Making phylogenetic networks orchard

Algorithms to determine if a phylogenetic network is orchard and to transform non-orchard to orchard networks

Bachelor End Project by
Merel Susanna

Delft University of Technology

TU Delft

Making phylogenetic networks orchard

Algorithms to determine if a phylogenetic network is orchard and to transform non-orchard to orchard networks

by

Merel Susanna

Student Name	Student Number
Merel Susanna	4907299

Supervisors:	Dr. ir. L.J.J. van Iersel & Dr. Y. Murakami
Thesis committee:	Dr. ir. L.J.J. van Iersel
	Dr. Y. Murakami
	Dr. J.W. van der Woude
Institution:	Delft University of Technology
Project Duration:	April, 2022 - June, 2022
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

TU Delft

Preface

While making the decision about the subject of my bachelor end project, I came across a project about phylogenetic networks. I never heard of phylogenetic networks, however the fact that is was an application of mathematics in the direction of discrete mathematics and optimization really interested me. It was great to see the application of the information obtained in the bachelor program of Applied Mathematics. The way I got to write algorithms and prove certain new theorems was great to see all the theoretical information applied in a project.

I would really like to thank my supervisors Leo van Iersel and Yuki Murakami for guiding me through the process of research. It was the first experience with doing research and proving new statements on my own, and they were of great help for doing this task in a good way. During the time I worked on this project, the weekly meetings with my supervisors helped to stay on track. The feedback I received during this project was really helpful for continuing and finishing this project.

Merel Susanna
Delft, July 2022

Abstract

Phylogenetic networks are used to represent evolutionary histories of a set of taxa. In this thesis, we look at a certain network class, called orchard networks. In the beginning of this thesis, definitions concerning phylogenetic networks and specifically orchard networks are introduced. This happens in Chapter 2. The characterization of orchard networks involves time-labelling of the vertices.
In Chapter 3 an algorithm is given to see if a given network is orchard. The last section of Chapter 3, explores a non-recursive labelling of a given network. How to label a network is given in a different paper [5], however there is not an explicit algorithm for this. An algorithm for the labelling is given in Section 3.3. We give the pseudo-code for finding such labels.
Chapter 4 is about non-orchard networks. It contains multiple actions that can be performed on the non-orchard networks in order to transform the non-orchard networks into orchard networks.

Contents

1 Introduction

At first sight, viruses seem to occur out of nowhere. One day people live their lives as they know it; in the next, a new virus hits and creates a pandemic. A great example of such events, was caused recently by the coronavirus (COVID). However, COVID was not 'born' out of nowhere. Viruses that we know can mutate into new viruses. Even if some small part of the RNA of a virus is different, a new virus can occur with whole new or different symptoms. This is an interesting process and can be represented by mathematical graphs [8].

The evolution from ape to human, a family tree or the evolution of viruses as described above are all examples of processes that can be visualized with phylogenetic networks. Networks are even capable of dealing with complex evolutionary scenarios, such as genetic recombination, hybridization and horizontal gene transfer. Vertices in a phylogenetic network represent evolutionary events, but they can also represent ancestral species. Besides the actions of splitting and combining properties, the information on the length between evolutionary scenarios can also be stored in phylogenetic networks, by means of edge lengths.

In this thesis, some properties of these kind of networks are studied. Before doing this, useful definitions are introduced in Chapter 2. In Chapter 3, we introduce orchard networks, which are networks that can be reduced to the common ancestor of all taxa in this network, by means of cherry reduction (explained is Chapter 2). Orchard network were first introduced for the computational benefits, however in a recent paper a biological motivation is found. This motivation has everything to do with vertex labelling [5]. Algorithms are stated in order to see if a given network is orchard and if so, a labeling can be applied that represents the length of an evolution. These algorithms fill a gap in the literature about vertex labelling. In the last chapter, networks that are not orchard are considered. Using some actions, like adding taxa or deleting some arcs of the network, the network can be transformed into an orchard network. The connection between the amount of taxa that are known and the amount of deletions that needs to happen is investigated.

2

Definitions

All the figures included in this thesis are created using Graph Online [2].

Definition 2.1 (Directed phylogenetic network) A directed binary phylogenetic network N is a directed graph on a set of taxa X containing the following categories of vertices and properties:
Categories:

- A root: a vertex with indegree-0 and outdegree-1.
- A tree vertex: a vertex with indegree-1 and outdegree-2.
- A reticulation: a vertex with indegree-2 and outdegree-1.
- A leaf: a vertex with indegree-1 and outdegree-0.

Properties:

- N can not contain any cycles.
- N can not contain nodes with indegree-1 and outdegree-1.
- N contains one root.
- All the leaves of N are labelled with an element of X. A node that is not a leaf/root is called a internal node.
- Every element of X is exactly one leaf.

From now on only binary phylogenetic networks are considered. This means that a vertex can only have maximal degree-3. [3] [9]

Definition 2.2 (Phylogenetic Tree) A phylogenetic network without reticulations is a phylogenetic tree. [3]

Definition 2.3 ((Reticulated) cherry) Let x and y be two leaves in N. If x and y have a parent in common (x, y) is called a cherry.
Let p_x denote a parent of x. If p_x is a reticulation and p_x and y have a parent in common, then (x, y) is a reticulated cherry and the incoming arcs of a reticulation are reticulation arcs. A cherry or a reticulated cherry is also called a reducible pair. [5]

A tree edge is an edge in a network such that it is not a reticulation edge.

