CONSTRUCTING LEVEL-2 PHYLOGENETIC NETWORKS FROM TRINETS

CONSTRUCTING LEVEL-2 PHYLOGENETIC NETWORKS FROM TRINETS

by

Sjors KOLE

to obtain the degree of Master of Science in Applied Mathematics,
for the specialisation Optimization,
at the faculty of Electrical Engineering, Mathematics and Computer Science,
at the Delft University of Technology,
to be defended publicly on Thursday February 6, 2020 at 13:00.

Student number:	4312279	
Thesis committee	Prof. dr. ir. K.I. Aardal,	Technische Universiteit Delft
	Dr. ir. L.J.J. van Iersel,	Technische Universiteit Delft
	Dr. M.M. de Weerdt,	Technische Universiteit Delft

Dr. ir. L.J.J. van Iersel has contributed significantly in the realization of this thesis.

Contents

PREFACE

The better part of the past year I have been researching the structure of phylogenetic networks. The result of this research is this document and the `TriL2Net` algorithm, which combined constitute my thesis. This thesis is the last requirement for fulfilling my master Applied Mathematics at the Delft University of Technology.

The algorithm allows for the construction of a rooted level-2 phylogenetic network from a set of level-2 trinets in polynomial time. This paper illustrates how this algorithm works and performs for different types of sets of trinets. Most importantly, I have shown that the minimal cut-arc sets of a level-2 phylogenetic network are precisely the minimal sink-sets of its auxiliary graph. Using this result, I have shown that the algorithm reconstructs a rooted level-2 phylogenetic trinet from its set of trinets.

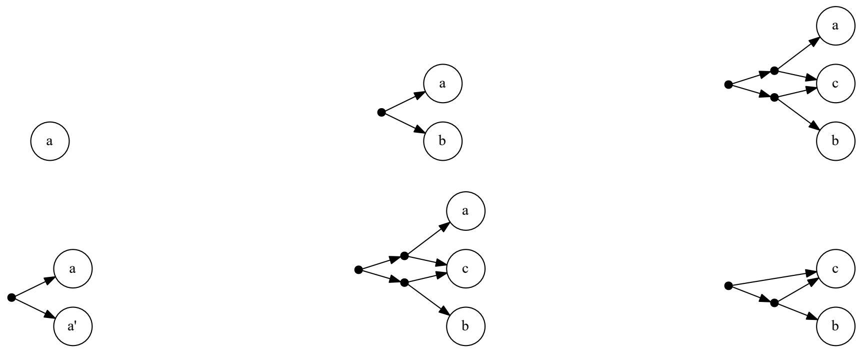
I would like to thank my supervisor Leo van Iersel for his guidance during this process. I would also like to thank my friends and family for the company and support.

Sjors Kole.

Delft, January 9th, 2020.

1 Introduction

In biology, phylogenetics is the study of the evolutionary history and relationship of a set of species or taxa. There are three main types of evolutionary events: speciation, reticulation and extinction. Firstly, speciation is the event of a single species evolving into two or more distinct species. Secondly, reticulation is the event of two or more distinct species forming a new species distinct from the originals. Reticulation events can be broken down into recombination, hybridisation and lateral gene transfer. Thirdly, extinction is the event of a species going extinct. In Figures 1.1a, 1.1b and 1.1c, examples of these events can be seen. Through these events, it is possible that a single species branches and grafts into many distinct species creating a "family tree" of species. The representation of such a family tree in a diagram is also called a phylogeny.



(a) Speciation (b) Reticulation (c) Extinction

Figure 1.1: The three types of evolutionary events.

For a set of species without any reticulation events, a rooted phylogenetic tree can be used as the phylogeny. A rooted phylogenetic tree is a rooted tree in which the leaves are labelled by, for example, a set of species. (Semple and Steel 2003). An example can be seen in Figure 1.2. An important question concerning these trees is how to construct them. A fundamental result in phylogenetics shows that every rooted phylogenetic tree is encoded by its triplets, see for example Dress et al. (2012). A triplet T' of a rooted phylogenetic tree T is a rooted phylogenetic tree on three species that can be deduced from T. This triplet T' is deduced by removing all other species and removing any arcs that lead nowhere iteratively. This process is described in more detail in Section 3.1. In Figure 1.3, two triplets of the tree in Figure 1.2 can be seen. This fundamental result shows that the evolutionary relationship of a set of species X without any reticulation events can be deduced from the evolutionary relationship of every subset of size three of X. Moreover, it is possible to reconstruct a phylogenetic tree from its set of triplets, as is shown by the algorithms introduced by Aho et al. (1981), Ranwez et al. (2007) and Scornavacca et al. (2008).

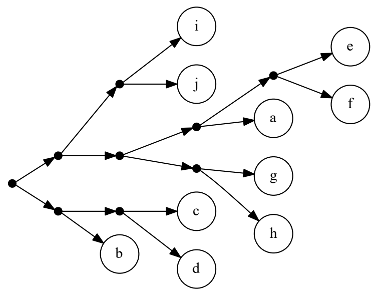


Figure 1.2: A phylogenetic tree.

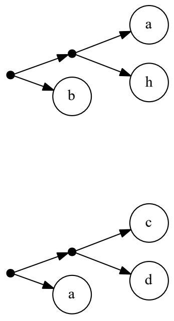


Figure 1.3: Two triplets of the tree on the left.

A shortcoming of phylogenetic trees is that they are not able to describe the aforementioned reticulation events. To tackle this, a generalized version of rooted phylogenetic trees which are able to represent these events was introduced: rooted phylogenetic networks. A rooted phylogenetic network is a rooted acyclic digraph in which the leaves are labelled by a set of species (Huson, Rupp, et al. 2010; Morrison 2011). An example can be seen in Figure 1.4.

Similarly to the aforementioned fundamental result, Huber and Moulton (2013) conjectured that every recoverable rooted phylogenetic network is encoded by its trinets. Here, as expected, a trinet is a rooted phylogenetic network on three species. A formal definition can be found in Figure 3.1. Two trinets of the network in Figure 1.4 can be seen in Figure 1.5. In fact, they proved that level-1 rooted phylogenetic networks are encoded by their trinets. Here ‘recoverable’ is a mild restriction and level-1 limits the complexity of the network; both are defined in more detail in Section 3.1. Using this result, an algorithm has been introduced by Huber and Moulton (2013) which recreates a recoverable level-1 phylogenetic network from its set of trinets. **TriLoNet** introduced the `TriLoNet` algorithm, which is a similar algorithm and is also able to work with noisy data. An algorithm that constructs a rooted level-1 phylogenetic network from a noisy set of triplets was introduced by Huber, van Iersel, et al. (2011).

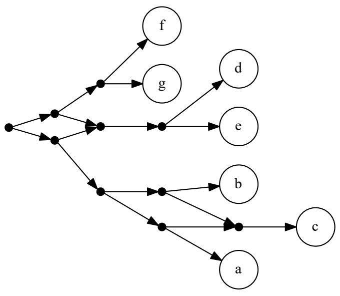


Figure 1.4: A phylogenetic network.

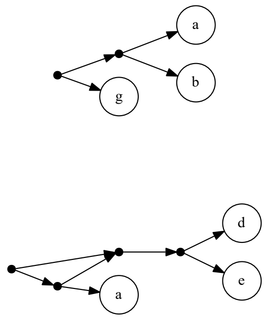


Figure 1.5: Two trinets of the network on the left.

Later, van Iersel and Moulton (2014) proved that recoverable rooted level-2 phylogenetic networks (a category of rooted phylogenetic networks allowing for more complex scenarios) are also encoded by their trinets. Various algorithms have been introduced for constructing rooted phylogenetic networks (Huson and Scornavacca 2012; Lot et al. 2009; Poormohammadi et al. 2014; Than et al. 2008). However, these algorithm do not reconstruct a level-2 network from its set of trinets. This thesis introduces `TriL2Net` (Trinet Level-2 Network Algorithm), which constructs a recoverable rooted level-2 phylogenetic network from a set of recoverable rooted level-2 trinets. In particular, given the set of trinets $\mathscr{T}$ that such a network N exhibits, it construct N. This means that the algorithm's output is correct if the input is too. Furthermore, in case the input set is distorted by either deleting or altering a certain percentage of its trinets, the output network will still be reasonably consistent with the input. For example, if 15% of the trinets are replaced with random other (level-2) trinets, then the trinet consistency scores in our experiment is 70% on average. Moreover, for two sets of generated from biological data, `TriL2Net` is able to construct networks which are at least as consistent with the input as the networks produced by `TriLoNet`.

Lastly, the algorithm's complexity is polynomial in the number of trinets t and number of species n: its runtime is at most $O(n \cdot t + n^5)$. A trinet set on 25 species, for example, is solved in under two minutes using an i7 CPU.

The algorithm utilizes a bottom up approach similar to that of `TriLoNet` and `Neighbour-Joining` (Saitou and Nei 1987). However, it differs from `TriLoNet` in the way it deals with contradicting trinets. Moreover, the heuristics used are different. The steps of the approach are illustrated using a phylogeny of sunflowers in Chapter 2. Thereafter, in Chapter 3, the necessary concepts are formalized. This includes different classifications and properties of rooted phylogenetic networks, such as stable ancestors and cut-arcs. In Chapter 4 several lemmas are introduced, which lead to a characterization of the cut-arc sets of recoverable rooted level-2 phylogenetic networks. Next, in Chapter 5 the different steps of the algorithm are described in detail using pseudocode. For each step, the time-complexity is computed. Moreover, it is shown that if the input is deduced from a recoverable rooted level-2 phylogenetic network N, then the algorithm reconstructs N. Additionally, it is explained which measures are taken to minimize the effect of noise in the input. In Chapter 6, the algorithm is tested on both sampled and biological data. The results of these tests are compared to the results of Oldman et al. (2016). Finally, in Chapter 7 the results are summarized and recommendations for further research are given.

The source code of the algorithm and instructions on how to use it can be found at https://github.com/KSjors/TriL2Net.

2 EXAMPLE

In order to illustrate the different steps of the algorithm, a phylogeny of a set of Helianthus, or sunflowers, is taken as an example (Timme et al. 2007).[1] This phylogeny is a recoverable rooted level-2 network and can be seen in Figure 2.1.

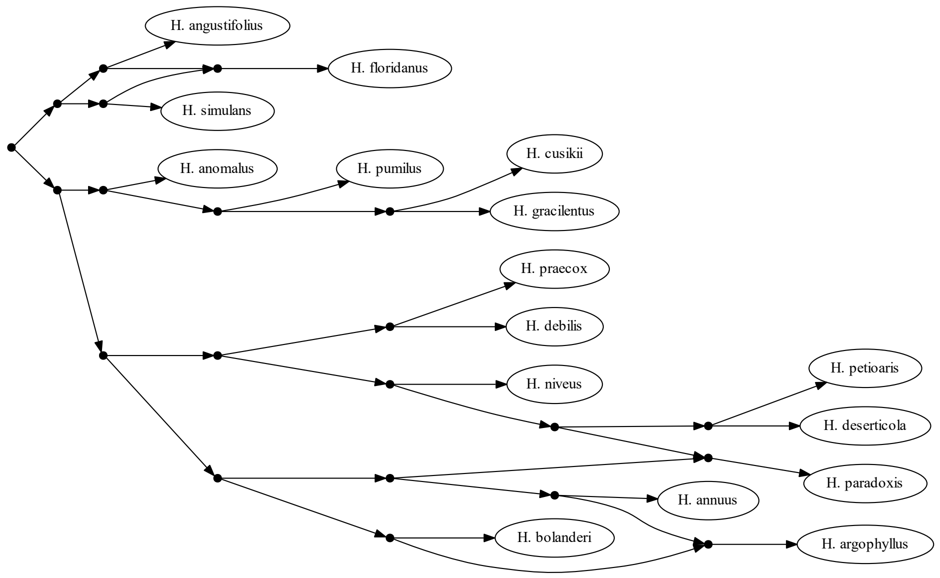


Figure 2.1: A phylogeny of Helianthus.

[1]Only the species in group 2 in the network are used, as this results in a concise and interesting network. Furthermore, the species H. laevigatus, H. schweinitzii, H. hirsutus, H. eggertii, and H. californicus have been removed as their evolutionary relation is unclear.

The input of the algorithm are all the networks on three types of sunflowers. These are called the networks exhibited trinets. Three examples of these trinets can be seen in Figure 2.2 below.

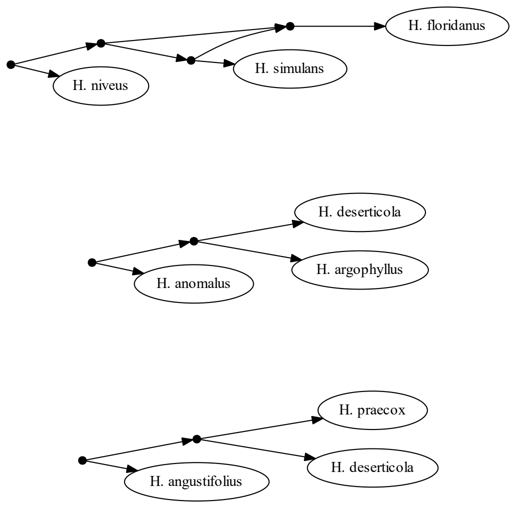


Figure 2.2: Three trinets of the Helianthus phylogeny.

Denote the network from Figure 2.1 using N, the set of sunflowers using X and the set of of trinets using $\mathscr{T}$. Then, the algorithm reconstructs N from $\mathscr{T}$ using a bottom up approach. Step one of this approach consists of identifying groups of species which form a sub-network at the bottom of N. Note this must be done not from N but from $\mathscr{T}$. In Figure 2.3 these sub-networks are illustrated using the red lines.

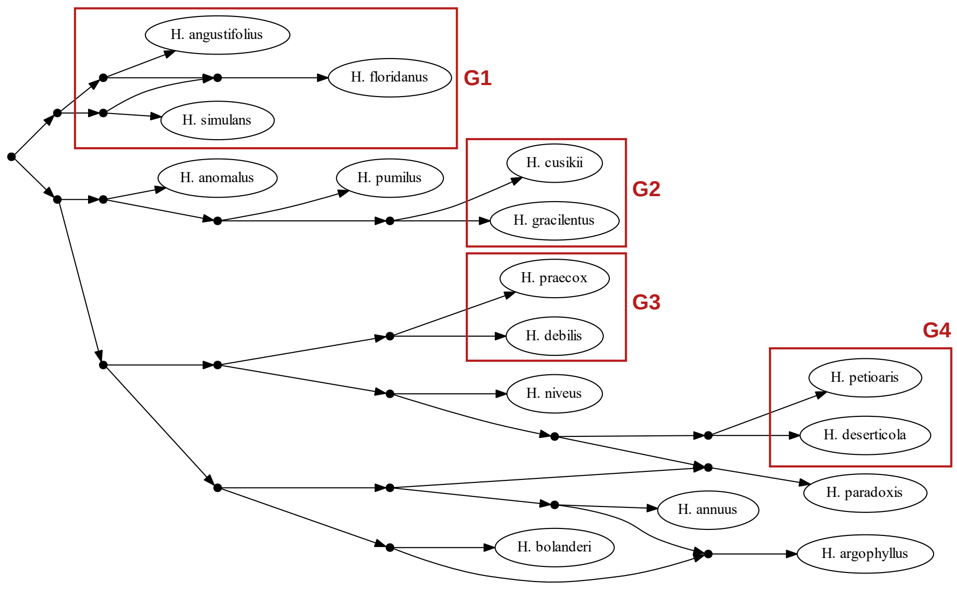


Figure 2.3: The sub-networks at the bottom of the Helianthus phylogeny.

The species attached to these sub-networks are called the minimal cut-arcs sets of N. For example, the minimal cut-arc set corresponding to G1 is:

$$\{H.\,angustifolius,\ H.\,Floridanus,\ H.\,simulans\}.$$

Next, for each of these cut-arc sets, the algorithm finds all the trinets which have information on this set. This is called the restricted trinet set. From the information in this restricted trinet set, the algorithm then computes what the sub-network of this cut-arc set must look like. It does this by iterating through the set and counting certain properties of the trinets. After this has been done for every cut-arc set, these sets are collapsed into leaves. The result of collapsing these sets can be seen in Figure 2.4.

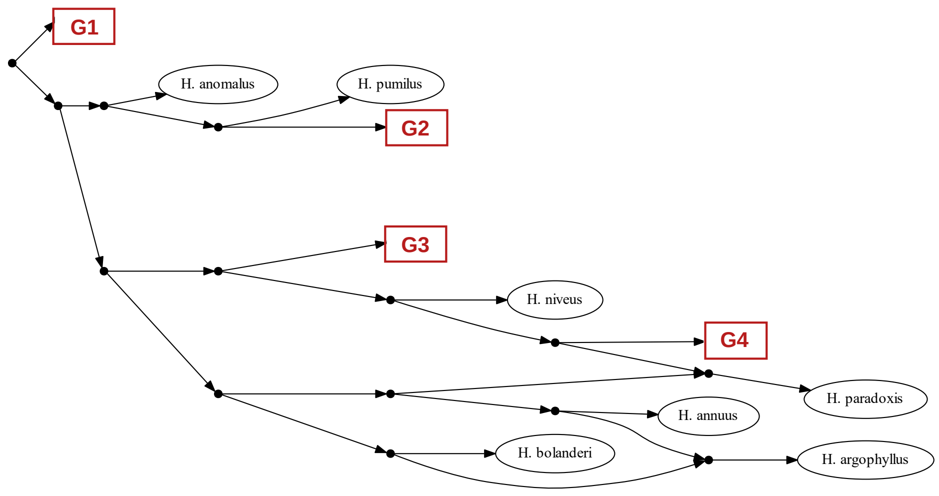


Figure 2.4: The Helianthus phylogeny after collapsing the minimal cut-arc sets $G1$, $G2$, $G3$ and $G4$.

As the original network N is unknown, this collapsing must be performed on the trinets in $\mathscr{T}$ instead. This creates a collapsed trinet set. The three trinets from Figure 2.2, for example, will become the trinets in Figure 2.5. Note that some of the trinets can collapse into binets or singletons, phylogenetic networks on two leaves and one leaf respectively. Such trinets will be discarded.

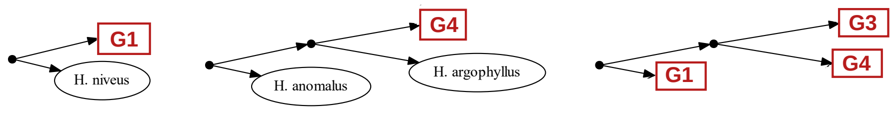


Figure 2.5: The collapsed version of the trinets of Figure 2.2.

New minimal cut-arc sets can be derived from this collapsed trinet set, for which the sub-networks must be computed. These cut-arc sets can again be collapsed into leaves creating a further collapsed version of the already collapsed trinet set. These steps are repeated until every trinet is collapsed into a binet or singleton. The last found sub-network then contains leaves which actually represent a set of leaves; such as the leaves $G1$ in the figures above. Such a leaf is then replaced with the sub-network it corresponds to. This sub-network may again contain leaves which corresponds to other sub-networks. The algorithm then repeats this process of replacing leaves with the corresponding sub-network until all no more of such leaves exist and all the leaves represent a single species in X.

This process can roughly be divided into the following steps:

- collapsing:
  - find the minimal cut-arc sets from $\mathscr{T}$;
  - find the corresponding sub-network from $\mathscr{T}$ for each cut-arc set;
    - ◇ find the number of reticulations in the sub-network;
    - ◇ find the underlying structure of the sub-network;
    - ◇ determine on which side each leaf is located;
    - ◇ determine the order of the leaves on each side.
  - collapse the minimal cut-arc sets in each trinet in $\mathscr{T}$ creating a new trinet set;
- expanding:
  - in each sub-network, find leaves which correspond to other sub-networks;
  - replace these leaves with these sub-networks.

