 422 400	1 464 596 709 120	229 383 137 571 840
Time	0.00s	0.02s	5.99s	1 693.39s	470 828.27s

Table 2 Time needed for the generation of all tree-child networks with $n = 5$ leaves using different implementations

Implementation in Python from Cardona and Zhang (2020)	9m19.249s
Implementation in Python from Cardona et al. (2019)	7m23.162s
Implementation in C of the current paper	0m00.056s

9 Conclusions

Phylogenetic networks model evolutionary relationships among organisms and overcome the limitations of using phylogenetic trees by allowing the representation of reticulate processes.

In this paper, we have considered the problem of the efficient and injective generation of all orchard and tree-child networks (with a given number of leaves and reticulations), two special classes of phylogenetic networks with biological relevance (Kong et al. 2022). Our method is based on considering sequences of pairs of integers that characterize those networks (Janssen and Murakami 2021) and finding a subset of those (called minimum complete reducible pairs sequences) that characterize the networks injectively.

To this end, we have first shown that such a sequence must end in a pair (m, n), where n is the desired number of leaves and $m < n$, and that we can iteratively extend the sequences by prepending new pairs to generate the sequences that encode the networks. This method is efficient since there is no need to construct the network itself in order to check if the candidate sequence effectively corresponds to an orchard (or tree-child) network.

The implementation of the algorithms described in the paper allows a fast generation of the sequences (and implicitly of the networks). For example, our implementation is capable of generating all the sequences corresponding to orchard networks with

4 leaves and at most 8 reticulations, of which there are about 6 billions of them, in approximately 6 s.[1] For tree-child networks, we have shown that our method is much faster than other methods previously published and implemented.

There are some natural questions that arise as a possible future work, mainly in the direction of extending our results to the generation of other classes of phylogenetic networks. One of the possible generalizations is removing the binary condition and generating semi-binary and non-binary (orchard and tree-child) networks. In this sense, the results in (Janssen and Murakami 2021) could be applied, considering more possible annotations of pairs, in order to cover the six different reductions that this paper considers. Another direction could be trying to use other topological conditions on the networks to be generated. For instance, and as we have commented at the end of Sect. 7, only a small change in our method is needed in order to generate stack-free orchard networks. Another potential subclass of networks where our methods could apply is the class of normal networks (Willson 2010), which is a subclass of tree-child networks where shortcuts are not allowed (that is, if two nodes are linked by an arc, then they cannot be connected by another path).

Acknowledgements GC and JCP were supported by Grant PID2021-126114NB-C44 funded by MCIN/AEI/ 10.13039/501100011033 and by "ERDF A way of making Europe".

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

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

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Aldous D (1996) Probability distributions on cladograms. In: Random discrete structures. Springer, pp 1–18. https://doi.org/10.1007/978-1-4612-0719-1_1

Bordewich M, Semple C (2016) Determining phylogenetic networks from inter-taxa distances. J Math Biol 73(2):283–303. https://doi.org/10.1007/s00285-015-0950-8

Cardona G (2023) PhyloNetwork, v.2.2. https://github.com/bielcardona/PhyloNetwork

Cardona G, Zhang L (2020) Counting and enumerating tree-child networks and their subclasses. J Comput Syst Sci 114:84–104. https://doi.org/10.1016/j.jcss.2020.06.001

Cardona G, Rossello F, Valiente G (2009) Comparison of tree-child phylogenetic networks. IEEE/ACM Trans Comput Biol Bioinf 6:552–569. https://doi.org/10.1109/TCBB.2007.70270

Cardona G, Pons JC, Scornavacca C (2019) Generation of binary tree-child phylogenetic networks. PLoS Comput Biol 15(9):e1007347. https://doi.org/10.1371/journal.pcbi.1007347

Erdős PL, Semple C, Steel M (2019) A class of phylogenetic networks reconstructable from ancestral profiles. Math Biosci 313:33–40. https://doi.org/10.1016/j.mbs.2019.04.009

[1] Computation performed on a computer with two processors Intel® Xeon® E5-2690 (3.00GHz), providing 40 CPUs.

Fuchs M, Gittenberger B, Mansouri M (2021) Counting phylogenetic networks with few reticulation vertices: exact enumeration and corrections. Australas J Comb 81(2):257–282

Fuchs M, Yu GR, Zhang L (2021) On the asymptotic growth of the number of tree-child networks. Eur J Comb 93:103278. https://doi.org/10.1016/j.ejc.2020.103278

van Iersel L, Janssen R, Jones M et al (2022) Orchard networks are trees with additional horizontal arcs. Bull Math Biol 84(8):76. https://doi.org/10.1007/s11538-022-01037-z

Janssen R, Liu P (2021) Comparing the topology of phylogenetic network generators. J Bioinform Comput Biol 19(06):2140012. https://doi.org/10.1142/S0219720021400126

Janssen R, Murakami Y (2021) On cherry-picking and network containment. Theor Comput Sci 856:121–150. https://doi.org/10.1016/j.tcs.2020.12.031

Kong S, Pons JC, Kubatko L et al (2022) Classes of explicit phylogenetic networks and their biological and mathematical significance. J Math Biol 84(6):47. https://doi.org/10.1007/s00285-022-01746-y

Pons JC, Scornavacca C, Cardona G (2019) Generation of level-*k* lgt networks. IEEE/ACM Trans Comput Biol Bioinf 17(1):158–164. https://doi.org/10.1109/TCBB.2019.2895344

Pons M, Batle J (2021) Combinatorial characterization of a certain class of words and a conjectured connection with general subclasses of phylogenetic tree-child networks. Sci Rep 11(1):21875. https://doi.org/10.1038/s41598-021-01166-w

Semple C, Simpson J (2018) When is a phylogenetic network simply an amalgamation of two trees? Bull Math Biol 80(9):2338–2348. https://doi.org/10.1007/s11538-018-0463-x

Steel M (2016) Phylogeny: discrete and random processes in evolution. SIAM, New Delhi

Willson SJ (2010) Properties of normal phylogenetic networks. Bull Math Biol 72:340–358. https://doi.org/10.1007/s11538-009-9449-z

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.