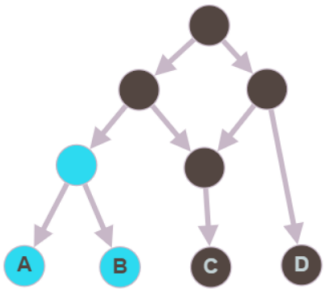

Figure 2.1: (A,B) is a cherry

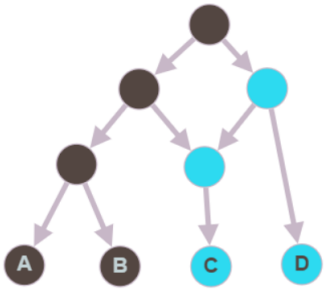

Figure 2.2: (C,D) is a reticulated cherry

Definition 2.4 (Cherry picking/reducing) Let (x, y) be a pair of leaves, p_x and p_y be the parents of x and y respectively. Cherry picking or reducing is an action of deleting nodes/arcs and in a network N using the following method:

- (x, y) is a cherry: delete node x and if p_x is a node with one incoming arc and one outgoing arc, then suppress p_x. Suppressing a node is the action of deleting the node and adding an arc from its in-neighbour to its out-neighbour.
- (x, y) is a reticulated cherry: delete the arc between p_y and p_x and suppress the nodes with one incoming- and one outgoing arc.
- (x, y) is not a cherry/reticulated cherry: do not do anything.

[5]

Definition 2.5 (Orchard network) A network N is called orchard if N can be reduced, according to a sequence of ordered pairs S, to a tree with one leaf. This sequence of ordered pairs denote the order in which the pair is picked and is called a cherry picking sequence. NS represents the network as a result of applying S to the network N. Therefore, N is orchard if there exists a sequence S such that NS is a tree with one leaf.

An property of networks is that they can be isomorphic. Two networks are isomorphic if there exists a bijective function f between the nodes such that there is an arc between node x and y if and only if there is an arc between $f(x)$ and $f(y)$. [5]

Example 2.1 (Cherry picking) Network N is an orchard network (the leftmost network in figure 2.3). This can be verified using cherry picking. First cherry (B, A) is reduced and so leaf B is deleted. Secondly reticulated cherry (C, A) is reduced by deleting the arc between the parent of A and the parent of C and suppress the nodes with one incoming and one outgoing arc. Then cherry (C, D) is reduced, followed by cherry (D, A). This process results in a tree with one leaf, A. Therefore N is an orchard network. This process is visualised in figure 2.3.

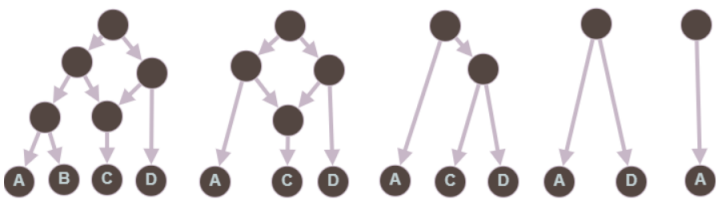

Figure 2.3: N is orchard and can be reduced by sequence S=(B,A)(C,A)(C,D)(D,A)

From now on, a binary phylogenetic network is denoted by binary network or just network.

3

Is a given network orchard?

In this chapter, an algorithm is constructed in order to check whether a network N is orchard or not. Before doing this, the knowledge of some theorems is necessary. These are stated in the beginning of this chapter.

3.1. Useful theorems

In order to get the following theorem, Theorem 3.1, some notation should be explained. CPN stands for cherry-picking network, which is the same as saying that the network is orchard. This is a network that can be reduced by some sequence consisting of pairs of leaves of the network that can be picked. Such sequences, which we call cherry-picking sequences, will be denoted by CPS in the next theorem. A partial CPS is a sequence of leaf pairs. It is a partial since it may not reduce some CPN.

Theorem 3.1 *Let N be a binary CPN, and S a partial CPS. If in each step of the reduction of N by S, the network is changed, then there exists a minimal CPS S' starting with S that reduces N.*

Theorem 3.1 says that the order in which cherries are picked does not matter. [6]

Theorem 3.2 *Let N be a phylogenetic network on a set of taxa X.Then N is orchard if and only if $N(x, y)$ is orchard for all $(x, y) \in X^2$.*

Proof. Let $(x, y) \in X^2$ and suppose that $N(x, y)$ is an orchard network. By the definition of orchard networks, there exists a sequence S consisting of pairs of leaves such that a pair in S forms a cherry or a reticulated cherry in $N(x, y)$. Therefore $N(x, y)S$ is a tree with one leaf. Since $N(x, y)$ is an orchard network, there are two possible cases:

- (x, y) is a cherry or a reticulated cherry. In this case, the pair (x, y) can first be added to $N(x, y)$ to get the network N. As a result of this, N can be reduced by the sequence $(x, y)S$ and therefore N is orchard.
- (x, y) is not a cherry nor a reticulated cherry. Then $N(x, y) = N$ and since S exists by assumption, S also reduces N, so N is orchard.

For proving the statement in the other direction, assume N is orchard and let $(x, y) \in X^2$. (x, y) can be a cherry or reticulated cherry, but it can also be neither. Therefore, two cases need to be considered.