These steps are explained in further detail in Chapter 5.

3
PRELIMINARIES

In order to derive the properties of a network, the evolutionary concepts need to be formalized. These are concepts such as lineage, children and ancestors. These concepts will be illustrated using the phylogenetic network in Figure 3.1. This is (part of) the renamed phylogeny of sunflowers from Chapter 2.

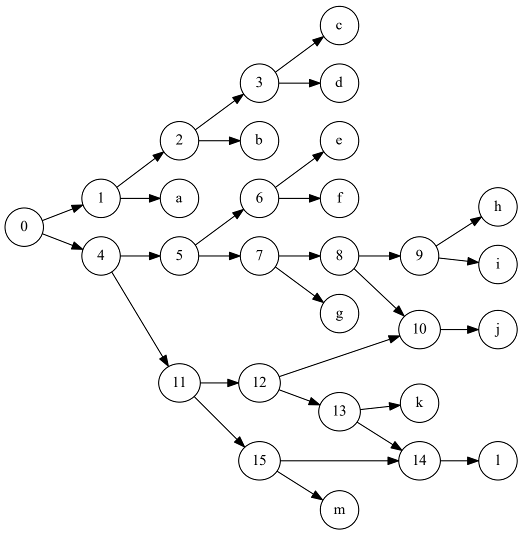


Figure 3.1: Example recoverable strictly level-2 phylogenetic network.

3.1. PHYLOGENETIC NETWORK

Formally, a rooted phylogenetic network is defined as follows.

Definition 3.1.1 (Rooted Phylogenetic Network)**.** For a set of species X, a *rooted phylogenetic network* $N = (V, E)$ on X is a rooted acyclic digraph with (finite) node set V and arc set E in which the leaves are bijectively labelled by X and each node v has either:

- $deg^+(v) = 0$ and $deg^-(v) \geq 2$ (the unique root node);
- $deg^+(v) = 1$ and $deg^-(v) \geq 2$ (tree node);
- $deg^+(v) \geq 2$ and $deg^-(v) = 1$ (reticulation node);
- $deg^+(v) = 1$ and $deg^-(v) = 0$ (leaf node),

where $deg^+(v)$ is the in-degree and $deg^-(v)$ the out-degree of v. Furthermore, duplicate arcs are not allowed in E.

The non-leaf nodes of N are also called the *internal nodes* of N. If the parent of a leaf is root node, tree node, or reticulation node, then the leaf is also called a *root leaf, tree leaf* or *reticulation leaf* respectively.

The *degree* of the network is the maximum of the sum of the in- and out-degrees of each node. A *binary* network has degree 3 for example, which is also the minimum degree of a network.

If N has no reticulation nodes, it is also called a *rooted phylogenetic tree.*

Furthermore, two rooted phylogenetic networks $N = (V, E)$ on X and $N' = (V', E')$ on X' are called *isomorphic* if there exists a bijection $\phi : V \mapsto V'$ such that $(v, w) \in E \iff (\phi(v), \phi(w)) \in E'$. They are called *equal* if additionally $\phi(x) = x$ for all $x \in X$.

Lastly, a rooted phylogenetic network on m leaves is also called an m-net. When $m = 2$ or when $m = 3$ then the network is also called a binet or trinet respectively.

In the example network:

- the unique root node is 0;
- the leaf nodes are $\{a, b, c, d, e, f, g, h, i, j, k, l, m\}$;
- the tree nodes are $\{1, 2, 3, 4, 6, 7, 8, 10, 11, 12, 13, 14, 15\}$;
- and the reticulation nodes are $\{10, 14\}$;
- the degree is 3, hence the network is binary.

From now on, it will be assumed that all phylogenetic trees and networks are rooted and binary, unless stated otherwise.

Important evolutionary concepts are lineage, ancestors and children. Their mathematical counterparts are called paths, higher nodes and lower nodes respectively and are defined below.

Definition 3.1.2 (Paths)**.** Let $N = (V, E)$ be a phylogenetic network, and let $u, v \in V$ be two nodes. Then a path from u to v in N of length $n \geq 1$ is a sequence of unique nodes $p_i \in V, i \in \{1, \dots, n\}$ such that:

$$\begin{aligned} p_1 &= u, \\ p_n &= v, \\ \big((p_i, p_{i+1}) \in E\big) \vee \big((p_{i+1}, p_i) \in E\big) & \qquad \forall i \in \{1, \dots, n-1\}. \end{aligned}$$

A path is called directed if the last condition can be replaced with:

$$(p_i, p_{i+1}) \in E \qquad \forall i \in \{1, \dots, n-1\},$$

and a backwards path if the last condition can be replaced with:

$$(p_{i+1}, p_i) \in E \qquad \forall i \in \{1, \dots, n-1\}.$$

If such paths exists, it is said that $\overleftrightarrow{P}(u, v) \in N$, $\overrightarrow{P}(u, v) \in N$ and $\overleftarrow{P}(u, v) \in N$ respectively. Furthermore, $\overleftrightarrow{P}_N(u, v)$, $\overrightarrow{P}_N(u, v)$ and $\overleftarrow{P}_N(u, v)$ are used to denote the set of all paths, dipaths, and backwards paths from u to v in N respectively.

Using these paths, it can be defined what higher and lower nodes are.

Definition 3.1.3 (Partial ordering)**.** Let $N = (V, E)$ be a phylogenetic network on X and let $u, v \in V$ be two nodes. Then u is said to be strictly below v if $\overrightarrow{P}(v, u) \in N$. Similarly, u is said to be strictly above v if $\overrightarrow{P}(u, v) \in N$. This is denoted with $u < v$ and $v > u$ respectively. This is also called the reachability relation, and is in fact a partial ordering of the nodes in V.

From this, a partial ordering of the arcs can be induced. Let $e, f \in E$ be arcs and denote $e = (e_1, e_2)$ and $f = (f_1, f_2)$. Then:

$$\begin{aligned} e &> f & \text{if} && e_1 &\geq f_2, \\ e &\geq f & \text{if} && e = f &\text{ or } e > f, \\ e &< f & \text{if} && e_2 &\leq f_1, \\ e &\leq f & \text{if} && e = f &\text{ or } e < f. \end{aligned}$$

Similarly, a partial ordering between arcs and nodes can be induced. Let $v \in V$ and $e \in E$. Again denote $e = (e_1, e_2)$. Then:

$$\begin{aligned} v &> e & \text{if} && v &> e_2, \\ v &\geq e & \text{if} && v &\geq e_2, \\ v &< e & \text{if} && v &< e_1, \\ v &\leq e & \text{if} && b &\leq e_1. \end{aligned}$$

Lastly, a partial ordering between the leaves can be induced. Let $x, y \in X$ be two leaves and let $v, w \in V$ be their respective parents. Then:

$$\begin{aligned} x \prec y &\quad \text{if} \quad v < w, \\ x \succ y &\quad \text{if} \quad v > w. \end{aligned}$$

In the example network:

- there is a directed path from node 5 to leaf i, but not from node 1 to node 10;
- node h is strictly below node 7;
- arc $(0,1)$ is neither below nor above edge $(4,11)$;
- and node 9 is below, but not strictly below, arc $(8,9)$.

An important type of ancestor is a stable ancestor. A stable ancestor of a species or set of species is an ancestor from which all genetic data, except mutations, can be traced back too.

Definition 3.1.4 (Stable Ancestors)**.** Let $N = (V, E)$ be a phylogenetic network and let $v \in V$ be a node. Then a stable ancestor of v is a node $s \in V$ such that s is on every directed path from the root ρ to v. The lowest stable ancestor of s is the lowest of all stable ancestors of v. The stable ancestors and lowest stable ancestor of s are denoted with $SA(s)$ and $LSA(s)$ respectively.

A stable ancestors of a set $S \subseteq V$ nodes is a node $s \in V$ that is a stable ancestor of all nodes in S. The lowest stable ancestor of S it the lowest of all stable ancestors of S. The stable ancestors and lowest stable ancestor of S are denoted with $SA(S)$ and $LSA(S)$ respectively.

Note that there always exists a stable ancestor, as the root is on every path from the root to a node. Also note that the lowest stable ancestor is unique.

In the example network:

- the lowest stable ancestors of leaf $\{i\}$ are the nodes 14, 12, 11 and 0;
- the lowest stable ancestor of leaves $\{b, c, g\}$ in the example network is the node 0;
- and the lowest stable ancestor of nodes $\{b, i\}$ is node 11.

Often a set of species is grouped in genera. These groups are often below the cut-arcs of the phylogenetic network and are called the network's cut-arc sets. These notions are defined below.

Definition 3.1.5 (Cut-arcs)**.** Let $N = (V, E)$ be a phylogenetic network and let ρ be its root. An arc $e = (e_1, e_2) \in E$ is called a *cut-arc* of N if for every node v below e_2 every path from ρ to v passes through e. A cut-arc is called a *trivial* cut-arc if its head is a leaf, that is, $e_2 \in X$. Furthermore, a non-trivial cut-arc is called a *minimal* cut-arc if there does not exist any non-trivial cut-arc below it. Lastly, if a phylogenetic network contains only trivial cut-arcs it is called a *biconnected phylogenetic network*.

Definition 3.1.6 (Cut-arc sets)**.** Let $N = (V, E)$ be a phylogenetic network on X and let $e = (e_1, e_2)$ be a cut-arc. Then the set of all species $S \subset X$ below e is called a *cut-arc set*. If $|S| = 1$ then S is called a *trivial* cut-arc set. If S is non-trivial and does not contain any other non-trivial cut-arc set, it is called a *minimal* cut-arc set. Furthermore, the set of all species X is also a cut-arc set. Therefore, in case N is biconnected, X is a minimal cut-arc set.

In the example network:

- the non-trivial cut-arcs in the example are edges $(0,1)$, $(1,2)$, $(2,3)$, $(0,4)$, $(5,6)$ and $(8,9)$;
- the three minimal cut-arcs are edges $(2,3)$, $(5,6)$ and $(8,9)$;
- minimal cut-arc $(2,3)$ has corresponding minimal cut-arc set $\{e, f\}$;
- cut-arc $(1,2)$ has corresponding cut-arc set $\{b, c, d\}$;

Observation 3.1.7. As the minimum size of such a set is two, a trinet can only have one minimal cut-arc set.

Similar to how cut-arc sets are sets of leaves below a cut-arc, biconnected components are sets of nodes 'between' cut-arcs.

Definition 3.1.8 (Biconnected components)**.** Let N be a phylogenetic network. Then the nodes that are connecte to each other after removing all the cut-arcs and leaves *from* N are called the *biconnected components*. A biconnected component is called *trivial* if it consists of a single node. A biconnected component is *below* an arc if all of its nodes are below it. The *leaf set* of a biconnected component is the set of all the leaves whose parents are in that component. In this case it is said that these leaves are *attached to* the biconnected component.

If a biconnected component contains the root ρ of N, then it is also called the *root biconnected component*. If a biconnected component has no outgoing non-trivial cut-arcs it is called a *leaf biconnected component*.

Often biconnected is left out when no confusion is possible.

Two special types of biconnected components are the one adding *almost no* information, and the one adding *no* information about the evolutionary relation. These components are called redundant and strongly redundant components respectively.

Definition 3.1.9 (Redundant components)**.** Let N be a phylogenetic network and let C be one of its non-trivial biconnected components. Then C is called *redundant* if it has only one outgoing arc. Furthermore, it is called *strongly redundant* if it also has no ingoing arcs. N is called *recoverable* if it contains no strongly redundant components.

In the example network:

- the only non-trivial biconnected component of the network is $\{4,5,7,8,10,11,12,13,14,15\}$;
- leaves $\{g, j, k, l, m\}$ are attached to this component;
- this component is not redundant nor strongly redundant;
- this component contains two reticulations.

Observation 3.1.10. For a phylogenetic network without redundant components, a cut-arc is trivial respectively minimal if and only if the corresponding cut-arc set is trivial respectively minimal.

The number of reticulations contained in a biconnected component describes how tree-like it is and is also called the level of the component.

Definition 3.1.11 (Level)**.** Let N be a phylogenetic network and let C be one of its non-trivial biconnected components. Then C is said to be *level-k* if contains no more than k reticulation nodes, and it said to be *strictly level-k* if it has exactly k reticulation nodes. Furthermore, N is said to be *level-k* if all of its biconnected components are of level-k, and it said to be *strictly level-k* if it is level-k and at least one of its biconnected components is strictly level-k.

Clearly, the example network is strictly level-2 as all of its non-trivial biconnected components are strictly level-2. Examples of a strict level-0 and a strict level-1 phylogenetic networks can be seen in Figure 1.2 and Figure 1.4 respectively.

Each of the biconnected components of a phylogenetic network has a certain underlying structure. This structure is called the generator of the component. The definition of a generator is slightly different to that of a phylogenetic network.

Definition 3.1.12 (Generator)**.** A *generator* $\mathscr{G} = (V, E)$ is a biconnected rooted acyclic digraph with (finite) node set V and edge set E in which each node has either:

- $deg^+(v) = 0$ and $deg^-(v) \geq 2$ (the unique root node);
- $deg^+(v) = 1$ and $deg^-(v) \geq 2$ (tree node);
- $deg^+(v) \geq 2$ and $deg^-(v) \leq 2$ (reticulation node).

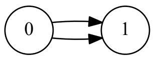
The *degree* of the generator is the maximum of the sum of the in- and out-degrees of each node. A *binary* generator has degree 3 for example, which is also the minimum degree of a generator. Furthermore, the number of reticulations of a generator is also called its *level*. A level-k generator is denoted using $\mathscr{G}^k$. If there are multiple such generators, an identifier is appended: e.g. $\mathscr{G}^{2a}$, $\mathscr{G}^{2b}$, $\mathscr{G}^{2c}$ and $\mathscr{G}^{2d}$ denote the four different level-2 generators.

The edges and out-degree 0 reticulation nodes are also called its *edge sides* and *reticulations sides* respectively. The set of all sides of a generator $\mathscr{G}$ denoted with $\mathscr{G}_S$, and the set of edge sides and the set reticulation sides are denoted with $\mathscr{G}_E$ and $\mathscr{G}_R$ respectively. A single side is referred to with s.

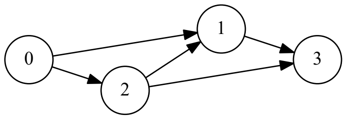
Two generators $\mathscr{G} = (V,E)$ and $\mathscr{G}' = (V',E')$ are called *isomorphic* if there exists an isomorphism $\phi : V \to V'$ such that $(v,w) \in E \iff (\phi(v),\phi(w)) \in E'$ for all $v,w \in V$. Moreover, they are called *equal* in case that $\phi(v) = v$ for all reticulation sides. Furthermore, if $\mathscr{G}$ and $\mathscr{G}'$ are equal, then it is said that the sides (v,w) and $(\phi(v),\phi(w))$ are in the same *set of symmetric sides*. Such a set is denoted using I.

A set of symmetric sides I is *crucial* if at least one leaf must be attached to one of the sides it contains in order for $\mathscr{G}$ to become a phylogenetic network. The set of all crucial sides is denoted with $\mathscr{G}_C$. Similarly $\mathscr{G}_{CE}$ and $\mathscr{G}_{CR}$ denote the sets of all *crucial edge sides* and *crucial reticulation sides* respectively.

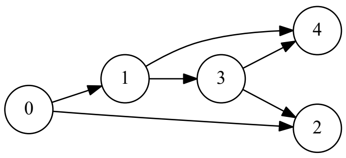
The differences between a generator and a phylogenetic network are that a generator does not contain any leaves, is always biconnected, can have duplicate edges, and that it has exactly the same amount of reticulations as its level. All the generators from up-to level 2 can be seen in Figure 3.2 (van Iersel and Moulton 2014).



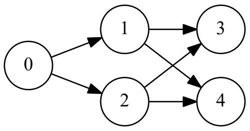
(a) Generator $\mathscr{G}^1$



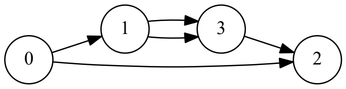
(b) Generator $\mathscr{G}^{2a}$



(c) Generator $\mathscr{G}^{2b}$



(d) Generator $\mathscr{G}^{2c}$



(e) Generator $\mathscr{G}^{2d}$

Figure 3.2: All level-1 and level-2 generators

In order to clarify the definition of the different types of sides of a generator, some of the sides of these generators are pointed out. For example, generator $\mathscr{G}^{2d}$ has six sides in total: five edge sides and one reticulation side. The sides $(1,3)$ and $(1,3)$ are equal, hence this generator contains five sets of symmetric sides: $\{(0,1)\}$, $\{(0,2)\}$, $\{(1,3),(1,3)\}$, $\{(3,2)\}$ and $\{2\}$. Note that for generator $\mathscr{G}^{2c}$ the reticulation sides 3 and 4 are not in the same set of symmetric sides. This is due to the fact the labelling of the reticulation sides needs to be preserved. For this generator, the non-singleton symmetric side sets are $\{(0,1),(1,2)\}$, $\{(1,3),(2,3)\}$ and $\{(1,4),(2,4)\}$.

The example network contains one component which is based on a generator other than generator $\mathscr{G}^0$: the nodes 4, 10, 11, 12, 14 and 15 together form generator $\mathscr{G}^{2b}$.

3.2. EXHIBITED NETWORKS, TREES AND CLUSTERS

Now it remains to be defined what an exhibited network is. An exhibited network can be seen as the evolutionary history on a set of species derived from a the evolutionary history of a larger set of species. The removal of information regarding one of the species not in this subset is called termination.

Definition 3.2.1 (Termination event)**.** Let $N = (V, E)$ be a phylogenetic network on X of degree d. Then a *termination event* reduces the number of species by one by *terminating* a leaf $x \in X$. That is, the following steps are executed:

1. remove x from X and V,
2. let p be the parent of x,
3. remove the edge (p,x) from E,
4. if p was the root node and now has a single outgoing arc
   (a) let c be the child of p,
   (b) remove edge (p,c) from E,
   (c) remove p from V.
5. if p was a tree node and now has a single outgoing arc:
   (a) let g and c be the parent and the child of p respectively,
   (b) remove edges (g,p) and (p,c) from E,
   (c) remove p from V,
   (d) if (g,c) is not an edge in E, add the edge (g,c) to E,
   (e) if (g,c) is an edge in E, follow steps (4), (5) and (6) from this list for $p \in \{g,c\}$.
6. if p was a reticulation node and now has no outgoing arcs:
   (a) let $\{g_i\}_{i=1}^{t}$ with $1 < t < d$ be the parents of p,
   (b) remove the edges $\{(g_i,p)\}_{i=1}^{t}$ from E,

   (c) remove p from V,
   (d) for each node g_i follow steps (4), (5) and (6) from this list with $p = g_i$.

7. remove all (strongly) redundant components C from N by removing all nodes and arcs in C from N and adding an arc between the node above C and the node below C, if both exist.

A network N' on X' is exhibited by a network N on X, if it is the network N with every leaf in $X \setminus X'$ terminated.

Definition 3.2.2 (Exhibited networks)**.** Let N and N' be two phylogenetic networks on X and $X' \subseteq X$ respectively. Then N' is an *exhibited network* of N if there exists a sequence of termination events that transform N into N'.

In Figure 3.3 two of the exhibited trinets of the example network can be seen.