- (x, y) is not a cherry nor a reticulated cherry. In this case N is equal to $N(x, y)$ since (x, y) does not have any influence on the network N. Since N is orchard, it follows automatically that $N(x, y) = N$ is also orchard.
- (x, y) is a cherry or a reticulated cherry. Since N is orchard, it can be reduced to a tree with one leaf by a sequence S. The order of picking the cherries does not matter (Theorem 3.1) and therefore it can be chosen that the sequence S starts with the pair (x, y), which can be written as $S = (x, y)A$ for some partial $CPSA$. So NS = $N(x, y)A$. As a consequence of this, it can be seen that A reduces $N(x, y)$ and from that it can be concluded that $N(x, y)$ is orchard.

Therefore, $N(x, y)$ is orchard in both cases. □

Theorem 3.3 *If N is orchard and contains at least two leaves, then N contains a cherry or a reticulated cherry.*

Proof. Let N be an orchard network. By definition, there exists a sequence S such that NS is a tree with one leaf. Suppose that every element in S reduces N in some way. Then the length of S is minimal. In that case, the first pair in this sequence should be a cherry or a reticulated cherry in N. It follows that each orchard network contains a cherry or a reticulated cherry. □

The trivial case is an exception to Theorem 3.3. The trivial case is the tree with one leaf. Such a network does not contain a cherry or a reticulated cherry, however we define a tree with one leaf to be an orchard network. The empty sequence S trivially reduces the network to a tree with one leaf.

3.2. Algorithm to check if a network is orchard

Given a network N, it is important to check whether this network is orchard or not, since an orchard network has multiple mathematical properties.

Let N be a given network.

Firstly, N could be the trivial case. This means that N is a tree with one leaf, and is therefore orchard.

Otherwise, the second step is to check whether the given network N contains a cherry or a reticulated cherry. This can be done by looking at the parents of the leaf nodes. Let (x, y) be an arbitrary ordered pair of two distinct leaves of N. The pair (x, y) is a cherry if leaf x and leaf y have a parent in common. If this is the case, then by definition, N contains a cherry.
In case of a reticulated cherry, it should be that the parent p_x of x is a reticulation and that this parent and p_y are connected, such that p_y is a parent of p_x. Checking this will verify whether there exists a reticulated cherry or not.

If it is verified that there does not exist a cherry nor a reticulated cherry, it follows from Theorem 3.3 that N is not an orchard network. Otherwise, there exists a cherry or a reticulated cherry. As a result of this, the existing cherry or reticulated cherry can be picked as described in Definition 2.3.

Now we have a network N with (x, y) picked. In particular, the order in which the cherries are picked does not matter due to Theorem 3.1. To see if N is orchard, these steps above need to be repeated until either the reduced network ends up in the trivial case (so N is an orchard network) or it can not be reduced anymore and it can be concluded that N is not orchard. This works since Theorem 3.2 is proven in the previous section. Namely, the reducing step creates a new network $N(x, y)$. Repeating the steps on $N(x, y)$ checks whether $N(x, y)$ is orchard. From this recursion, it can be observed that N is orchard if and only if the output network at the end of the algorithm is a tree with one leaf. So the last step is proving that NS is orchard for some sequence S. But from that it follows by the exception after Theorem 3.3 that N is orchard as well.

This process is written in two algorithms below. Algorithm 1 returns a cherry or reticulated cherry if there exists one in N otherwise it returns 'None' and Algorithm 2 checks if a given network is orchard.

Algorithm 1: FindCherriesOfNetwork(N)

Data: A network N on a set of taxa X
Result: A cherry or reticulated cherry in N if there exists one, otherwise 'None'
for *A leaf* x *in* N **do**
  if p_x*, the parent of x, is a tree vertex* **then**
    if *There exists a leaf child* y*, unequal to* x*, of* p_x **then**
      return (x, y)
    end
    else if *There is a tree vertex parent* p_y *of* p_x*, where* p_y *has a leaf child* y **then**
      return (x, y)
    end
  end
end
return *'None'*

Algorithm 2: ReduceNetwork(N)

Data: A network N
Result: A sequence S that reduces N to a tree with one leaf if N is orchard, otherwise return 'No' and sequence S that reduces N until there are no (reticulated) cherries left.

$S = ()$
$(x, y) =$ FindCherriesOfNetwork(N)
while (x, y) *is not 'None'* **do**
  $N = N(x, y)$
  $S = S(x, y)$
  $(x, y) =$ FindCherriesOfNetwork(N)
end
if N *is a tree with one leaf* **then**
  return S
else
  return *'No' and* S
end

3.3. Labelling

Networks usually have labels, from which the branch length can be worked out. For a phylogenetic network that indicates the evolutionary history of species, the branch length indicates how long this evolution takes. In Section 3.2, an algorithm is given on determining if a network is orchard. If it is determined that a network is orchard, the network can be labelled.

A type of labelling that we use in this section is HGT-consistent labelling.

Definition 3.1 (HGT-consistent labelling) Let N be a binary network on a set of taxa X. HGT-consistent labelling is a function $t : X \longrightarrow \mathbf{R}$ with the following properties:

- $t(x) \leq t(y)$ for all arcs from x to y and equality is only allowed if y is a reticulation.
- For each internal node x, there is a child y of x such that $t(x) < t(y)$.
- For each reticulation r with parents x and y, exactly one of $t(x) = t(r)$ and $t(y) = t(r)$ holds. [5]

HGT stands for horizontal gene transfer and it shows that orchard networks are trees with horizontal arcs added to them. An arc uv is called horizontal with respect to the HGT-consistent labelling if $t(u) = t(v)$. Since in this section it is assumed that N is orchard, the following theorem, Theorem 3.4, states that an HGT-consistent labelling can be used.