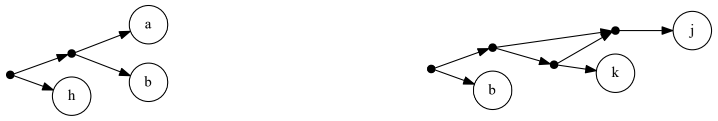


Figure 3.3: Two exhibited trinets of the example network.

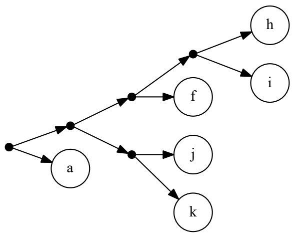
If the reticulation nodes of a phylogenetic network N are suppressed, the network becomes a tree. Here suppressing a reticulation node r is the removal of all but one of its incoming arcs.

Definition 3.2.3 (Reticulation suppression event)**.** Let $N = (V, E)$ be a phylogenetic network on X of degree d. Then a *reticulation suppression event* reduces the number of reticulations by one by *suppressing* a reticulation node $r \in V$. That is, the following steps are executed:

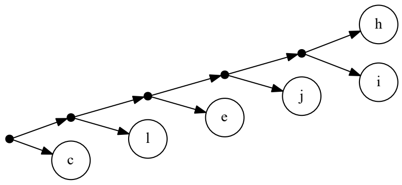
1. let $\{p_i\}_{i=1}^{t}$ with $1 < t < d$ be the parents of r,
2. for each parent p_i of r create a network N_i by performing the following:
   (a) remove all arcs (p_j, r) for $j \neq i$ from N,
   (b) for each node p_j, $j \neq i$ perform steps (4), (5), (6) and (7) from Definition 3.2.1

Definition 3.2.4 (Exhibited tree)**.** Let N and T be a phylogenetic network and a phylogenetic tree on X respectively. Then T is an *exhibited tree* of N if there exists a sequence of termination and reticulation suppression events that transform N into T.

Note that suppressing a reticulation node in a phylogenetic network N on X generates at most $d-1$ phylogenetic networks, where d is the degree of N. Hence, if N contains k reticulation nodes, then there exist at most $(d-1)^k$ exhibited trees of N on X. This is an upper limit due to the fact that suppressing one reticulation node can lead to the termination of others. Two trees exhibited by the example network can be seen in Figure 3.4.



(a) To get this tree, reticulation node 10 is suppressed by removing arc $(8,10)$. Which arc entering reticulation node 14 is removed does not matter.



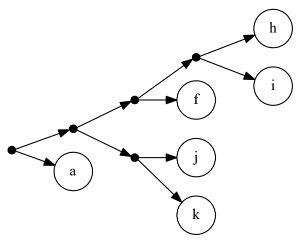
(b) To get this tree, reticulation node 10 is suppressed by removing arc $(12,10)$. Which arc entering reticulation node 14 is removed does not matter.

Figure 3.4: Two exhibited trees of the example network.

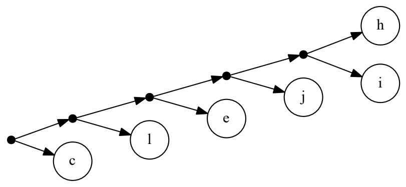
A special type of exhibited tree is the exhibited cluster. An exhibited cluster T of a network $N=(V,E)$ is the tree created from N by suppressing all reticulations nodes and then removing all arcs and leaves not below a certain node $v \in V$ from N.

Definition 3.2.5 (Exhibited cluster)**.** Let $N=(V,E)$ and T be a phylogenetic network on X and a phylogenetic tree on $X \subseteq'$ respectively. Then T is an *exhibited cluster* of N if it is an exhibited tree and X' consists of all leaves below a certain node $v \in V$.

A phylogenetic tree N containing n nodes and leaves exhibits exactly n clusters. A phylogenetic network N on X of degree d containing n nodes and leaves and k reticulations can exhibit at most $(d-1)^k \cdot n$ clusters: n for each of the at most $(d-1)^k$ trees on X it exhibits. Two clusters of the example network can be seen in Figure 3.5.



(a) To get this tree, reticulation node 10 is suppressed by removing arc $(8,10)$. Which arc entering reticulation node 14 is removed does not matter.



(b) To get this tree, reticulation node 10 is suppressed by removing arc $(12,10)$. Which arc entering reticulation node 14 is removed does not matter.

Figure 3.5: Two exhibited clusters of the example network

These exhibited networks, trees and clusters can be aggregated in a set.

Definition 3.2.6 (Network sets, tree sets and cluster sets)**.** A set of unique networks including a multiplicity per network is called a *network set* and is denoted using $\mathscr{N}$. If all networks in the set have the same amount of species m, then this set is called an *m-net set.* If a network N is added to a network set $\mathscr{N}$ and $\mathscr{N}$ already contains a network N' equal to N, then the multiplicity of N' is increased by one. The *size* of a network set is the number of distinct networks it contains and is denoted using $size(N)$. The *volume* of a network N is the sum of all the multiplicities of the networks it contains and is denoted using $volume(N)$.

If a network set contains two distinct networks on the same set of species, then it is said that these networks *contradict* each other.

If $m = 3$, a network set it is also called a *trinet set* and is denoted with $\mathscr{T}$. If $m = 2$ it is called a *binet set* and is denoted with $\mathscr{B}$. The set of m-nets exhibited by a network is called an *induced m-net set* or the m-net set induced by N, and is denoted with $\mathscr{N}_m(N)$. In case $m = 2$ or $m = 3$ then this is denoted using $\mathscr{B}(N)$ and $\mathscr{T}(N)$ respectively.

Furthermore, the set of species $\mathscr{N}$ describes is denoted with $\mathscr{N}_X$:

$$\mathscr{N}_X = \bigcup_{N \in \mathscr{N}} N_X,$$

where N_X is the set of species of network N.

A set of m species in also called an *m-set.* The number of m-sets of $\mathscr{N}_X$ that are represented by a network in $\mathscr{N}$ is called the *coverage* of $\mathscr{N}$ and is denoted using $coverage(\mathscr{N})$.

Tree sets and *cluster sets* are defined in a similar way.

There are several methods to iterate over such sets. These methods are different in the way they approach the multiplicity of the contained networks.

Definition 3.2.7 (Network set iterating methods)**.** Let $\mathcal{N}$ be an m-net set on n leaves. Denote all the m-sets of $\mathcal{N}_X$ using $M_1, \dots M_{\binom{n}{m}}$ and denote the networks $N \in \mathcal{N}$ that have $N_X = S_i$ using $N_i^2, \dots$. Denote their corresponding multiplicities using $m_i^1, m_i^2, \dots$. Then a *network set iterator* iterates over all sets S_i and yields some of the networks N_i^j together with a weight w_i^j.

The first iterator is the *maximum multiplicity* iterator, which yields for each m-set M_i the network N_i^j which has the highest multiplicity. If there are two or more networks with the highest multiplicity, one of them is picked at random. The weights yielded with these networks are always one.

The second iterator is the *weighted average* iterator, which yields all networks N_i^j. The weight of a network is its multiplicity divided by the sum of the multiplicities of all networks on M_i, that is:

$$w_i^j = \frac{m_i^j}{\sum_h m_i^h}.$$

The third and last iterator is called the *weighted sum* iterator. This iterator simply yields all networks N_i^j together with weights $w_i^j = m_i^j$.

Note that these iterators and methods yield the same networks and weights for induced network sets, but do not necessarily yield the same for induced tree and cluster sets. Furthermore, in `TriLoNet` the weighted sum iterator is used (Oldman et al. 2016).

3.3. Collapsed networks

Another method for deducing one phylogenetic from another is by collapsing a set of leaves to a single leaf.

Definition 3.3.1 (Collapsed phylogenetic network)**.** Let N be a phylogenetic network on X, let $S \subseteq X$ be a set of leaves and let l be a species not in X. Then *collapsing* the set S in N to l results in the a network N_s for each leaf $s \in S$. That is, N_s is the phylogenetic network on $X' = (X \setminus S) \cup \{l\}$ created from N by terminating all leaves in S except s, and renaming s to l. These networks N_s are also called *collapsed networks*. Note that these collapsed networks do not have to be unique. Moreover, if S is a cut-arc set of N, then all the collapsed networks are the same.

Collapsed versions of the Helianthus phylogeny of Figure 2.1 and its trinets in Figure 2.2 can be found in Figures 2.4 and 2.5 respectively.

3.4. The auxiliary graph

For the bottom up approach, the algorithm needs to determine the minimal cut-arc sets for the resulting phylogenetic network from the set of trinets. This is done using an auxiliary graph.

Definition 3.4.1 (Auxiliary graph)**.** Let $\mathcal{T}$ be a set of trinets on a set of species X. Then the *auxiliary graph* $\Omega(\mathcal{T})$ has vertex set $\mathcal{T}_X$ and (directed) edge set consisting of those $(x, y) \in (\mathcal{T}_X \times \mathcal{T}_X)$ such that every trinet $T \in \mathcal{T}$ with $\{x, y\} \subset T_X$ has y in in its minimal cut-arc set.

Sometimes it is easier to look at the contrapositive of the definition of the edges: if a trinet $T \in \mathcal{T}$ on $\{x, y, z\}$ has minimal cut-arc set $\{x, y\}$, then $(x, z) \notin \Omega$ and $(y, z) \notin \Omega$.

Below we define *minimal sink sets* of the auxiliary graph. In Section 4.4, this property will be shown to correspond to the minimal cut-arc sets of a phylogenetic network.

Definition 3.4.2 (Sink-set)**.** Let $\Omega = \Omega(\mathcal{T})$ be an auxiliary graph and let $S \subseteq \mathcal{T}_X$. Then S is a *sink-set* if it has no outgoing arcs in Ω. The set of all species $\mathcal{T}_X$ is called the *complete* sink-set. Furthermore, S is a *minimal* sink-set if it consists of at least two vertices and does not contain any other sink-sets.

In Figure 3.6, the auxiliary graph of the trinet set induced by the example network from Section 3.1 can be seen. Note that the minimal sink-sets of the auxiliary graph indeed correspond to the minimal cut-arc sets.

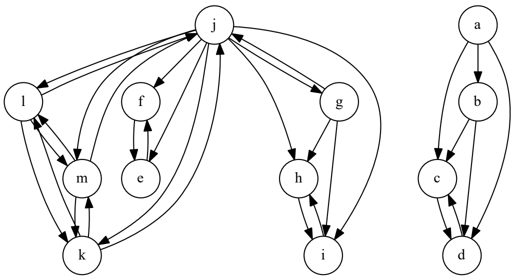


Figure 3.6: The auxiliary graph of the trinet set induced by the example network.

4 Theory

Finding the minimal cut-arc sets of a phylogenetic network through its trinets is the most mathematically complex part of the problem. This complexity arises from the fact that the information deduced from a single trinet only excludes the possibility of certain cut-arc sets in the complete network. For example, assume that T is a trinet on species x, y and z with minimal cut-arc set $\{x, y\}$. Then the only fact that can be deduced is that there exists no minimal cut-arc set containing both x and z or both y and z. If no trinet is found that excludes the possibility that x and z are in the same minimal cut-arc set, then they must be in the minimal cut-arc set together.

The auxiliary graph from Definition 3.4.1 is defined to capture all this information. The final theorems in this chapter show that its minimal sink-sets indeed characterize the minimal cut-arc sets of a phylogenetic network. The first of these theorems (Theorem 4.4.2) holds for level-2 phylogenetic networks only. This is due to the condition in Lemma 4.2.3. It is conjectured that this lemma also holds for networks of any level, and thus that Theorem 4.4.2 also holds for any level.

To prove these theorems, it is first analysed what happens to the cut-arc sets of a network when some of its leaves are terminated. This is because terminating leaves creates trinets, and these trinets induce the auxiliary graph.

4.1. STRUCTURE OF PHYLOGENETIC NETWORKS

It is important to know what happens to structure of a phylogenetic network when some of its leaves are terminated. Here, structure refers to the partial ordering of the nodes, lowest stable ancestors of nodes and the networks cut-arc sets.

First a rather intuitive result: the partial ordering of the nodes of a phylogenetic network is preserved when one of its leaf is terminated.

Lemma 4.1.1. *Let $N = (V,E)$ be a phylogenetic network and let $N' = (V',E')$ be the phylogenetic network on $X' = X \setminus \{x\}$ created by terminating a leaf $x \in X$ in N. Then $u \leq_{N'} v$ if and only if $u \leq_N v$ for any $u, v \in V'$.*

Proof. Let $u, v \in V'$ and assume that $u \leq_{N'} v$. It will be shown that $u \leq_N v$. By definition, there exists a directed path P' from u to v in N'. Clearly, if only arcs are removed during the termination process, then P' is also a path in N. However, there is one step in this process in arcs are added to the network: step *5(d)* adds the arc (g,c) to E'. If P' contains an arc (g,c) that was was added during termination, then just before the adding of the arcs two other arcs (g,p) and (p,c) were removed. Hence replacing (g,c) with the arcs (g,p) and (p,c) creates a longer path from u to v in N. Hence, there exists a (possibly longer) path P from u to v in N. So $u \leq_N v$.

Next, let $u, v \in V'$ and assume that $u \leq_N v$ in N. It will now be shown that $u \leq_{N'} v$. By definition, there exists a directed path P from u to v in N. Clearly, if only arcs are added during the termination process, then P is also a path in N'. However, there are three steps in this process which remove arcs from the network: steps *4(b)*, *5(b)* and *6(b)i*. Steps *4(b)* and *6(b)* are not of interest, as they remove edges between nodes of which at least one is not in V'. Step *5(b)* removes the arc (g,p) and (g,c) from E. If either of these arcs are in P, then so must the other. Now in case these arcs are removed during termination, this part of the path is replaced with a direct edge from g to c, creating a shorter path P' from u to v in N'. Hence $u \leq_{N'} v$. □

Note that the partial ordering of the network induces the lowest stable ancestor of a set of leaves. Therefore, if this lowest stable ancestor is not removed during termination, it should stay the lowest stable ancestor of this set. The following lemma shows that this is indeed the case for the lowest stable ancestor of a set of two leaves.

Lemma 4.1.2. *Let $N = (V,E)$ be a phylogenetic network on X and let $N' = (V',E')$ be the phylogenetic network on $X' = X \setminus \{x\}$ created by terminating a leaf $x \in X$ in N. Furthermore, let $S \subseteq X$ contain two leaves and let l and l' denote the lowest stable ancestor of S in N and N' respectively. If $x \notin S$ then $l = l'$.*

Proof. Assume that $x \notin S$ and denote $S = \{s_1, s_2\}$. Note that $V' \subseteq V$ and thus that $l' \in V$. From Lemma 4.1.1, $l' \leq_{N'} s_1$ and $l' \leq_{N'} s_2$ it follows that $l' \leq_N s_1$ and $l' \leq_N s_2$. Therefore, it holds that $l' \leq_N l$. If $l' = l$ then the statement holds. For the purpose of contradiction assume that $l' <_N l$. By Lemma 4.1.1, it then follows that $l' <_{N'} l$. But then l must be removed during the termination event as otherwise l would have to be the lowest stable ancestor of S in N'. There are three cases in which l might be removed: if l is the root node, a tree node or a reticulation node in N.

If l is a root node then it is not possible that $l' <_N l$.

If l is a tree node, then denote its two children with c_1 and c_2. Now l is only removed when all leaves below c_1 or c_2 are terminated. Without loss of generality, it can be assumed that the leaves below c_1 are terminated. Hence the leaves in S must be below c_2. Then $l <_N c_2$, $c_2 \leq_N s_1$ and $c_2 \leq_N s_2$. A contradiction with the fact that l is the lowest stable ancestor of S in N.

If l is a reticulation node then s_1 and s_2 are both below the single outgoing arc of l in N. Denote this arc with (l, c). Then $l <_N c$, $c \leq_N s_1$ and $c \leq_N s_2$. A contradiction with the fact that l is the lowest stable ancestor of S in N.

So none of these cases are possible. Hence it holds that $l = l'$. □

These last two lemmas show that the structure of phylogenetic networks can be removed but not altered when leaves are terminated. In the following lemma, these results are combined and translated to another structural property of these networks: its cut-arc sets.

Lemma 4.1.3. *Let $N = (V, E)$ be a phylogenetic network on X, let $\mathcal{T} = \mathcal{T}(N)$ be its trinet set and let $\Omega = \Omega(\mathcal{T})$ be its auxiliary graph. Furthermore, let $x, y, z \in X$ and assume that x and y are part of the same cut-arc set of which z is not a part of. Then the minimal cut-arc set of $T \in \mathcal{T}$ on $\{x, y, z\}$ is $\{x, y\}$.*

Proof. Assume that there exists a cut-arc $\mathbf{e} = (e_1, e_2)$ such that $x \leq_N \mathbf{e}$ and $y \leq_N \mathbf{e}$ but not $z \leq_N \mathbf{e}$. It follows that the lowest stable ancestor l of $\{x, y\}$ is below or equal to e_2 in N. It also follows that z is not below l in N.

Now let $T = (V_T, E_T)$ be the trinet on $\{x, y, z\}$ of N created through termination sequence $\{\epsilon_i\}$. From Lemma 4.1.2 it follows that $l \in V_T$, and from Lemma 4.1.1 it follows that x and y are also below l in T and that z is not below l in T. Therefore, l must be the head of a cut-arc. Now the cut-arc set corresponding to l is minimal and contains x and y but not z. □

4.2. BICONNECTED PHYLOGENETIC NETWORKS

Using the knowledge of how the structural properties of a phylogenetic network change when leaves are terminated, it can now be analysed how non-trivial cut-arcs arise in biconnected phylogenetic network during such termination events. The following lemma shows that a biconnected network can only become non-biconnected if a reticulation leaf is terminated.

Lemma 4.2.1. *Let $N = (V, E)$ be a biconnected phylogenetic network on X with $|X| \geq 2$. Let N' be the network created from N by terminating a leaf $x \in X$. If N' is not biconnected, then x is reticulation leaf.*

Proof. Let $N = (V, E)$ be a biconnected phylogenetic network on X. Also let $N' = (E', V')$ be the network created from N by terminating x. Now assume that N' is not biconnected. By definition, N' contains a non-trivial cut-arc $e = (e_1, e_2) \in E'$. Note that $e_1, e_2 \in V$ but not necessarily $e \in E$. By definition, for every node v below e_2, every path from the root ρ' of N' to v must pass through e. It follows that e_2 can not be a reticulation node in N'. Therefore, e_2 can not be a reticulation node in N either.

Now let v be a node below e_2 in N', then it follows from Lemma 4.1.1 that v is also below e_2 in N. Now it is possible to let v be a reticulation node in N, as otherwise e_2 would be the head of a non-trivial cut-arc in N. Next, let P and Q denote two distinct directed paths from the root ρ of N to v. Without loss of generality, it can be assumed that path P does not pass through e_2.

Now note that terminating any tree nodes will only shorten path P. For example, terminating a tree node P_i from the path will result in the nodes P_{i-1} and P_{i+1} to be connected. Moreover, this shortened path P still does not pass through e_2. Furthermore, note that the only moment a reticulation node is terminated is when its out-degree becomes zero, which happens only when a leaf node or a reticulation node below it is terminated. Hence, if x is not a reticulation leaf, no reticulation nodes will be terminated and the path P will still be a (possibly shorter) path in N' that still does not pass through e_2. It follows that e is not a cut-arc in N': a contradiction. Therefore, x must be a reticulation leaf. □

The next lemma shows that for every tree leaf x there exists a reticulation leaf x_r such that these two leaves will still be attached to the same component after any leaf other than x and x_r is terminated. Furthermore, this even holds if any number of leaves other than x and x_r are terminated. In particular, if all leaves but x, x_r and a third leaf y are terminated, then the network becomes a trinet. It follows that x and x_r must be contained in the minimal cut-arc set of this trinet.

Lemma 4.2.2. *Let $N = (V, E)$ be a biconnected phylogenetic network on X with $|X| \geq 4$. Denote the set of reticulation leaves of N with R. Let $x \in X \setminus R$ be a tree leaf. Then there exists a reticulation leaf $x_r \in R$ such that all exhibited trinets on $X' = \{x, x_r, y\}$ of N with $y \in X \setminus \{x, x_r\}$ have x, x_r in its minimal cut-arc set.*

Proof. It will be shown that letting x_r be a reticulation leaf below the parent of x fulfils this condition. Denote with p the parent of x. Then there exists a reticulation node below p. Let $v_r \in V$ be such a reticulation node, but without any reticulations below it. Since there are no reticulations below it and N does not contain any non-trivial cut-arcs, the arc leaving v_r must be a trivial cut-arc. Hence the single child of v_r is a leaf. Denote this leaf with x_r.

Now it will be shown that every exhibited trinet on $X' = \{x, x_r, y\}$ of N with $y \in X \setminus \{xr, x_r\}$ has x and x_r in its minimal cut-arc set. To this end, fix $y \in X \setminus \{x, x_r\}$ and let T be the trinet on $X = \{x, x_r.y\}$ created through termination sequence $\{\epsilon_i\}_{i=1}^{n}$. Denote with N_i the resulting network of applying the termination event ϵ_i to network N_{i-1}, where $N_0 = N$ and $N_n = T$.

It will be shown through induction that there exist two edge-disjoint directed paths in N_i from the root ρ_i of N_i to v_r, exactly one of which passes through the parent p of x. This clearly holds for N_0 as N is biconnected and v_r is a reticulation node. Now assume that for a fixed $1 \leq k < n$ there exist two edge-disjoint directed paths in N_k from the root ρ_k of N_k to v_r, exactly one of which passes through p. Now applying ϵ_k to N_k will terminate a leaf other than x and x_r. This event will not remove p, as p is above both x and x_r, nor will it remove v_r as it is above x_r. Therefore, each of the two previous paths may be shortened when one of the tree nodes it passes through is removed, but they will remain distinct as p is on exactly one of them. So indeed the induction hypothesis holds. Therefore, T contains two edge-disjoint directed paths from its root to v_r, exactly on which passes through p.

From this, it follows that x and x_r are attached to the same biconnected component in T. Now let S be the minimal cut-arc set of T. As $|S| \geq 2$ it follows that either x or x_r in S. As x and x_r are attached to the same component, it follows that then also x_r or x in S respectively. So indeed $\{x, x_r\} \subseteq S$. □

The reticulation leaf x_r in the proof above is also called a *limit leaf* of x. Note that a leaf can have multiple limit leaves. Using these lemmas, it will now be shown that the auxiliary graph of a biconnected level-2 phylogenetic network has only one minimal sink-set: the complete sink-set. Using Lemma 4.2.2 it will be shown in Lemma 4.2.3 that every tree leaf is connected to its limit leaf in the auxiliary graph and vice versa. Using Lemma 4.2.1 it will be shown, also in Lemma 4.2.3, that if there are two reticulation leaves these are connected to each-other too.

Lemma 4.2.3. *Let $N = (V, E)$ be a biconnected level-2 phylogenetic network on X with $|X| \geq 2$. Furthermore, let $\mathcal{T} = \mathcal{T}(N)$ be its trinet set and let $\Omega = \Omega(\mathcal{T})$ be its auxiliary graph. Then for every two leaves $x_1, x_2 \in X$ there exists a sequence of leaves $x_1 = y_1, y_2, \dots y_m = x_m$ such that (y_i, y_{i+1}) and $(y_{i+1}, y_i) \in \Omega$ for all $i \in (1, 2, \dots (m-1))$.*

Proof. As N is biconnected it contains at most one non-trivial biconnected component: the root component.

If the root component has no reticulations, then the only way for N to be biconnected is if $|X| = \{x_1, x_2\}$. But then $\mathcal{T} = \emptyset$ and thus for x_1, x_2 there exists no trinet such that x_1 or x_2 is not in its minimal cut-arc set. Therefore (x_1, x_2) and $(x_2, x_1) \in \Omega$.

Now assume that the root component has one or two reticulations. It will first be shown that for every tree leaf t there exists a reticulation leaf r such that (t, r) and $(r, t) \in \Omega$. To this end, fix a tree leaf t and let r be a reticulation leaf as in Lemma 4.2.2. Now let $X' = \{t, r, z\}$ where $z \in X \setminus \{t, r\}$ arbitrary and let $T \in \mathcal{T}$ be the trinet with leaf set X'. Then, according to this lemma, the minimal sink-set of T contains r and t. As this holds for every z, by definition (t, r) and $(r, t) \in \Omega$.

The proof is finished in case there is only one reticulation leaf, as every tree leaf is connected to this reticulation leaf.

Now assume that the root component has two reticulation leaves: r_1 and r_2. It remains to be shown that (r_1, r_2) and (r_2, r_1) in Ω. To this end, let $X' = \{r_1, r_2, z\}$ where $z \in X \setminus \{r_1, r_2\}$ arbitrary. Now let $T \in \mathcal{T}$ be the trinet with leaf set X'. Then T is created from N by terminating only tree leaves. From Theorem 4.2.1 it follows that T is biconnected. Therefore, by definition, X' is the only minimal cut-arc set of T. As this holds for every z, it follows that both (r_1, r_2) and $(r_2, r_1) \in \Omega$. Hence, the statement holds. □

Applying the above lemma to every leaf of a biconnected phylogenetic network gives the following corollary.

Corollary 4.2.4. *Let $N = (V, E)$ be a biconnected level-2 phylogenetic network on X with $|X| \geq 2$. Furthermore, let $\mathcal{T} = \mathcal{T}(N)$ be its trinet set and let $\Omega = \Omega(\mathcal{T})$ be its auxiliary graph. Then X is a minimal sink-set in Ω.*

Note that the limitation to level-2 phylogenetic networks comes from the last part of the proof of the previous lemma. We conjecture that this part can be generalized to hold for any level. In particular, by showing that there exists a directed graph on the reticulation leaves such that each two adjacent reticulation leaves are connected to each-other in the auxiliary graph.

Conjecture 4.2.5. Let $N = (V, E)$ be a biconnected phylogenetic network on X with $|X| \geq 2$ with $n \geq 2$ reticulation leaves. Furthermore, let $\mathscr{T} = \mathscr{T}(N)$ be its trinet set and let $\Omega = \Omega(\mathscr{T})$ be its auxiliary graph. Then for every two reticulation nodes r, s there exits a sequence of $m \leq n$ reticulation leaves $r = r_1, r_2, \ldots, r_m = s$ such that $(r_i, r_{i+1}) \in \Omega$ and $(r_{i+1}, r_i) \in \Omega$ for $i = i, \ldots, m-1$.

4.3. COLLAPSED NETWORKS

In order to extend Corollary 4.2.4 to any level-2 phylogenetic network, it needs to be analysed how the exhibited trinet set $\mathscr{T}$ of a network N translates to the exhibited trinet set $\mathscr{T}'$ of the network N' in which a cut-arc set has been collapsed. The following lemma shows that the $\mathscr{T}'$ can be deduced from $\mathscr{T}$.

Lemma 4.3.1. *Let N be a recoverable phylogenetic network on X, let $\mathscr{T} = \mathscr{T}(N)$ be its trinet set and let S be a minimal cut-arc set of N. Furthermore, let N^c be the network created from N by collapsing S to l and let $\mathscr{T}(N^c)$ be its trinet set. Lastly, let $\mathscr{T}^c$ denote the union of all collapsed version of the trinets in in $\mathscr{T}$ containing three leaves. Then $\mathscr{T}^c = \mathscr{T}(N^c)$.*

Proof. Denote the cut-arc corresponding to the S with e.

Let T^c be a trinet in $\mathscr{T}^c$. It will be shown that $T^c \in \mathscr{T}(N^c)$. If $l \notin T^c_X$, then clearly $T^c \in \mathscr{T}$. Furthermore, only nodes and edges below cut-arc e are removed from N to create N^c. As T^c does not contain leaves that are below e in N, it follows that $T^c \in \mathscr{T}(N^c)$. If $l \in T^c_X$, let $T \in \mathscr{T}$ denote the trinet from which T^c has been derived. It holds that T contains precisely one leaf s in S. As this is the only leaf that T contains that is below e, it follows that the trinet created from T by replacing s with l is exhibited by N^c. Hence $T^c \in \mathscr{T}(N^c)$.

It remains to be shown that $\mathscr{T}(N^c) \subseteq \mathscr{T}^c$. To this end, let T^c be a trinet in $\mathscr{T}(N^c)$. It will be shown that $T \in \mathscr{T}^c$. If $l \notin T^c_X$, then clearly $T^c \in \mathscr{T}$. As T has no leaves that are in S, it follows that T is added to $\mathscr{T}^c$. Hence $T^c \in \mathscr{T}^c$. If $l \in T^c_X$, then there exists a leaf $s \in S$ such that the trinet T created from T^c by replacing l with s is exhibited by N. As l is the only leaf in T^c that is below cut-arc e in $\overline{N}$, it follows that l is replaced by y to create $T"$ which is added to $\mathscr{T}^c$. Due to the recoverability of N it holds that $T" = T$. Hence it follows that $T \in \mathscr{T}^c$.

It follows that $\mathscr{T}(N^c) = \mathscr{T}^c$. □

It can be shown analogously that this also holds for binet sets.

Corollary 4.3.2. *Let N be a recoverable phylogenetic network on X, let $\mathscr{B} = \mathscr{B}(N)$ be its binet set and let S be a minimal cut-arc set of N. Furthermore, let N^c be the network created from N by collapsing S to l and let $\mathscr{B}(N^c)$ be its binet set. Lastly, let $\mathscr{B}^c$ denote the union of all collapsed version of the binets in in $\mathscr{T}$ containing two leaves. Then $\mathscr{B}^c = \mathscr{B}(N^c)$.*

4.4. CUT-ARC SETS AND SINK-SETS

Section 4.2 shows how the non-trivial cut-arc sets of biconnected phylogenetic network convert to the minimal cut-arc sets the networks exhibited trinets. As the auxiliary graph is derived from these minimal cut-arc sets, it should now be possible to prove how these non-trivial cut-arc sets convert to sink-sets in the auxiliary graph.

Lemma 4.4.1. *Let $N = (V,E)$ be a phylogenetic network on X, let $\mathcal{T} = \mathcal{T}(N)$ be its trinet set and let $\Omega = \Omega(\mathcal{T})$ be its auxiliary graph. If $S \subseteq X$ is a non-trivial cut-arc set of N then S is a sink-set in Ω.*

Proof. Assume that $S \subseteq X$ is a non-trivial cut-arc set of N. To show that S is a sink-set in Ω it needs to be shown that there are no edges (s,t) in Ω such that $s \in S$ and $t \notin S$.

To this end, fix $s \in S$ and $t \notin S$ arbitrarily. It must be shown that $(s,t) \notin \Omega$. By definition, this means that it must be shown that there exists a leaf $z \in X \setminus \{s,t\}$ such that the minimal cut-arc set of the trinet $T \in \mathcal{T}$ with leaf set $\{s,t,z\}$ does not contain t. It will be shown that any $z \in S \setminus \{s\}$ suffices.

Fix $z \in S \setminus \{s\}$ arbitrarily, let $X' = \{s,t,z\}$ and let T be the trinet on X' of N. Note that such a z exists as S is a non-trivial cut-arc set. From Lemma 4.1.3 it follows that $t \notin S$. From this, it follows that $(s,t) \notin \Omega$ for any $t \notin S$. Therefore, S is a sink-set in Ω. □

The following theorem shows that the minimal sink-sets of an auxiliary graph are minimal cut-arc sets in the corresponding phylogenetic network. This is done using induction on the number of non-trivial biconnected components of the network.

Theorem 4.4.2. *Let N be a level-2 phylogenetic network on X and let $\mathcal{T} = \mathcal{T}(N)$ be its trinet set and let $\Omega = \Omega(\mathcal{T})$ be its auxiliary graph. If $S \subseteq X$ is a minimal sink-set in Ω, then S is a minimal cut-arc set of N.*

Proof. Assume that $S \subseteq X$ is a minimal sink-set in Ω. It will first be proven that S is a non-trivial cut-arc set via induction on the number of non-trivial biconnected components n of N.

Let $n = 1$, then N is biconnected. By definition, X is a minimal cut-arc set. From Corollary 4.2.4 it follows that X is also a minimal sink-set. So the induction basis holds.

Now assume the hypothesis holds for $n \leq k$. It will be shown that it also holds for $n = k+1$. To this end, suppose that N has $k+1$ non-trivial biconnected components denoted with $B_1, \ldots B_{k+1}$.

Firstly, assume that there exists a leaf biconnected component L of N to which no leaves of S are attached. Denote the leaves attached to L with X_L. Now let N^c denote the network obtained after collapsing X_L to l in N. Moreover, denote its trinet set with $\mathcal{T}^c$ and its auxiliary graph with Ω^c. Now it holds that this network N^c satisfies the properties of the induction hypothesis, and thus every minimal sink-set of Ω^c is a non-trivial cut-arc set of N^c.

First assume that $S \cap X_L = \emptyset$, it will now be shown that S is a minimal sink-set in Ω^c. To this end, let $s \in S$ and $t \notin S$. As S is a sink-set in Ω it follows that $(s,t) \notin \Omega$. Hence there exists a trinet $T \in \mathcal{T}$ on $\{s,t,x\}$ with $x \in X$ for which t is not in its minimal cut-arc set. It will be shown that such a trinet also exists in Ω^c. If $x \in X_L$, take the trinet $T^c \in \mathcal{T}^c$ on $\{s,t,l\}$. Now T^c is the same as T, except that x is named l in T^c. If $x \notin X_L$ simply take the trinet $T^c \in \mathscr{T}^c$ on $\{s,t,x\}$, which is the same trinet as T. Thus in both cases t is not in the minimal cut-arc set of T. Therefore, $(s,t) \notin \Omega^c$ and thus S is a sink-set in Ω^c.

Next, it will be shown that S is also a *minimal* sink-set in Ω^c. This will be done by showing that $(s_1,s_2) \in \Omega^c$ if $(s_1,s_2) \in \Omega$ for all $s_1,s_2 \in S$. If this holds, then the edges between leaves in S in Ω^c are also in Ω. For the purpose of contradiction, fix $s_1,s_2 \in S$ and assume that $(s_1,s_2) \in \Omega$ but $(s_1,s_2) \notin \Omega^c$. By definition, there must exist a trinet $T^c \in \mathscr{T}^c$ containing s_1 and s_2 such that s_2 is not in this trinet's minimal cut-arc set. Denote the third leaf of this trinet with x. If $x = l$, take the trinet $T \in \mathscr{T}^c$ on $\{s,t,z\}$ with z any leaf in X_L. Now T is the same as T^c except that x is named z in T. If $x \neq l$ simply take the trinet $T \in \mathscr{T}$ on $\{s,t,x\}$, which is the same as T^c. So in both cases s_2 is not in the minimal cut-arc set of T. Therefore $(s_1,s_2) \notin S$: a contradiction. So $(s_1,s_2) \in \Omega^c$. Thus indeed, S is also a minimal sink-set in Ω^c.

Secondly, assume that there exists at least one leaf biconnected component to which a leaf of S is attached. Denote this biconnected component with L and the leaves that are attached to it with X_L. Then X_L is a minimal cut-arc set in N and by Theorem 4.4.1 a sink-set in Ω. It will be shown that X_L is a minimal sink-set in Ω. From this it will follow that $S = X_L$ and thus that S is a minimal cut-arc set in N.

Let N^r be the exhibited network on X_L. Also denote its trinet set with $\mathscr{T}^r$ and its auxiliary graph with Ω^r. Now N^r is as in Lemma 4.2.3, hence for every node $x \in X_L$ there exists a node $y \in X_L$ such that (x,y) and $(y,x) \in \Omega^r$. Now let $T \in \mathscr{T}$ be a trinet on $\{x,y,z\}$, where $z \in X$ arbitrary. If $z \in X_L$ then clearly $T \in \mathscr{T}^r$ and, by definition of Ω^r, both x and y are in the minimal cut-arc set of T. If $z \notin X_L$, then x and y are in a cut-arc set that z is not in. From Lemma 4.1.3 it follows that $\{x,y\}$ is the minimal cut-arc set of T. Again, both x and y are in the minimal cut-arc set of T. As this holds for all trinets on x and y it follows that (x,y) and $(y,x) \in \Omega$. So Ω contains at least the same edges in X_L as Ω^r. As X_L is a minimal-sink set in Ω^r and a sink-set in Ω, it follows that it must be a minimal sink-set in Ω.

To sum up, X_L and S are minimal sink-sets in Ω and $X_L \cap S \neq \emptyset$. It follows that $S = X_L$. As X_L is a minimal cut-arc set, it follows that so is S.

Finally, it remains to show that S is a minimal cut-arc set in N. For the purpose of contradiction, assume that S is not minimal and hence strictly contains a non-trivial cut-arc set S'. From Theorem 4.4.1 it follows that S' is a sink-set in Ω. Hence $S' \subset S$: a contradiction.

So indeed, S is a minimal cut-arc set in N. □

The next theorem shows that the converse is also true, which follows directly from the inclusive properties of cut-arc sets and sink-sets.

Theorem 4.4.3 (Minimal cut-arc set implies minimal sink-set)**.** *Let N be a level-2 phylogenetic network, let $\mathscr{T} = \mathscr{T}(N)$ be its set of trinets and let $\Omega = \Omega(\mathscr{T})$ be its auxiliary graph. If $S \subseteq X$ is a minimal cut-arc set in N, then S is a minimal sink-set in Ω.*

Proof. Let S be a minimal cut-arc set in N. Then from Lemma 4.4.1 it follows that S is a sink-set in Ω. It will now be shown that S is a minimal sink-set. For the purpose of contradiction, assume that S is not minimal, and thus strictly contains a sink-set in Ω and hence also a minimal sink-set S'. By Theorem 4.4.2, S' is in fact a minimal cut-arc set. Since $S' \subseteq S$, this contradicts the assumption that X is a minimal cut-arc set. □

By combining Theorems 4.4.2 and 4.4.3 we obtain the following corollary.

Corollary 4.4.4. *Let N be a level-2 phylogenetic network, let $\mathscr{T} = \mathscr{T}(N)$ be its set of trinets and let $\Omega = \Omega(\mathscr{T})$ be its auxiliary graph. Then $S \subseteq X$ is a minimal cut-arc set in N if and only if S is a minimal sink-set in Ω.*

Note that Huber, Moulton, and Wu (2019) have proven a similar theorem for 2-terminal rooted phylogenetic networks through hierarchies of SN-sets, closed sets and minimal closed sets. 2-terminal rooted phylogenetic networks are defined as rooted phylogenetic networks containing at most two reticulation *leaves* per biconnected component. Closed sets are defined as sets of leaves S for which the leaves below the lowest stable ancestor of S are exactly the leaves in S.

5 ALGORITHM

`TriL2Net` takes in a set of trinets $\mathscr{T}_1$ on a set of species X_1 with $|X_1| \geq 3$. It then computes a level-2 recoverable phylogenetic network N_1 from this trinet set. It does this using two main functions: `collapse` and `expand`.