Theorem 3.4 *A binary network N is orchard if and only if it admits an HGT-consistent labelling. [5]*

To find an HGT-consistent labelling for an orchard network N, a sequence that reduces N is used. The labelling for N is obtained by constructing the network back from the sequence. Constructing a network works as follows.
There is a picking sequence $S = (x_1, y_1)(x_2, y_2)\dots(x_m, y_m)$ for the network N, since N is assumed to be orchard. Reconstructing starts with a one-leaf tree with leaf y_m (by definition of cherry picking). The first step is adding the pair (x_m, y_m) to this tree. This is done by adding a node on the arc between the root and y_m and connecting this new node to a new leaf x_m. Then for adding the pair (x_{m-1}, y_{m-1}) first is checked if x_{m-1} is already a leaf. If this is the case, then two nodes are added, $p_{x_{m-1}}$ between x_{m-1} and its parent and $p_{y_{m-1}}$ between y_{m-1} and his parent. Between these two new nodes, an arc $(p_{y_{m-1}}, p_{x_{m-1}})$ is added. This arc forms a reticulation arc. If x_{m-1} is not already an existing leaf, the action of adding (x_m, y_m) is mirrored. If this is done for the complete sequence S, the network is finished. This process is visualized in Figure 3.1.

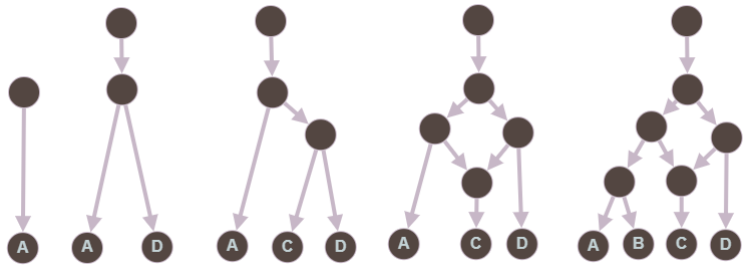

Figure 3.1: N is constructed by repeatedly adding pairs from the sequence S=(B,A)(C,A)(C,D)(D,A) backwards.

With the following definition and theorem, Definition 3.2 and Theorem 3.5, it can be concluded that the network that is constructed is unique.

Definition 3.2 (Reconstructible) *A subset of orchard networks is called reconstructible if for any two networks N, N' in this subset, with a common minimal CPS, we have that N and N' are isomorphic. [6]*

Theorem 3.5 *A binary network is reconstructible. [6]*

Only binary networks are considered and those are reconstructible by Theorem 3.5. From Definition 3.2, it can be concluded that networks reconstructed from the same S are isomorphic and therefore the same. Now the reconstructing part is done, the labelling of a network can be introduced.

Let $S = (x_1, y_1)(x_2, y_2)\dots(x_m, y_m)$ be a sequence that reduces an orchard network N. Now the algorithm for finding an HGT-consistent labelling for an orchard network is given. First label the root node r with $t(r) = 0$, followed by labelling the leaves l_j with $t(l_j) = m + 1$. This makes sure that all the leaves have the same label. Leaves do not necessarily have to have the same label, however in this paper it is chosen to label them the same. The label is $m + 1$ since the leaves need to have higher labelling than any other nodes in the network. Then the internal nodes are left to label. This is done in the following way. Reattaching pair (x_i, y_i) gives label $t(x_i) = m + 1 - i$. This is indeed HGT-consistent labelled proven in [5]

This is formulated in the following pseudo code, Algorithm 3. In this algorithm a new notation is used for adding a pair (x, y) to a network, this is denoted by $N = N + (x, y)$

Algorithm 3: HGTLabellingOfN(S)

Data: A sequence of ordered pairs $S = (x_1, y_1)...(x_m, y_m)$ of length m
Result: Network N that can be reduced by S with HGT-consistent labelling and return the labelling t

Let N be a tree with one leaf y_m.
Let p_{y_m} be the parent of y_m.
Set $t(p_{y_m}) = 0$
$t(y_m) = m + 1$
$i = m$
while $1 \leq i \leq m$ **do**
  $N = N + (x_i, y_i)$
  Let p_{x_i} and p_{y_i} be the parent of x_i and y_i respectively.
  if x_i *is an existing leaf* **then**
    $t(p_{x_i}) = m + 1 - i$ and $t(p_{y_i}) = m + 1 - i$
  else
    $t(x_i) = m + 1$ and $t(p_{x_i}) = m + 1 - i$
  end
  $i = i - 1$
end
return N and t

4

From a non-orchard network to an orchard network

Until now, networks that are not orchard are not discussed. Reducing a non-orchard network seems impossible at first. However, the actions of adding leaf nodes or deleting arcs are possible solutions to this problem. In this chapter we explain how these actions can transform a non-orchard network into an orchard network, making use of a couple of techniques that can be performed on non-orchard networks.