Given $\mathscr{T}^1$, `collapse` identifies a non-singleton subset Y_1 of X_1 that will be a minimal cut-arc set in the output phylogenetic network N_1. Next, it restricts $\mathscr{T}_1$ to the leaves in Y_1 to create the restricted trinet set $\mathscr{T}_1^r$. From this set $\mathscr{T}_1^r$, it constructs a biconnected level-2 phylogenetic network M_1 on Y_1. Then, it creates a new trinet-set $\mathscr{T}_2$ on species X_2 from $\mathscr{T}_1$ by collapsing the leaves in Y_1 to a single leaf c_2. Here, $X_2 = (X \setminus Y_1) \cup \{c_1\}$. If $|X_2| \geq 3$, then `collapse` constructs a biconnected level-2 phylogenetic network M_2 on a non-singleton subset of Y_2 from $\mathscr{T}_2$ and continues like this until $|X_i| = 1$ for some iteration i.

The `expand` function takes this last phylogenetic network M_i and denotes it with N_i. Next, it creates the expansion N_{i-1} of N_i by replacing the leaf c_i in this phylogenetic network with the phylogenetic network M_{i-1}. It continues like this until the phylogenetic network N_1 is created, which is the output of the algorithm.

Note that each phylogenetic network N_i is the result of running `Tril2Net` on the trinet set $\mathscr{T}_i$.

The `expand` function is relatively straightforward, but the `collapse` function consists of three complex steps: finding the minimal cut-arc set, constructing a biconnected level-2 phylogenetic network for each minimal cut-arc set, and collapsing the trinet set. These steps will be explained in more detail below. In these steps, variables with a hat above them denote the output of a part of the algorithm. Similarly, if there is a desired output of the algorithm, it is denoted with a variable with a line above it.

For example, generators are denoted with $\mathscr{G}$. If the algorithm decides that the output phylogenetic network should be based on a specific generator, this is denoted with $\hat{\mathscr{G}}$. Furthermore, if the input of the algorithm is induced by a network, then the generator of that phylogenetic network the desired output and is denoted with $\overline{\mathscr{G}}$. Clearly if $\overline{\mathscr{G}}$ is known then the goal is to have $\hat{\mathscr{G}} = \overline{\mathscr{G}}$. Furthermore, statistics derived from the trinet set are denoted using subscripted p's q's and r's. For example, the portion of trinets that have k reticulations is denoted with p_k.

Lastly, the subscripts denoting the iteration are left out in the following section as they are unnecessary. That is, let $\mathscr{T}$ will denote the trinet set on species X of a certain iteration j and the first step consists of finding a minimal cut-arc sets $Y \subseteq X$ of the output phylogenetic network $\hat{N}$.

5.1. FINDING THE MINIMAL CUT-ARC SETS

The minimal cut-arc sets are assumed to be the minimal sink-sets in the auxiliary graph $\Omega = \Omega(\mathscr{T})$. If $\mathscr{T}$ is an induced trinet set, then it follows from Corollary 4.2.4 that Ω must at least one non-trivial sink-set. However, if $\mathscr{T}$ is not induced, it can happen that Ω contains no arcs, resulting in each leaf being a trivial sink-set. A way to tackle this is using the augmented auxiliary graph from Oldman et al. (2016).

Definition 5.1.1 (Augmented auxiliary graph)**.** Let $\mathscr{T}$ be a trinet set on a set of species X. Then the i^{th} *augmentation of the auxiliary graph* $\Omega(\mathscr{T})$ is the graph $\Omega_i(\mathscr{T})$ with vertex set $\mathscr{T}_X$ and (directed) edge set consisting of those $(x, y) \in (X \times X)$ such that at most i trinets $T \in \mathscr{T}$ with $\{x, y\} \subset T_X$ do not have y in their minimal cut-arc set.

The goal of this augmented graph is to find sets of species which best resembles minimal cut-arc sets. This can be done in two ways. The first method, conform Oldman et al. (2016), lets $\hat{i}_1$ be the minimal value of i for which $\Omega_i(\mathscr{T})$ contains at least one arc and finds the strongly connected components of $\Omega_{\hat{i}_1}(\mathscr{T})$. Then, if there exist strongly connected components that are also minimal sink-sets it returns one of minimal size. If no such a component exists, it takes a node p in $\Omega_i(\mathscr{T})$ which is the parent of at least one, but a minimal number, of other nodes c_i. It then returns the union of p and its children c_i.

The second method lets $\hat{i}_2$ be the minimal value of i for which $\Omega_i(\mathscr{T})$ contains at least one strongly connected component that is also a minimal sink-set. It then returns the smallest of such sets.

To aid in these processes, the weighted auxiliary graph $\boldsymbol{\Omega}$ and the condensed graph Ω^*_w are introduced.

Definition 5.1.2 (Weighted auxiliary graph)**.** Let $\mathscr{T}$ be a trinet set on a set of species X. Then the *weighted auxiliary graph* is the complete arc-weighted digraph $\mathbf{\Omega}$ with vertex set $\mathscr{T}_X$ and arc weights:

$$\begin{aligned}\boldsymbol{\omega}((x,y)) = {} & \text{the number of trinets } T \in \mathscr{T} \text{ with } \{x,y\} \subset T_X \\ & \text{that do no have } y \text{ in their minimal cut-arc set.}\end{aligned}$$

Definition 5.1.3 (Condensed graph)**.** Let Ω_w be an augmented auxiliary graph of a trinet set $\mathscr{T}$ on a set of species X and let S_i, $i = 1,2,\dots$, be the strongly connected components of Ω_w. Then the *condensed graph* is the digraph Ω^*_w with vertex set $\{S_1, S_2, \dots\}$. This graph contains an arc (S_i, S_j) if and only if there exists nodes $v \in S_i$ and $w \in S_j$ such that (v, w) is an arc in Ω_w.

Note that these graphs are the same for all network set iterating methods if and only if the trinet set is induced. Moreover, in case $\mathscr{T}$ contains multiple trinets on the same leaf set and the weighted average method is used, it is possible that the weights of the weighted auxiliary graph are not integer.

The pseudo-code for these two methods can be seen in Algorithm 1 and Algorithm 2 respectively. The following two lemmas show that both algorithms are correct if $\mathscr{T}$ is induced by a recoverable level-2 phylogenetic network $\overline{N}$: that is, they both return one of the minimal cut-arc sets of $\overline{N}$.

Lemma 5.1.4. *Let $\mathscr{T}$ be a non-empty trinet set on n leaves containing t trinets. Then Algorithm 1 computes a minimal sink-sets of $\Omega_{\hat{i}_1}$ in $O(t + n^2)$.*

Furthermore, in case $\mathscr{T}$ is induced by a recoverable level-2 phylogenetic network $\overline{N}$, then the set it returns is a minimal cut-arc set of $\overline{N}$.

Proof. First, $\boldsymbol{\omega}$ can be computed in $O(t)$ by iterating over all t trinets in $\mathscr{T}$. Finding $\hat{i}_1$ can be done by iterating over all arcs in $\mathbf{\Omega}$. Note that $\hat{i}_1$ exists as Ω_n is a complete graph. Next, $\Omega_{\hat{i}_1}$ can be computed by, again, iterating over all arcs in $\mathbf{\Omega}$. The strongly connected components of $\Omega_{\hat{i}_1}$ and hence the condensed graph $\Omega^*_{\hat{i}_1}$ can be computed using Tarjan's algorithm in $O(n^2)$ (Tarjan 1972). Note that there are at most n strongly connected components. Dividing the set of nodes u of $\Omega^*_{\hat{i}_1}$ in leaves and internal nodes can be done by iterating over the at most n nodes. Checking if there is a non-singleton set π_u and finding such a set of minimal size can be done by iterating over the at most n nodes too. Hence lines 1-10 have time-complexity $O(t + n^2)$. Next, note that π'_u can be constructed using breadth first search in $O(n^2)$. Finding a minimum size π'_u can be done in $O(n)$. Hence the full algorithm has time-complexity $O(t + n^2)$.

Next, it must be shown that the algorithm returns a minimal sink-set of $\Omega_{\hat{i}_1}$. Clearly, a strongly connected component with out-degree zero containing at least two elements is a minimal sink-set. It follows that if the algorithm terminates in line 9, then it returns a minimal sink-set π_u of $\Omega_{\hat{i}_1}$. If, on the other hand, it terminates in line 15, then it returns the set π'_u containing all its descendant. Hence, it returns a sink-set. This sink-set must be minimal, as otherwise there exists a sink-set π'_v such that $\pi'_v \subset \pi'_u$, a contradiction with the fact that $|\pi'_u| \leq |\pi'_v|$.

Lastly, assume that $\mathcal{T}$ is induced by a recoverable level-2 phylogenetic network $\overline{N}$ and let C_i be the minimal cut-arc sets of $\overline{N}$. By Corollary 4.2.4, it follows that the C_i are also minimal sink-sets in Ω. Hence Ω_0 contains an arc and $\hat{i}_1 = 0$. It follows that Algorithm 1 return a minimal sink-set of Ω_0, which is a minimal cut-arc set of $\overline{N}$. □

Algorithm 1: minimal cut-arc sets using $\hat{i}_1$

```
  Input:
  - Trinet set $\mathcal{T}$
  Output:
  - Set of minimal sink-sets $MSS$
1 $\omega \leftarrow$ weighted auxiliary graph of $\mathcal{T}$
2 $\hat{i}_1 \leftarrow$ minimum of weights of the arcs in $\omega$
3 $\Omega_{\hat{i}_1} \leftarrow \hat{i}^{th}$ augmentation of $\Omega$
4 $\Omega^*_{\hat{i}_1} \leftarrow$ condensed graph of $\Omega_{\hat{i}_1}$
5 $\pi_u \leftarrow$ set of nodes contained in strongly connected component $u \in \Omega^*_{\hat{i}_1}$
6 $A \leftarrow$ set of leaves of $\Omega^*_{\hat{i}_1}$
7 $B \leftarrow$ set of internal nodes of $\Omega^*_{\hat{i}_1}$
8 if there exists $u \in A$ such that $|\pi_u| \geq 2$ then
9     return $\pi_u$ such that $|\pi_u| \geq 2$ and $|\pi_u| \leq |\pi_v|$ for all $v \in A$
10 end
11 else
12     for each $u$ in $B$ do
13         $\pi'_u \leftarrow$ union of $\pi_u$ and $\pi_v$ for all descendents $v$ of $u$
14     end
15     return $\pi'_u$ such that $|\pi'_u| \geq 2$ and $|\pi'_u| \leq |\pi'_v|$ for all $v \in B$
16 end
```

Lemma 5.1.5. *Let $\mathcal{T}$ be a non-empty trinet set on n leaves containing t trinets. Then Algorithm 2 computes a minimal sink-set of $\Omega_{\hat{i}_2}$ in $O(t + n^4)$.*

Furthermore, in case $\mathcal{T}$ is induced by a recoverable level-2 phylogenetic network $\overline{N}$, then the minimal-sink set it returns is a minimal cut-arc set of $\overline{N}$.

Proof. First, ω can be computed in $O(t)$ by iterating over all trinets in $\mathcal{T}$. Sorting the weights can be done using quick-sort with time-complexity $O(n^2)$ (Musser 1997). As ω contains $n(n-1)$ arcs there are at most $n(n-1)$ weights and hence the for-loop in lines 3-10 is run at most $n(n-1)$ times. Computing the w^{th} augmentation of $\Omega(\mathcal{T})$ can be done by iterating over all $n(n-1)$ arcs. Finding the strongly connected components and hence the condensed graph Ω^*_w of Ω_w can be done using Tarjan's algorithm in $O(n^2)$ (Tarjan 1972). Finding the leaves can be done by iterating over the at most n nodes of Ω^*_w. Checking if there is a non-singleton set π_u and finding such a set of minimal size can be done by iterating over the at most n nodes, too. Hence the algorithm has time-complexity $O(t + n^4)$.

Next, it must be shown that the algorithm returns a minimal sink-set of $\Omega_{\hat{i}_2}$. Clearly, if the algorithm terminates in line 9, then it returns a strongly connected component of Ω_w containing at least two elements that has no outgoing arcs. Such a set is a minimal-sink set of Ω_w. As the weights are iterated over in ascending order, it follows that if $\hat{i}_2$ exists then the algorithm returns a minimal sink-set of $\Omega_{\hat{i}_2}$. Now note that for the maximum weight w_{max} of an arc in $\boldsymbol{\omega}$ that $\Omega_{w_{max}}$ is a full graph. Hence $\mathscr{T}_X$ is a minimal sink-set in $\Omega_{w_{max}}$. Therefore, $\hat{i}_2 \leq w_{max}$.

Lastly, assume that $\mathscr{T}$ is induced by a recoverable level-2 phylogenetic network $\overline{N}$ and let C_i be the minimal cut-arc sets of $\overline{N}$. By Corollary 4.2.4, it follows that the C_i are also minimal sink-sets in Ω_0. As a minimal sink-set is a strongly connected component, it follows that $\hat{i}_2 = 0$. It follows that Algorithm 2 returns a minimal sink-set of Ω_0, which is a minimal cut-arc sets of $\overline{N}$. □

Algorithm 2: minimal cut-arc sets using $\hat{i}_2$

Input:
- Trinet set $\mathscr{T}$

Output:
- Set of minimal sink-sets C

```
1  $\boldsymbol{\omega} \leftarrow$ weighted auxiliary graph of $\mathscr{T}$
2  $W \leftarrow$ ascending lists of weights of the arcs in $\boldsymbol{\omega}$
3  for $w$ in $W$ do
4      $\Omega_w \leftarrow w^{th}$ augmentation of $\Omega(\mathscr{T})$
5      $\Omega^*_w \leftarrow$ condensed graph of $\Omega_w$
6      $\pi_u \leftarrow$ set of nodes contained in strongly connected component $u \in \Omega^*_w$
7      $A \leftarrow$ set of leaves of $\Omega^*_{\hat{i}_1}$
8      if there exists $u \in A$ such that $|\pi_u| \geq 2$ then
9          return $\pi_u$ such that $|\pi_u| \geq 2$ and $|\pi_u| \leq |\pi_v|$ for all $v \in A$
10     end
11 end
```

5.2. CONSTRUCTING A BICONNECTED LEVEL-2 NETWORK

Let $Y \subseteq X$ denote one of the cut-arc sets found in the previous section. The next step is to compute a biconnected level-2 phylogenetic network $\hat{M}$ on Y from the input trinet $\mathscr{T}$. Clearly, not all trinets in $\mathscr{T}$ contain information on Y. Hence, a new set $\mathscr{T}^r$ is constructed from $\mathscr{T}$ by discarding all trinets T that do not have all leaves in Y. This procedure is called *restricting* $\mathscr{T}$ to Y.

In case $\mathscr{T}$ is induced by a phylogenetic network $\overline{N}$, then, by Lemma 5.1.4 or Lemma 5.1.5, Y is a minimal cut-arc set of $\overline{N}$. The next lemma shows that $\mathscr{T}^r$ is induced by $\overline{M}$ if $\overline{M}$ is the restriction of $\overline{N}$ to Y.

Lemma 5.2.1. *Let $\mathscr{T}$ be a trinet set containing at most t trinets. Then the restriction $\mathscr{T}^r$ of $\mathscr{T}$ to $Y \subseteq X$ can be computed with time-complexity $O(t)$.*

Furthermore, let $\overline{N}$ be a recoverable phylogenetic network on X, let Y be a minimal cut-arc set of $\overline{N}$ and let $\overline{M}$ be the restriction of $\overline{N}$ to Y. If $\mathscr{T}$ is induced by $\overline{N}$ then it holds that $\mathscr{T}^r = \mathscr{T}\left(\overline{M}\right)$.

5

Proof. The restriction $\mathscr{T}^r$ of $\mathscr{T}$ to Y can be computed by iterating over all t trinets T in $\mathscr{T}$ and checking for each T if $T_X \subseteq Y$. As T_X contains at most three leaves, this check can be done with constant time-complexity. It follows that the time-complexity of restricting is $O(t)$.

Now for the second part, let $\overline{N}$ be a recoverable phylogenetic network on X, let Y be a minimal cut-arc set of $\overline{N}$ and let $\overline{M}$ be the restriction of $\overline{N}$ to Y.

First, let $T^r \in \mathscr{T}^r$ be a trinet. It will be shown that $T^r \in \mathscr{T}(\overline{M})$. To this end, let $T \in \mathscr{T}$ denote a trinet from which T^r has been derived. As T contains three leaves, it follows that $T^r_X \subseteq Y$. Hence T^r is exhibited by $\overline{M}$. It follows that $T^r \in \mathscr{T}(\overline{M})$.

Secondly, let $T^r \in \mathscr{T}(\overline{M})$ be a trinet. It will be shown that $T^r \in \mathscr{T}^r$. Clearly $T^r_X \subseteq Y$. Hence T^r is exhibited by $\overline{N}$. As T^r only contains leaves in Y, none of its leaves will be terminated during the restriction. Hence $T^r \in \mathscr{T}^r$.

It follows that $\mathscr{T}^r = \mathscr{T}(\overline{M})$. □

Now $\hat{M}$ is constructed from $\mathscr{T}^r$ in several steps. Firstly, it is determined what the strict level $\hat{k}$ of $\hat{M}$ will be. Secondly, a generator $\hat{\mathscr{G}}$ of this level is chosen that $\hat{M}$ will be based on. Thirdly, the leaves in Y are partitioned into groups $\hat{\eta}_I$ that correspond to sets of symmetric sides I of $\hat{\mathscr{G}}$. These groups are then partitioned further into groups $\hat{\theta}_s$ that correspond to the sides s of I. Next, each of these groups is ordered, resulting in an ordered list of leaves $\hat{\sigma}_s$ for each side s. Lastly, if $\hat{\mathscr{G}}$ is generator $\mathscr{G}^{2c}$ of Figure 3.2, then an ordering ψ_I of the sides s in I for each set of symmetric sides must be determined. For each of these steps it is shown what the worst-case time-complexity is. Furthermore, in case $\mathscr{T}^r$ is induced by a network $\overline{M}$, then it is shown that $\hat{M} = \overline{M}$.

In the remainder of this section the superscripts ^{r} are left out for readability. This is possible, as this section is only concerned with restricted trinet sets.

5.2.1. NUMBER OF RETICULATIONS

First, the number of reticulations $\hat{k}$ of the biconnected level-2 phylogenetic network $\hat{M}$ must be chosen. To this end, let:

$$p_k, \tag{5.1}$$

be the fraction of trinets in $\mathscr{T}$ which are strictly level-k. Note that here $\mathscr{T}$ is the trinet set restricted to Y, as the superscript ^{r} have been dropped. Furthermore, let $th_k(m)$ be thresholds for each value of k. These thresholds may depend on the number of elements m of Y. Now $\hat{k}$ is chosen to be the highest value of k for which the fraction p_k breaches the corresponding threshold $th_k(m)$. More precisely:

$$\hat{k} = \max\{k \in 0,1,2 : p_k \geq th_k(m)\}. \tag{5.2}$$

This threshold is a heuristic for limiting the effect of noise in the data set. The threshold values of the algorithm are:

$$\begin{aligned} th_0(m) &= 0, \\ th_1(m) &= \begin{cases} 0.5 \cdot \dfrac{\binom{m-1}{2}}{\binom{m}{3}}, & m = 2 \\ 0, & m \geq 3 \end{cases}, \\ th_2(m) &= 0.5 \cdot \min\left\{ \dfrac{\binom{m-1}{2}}{\binom{m}{3}}, \dfrac{m-2}{\binom{m}{3}} \right\}. \end{aligned} \tag{5.3}$$

These are chosen such that the level for $m \geq 3$ is at least one. It will now be shown that this method of computing the level $\hat{k}$ of $\hat{N}$ is correct.

Lemma 5.2.2. *Let $\mathscr{T}$ be a non-empty trinet set on m leaves containing t trinets. Then $\hat{k}$ from Equation (5.2) with threshold defined in Equation (5.3) can be computed with time-complexity $O(t)$.*

Furthermore, if $\mathscr{T}$ is induced by a biconnected level-2 network $\overline{M}$, then $\hat{k} = \overline{k}$.

Proof. Clearly, all values $p_k(\mathscr{T})$ can be computed with time-complexity $O(t)$ by iterating over the all t trinets. Furthermore, all $th_k(m)$ can be computed in constant time. Therefore, $\hat{k}$ can be computed with time-complexity $O(t)$.

Next, assume that $\mathscr{T}$ is induced by a biconnected level-2 network $\overline{M}$. It will be shown that $\hat{k}$ equals the number of reticulations $\overline{k}$ of $\overline{M}$. Then $\mathscr{T}$ contains the following fraction of trinets:

$\overline{k}$	0	1	2	
$\lvert\overline{g}_{CR}\rvert$	0	1	1	2
$p_{\overline{k}}(s)$	1	$\frac{\binom{m-1}{2}}{\binom{m}{3}}$	$\frac{\binom{m-1}{2}}{\binom{m}{3}}$	$\frac{m-2}{\binom{m}{3}}$

Here $\overline{g}_{CR}$ is the set of all crucial reticulation sides of generator $\overline{g}$. For example, if $\overline{M}$ is a network with two reticulations and one crucial reticulation side, then the portion of trinets in $\mathscr{T}$ that is level two is:

$$\frac{m-1}{\binom{m}{3}}.$$

The formulas in this table follow directly from the fact that an exhibited trinet T of a biconnected phylogenetic network N only has the same amount of reticulations if it contains all reticulation leaves of N. Hence it clearly holds that:

$$p_k \geq th_k(m),$$

for $k = \overline{k}$. Therefore, the threshold $th_{\overline{k}}(m)$ is breached. Furthermore, a trinet in $\mathscr{T}$ can not contain more reticulations than $\overline{k}$. Therefore, no threshold $th_k(m)$ with $k \geq \overline{k}$ is breached. It follows that $th_{\overline{k}}(m)$ is the highest threshold that is breached. Hence $\hat{k} = \overline{k}$. □

5.2.2. GENERATOR

Secondly, a generator $\hat{\mathscr{G}}$ of level $\hat{k}$ must be chosen for $\hat{M}$. Let

$$p_{\mathscr{G}}, \tag{5.4}$$

be the fraction of trinets in $\mathscr{T}$ that have generator $\mathscr{G}$. Then the generator $\hat{\mathscr{G}}$ of $\hat{M}$ is chosen to be any of the generators which have the highest percentage of all level $\hat{k}$ generators. That is:

$$\hat{\mathscr{G}} \in \arg\max_{\mathscr{G}\in\mathscr{G}^k} \{p_{\mathscr{G}}\}. \tag{5.5}$$

It will now be shown that this method of computing $\hat{\mathscr{G}}$ is correct.

Lemma 5.2.3. *Let $\mathscr{T}$ be a non-empty trinet set on m leaves containing t trinets and let $\hat{k} \in \{0,1,2\}$ be an integer. Then $\hat{\mathscr{G}}$ from (5.5) can be computed in $O(t)$.*

Furthermore, if $\mathscr{T}$ is induced by a biconnected strictly level-$\hat{k}$ phylogenetic network $\overline{M}$ on m species, then $\hat{\mathscr{G}} = \overline{\mathscr{G}}$.

Proof. Clearly, all values $p_{\mathscr{G}}$ can be computed with time-complexity $O(t)$ by iterating over all t trinets. Therefore, $\hat{\mathscr{G}}$ can be computed with time-complexity $O(t)$.

Next, assume that $\mathscr{T}$ is induced by a biconnected strictly level-$\hat{k}$ phylogenetic network $\overline{M}$ on m species. It will be shown that $\hat{\mathscr{G}}$ is the generator of $\overline{M}$. To this end, let $\overline{\mathscr{G}}$ be the generator of $\overline{M}$. Then $\mathscr{T}$ contains precisely:

$$\binom{s - |\overline{\mathscr{G}}_{CR}|}{3 - |\overline{\mathscr{G}}_{CR}|},$$

trinets of level-$\hat{k}$. Here $\overline{\mathscr{G}}_{CR}$ is the set of all crucial reticulation sides of $\overline{\mathscr{G}}$. Additionally, all of these trinets must have generator $\overline{\mathscr{G}}$. Hence the only $p_{\mathscr{G}}$ which is strictly positive is $p_{\overline{\mathscr{G}}}$. It follows that $\hat{\mathscr{G}} = \overline{\mathscr{G}}$. □

5.2.3. GROUP OF LEAVES PER SET OF SYMMETRIC SIDES

For each set of symmetric sides I of this generator $\hat{\mathscr{G}}$, it must be determined which leaves are attached to it. To this end, let:

$$p_{x,I}, \tag{5.6}$$

be the portion of trinets in $\mathscr{T}$ that have leaf $x \in Y$ on one of the sides in the set of symmetric sides I of generator $\hat{\mathscr{G}}$. Now remember that there are different types of sides: all sides $\hat{\mathscr{G}}_S$; optional sides $\hat{\mathscr{G}}_O$; crucial sides $\hat{\mathscr{G}}_C$; crucial reticulation sides $\hat{\mathscr{G}}_{CR}$; and crucial edge sides $\hat{\mathscr{G}}_{CE}$. The crucial reticulation sides must be assigned exactly one leaf each and the crucial edge sides must be assigned at least one leaf each. The optional sides do not impose any constraints on the partitioning of the leaves. This can be summarized as follows:

$$\begin{aligned} &\sum_{x \in s} \gamma_{x,I} = 1, && \forall I \in \hat{\mathscr{G}}_{CR}, \\ &\sum_{x \in s} \gamma_{x,I} \geq 1, && \forall I \in \hat{\mathscr{G}}_{CE}, \\ &\sum_{I \in \hat{g}_S} \gamma_{x,I} = 1, && \forall x \in Y, \\ &\gamma_{x,I} \in \{0,1\}, && \forall x \in Y, \forall I \in \hat{\mathscr{G}}_C. \end{aligned} \tag{5.7}$$

Here $\gamma_{x,I}$ indicates whether leaf s is attached to a side in the set of symmetric sides I. To find a feasible solution $(\gamma_{x,I})$, an algorithm using a greedy approach is introduced. It is greedy in the sense that it sets $\gamma_{x,I} = 1$ if $p_{x,I}$ is maximal. The next lemma introduces such a

Lemma 5.2.4. *Let $\mathscr{T}$ be a non-empty trinet set on m leaves containing t trinets, let $\hat{k} \in \{0,1,2\}$ be an integer and let $\hat{\mathscr{G}}$ be a generator of level $\hat{k}$. Then a feasible solution $(\hat{\gamma}_{x,I})$ to Equation* (5.7) *can be computed in $O(t+m^2)$ time.*

Furthermore, assume $\mathscr{T}$ is induced by a biconnected strictly level-$\overline{k}$ network $\overline{M}$ on generator $\overline{\mathscr{G}}$ such that $\hat{k} = \overline{k}$ and $\hat{\mathscr{G}} = \overline{\mathscr{G}}$. Furthermore, let $(\overline{\gamma}_{x,I})$ indicate whether leaf x is placed on a side in the set of symmetric sides I. Then it holds that $(\hat{\gamma}_{x,I}) = (\overline{\gamma}_{x,I}) = p_{x,I}$.

Proof. Let $\mathscr{T}$ be a trinet set, let $\hat{k} \in \{0,1,2\}$ be an integer and let $\hat{\mathscr{G}}$ be a generator of level $\hat{k}$. Now a feasible solution to Equation (5.7) will be constructed in a greedy manner.

For every set C of Y containing $|\hat{\mathscr{G}}_C|$ leaves and bijection $\phi : C \to \hat{g}_C$ define a solution $(\gamma_{x,I})$ such that:

$$\gamma_{x,I} = \begin{cases} 1 & \phi(x) = I \\ 0 & \text{else} \end{cases}, \qquad \forall x \in Y.$$

Next, choose a set $\hat{C}$ and bijection $\hat{\phi}$ such that the corresponding solution $(\hat{\gamma}_{x,I})$ maximises:

$$\sum_{I \in \hat{g}_C} \sum_{x \in \hat{C}} p_{x,I} \cdot \hat{\gamma}_{x,I}. \tag{5.8}$$

This solution satisfies the first two and the last constraints of Equation (5.7), and the third only for every $x \in \hat{C}$. In order to satisfy the third constraint for the leaves $x \in Y \setminus \hat{C}$, redefine:

$$\hat{\gamma}_{x,I} = 1,$$

for any:

$$I \in \arg\max_{I \notin \hat{g}_{CR}} \{p_{x,I}\}.$$

Now clearly all constraints of Equation (5.7) are satisfied. Therefore, $\left(\hat{\gamma}_{x,I}\right)$ satisfies Equation (5.7).

Now let $\mathscr{T}$ be induced by a biconnected strictly level-$\overline{k}$ network $\overline{M}$ with generator $\overline{g}$ on m species. It will be shown that $\left(\hat{\gamma}_{x,I}\right)$ defines leaf x to be on a side in I if and only if leaf x is on a side in I in network $\overline{M}$. To this end, let $\left(\overline{\gamma}_{x,I}\right)$ denote the set of symmetric sides I which contains the side s to which leaf x is attached in $\overline{M}$. It holds that:

$$p_{x,I} = \begin{cases} 1 & \overline{\gamma}_{x,I} = 1 \\ 0 & \text{else} \end{cases}. \tag{5.9}$$

So $p_{x,I} = \overline{\gamma}_{x,I}$. As $\left(\hat{\gamma}_{x,I}\right)$ is chosen to maximise Equation (5.8), it follows that:

$$\hat{\gamma}_{x,I} = \begin{cases} 1 & p_{x,I} = 1 \\ 0 & \text{else} \end{cases} \qquad \forall x \in \hat{C}. \tag{5.10}$$

So $\hat{\gamma}_{x,I} = p_{x,I}$ for all $x \in \hat{C}$. It follows from the way that $\left(\hat{\gamma}_{x,I}\right)$ is redefined that:

$$\hat{\gamma}_{x,I} = \begin{cases} 1 & p_{x,I} = 1 \\ 0 & \text{else} \end{cases} \qquad \forall x \in Y \setminus \hat{C}. \tag{5.11}$$

Hence $\hat{\gamma}_{x,I} = p_{x,I}$ for all x. Combining Equations (5.9), (5.10) and (5.11) gives:

$$\left(\hat{\gamma}_{x,I}\right) = \left(\overline{\gamma}_{x,I}\right) = p_{x,I}.$$

Lastly, it will be shown that time-complexity of finding this solution is $O(t)$. To this end, note first that the values $p_{x,I}$ can be computed with time-complexity $O(t)$ by iterating over all t trinets. Next, note that there are:

$$P(m, |\hat{g}_C|) = \frac{m!}{(m - |\hat{g}_C|)!},$$

combinations of sets C and bijections ϕ. Also note that $|\hat{\mathscr{G}}_C| \leq 2$ for $\hat{k} \leq 2$. Hence there are at most m^2 combinations. Furthermore, calculating the value of Equation (5.8) has time-complexity $O(|\hat{\mathscr{G}}_C|^2)$, which is independent of m and can thus be seen as constant. Therefore, choosing the best combination can be done in $O(m^2)$. Next, redefining the solution for the leaves $x \in Y \setminus C$ consists of finding the maximum of a set containing at most $|\mathscr{G}_S|$ elements. As $|\hat{\mathscr{G}}_S| \leq 8$ for $\hat{k} \leq 2$, it follows that this redefining has time-complexity $O(m)$.

Therefore, finding this feasible solution has time-complexity $O(t + m^2)$. □

This solution $(\hat{\gamma}_{x,I})$ is rewritten as a set of leaves for each set of symmetric sides I in $\hat{\mathscr{G}}$:

$$\hat{\eta}_I = \{x : \hat{\gamma}_{x,I} = 1\}, \qquad \forall I \in \hat{\mathscr{G}}_S.$$

5.2.4. GROUP OF LEAVES PER SIDE

Each of these sets of leaves $\hat{\eta}_I$ must be split in to groups θ_s, for each side s in the set of symmetric sides I. To this end, let:

$$p_{x,y}, \tag{5.12}$$

be the number of biconnected level-1 and level-2 trinets such that:

- x nor y is a reticulation leaf;
- the reticulation leaves are as specified by $\hat{\eta}_I$;
- $x \prec y$.

A trinet has $x \prec y$ if it contains both x and y and the parent of x is strictly above the parent of y. Now:

$$q_{x,y} = \begin{cases} p_{x,y} + p_{y,x} & x \neq y \\ 1 & x = y \end{cases}, \tag{5.13}$$

is a measure for how likely x and y are one the same side of $\hat{\mathscr{G}}$. From this, the amount of leaves that are likely to be on the same side as x and on the same of side as y can be computed:

$$\sum_{z \in \hat{\eta}_I} \min\{q_{x,z}, q_{y,z}\}.$$

Subtracting the amount of leaves that are likely to be on a the same side as x but not on the same side as y gives the following measure:

$$r_{x,y} = \begin{cases} 3 \cdot \sum_{z \in \hat{\eta}_I} \min\{q_{x,z}, q_{y,z}\} - \sum_{z \in \hat{\eta}_I} q_{x,z} - \sum_{z \in \hat{\eta}_I} q_{y,z} & x \neq y \\ -\infty & x = y \end{cases}. \tag{5.14}$$

Using this measure, the leaves in $\hat{\eta}_I$ are added to a group θ_s iteratively. This process can be seen in Algorithm 3. The next lemma shows that this algorithm is correct.

Lemma 5.2.5. *Let $\mathcal{T}$ be a non-empty trinet set on m leaves containing t trinets, let $\hat{\mathcal{G}}$ be a generator and let $\hat{\eta}_I$ be sets of leaves for every I in $\hat{\mathcal{G}}_I$. Then Algorithm 3 returns sets of leaves $\hat{\theta}_s$ such that $\hat{\eta}_I = \bigcup_{s\in I} \hat{\theta}_s$ for each $I \in \hat{\mathcal{G}}_S$. It does this with time-complexity $O(t+m^2)$.*

Furthermore, if $\mathcal{T}$ is induced by a biconnected network $\overline{M}$ on generator $\hat{\mathcal{G}}$ such that $\hat{\mathcal{G}} = \overline{\mathcal{G}}$ and $\hat{\eta}_I = \overline{\eta}_I$, then $\hat{\theta}_s \cong \overline{\theta}_s$.

Proof. First the time-complexity will be derived. To this end, let m_I denote the number of leaves in $\hat{\eta}_I$.

Clearly, all values $q_{x,y}$ can be computed with time-complexity $O(t)$ by iterating over all t trinets. From $q_{x,y}$ all values of $r_{x,y}$ can be derived by iterating over all m^2 pairs of nodes. The set C in line 3 contains precisely m_I number of singletons. In each of the iterations of the while-loop in lines 4-11 this number is decreased by one. Hence, this while-loop is run at most $m_I - 1$ times. Finding two leaves that maximise $r_{x,y}$ requires iterating over all $|C|^2$ pairs of clusters. As there are $|C|-2$ new values of $r_{z,w}$, these can be calculated in $O(|C|)$. It follows that the while-loop in lines 4-11 has the following time-complexity:

$$\sum_{h=m_I,(m_I-1),\ldots,1} O(h^2) = O(m_I^3).$$

The for-loop from lines 1-14 is run once for each set of symmetric sides in $\hat{g}_S$. It follows that this for-loop has time-complexity:

$$O\left(\sum_{I\in\hat{g}_S} m_I^3\right).$$

As the m_I add up to m it follows that this can be simplified to $O(m^3)$. Combined, this results in an overall time-complexity of $O(t+m^3)$.

Now it will be shown that the algorithm is correct. Clearly, the algorithm returns sets of leaves $\hat{\theta}_s$ such that $\hat{\eta}_I = \bigcup_{s\in I} \hat{\theta}_s$ for each $I \in \hat{\mathcal{G}}_S$. Next, assume that $\mathcal{T}$ is induced by a biconnected network $\overline{M}$ such that $\hat{\mathcal{G}} = \overline{\mathcal{G}}$ and $\hat{\eta}_I = \overline{\eta}_I$, and let $\overline{\theta}_s$ denote the set of leaves that is attached to side s in $\overline{M}$.

First, note for $x, y \in \overline{\eta}_I$ it holds that:

$$\{x,y\} \subseteq \overline{\theta}_s \iff q_{x,y} > 1.$$

As $\{x,y\} \subseteq \overline{\theta}_s$ is a transitive relation, that is:

$$\{x,y\} \subseteq \overline{\theta}_s \wedge \{y,z\} \subseteq \overline{\theta}_s \implies \{x,z\} \subseteq \overline{\theta}_s,$$

it follows that:

$$\{x,y\} \subseteq \overline{\theta}_s \iff r_{x,y} = 3|\overline{\theta}_s| > 0,$$
$$\{x,y\} \not\subseteq \overline{\theta}_s \iff r_{x,y} < 0.$$

Hence, the first iteration picks two leaves x and y that are in the same set $\overline{\theta}_s$. Now let X denote the set of leaves that a leaf x represent. So in this case $Z = \{x, y\}$ and $Z \subseteq \overline{\theta}_s$.

Moreover, the new defined values $r_{z,w}$ are the weighted average of the values $r_{x,w}$ and $r_{y,w}$.

Next, from the transitivity of $\{x, y\} \subseteq \overline{\theta}_s$ it follows that:

$$r_{z,w} = r_{x,w} = r_{y,w}.$$

Hence, in any of the following iterations, if there exists a positive value of r, two leaves x and y are picked such that both $X \subset \overline{\theta}_s$ and $Y \subset \overline{\theta}_s$. It follows that $Z = X \cup Y \subseteq \overline{\theta}_s$. If no such positive value of $r_{x,y}$ exists, then there are no leaves x and y such that the sets X and Y they represent are in the same set $\overline{\theta}_s$. Therefore, for each x it must hold that there exists a side s such that $X = \overline{\theta}_s$. It follows that C contains an equal or smaller number of elements than there are sets s in I. Hence the while-loop in lines 4-11 terminates and the sets X that the elements $x \in C$ represent are assigned to a set $\hat{\theta}_s$ arbitrarily. It follows that $\hat{\theta}_s \cong \overline{\theta}_s$. □

Algorithm 3: dividing leaves in a set of symmetric sides over sides

```
Input:
- Trinet set $\mathscr{T}$,
- generator $\hat{g}$,
- Set of leaves $\eta_I$ for every $I \in \hat{g}_S$,
Output:
- Group $\hat{\theta}_s$ for every side $s$ in $I$ for every $I \in \hat{\mathscr{G}}_S$
for each set of symmetric sides $I$ in $\hat{\mathscr{G}}_S$ do
    $r_{x,y} \leftarrow$ as in Equation (5.14) for the leaves in $\eta_I$
    $C \leftarrow$ set containing the singletons $\{x\}$ for every leaf in $\eta_I$
    while there exists a strictly positive element $r_{x,y}$ or $C$ contains more elements
      then there are sides in $I$ do
        let $x, y$ be two leaves which maximise $r_{x,y}$
        define a new leaf $z$ which represents $x$ and $y$
        remove $x$ and $y$ from $C$ and add $z$
        let $L_x$ and $L_y$ denote the amount of leaves that $x$ and $y$ represent
        $r_{z,w} \leftarrow \frac{L_x \cdot r_{x,w} + L_y \cdot r_{y,w}}{L_x + L_y}$, $r_{w,z} \leftarrow r_{z,w}$ for each other leaf $w$
        undefine $r_{x,w}$, $r_{w,x}$, $r_{y,w}$ and $r_{w,y}$ for each other leaf $w$
    end
    assign the sets of leaves the singletons represent to different sides $\hat{\theta}_s$ for $s \in I$,
      if no set is assigned to a side $s$, it remains empty
end
return $\hat{\theta}_s$
```

5.2.5. Ordered group of leaves per side

Now it is known which leaves are attached to each side s of $\hat{\mathscr{G}}$, the next step is to order each of these groups $\hat{\theta}_s$. Denote these orderings with $\hat{\sigma}_s$. This ordering is based on $p_{x,y}$ from Equation (5.12). This can be seen as the problem of finding a ranking in a graph-theoretical tournament. There are several methods to find a ranking in such a tournament. The default ordering method of `TriL2Net` uses the difference of normalized values of $p_{x,y}$, that is:

$$\tilde{p}_{x,y} = \frac{p_{x,y} - p_{y,x}}{q_{x,y}}, \tag{5.15}$$

where $q_{x,y}$ is as in Equation (5.13). Now a leaf $x \in \hat{\theta}_s$ that maximises:

$$\sum_{y \in \hat{\theta}_s} \tilde{p}_{x,y},$$

is chosen to be 'highest' leaf. That is, the leaf which is attached highest to side s and hence is the first leaf of the ordered set $\hat{\sigma}_s$. Now the next leaf is chosen to be a leaf x that maximises:

$$\sum_{y \in \hat{\theta}_s \setminus (\hat{\sigma}_s)} \tilde{p}_{x,y}, \tag{5.16}$$

is appended to $\hat{\sigma}_s$ iteratively.