4.1. Adding leaves to a network

Consider a network that is not reducible to a tree with one leaf with just picking cherries and reticulated cherries. This means that at one point, after cherry-picking, the resulting network (which is not a trivial tree) does not contain any cherries or reticulated cherries. It would be very useful to still reduce this network completely. A possible choice of doing this, is by adding leaves to the network to construct reticulated cherries. In order to create a reticulated cherry that can be picked, a leaf should be added to the network. The action of adding a leaf l to a network can be described as follows: first, find a leaf of the network with a reticulation as parent. Such a leaf must exist by considering a lowest reticulation vertex in a network, which can be two things, either a parent of a leaf or there is a cherry below it. The network that is considered has no cherries, so it must be the former. This reticulated parent is denoted by r. Then, pick one of the incoming edges incident to r and name this edge $e = uv$. Subdivide e with a node x by deleting edge uv and adding ux and xv. Then finally, the the new leaf l can be added and connected to x with edge uv.

Figure 4.1 shows the process of adding a leaf to a non-orchard network. Now the network on the right can be reduced by the sequence $S = (B, C)(A, B)(A, C)(B, C)$.

However, if a leaf is added to a bottom most tree edge of the network, a cherry is created instead of a reticulated cherry. In this case, if that particular cherry is picked, there is no progress since the same network as before is obtained. This can be explained by Theorem 4.1.

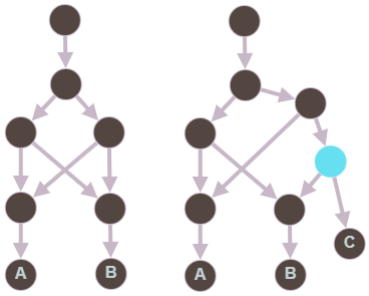

Figure 4.1: The blue leaf is added to a non-orchard network (the network on the left is non-orchard since there are two leaves, but no (reticulated) cherries), such that there is now a reticulated cherry (B, C) that can be picked.

Theorem 4.1 *A network N is orchard if and only if the network obtained by adding a leaf to a tree edge of N is orchard.*

Proof. Assume a network N is an orchard network. This means, by Definition 2.5, that there exists a sequence $S = (x_1, y_1)(x_2, y_2)...(x_n, y_n)$ such that NS is a tree with one leaf. Let l be the leaf that is added to a tree edge and call this new network Nl. The parent of l is x, so $p_l = x$. Since N is an orchard network, all the nodes below x can be reduced. Let $T = (x_1, y_1)...(x_i, y_i)$ be the sequence that reduces all nodes below x. Since x is a parent of two leaves l and y_i in the network NT, with y_i being a leaf of N, it follows that NT contains a cherry (l, y_i). This cherry can be picked. After this, we can continue reducing the resulting network according to sequence S. So Nl can be reduced to a tree with one leaf by sequence $S = (x_1, y_1)(x_2, y_2)...(l, y_i)...(x_n, y_n)$ and it is therefore orchard.

Assume N is a network. Let Nl be an orchard network such that Nl is the network N with leaf l added to a tree edge. Since Nl is an orchard network, there exists a sequence S that reduces Nl to a tree with one leaf. S contains a pair containing leaf l. This is a normal cherry and not a reticulated cherry since l is added to a tree edge instead of a reticulation edge. Choose S to be a sequence that deletes the cherry involving l with l as the first coordinate, the first time l appears. Reduce the network N according to S until the cherry with leaf l occurs. Since network N does not contain leaf l, picking the cherry in Nl containing l makes sure that l is deleted. N already does not contain leaf l, from that point on N can be reduced according to S again. Therefore, N can be reduced by the sequence S without the pair containing l, so N is orchard.

□

From Theorem 4.1 we can say that adding leaves to tree edges in non-orchard networks results in non-orchard networks. Therefore, leaves should be added to reticulation edges in order to reduce non-orchard networks with cherry reduction and leaf additions.

4.2. Deleting edges from a network

Now the action of deleting edges is explained. Deleting an edge is not so straightforward as it might seem. In particular, we wish to define edge deletions so that the resulting graph remains a network. There are a few cases that should be considered.
First, an incoming arc of a tree node cannot be deleted. If this would be possible, there are two cases that can occur that are not possible in a network. One option is that the network

is divided into two separate components. The other option is that there arises a node with indegree-0 and outdegree-2 and that is not the root node. Since it is only possible that the root node has no incoming arcs, this option would not work as well.
The second kind of arc that cannot be deleted exists because of a similar consequence seen in the first case. A reticulation node has indegree-2 and outdegree-1 since only binary networks are considered. So if the outgoing arc of a reticulation is deleted, a node is created with indegree-2 but outdegree-0. The only possibility for a node to have outdegree-0 is for leaves, however there cannot be any leaves created through the process of deleting arcs. Therefore, the outgoing arc of a reticulation cannot be deleted.
Lastly, the incoming arc of a leaf and the outgoing arc of the root cannot be deleted.

These are the cases in which an edge cannot be deleted, so now we state the situation in which an arc can be deleted.
An arc uv (with u and v being nodes in a network) can be deleted from a network only if the outdegree of u is equal to two and v is a reticulation. After deleting an edge, nodes with indegree-1 and outdegree-1 are suppressed. If it occurs that there are parallel edges, deleting one of those counts as deleting an edge, so it does not count as suppressing and edge [7].

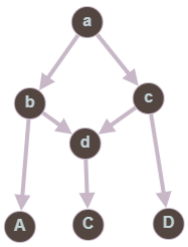

Figure 4.2: Some of the arcs that cannot be deleted in this network are: ab (b will be a node with indegree-0) and dC (d will be a node with outdegree-0 and c will be totally separate of the network). An example of an edge that can be deleted is cd.

4.3. Comparison between adding leaves and deleting edges

Let L denote the minimal number of leaves that need to be added to a non-orchard network to make it orchard and let E be the minimal number of edges that need to be deleted from a non-orchard network to make it orchard. At first instinct it seemed reasonable for L and E to be the same. Since adding a leaf ensures that the reticulation number of a network decreases by at least one, it looks the same as just deleting the reticulation edge. However, the leaves that are added to the network in the process can possibly be used later on to pick a new cherry or reticulated cherry. To see this, an example is shown in Figure 4.3.

First, since there does not exist a cherry or reticulated cherry in the leftmost graph, it is a non-orchard network. We add a leaf C to arc uv. We then reduce the reticulated cherry (A, C). The result (the middle graph) is still a non-orchard network and therefore we add a new leaf D to the edge wu. Now, it can be seen that after adding D, the resulting network (the rightmost graph) can be picked and is therefore orchard. So $L = 2$. Now it is shown that $L = 2$ is an upper-bound for the amount of leaves to add. However, this is also the lowest amount. L cannot be equal to zero since the starting network is non-orchard. Then it should be investigated that $L = 1$ is also not possible. Since this network is symmetric, adding the leaf to the other edge incident to v would not make a difference.

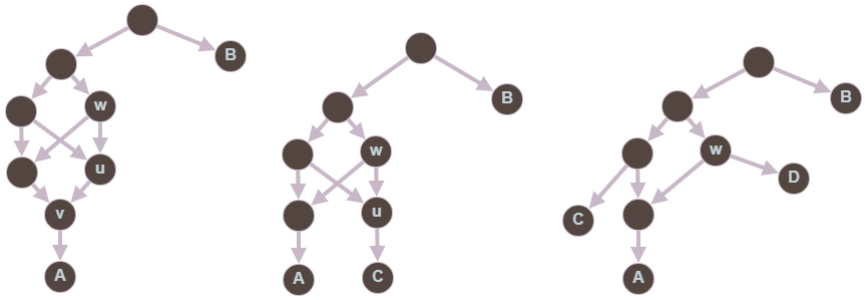

Figure 4.3: The left graph is a non-orchard network. After adding leaf C to edge uv and picking (A, C) the middle network is the network that is left. After adding leaf D to edge wu and picking (C, D) the graph on the right occurs and is orchard. So $L \leq 2$

In Figure 4.4 the process of arc deletion is shown for the same graph as in Figure 4.3. After the action of first deleting wx, followed by yu, a graph is left with a pair of parallel edges. One of these edges should be deleted in order to create an orchard network (right in Figure 4.4). Therefore, $E \leq 3$. Here $E \leq 3$ is found as an upper-bound, however is it also the lowest amount? Again, $E \neq 0$, since the network in non-orchard. Deleting one edge is also not enough since the network obtained by deleting wx is also non-orchard. Deleting any other edge will not result in an orchard network. In any case, there will be a parallel set of edges and one of those should always be deleted.

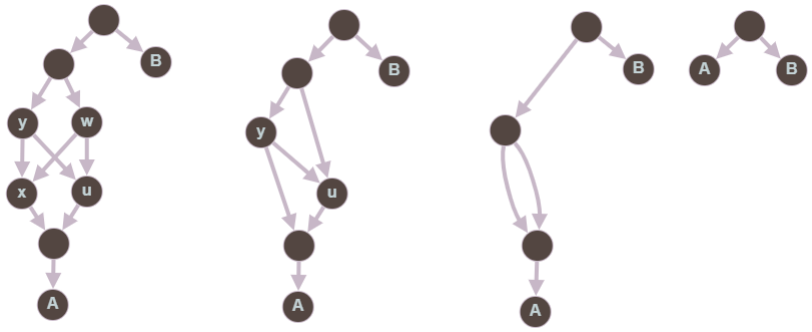

Figure 4.4: The left graph is the same non-orchard network as in Figure 4.3. First, the edge wx is deleted. After that, yu is deleted. At last, one of the two parallel edges is deleted. The resulting network on the right is orchard.

Another network class that has been considered in literature is the class of tree-based networks. The definition of a tree-based network is given by:

A binary network N is tree-based with base tree T in N can be obtained from T in the following steps [5]:

1. Replace some arcs of T by paths, whose internal nodes are called attachment points; each attachment point is of degree-1 and outdegree-1.
2. Place arcs between attachment points, called linking arcs, so that the network remains binary and acyclic.
3. Suppress all attachment points not incident to any linking arcs.

It is true that all orchard networks are tree-based [4] and a tree-based network can have multiple different base trees. [1].

In Figure 4.5, the same network is shown twice. The right network however has blue nodes and edges. These blue parts of the network form a tree containing all nodes in the network. The grey arcs are called the linking arcs.

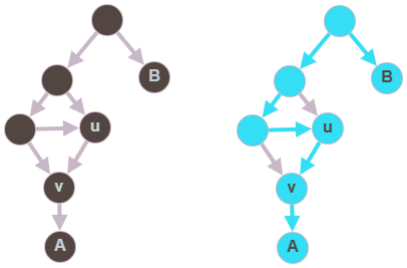

Figure 4.5: On the left a non-orchard network. The blue nodes and edges in the right network form a tree that contains all nodes.