Note that `TriLoNet` orders the leaves concurrently with assigning the leaves to the different sides in a symmetric side set. That is, `TriLoNet` picks a leaf x which maximises the value of:

$$\sum_{y \in \hat{\theta}_s} p_{x,y},$$

and assigns it to a side $s \in I$ based on the amount of descendants x and the leaves $y \in \theta_s$ share. The ordering of the leaves on the sides follows directly from the order the leaves are assigned to each side. Hence, it is possible for two leaves x_1, x_2 that are placed on side $s \in I$ that the following holds:

$$\sum_{y \in \hat{\theta}_s} p_{x_1,y} > \sum_{y \in \hat{\theta}_s} p_{x_2,y},$$
$$\sum_{y \in s} p_{x_1,y} < \sum_{y \in s} p_{x_2,y}.$$

This means that x_1 will be placed above x_2 on side s by `TriLoNet`. The method that `TriL2Net` uses, on the other hand, will place x_2 above x_1. The `TriLoNet` method produces the correct result in case the input trinet set is induced (Oldman et al. 2016), as does the `TriL2Net` method. The latter is shown in the following lemma.

Lemma 5.2.6. *Let $\mathcal{T}$ be a non-empty trinet set on m leaves containing t trinets, let $\hat{g}$ be a generator and let $\hat{\theta}_s$ be a set of leaves for each side s of $\hat{g}$. Then $\hat{\sigma}_s$ can be computed with time-complexity $O(t+m^2)$.*

Furthermore, if $\mathcal{T}$ is induced by a biconnected network $\overline{M}$ on generator $\hat{\mathcal{G}}$ such that $\hat{\mathcal{G}} = \overline{\mathcal{G}}$ and $\hat{\theta}_s \cong \overline{\theta}_s$, then $\hat{\sigma}_s \cong \overline{\sigma}_s$.

Proof. First the time-complexity will be derived: $p_{x,y}$ can be computed by iterating over all t trinets in $\mathcal{T}$ and hence has time-complexity $O(t)$. $q_{x,y}$ and $\tilde{p}_{x,y}$ can be computed by iterating over all pairs of leaves x, y. As there are $m(m-1)$ such pairs, it follows that this computation can be done with time-complexity $O(m^2)$. Next, computing each Equation (5.16) for a leaf x requires iterating over all leaves in $\hat{\theta}_s$ in the worst case. Denoting the number of elements in $\hat{\theta}_s$ with m_s, it follows that these values can be calculated in $O(m_s^2)$. Moreover, finding the maximum of these values can be done in $O(m_s)$. Note that calculating these values and finding the maximum needs to be done for every one of the n_s leaves. Hence, this method of ordering has time-complexity $O(m_s^3)$. As this needs to be done for every side s, and as:

$$\sum_{s\in i} m_s = m_I, \qquad \sum_{I\in \hat{g}} m_I = m,$$

it follows that this amounts to a time-complexity of $O(m^3)$ for ordering all sides of $\hat{\mathcal{G}}$.

Secondly, assume that $\mathcal{T}$ is induced by a biconnected network $\overline{M}$ on generator $\hat{\mathcal{G}}$ such that $\hat{\mathcal{G}} = \overline{\mathcal{G}}$ and $\hat{\theta}_s \cong \overline{\theta}_s$. It will be shown that $\hat{\sigma}_s \cong \overline{\sigma}_s$. To this end, let ϕ be the homomorphism between the sides s of $\hat{\mathcal{G}}$ such that:

$$\hat{\theta}_s = \overline{\theta}_{\phi(s)}, \qquad \forall s.$$

It will be shown that:

$$\hat{\sigma}_s = \overline{\sigma}_{\phi(s)}, \qquad \forall s.$$

To this end, fix s. Then by the assumption that $\mathcal{T}$ is induced, it follows for all $x, y \in \hat{\theta}_{\phi(s)}$ that $\tilde{p}_{x,y} = 1$ if and only if x comes before y in $\overline{\sigma}_s$. By construction of $\hat{\sigma}_{\phi(s)}$, it follows that x comes before y in $\hat{\sigma}_{\phi(s)}$ if and only if x comes before y in $\overline{\sigma}_s$. It follows that $\hat{\sigma}_s = \overline{\sigma}_{\phi(s)}$. As this holds for all s, it holds that $\hat{\sigma}_s \cong \overline{\sigma}_s$. □

Other methods for ordering the leaves are mentioned in the discussion.

5.2.6. ALIGNMENT OF THE SIDES

For generator $\mathcal{G}^{2c}$, an extra step needs to be performed as this generator contains another symmetry. To this end, name the sides in Figure 3.2 as follows:

$$\begin{aligned}\hat{g}_E &= \{I_1, I_2, I_3\}\\ &= \{\{s_1, s_2\}, \{s_3, s_4\}, \{s_5, s_6\}\}.\end{aligned}$$

Furthermore, these sides can be split in two groups: the ones adjacent to node 1 and the ones adjacent to node 2. This can be seen as the left (L) and the right (R) group:

$$L = \{s_1, s_3, s_5\}, \qquad R = \{s_2, s_4, s_6\}.$$

If $\hat{\mathscr{G}}$ is this generator, then having $\hat{\sigma}_s \cong \overline{\sigma}_s$ is not sufficient. This is due to the fact that swapping edge sides s_1 and s_2 of this generator, for example, will result in a different network $\hat{M}$. Therefore, the corresponding homomorphism ϕ must be determined.

To this end, define a measure for how likely it is that two sides $s \in I$ and $s' \in I'$ are aligned. That is, for two sets of symmetric sides $I \neq I'$ of $\hat{\mathscr{G}}$ let:

$$u_{s,s'} = \sum_{x \in \hat{\sigma}_s} \sum_{y \in \hat{\sigma}_{s'}} \left(q_{x,y}\right) - |\hat{\sigma}_s||\hat{\sigma}_{s'}|. \tag{5.17}$$

Here $q_{x,y}$ is from Equation (5.13). Now let L_i be one of the two sides from each set of symmetric sides I_i and let R_i be the other of the two sides. This must be done such that:

$$\mathbf{u} = \left(u_{L_1,L_2} + u_{L_2,L_3} + u_{L_3,L_1}\right) + \left(u_{R_1,R_2} + u_{R_2,R_3} + u_{R_3,R_1}\right), \tag{5.18}$$

is maximized. Now let $\hat{\phi}$ be such that:

$$\begin{aligned} \hat{\phi}(L_1) &= s_1, & \hat{\phi}(R_1) &= s_2, \\ \hat{\phi}(L_2) &= s_3, & \hat{\phi}(R_2) &= s_4, \\ \hat{\phi}(L_3) &= s_5, & \hat{\phi}(R_3) &= s_6. \end{aligned} \tag{5.19}$$

5

The next lemma shows that this methods is correct.

Lemma 5.2.7. *Let $\mathscr{T}$ be a non-empty trinet set on m leaves containing at most t trinets, let $\hat{g}$ be generator* 2.3 *and let $\hat{\theta}_s$ be a list of ordered leaves per side s of $\hat{\mathscr{G}}$. Then $\hat{\phi}$ from Equation* (5.19) *can be computed with time-complexity $O(t)$.*

Furthermore, if $\mathscr{T}$ is induced by a biconnected network $\overline{M}$ such that $\hat{\mathscr{G}} = \overline{\mathscr{G}}$ and $\hat{\sigma}_{\overline{\phi}(s)} = \overline{\sigma}_s$, then $\hat{\phi}$ is such that:

$$\hat{\sigma}_{\hat{\phi}(s)} = \overline{\sigma}_s,\ \forall s \qquad \text{or} \qquad \hat{\sigma}_{\psi \circ \hat{\phi}(s)} = \overline{\sigma}_s,\ \forall s, \tag{5.20}$$

where:

$$\begin{aligned} \psi(s_1) &= s_2, & \psi(s_2) &= s_1, \\ \psi(s_3) &= s_4, & \psi(s_4) &= s_3, \\ \psi(s_5) &= s_6, & \psi(s_6) &= s_5. \end{aligned}$$

Proof. First it is shown that the time-complexity of computing $\hat{\phi}$ is $O(t)$. Clearly, the values $u_{s,s'}$ can be computed with time-complexity $O(t)$ by iterating over all t trinets. Next, three sides $s_{I_1}, s_{I_2}, s_{I_3}$ must be chosen for which Equation (5.18) is maximised. This can be done by computing the sum for each of the 2^3 possibilities. It follows that $\hat{\phi}$ can be computed with time-complexity $O(t)$.

Now assume that $\mathscr{T}$ is induced by a biconnected recoverable network $\overline{M}$ on generator $\overline{\mathscr{G}}$ such that $\hat{\mathscr{G}} = \overline{\mathscr{G}}$ and $\hat{\sigma}_{\overline{\phi}(s)} = \overline{\sigma}_s$.

As $\mathscr{T}$ is induced by a network with generator $\mathscr{G}^{2c}$, it holds that:

$$\begin{aligned} u_{s,s'} &= \sum_{x \in \hat{\sigma}_s} \sum_{y \in \hat{\sigma}_{s'}} \left(q_{x,y}\right) - |\hat{\sigma}_s||\hat{\sigma}_{s'}| \\ &= \sum_{x \in \overline{\sigma}_{\overline{\phi}(s)}} \sum_{y \in \overline{\sigma}_{\overline{\phi}(s')}} \left(q_{x,y}\right) - |\overline{\sigma}_{\overline{\phi}(s)}||\overline{\sigma}_{\overline{\phi}(s')}|. \end{aligned}$$

Furthermore, if $\overline{\phi}(s), \overline{\phi}(s') \in L$ or $\overline{\phi}(s), \overline{\phi}(s') \in R$, then:

$$\begin{aligned} u_{s,s'} &= \sum_{x \in \overline{\sigma}_{\overline{\phi}(s)}} \sum_{y \in \overline{\sigma}_{\overline{\phi}(s')}} (q_{x,y}) - |\overline{\sigma}_{\overline{\phi}(s)}||\overline{\sigma}_{\overline{\phi}(s')}| \\ &= |\overline{\sigma}_{\overline{\phi}(s)}||\overline{\sigma}_{\overline{\phi}(s')}| \\ &= |\hat{\sigma}_s||\hat{\sigma}_{s'}|. \end{aligned}$$

This follows from the fact that each trinet T on $\{x, y, z\}$ with:

$$x \in \overline{\sigma}_{\overline{\phi}(s)}, \qquad y \in \overline{\sigma}_{\overline{\phi}(s')}, \qquad z \in \overline{g}_{CR},$$

has $x \prec y$ or $y \prec x$. As this are all trinets that have $x \prec y$ or $y \prec x$, it follows that there are:

$$|\overline{\mathscr{G}}_{CR}||\overline{\sigma}_{\overline{\phi}(s)}||\overline{\sigma}_{\overline{\phi}(s')}| = 2|\overline{\sigma}_{\overline{\phi}(s)}||\overline{\sigma}_{\overline{\phi}(s')}|,$$

such trinets in total.

Similarly, if $\overline{\phi}(s) \in L$ and $\overline{\phi}(s') \in R$ (or vice versa) such that $\overline{\phi}(s)$ and $\overline{\phi}(s')$ are not in the same set of symmetric sides I, then:

$$u_{s,s'} = -|\hat{\sigma}_s||\hat{\sigma}_{s'}|.$$

Now remember that m_I denotes the number of elements in $\hat{\theta}_I$. In case $m_{I_i} = 0$ for at least two $i \in \{1,2,3\}$, it holds that any $\hat{\phi}$ satisfies Equation (5.20). In case $m_{I_i} \neq 0$ for at least two $i \in \{1,2,3\}$, it can be assumed without loss of generality that $m_{I_1} > 0$ and $m_{I_2} > 0$ due to the symmetry of the problem. Now let L_1, L_2, R_1 and R_2 be such that $\mathbf{u}$ is maximised. Now note that:

$$u_{\overline{\phi}(s_1),\overline{\phi}(s_3)} + u_{\overline{\phi}(s_2),\overline{\phi}(s_4)} > u_{\overline{\phi}(s_1),\overline{\phi}(s_4)} + u_{\overline{\phi}(s_2),\overline{\phi}(s_3)}.$$

The reason that this is a *strict* inequality is that $m_{I_1} > 0$ and $m_{I_2} > 0$. Hence $L_1 = \overline{\phi}(s_1), L_2 = \overline{\phi}(s_3)$ and $R_1 = \overline{\phi}(s_2), R_2 = \overline{\phi}(s_4)$ (or vice versa). Without loss of generality, it can be assumed that $L_1 = \overline{\phi}(s_1), L_2 = \overline{\phi}(s_3)$ and $R_1 = \overline{\phi}(s_2), R_2 = \overline{\phi}(s_4)$. By definition of $\hat{\phi}$ it holds that:

$$\hat{\phi}(\overline{\phi}(s_1)) = s_1, \qquad \hat{\phi}(\overline{\phi}(s_2)) = s_2, \qquad \hat{\phi}(\overline{\phi}(s_3)) = s_3, \qquad \hat{\phi}(\overline{\phi}(s_4)) = s_4.$$

As $\overline{\phi}^{-1} = \overline{\phi}$, it holds that $\hat{\phi}(s) = \overline{\phi}(s)$ for $s \in \{s_1, s_2, s_3, s_4\}$ and hence:

$$\hat{\sigma}_{\hat{\phi}(s)} = \hat{\sigma}_{\overline{\phi}(s)} = \overline{\sigma}_s, \qquad \forall s \in \{s_1, s_2, s_3, s_4\}.$$

If $m_{I_3} = 0$, then for any choice for L_3 and R_3 it holds that:

$$\hat{\sigma}_{\hat{\phi}(s)} = \hat{\sigma}_{\overline{\phi}(s)} = \overline{\sigma}_s, \qquad \forall s \in \{s_5, s_6\}.$$

If $m_{I_3} > 0$, then it follows that the following strict inequalities hold:

$$\begin{aligned} u_{\overline{\phi}(s_1),\overline{\phi}(s_5)} + u_{\overline{\phi}(s_2),\overline{\phi}(s_6)} &> u_{\overline{\phi}(s_1),\overline{\phi}(s_6)} + u_{\overline{\phi}(s_2),\overline{\phi}(s_5)}, \\ u_{\overline{\phi}(s_3),\overline{\phi}(s_5)} + u_{\overline{\phi}(s_4),\overline{\phi}(s_6)} &> u_{\overline{\phi}(s_3),\overline{\phi}(s_6)} + u_{\overline{\phi}(s_4),\overline{\phi}(s_5)}. \end{aligned}$$

As $\hat{\phi}(s) = \overline{\phi}(s)$ for $s \in \{s_1, s_2, s_3, s_4\}$, it holds that:

$$u_{\hat{\phi}(s_1),\overline{\phi}(s_5)} + u_{\hat{\phi}(s_2),\overline{\phi}(s_6)} > u_{\hat{\phi}(s_1),\overline{\phi}(s_6)} + u_{\hat{\phi}(s_2),\overline{\phi}(s_5)},$$
$$u_{\hat{\phi}(s_3),\overline{\phi}(s_5)} + u_{\hat{\phi}(s_4),\overline{\phi}(s_6)} > u_{\hat{\phi}(s_3),\overline{\phi}(s_6)} + u_{\hat{\phi}(s_4),\overline{\phi}(s_5)}.$$

It follows that $L_3 = \overline{\phi}(s_5)$ and $R_3 = \overline{\phi}(s_6)$ must hold in order for $\mathbf{u}$ to be maximised. By definition of $\hat{\phi}$, it holds that:

$$\hat{\phi}(\overline{\phi}(s_5)) = s_5, \qquad \hat{\phi}(\overline{\phi}(s_6)) = s_6.$$

Hence:

$$\hat{\sigma}_{\hat{\phi}(s)} = \hat{\sigma}_{\overline{\phi}(s)} = \overline{\sigma}_s, \qquad \forall s \in \{s_5, s_6\}.$$

Therefore, in any case:

$$\hat{\sigma}_{\hat{\phi}(s)} = \hat{\sigma}_{\overline{\phi}(s)} = \overline{\sigma}_s, \qquad \forall s \in \{s_1, s_2, s_3, s_4, s_5, s_6\},$$

and thus Equation (5.20) holds.

In case it was assumed that $L_1 = \overline{\phi}(s_2)$, it would have followed that:

$$\hat{\sigma}_{\psi\circ\hat{\phi}(s)} = \hat{\sigma}_{\overline{\phi}(s)} = \overline{\sigma}_s, \qquad \forall s \in \{s_1, s_2, s_3, s_4, s_5, s_6\}.$$

□

5.2.7. COMBINING THE STEPS TO CONSTRUCT A BICONNECTED NETWORK

Performing all the steps in this section creates a biconnected level-2 phylogenetic network $\hat{M}$ from a trinet set $\mathcal{T}$. The following theorem shows what the time-complexity is and that this process is correct. That is, if the trinet set $\mathcal{T}$ is induced by a biconnected level-2 phylogenetic network $\overline{M}$, then these steps recreate $\overline{M}$.

Theorem 5.2.8. *Let $\mathcal{T}$ a non-empty trinet set on m leaves containing t trinets. Then a biconnected rooted level-2 phylogenetic network $\hat{M}$ on m leaves can be constructed with time-complexity $O(t+m^2)$.*

Furthermore, in case $\mathcal{T}$ and $\mathcal{B}$ are induced by a biconnected level-2 phylogenetic network $\overline{M}$, then $\hat{M} = \overline{M}$.

Proof. Using the methods in Lemmas 5.2.2, 5.2.3, 5.2.4, 5.2.5, 5.2.6 and 5.2.7 a generator $\hat{\mathcal{G}}$ of strict level $\hat{k}$ together with ordered lists $\hat{\sigma}_s$ for each side s of $\hat{\mathcal{G}}$ are computed with time-complexity $O(t+m^2)$. Furthermore, if $\hat{\mathcal{G}} = \mathcal{G}^{2c}$, then a homomorphism $\hat{\phi}$ is computed with time-complexity $O(t)$. Together, $\hat{\mathcal{G}}$, $\hat{\sigma}_s$ and $\hat{\phi}$ define a biconnected recoverable network $\hat{M}$ up to an isomorphism. Hence, the network $\hat{M}$ is computed with time-complexity $O(t+m^2)$.