The left network in Figure 4.3 is not tree-based since there is no possible tree that contains all the nodes. So Figures 4.3 and 4.4 show that $L \neq E$ for networks that are not tree-based. The claim $L = E$ is also not true for networks that are tree-based. For this, a counterexample is given in in Figures 4.6 and 4.7.

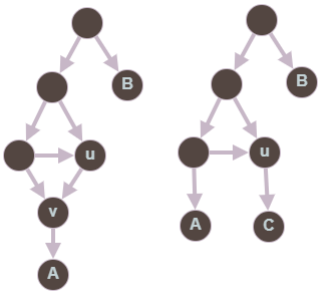

Figure 4.6: The graph on the left in this figure is tree-based and non-orchard. By adding one leaf C to the edge uv and picking (A, C), the network is transformed into an orchard network (on the right). For this network it holds that $L = 1$.

Figure 4.6 shows again the action of adding leaves, but now to a tree-based network (shown in the left in the figure). A new leaf C is added to the edge uv since adding that leaf creates a reticulated cherry (A, C). This reticulated cherry can be picked, creating the network on the right in the figure. This resulting network is an orchard network. Therefore $L = 1$.

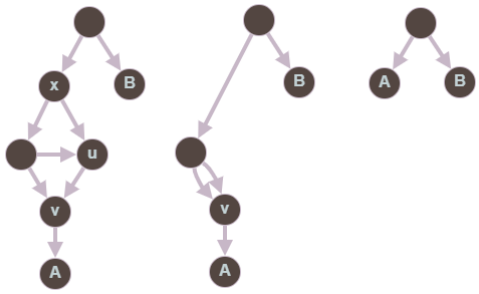

Figure 4.7: The graph on the left in this figure is the same tree-based non-orchard network as in Figure 4.6. In order to make this network orchard, edge xu is deleted. After this, one of the parallel edges is deleted. So $E \leq 2$.

To see if L and E are not the same, Figure 4.7 displays the action of edge deletion. Edge xu is deleted first. The middle network in the Figure 4.7 shows the result of this edge deletion. This network contains again parallel edges, from which one should be deleted. It follows that $E \leq 2$. Is it possible that $E = 1$? This is not possible, since the only other possible edge that can be removed is the incoming arc of v (not the one coming from u). However, if this edge is removed, there exist again a parallel set of edges. On of these parallel edges should be removed, so $E = 2$.

From these counterexamples it can be concluded that the amount of leaves that should be added to a non-orchard network in order to make it orchard is not the same or an upper bound for the amount of edges that should be deleted to make the network orchard.

4.4. Comparison between deleting edges and linking arcs

In the previous section, Section 4.3, it is shown that there is not an obvious connection between the action of adding arcs to a non-orchard network and the action of deleting edges from a non-orchard network in order to make the orchard. In this section the comparison between the amount of edges to delete and the amount of linking arcs that a non-orchard network contains is considered.

For this section, tree-based networks are considered. As stated in Section 4.3, a network is tree-based if there exists a spanning tree in the network, whose leaves coincide with the leaves of the network.

Besides the fact that a network is orchard if there exists a sequence S such that NS is a tree with one leaf, there is another characteristic of a network such that it is orchard. A network is orchard if it is tree based and that the linking arcs are horizontal.

Since another definition of orchard networks is given and linking arcs are introduced, the following theorem can be proved. This theorem shows a comparison between the minimum edges that needs to be deleted and the forward-in-time linking arcs that a network contains. This linking arcs should be forward in time and non-horizontal.

Theorem 4.2 *Let N be a tree-based network such that it is not orchard. The minimum amount of edges to delete to make N orchard is less than or equal to the minimum number of non-horizontal linking arcs, forward in time, existing in N.*

Before proving Theorem 4.2, a new kind of labelling is introduced. Let N be a tree-based network. Label the nodes as follows:
Label the root node with label zero. Let v be a node in the network, that is not the root node. Now label v with the length of a longest path from the root to this node v. That is, label v with the maximal number of arcs that need to be used to go from the root to v. Repeat this for every non-root vertex. This labelling makes sure that no arc is going back in time and no arcs are horizontal. To visualize this labelling, an example is given in Figure 4.8. The figure shows a tree-based network. The digits inside the nodes represent the label that they have. The figure shows a arc that looks horizontally, however by the labelling can be seen that it is forward in time.

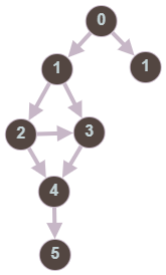

Figure 4.8: This tree-based network has the labelling based on path lengths. One linking looks horizontal, but the labelling shows that it is in fact non-horizontal.

Now that the labelling based on path length is shown, we can find labellings of tree-based networks where most arcs go forward in time and some are horizontal. Now for Theorem 4.2, the minimum number of non-horizontal linking arcs are searched over all such labellings. Theorem 4.2 can now be proved:

Theorem 4.2 *Let N be a tree-based network such that it is not orchard. The minimum amount of edges to delete to make N orchard is less than or equal to the minimum number of non-horizontal linking arcs, forward in time, existing in N.*

Proof. Let N be a tree-based network such that it is not orchard. The labelling based on path length shows that there exists a network with non-horizontal linking arcs forward in time. Now it is showed that such networks exists, let N be a network with k non-horizontal linking arcs forward in time, such that this is the minimal over all possible labels. It is known that a network is orchard if all the linking arcs are horizontal. Therefore, deleting all the non-horizontal arcs from a tree-based non-orchard network makes sure that the resulting network is orchard. Here we use Theorem 3.4. So k is an upper bound for the amount of edges that needs to be deleted in order to make N orchard. Therefore, The minimum amount of edges to delete to make N orchard $\leq$ the minimum number of non-horizontal linking arcs, forward in time, existing in N.

□

5

Conclusion

In this thesis, algorithms are stated in order to check if a given network is orchard or not and for finding a time-labelling for the orchard networks. Besides considering orchard networks, non-orchard networks are also considered. For these non-orchard networks, three measures are considered about transforming non-orchard networks into orchard networks.

In order to create an algorithm that returns if a network is orchard or not, another algorithm is necessary. It is necessary to create an algorithm that returns a cherry contained in a certain network. If such cherry exists, it is used to see if a network is orchard. If the algorithm to find cherries does not return a cherry, it can be concluded that the network is not orchard. The only exception on this statement is the trivial tree, which does not contain a cherry, but is orchard.

If a network is orchard, HGT-consistent labelling is a type of labelling that would always work. To get this type of labelling for a given network, an algorithm in pseudo-code is found, using the sequence that reduces the network to a tree with one leaf. It is used that the root node has label 0 and that all the leaves have the same label.

Non-orchard networks cannot be reduced to a tree with one leaf immediately. Some actions are required for that. The action of adding a leaf to a network is used to create a reticulated cherry that can be picked such that the network can be reduced easier. For this action it is shown that the addition of leaves to a tree edge is not use full for transforming non-orchard networks to orchard networks. Therefore the leaves should be added to reticulation edges. Deleting edges is another action that can help in reducing non-orchard networks. There are some restrictions on the edges that can be deleted in order for the network to remain valid.

In the last part of the thesis, a prove is given for the following statement: the minimum amount of edges to delete to make N orchard is less than or equal to the minimum number of non-horizontal linking arcs, forward in time, existing in N.

6 Discussion

Since the time was limited for this thesis, further research into the subject of this thesis, orchard and non-orchard networks, can be useful. There are a couple of things that are not investigated due to time limitations, but that are interesting to look at for further research.

The algorithms stated in Sections 3.2 and 3.3 are not optimized. So it is possible that the algorithms stated in this thesis are not the most optimal concerning the running time. This a point that would be worth to further investigate.

Another thing that is good to point out is the fact that there is not an upper bound found for the number of leaves that should be added to make a non-orchard network orchard. The same holds for the number of edges that need to be deleted in comparison to the action of adding leaves. However, maybe there exists a clear connection between them.

Lastly, further investigation into the structures of the networks could be useful. During the action of deleting edges, some restrictions were introduced in order to maintain a correct network. However, maybe it is possible to set other restrictions such that it is more effective. The same holds for adding leaves. Adding leaves to non-orchard networks can have as a result that a certain structure in the non-orchard network is deleted. Maybe some of these kinds of structures can be forbidden in case of orchard networks.

References

[1] A. Francis and M. Steel. “Which phylogenetic networks are merely trees with additional arcs?” In: *Systematic Biology* 64.5 (2015), pp. 768–777. URL: https://academic.oup.com/sysbio/article/64/5/768/1686032.

[2] *Graph Online*. URL: https://graphonline.ru/en/.

[3] L. van Iersel. “Hoe zijn ze verwant?” In: (2015). URL: http://www.nieuwarchief.nl/serie5/pdf/naw5-2015-16-3-174.pdf.

[4] L. van Iersel et al. “A unifying characterization of tree-based networks and orchard networks using cherry covers”. In: *Advances in Applied Mathematics* 129 (2021). URL: https://www.sciencedirect.com/science/article/pii/S0196885821000609.

[5] L. van Iersel et al. “Orchard Networks are Trees with Additional Horizontal Arcs”. In: *Bulletin of Mathematical Biology* 84.76 (2022). URL: https://link.springer.com/article/10.1007/s11538-022-01037-z#citeas.

[6] R. Janssen and Y. Murakami. “On cherry-picking and network containment”. In: *Theoretical Computer Science* 856 (2021), pp. 121–150. URL: https://repository.tudelft.nl/islandora/object/uuid%3A6fdc6897-0883-4fc1-b3c3-f62477324d49.

[7] S. Kong et al. “Classes of explicit phylogenetic networks and their biological and mathematical significance”. In: *Journal of Mathematical Biology* 84.47 (2022). URL: https://link.springer.com/article/10.1007/s00285-022-01746-y.

[8] R. Wallin et al. “Applicability of several rooted phylogenetic network algorithms for representing the evolutionary history of SARS-CoV-2”. In: *BMC Ecology and Evolution* 21.220 (2021). URL: https://bmcecolevol.biomedcentral.com/articles/10.1186/s12862-021-01946-y.

[9] S. Willson. “Properties of Normal Phylogenetic Networks”. In: *Bulletin of Mathematical Biology* 72 (2009), pp. 340–358. URL: https://link.springer.com/article/10.1007/s11538-009-9449-z?utm_source=getftr&utm_medium=getftr&utm_campaign=getftr_pilot.