Now assume that $\mathcal{T}$ is induced by a biconnected recoverable level-2 phylogenetic network $\overline{M}$ with generator $\overline{\mathcal{G}}$ of strict level $\overline{k}$ together with ordered lists $\overline{\sigma}_s$ for each side s of $\overline{\mathcal{G}}$. Then it follows from the same lemmas that:

$$\hat{\mathcal{G}} = \overline{\mathcal{G}},$$
$$\hat{\gamma_{x,I}} = \overline{\gamma_{x,I}}, \qquad \forall I \in \hat{\mathcal{G}},$$
$$\hat{\sigma}_s \cong \overline{\sigma}_s, \qquad \forall s.$$

Furthermore, if $\hat{\mathscr{G}} = \mathscr{G}^{2c}$, then the homomorphism $\hat{\phi}$ is such that:

$$\hat{\sigma}_{\hat{\phi}(s)} = \overline{\sigma}_s, \ \forall s \qquad \text{or} \qquad \hat{\sigma}_{\psi \circ \hat{\phi}(s)} = \overline{\sigma}_s, \ \forall s, \tag{5.21}$$

where:

$$\begin{aligned} \psi(s_1) &= s_2, & \psi(s_2) &= s_1, \\ \psi(s_3) &= s_4, & \psi(s_4) &= s_3, \\ \psi(s_5) &= s_6, & \psi(s_6) &= s_5. \end{aligned}$$

It follows that there exists an isomorphism between the nodes of $\hat{M}$ and $\overline{M}$ that preserves the naming of the leaves. Hence $\hat{M}$ and $\overline{M}$ are equal. □

5.3. COLLAPSING THE TRINET AND BINET SET

After a biconnected network $\hat{M}$ has been constructed for a minimal sink set Y of $\Omega_{\hat{i}}(\mathscr{T})$ using the restricted trinet set $\mathscr{T}^r$, the set Y must be collapsed to a leaf l in every trinet $\mathscr{T}$ to create the new trinet set $\mathscr{T}^c$. This procedure is detailed in Algorithm 4. The time-complexity is analysed in the following lemma. That this procedure is correct follows from Lemma 4.3.1.

Lemma 5.3.1. *Let $\mathscr{T}$ be a non-empty trinet set on X with $|X| = n$ containing t trinets. Furthermore, let $Y \subseteq X$ be any leaf set, then Algorithm 4 has time-complexity $O(t^2)$.*

Proof. The for-loop in lines 3-14 is run t times. Clearly, the sets K and L in lines 4 and 5 can be computed in constant time as T contains at most three leaves. As a trinet has exactly three leaves it follows that checking if a trinet is equal to another trinet has constant time-complexity. As $|\mathscr{T}^c| \leq |\mathscr{T}|$ it follows that looking for an equal trinet in $\mathscr{T}^c$ has time-complexity $O(t)$. Hence lines 7 and 12 have time-complexity $O(t)$. Therefore, the time-complexity of collapsing is $O(t^2)$. □

Algorithm 4: collapsing

```
Input:
- Trinet set $\mathscr{T}$,
- set of leaves $Y$,
Output:
- Trinet set $\mathscr{T}^c$,
- leaf $l$,
1  $\mathscr{T}^c \leftarrow$ empty set
2  $l \leftarrow$ leaf representing $Y$
3  for trinet $T$ in $\mathscr{T}$ do
4      $K \leftarrow$ leaves of $T$
5      $L \leftarrow K \cap Y$
6      if $L$ is empty then
7          add $T$ to $\mathscr{T}^c$
8      end
9      if $|L| = 1$ then
10         $x \leftarrow$ leaf in $L$
11         replace $x$ with $l$
12         add $N^c$ to $\mathscr{T}^c$
13     end
14 end
15 return $\mathscr{T}^c$ and $l$
```

5.4. CONSTRUCTING A RECOVERABLE LEVEL-2 NETWORK

Given a trinet set $\mathscr{T}$ on species X, it is now known how to determine a minimal cut-arc set $Y \subseteq X$, how to compute a biconnected level-2 phylogenetic network $\hat{M}$ on species Y from the restricted trinet set $\mathscr{T}^r$ and how to collapse Y in $\mathscr{T}$ to create a new trinet set $\mathscr{T}^c$. This can be done iteratively until every leaf in X has been described by a biconnected level-2 network. These biconnected level-2 networks can now be combined to create the output network $\hat{N}$ in the `expand` function. The whole process is detailed in Algorithm 5. The lines 1-18 are the `collapse` part of the algorithm and the lines 19-25 are the `expand` part. The following theorem shows what the algorithm's time-complexity is and that it reproduces the network $\overline{N}$ if the input trinet set $\mathscr{T}$ induced by $\overline{N}$.

Theorem 5.4.1. *Let $\mathscr{T}$ be non-empty set of recoverable level-2 trinets on species X with n leaves containing t trinets. Then Algorithm 5 constructs a level-2 network $\hat{N}$ on X with time-complexity $O(n \cdot t^2 + n^5)$.*

Furthermore, in case $\mathscr{T}$ is induced by a recoverable level-2 phylogenetic network $\overline{N}$, then $\hat{N} = \overline{N}$.

Proof. First it will be shown that the Algorithm 5 is correct and that the time-complexity is $O(n \cdot t^2 + n^5)$.

By definition, a minimal sink-set of $\Omega_{\hat{i}}(\mathscr{T})$ contains at least two elements. Hence:

$$|X| < |(X \setminus Y) \cup \{c\}|,$$

and thus the while-loop in lines 4-12 runs at most n times.

Next, as the trinet sets $\mathscr{T}_i$ and $\mathscr{T}_i^r$ are collapsed and restricted trinet sets respectively, it holds that:

$$|\mathscr{T}_i| \leq |\mathscr{T}|, \qquad |\mathscr{T}_i^r| \leq |\mathscr{T}|.$$

Hence each trinet set $\mathscr{T}_i$ and $\mathscr{T}_i^r$ contains at most t trinets. Furthermore, each minimal sink-set Y_i is a subset of X_i. Hence each minimal sink-set Y_i contain at most n leaves.

Hence, it follows from Lemma 5.1.4 and Lemma 5.1.5 that a minimal sink-set Y_i exists and can be computed in $O(t + n^4)$. Furthermore, from Lemma 5.2.1 it follows that the restrictions $\mathscr{T}_i^r$ of $\mathscr{T}_i$ can be computed with time-complexity $O(t)$. Moreover, by Theorem 5.2.8, it holds that the biconnected network M_i can be computed with time-complexity $O(t + n^2)$. By Lemma 5.3.1, it hold that the collapsed trinet set $\mathscr{T}_{i+1}$ of $\mathscr{T}_i$ can be computed with time-complexity $O(t^2)$. As X_i and Y_i both contain at most n leaves, it follows that computing X_{i+1} has time-complexity $O(n^2)$. Hence, lines 4-12 have time-complexity $O(n \cdot t^2 + n^5)$.

Computing $\mathscr{B}$ in line 14 requires iterating through the at most t trinets in $\mathscr{T}_{i-1}$. Creating the induced binet set for each of these trinets has constant time. Taking the union of all these binet sets consists of iterating through all their binets and checking if it is already present in the resulting binet set. As each trinet exhibits three binets, it follows that there are at most $3t$ binets. Hence, $\mathscr{B}$ can be computed with time-complexity $O(t^2)$. Removing all binets which do not have all leaves in X_i can be done by iterating through the at most $3t$ binets and hence has time-complexity $O(t)$. Finding the binet with the highest multiplicity again requires iterating through the at most $3t$ binets. Hence, lines 13-18 have time-complexity $O(t^2)$.

Next, from Observation 2.4 of Janssen et al. (2018) it follows that all binary biconnected level-2 networks with n leaves contain $2n+4$ nodes and $2n+5$ edges. It follows that replacing a leaf with a network has time-complexity $O(n)$. Lastly, it holds that the while-loop in lines 21-24 runs at most n times. Hence this loop has time-complexity $O(n^2)$.

Combined, this gives a time-complexity of $O(n \cdot t^2 + n^4)$.

Now it must be shown that l_{i-1} is in fact a leaf of network N_i. To this end, let R be the highest index for which M_i exists. Then it holds that $l_{R-1} \in M_R$. Furthermore, it follows from:

$$l_i \in X_{i+1}, \qquad X_{i+1} = (X_i \setminus Y_i) \cup \{l_i\}, \qquad Y_i \subset X_i,$$

for all $i \in \{1, \ldots R\}$ that:

$$\{l_i, \ldots, l_{R-2}\} \subseteq \left(\bigcup_{j=i+1}^{R-1} Y_j \right), \qquad \forall i \in \{1, \ldots R-2\}.$$

As $N_R = M_R$, it follows that l_{R-1} is a leaf of N_R. Now let N_{R-1} be the network created from N_R by replacing this leaf l_{R-1} with the network M_{R-1}. It now holds that N_{R-1} is a network on $(Y_R \cup Y_{R-1}) \setminus \{r_{R-1}\}$. Hence r_{R-2} is a leaf of N_{R-1}. Continuing this line of reasoning, it follows that r_{i-1} is in fact a leaf of network M_i. Therefore, the algorithm completes.

It follows that Algorithm 5 is correct.

Secondly, assume that $\mathcal{T}_1$ and $\mathcal{B}_1$ are induced by a recoverable level-2 phylogenetic network $\overline{N}_1$. It will be shown that the output of the algorithm is equal to $\overline{N}_1$.

By Lemma 5.1.4 or Lemma 5.1.5, it holds that Y_1 of line 5 is a minimal cut-arc set of $\overline{N}_1$. Let $\overline{M}_1$ denote the restriction of $\overline{N}_1$ to Y_1. It follows from 5.2.1 that $\mathcal{T}_1^r$ is induced by $\overline{M}_1$. It follows from 5.2.8 that $M_1 = \overline{M}_1$. Next, let $\overline{N_2}$ be the network created from $\overline{N}_1$ by collapsing Y_1 to l_1. By Lemma 4.3.1 and Corollary 4.3.2, it holds that $\mathcal{T}_2$ and $\mathcal{B}_2$ are induced by $\overline{N}_2$.

Continue this line of reasoning until all M_i's have been computed and let R denote the last index for which M_i is defined. Moreover, let $\overline{M}_i$ denote the component of $\overline{N}_i$ restricted to Y_i. Furthermore, let $\overline{N}_{i+1}$ be the network created from $\overline{N}_i$ by collapsing Y_i to l_i. It now holds for each i with $1 \leq i \leq R$ that Y_i is a minimal cut-arc set of $\overline{N}_i$, that $\mathcal{T}_i^r$ and $\mathcal{B}_i^r$ are induced by $\overline{M}_i$, that $M_i = \overline{M}_i$, and that $\mathcal{T}_i$ and $\mathcal{B}_i$ are induced by $\overline{N}_i$.

It will now be shown that $N_i = \overline{N}_i$ for all $i \in \{1, \ldots R\}$. First note that $N_R = M_R = \overline{M}_R$. Next, note that the networks $\overline{M}_R$ and $\overline{N}_R$ must be equal. So $N_R = \overline{N}_R$. Furthermore, it holds that $\overline{N}_R$ is the network created from the network $\overline{N}_{R-1}$ by collapsing Y_{R-1} to l_{R-1}. Note that this is the same as replacing the component $\overline{M}_{R-1}$ in $\overline{N}_{R-1}$ with the leaf l_{R-1}. Hence, $\overline{N}_{R-1}$ can be created by replacing leaf l_{R-1} in $\overline{N}_R$ with the network $\overline{M}_{R-1}$. As $N_R = \overline{N}_R$ and $M_{R-1} = \overline{M}_{R-1}$, it follows that $N_{R-1} = \overline{N}_{R-1}$. Continuing this line of reasoning, it follows that $N_1 = \overline{N}_1$, as promised. □

Algorithm 5: Fit Network

```
Input:
- Set of species $X$,
- Trinet set $\mathscr{T}$ on $X$,
Output:
- Network $\hat{N}$ on $X$.
1  $X_1 \leftarrow X$
2  $\mathscr{T}_1 \leftarrow \mathscr{T}$
3  $i \leftarrow 1$
4  while $X_i$ contains more than three elements do
5      $Y_i \leftarrow$ minimal sink-set of $\Omega_{\hat{i}}(\mathscr{T}_i)$
6      $\mathscr{T}_i^r \leftarrow \mathscr{T}_i$ restricted to $Y_i$
7      $M_i \leftarrow$ biconnected network on $Y_i$ derived from $\mathscr{T}_i^r$
8      $\mathscr{T}_{i+1} \leftarrow Y_1$ collapsed in $\mathscr{T}_i$
9      $l_i \leftarrow$ leaf that represents the leaves in $Y_i$ after collapsing
10     $X_{i+1} \leftarrow (X_i \setminus Y_i) \cup \{l_i\}$
11     $i \leftarrow i+1$
12 end
13 if $X_i$ contains two elements then
14     $\mathscr{B} \leftarrow$ union of all binet sets induced by trinets in $\mathscr{T}_{i-1}$
15     $\mathscr{B}' \leftarrow$ all binets which have all leaves in $X_i$
16     $M_i \leftarrow$ binet in $\mathscr{B}$ with highest multiplicity
17     $i \leftarrow i+1$
18 end
19 $i \leftarrow i-1$
20 $N_i \leftarrow M_i$
21 while $i \geq 2$ do
22     $N_{i-1} \leftarrow$ network created from $N_i$ by replacing $c_{i-1}$ with $M_{i-1}$
23     $i \leftarrow i-1$
24 end
25 $\hat{N} \leftarrow N_1$
26 return $\hat{N}$
```

5.5. IDENTIFYING TRINETS AND BINETS

Most steps of the algorithm require some properties of the networks in $\mathscr{T}$. These are properties such as a networks cut-arcs sets and generator. To optimize computing these properties, all recoverable level-2 trinets are constructed and their properties are derived. This list of networks and their properties are saved in a reference data set. Then, for each trinet in the input, a trinet in the reference data is found such that they are isomorphic. Furthermore, an isomorphism between them is constructed such that the properties of the reference trinet can be translated to the input trinet.

Constructing all trinets and binets is done by first constructing all recoverable biconnected level-2 trinets and binets from the generators in Figure 3.2. Next, the other recoverable level-2 trinets are constructed by combining two (possibly equal) biconnected level-2 binets. That is, take two such binets and replace a leaf in one of these binets with the other binet. Note that there are no other recoverable level-2 binets, as a recoverable binet must be biconnected.

Given two networks, it is possible to check if they are isomorphic using Algorithm 6. Note that this algorithm can be optimized by only looking for one isomorphism, or using properties such as the networks cut-arc sets if they are available.

Lemma 5.5.1. *Let $N = (V, E)$ and $M = (W, F)$ be phylogenetic networks with the same amount of nodes, leaves, and arcs. Then Algorithm 6 returns a list of all the isomorphism between N and M.*

Proof. First some notation for differentiating between different recursion steps is introduced. If R is a recursion step, then the recursive calls made during R are denoted with R_i. Similarly, the inputs and output of a recursive call R_i are denoted with v_i, w_i, L_i and Λ_i.

First, the notion of a 'sub-bijection' is introduced. A bijection $\Psi : V' \to W'$ is a *sub-bijection* of a bijection $\Phi : V \to W$ if:

$$\begin{aligned} V' &\subseteq V, & W' &\subseteq W, \\ \Psi(x) &= \Phi(x) & \forall x &\in V'. \end{aligned} \tag{5.22}$$

It is also said that Ψ can be *extended* Φ. Furthermore, if $\Psi' : V" \to W"$ is also a sub-bijection of Φ, and $V" \supseteq V'$, then Ψ' is called an *extension* of Ψ *respecting* Φ. Additionally, if V' is a proper subset of $V"$, then Ψ is called a *proper sub-bijection* of Φ. Lastly, if Φ is an isomorphism, than Ψ is also called a *sub-isomorphism.*

Now for the original statement, assume that N and M are isomorphic and let Φ be one of the isomorphism from N to M. It will be shown that the output $\overline{\Lambda}$ of the initial call $\overline{R}$ contains Φ.

It will be shown that if a recursive call R has a sub-isomorphism Ψ of Φ such that $\Psi(v) = w$ in its input list L then:

if v is not a leaf: there exists a recursive call R_{i_1} that has an extension Ψ_{i_1} of Ψ respecting Φ in its input list L_i

if v is a leaf: the list Λ that R returns equals L.

To this end, assume that for a recursive call R there exists a sub-isomorphism Ψ of Φ such that $\Psi(v) = w$. Clearly the condition in line 1 of the algorithm is false as L is not empty. Next, it follows from $\Psi(v) = w$ and the fact that Ψ is a sub-isomorphism that v and w have the same in and out-degrees. Therefore, if v is a leaf it follows that w is also a leaf and thus that the recursive call R returns L. If v is not a leaf the condition in line 9 is false. Additionally, by assumption Ψ can be extended to an isomorphism between N and M. From this, it follows that there exists a bijection β in B which is not in contradiction with Φ nor Ψ. Now let $b = \beta$ in line 14 and let $\phi = \Psi$ in line 15. Then the condition in line 16 is false. Furthermore, ϕ' in line 24 is an extension of Ψ respecting Φ. Therefore, it is possible to let $\Psi_{i_1} = \phi'$. It now follows that in the first iteration of the loop in lines 22-25 that $L_{current} = [\Psi_{i_1}]$: creating a recursive call R_{i_1} in line 24 with $L_{i_1} = [\Psi_{i_1}]$ as requested.

As R_{i_1} satisfies the same condition as R, the sub-isomorphism Ψ recurses through the nodes below v in a depth-first way whilst being extended on the way.

Now note that the initial list of bijections $\overline{L}$ contains a single bijection $\overline{\phi}$ which is a sub-isomorphism for any $\Phi = \Phi_j$. Therefore, this initial sub-isomorphism $\overline{\phi}$ recurses through all the nodes v below $\overline{v} = \rho$. Moreover, as in each recursive call $\overline{\phi}(v) = w$, it follows that $\overline{\phi}$ is extended until it becomes equal to Φ. So indeed, Φ is contained in the output $\overline{\Lambda}$ of the initial call $\overline{R}$. As this holds for any $\Phi = \Phi_j$, it follows that $\overline{\Lambda}$ contains all isomorphisms Φ_j.

Finally, it must be shown that $\overline{\Lambda}$ contains only isomorphisms. To this end, let Φ be any returned bijection. It must be shown that Φ is in fact an isomorphism, that is: (x, y) is an arc in N if and only if $(\Phi(x), \Phi(y))$ is an arc in M. Note that by the symmetry of the algorithm that it is sufficient to show that $(\Phi(x), \Phi(y))$ is an arc in M if (x, y) is an arc in N.

To this end, assume that (x, y) is an arc in N. Note that there must exists a sequence of recursive calls R_i and a corresponding sequence of sub-bijections Ψ_i through which Φ is created. Now let R_{i_x} be the recursive call in which the input node v is x and denote the input node w with z. Then the sub-bijection Ψ'_{i_x}, the first extension of Ψ_{i_x}, must contain a bijection between the children of x and z. As y is a child of x, it follows that $\phi(y)$ is a child of z. Now it follows from $z = \phi(x)$ that $\phi(y)$ is a child of $\phi(x)$. So indeed, $(\phi(x), \phi(y))$ is an arc in M.

□

Algorithm 6: find_isomorphisms

```
Input:
- Recoverable level-2 phylogenetic network $N = (V, E)$ with root $\rho$,
- Recoverable level-2 phylogenetic network $M = (W, F)$ with root $\kappa$,
- node $v$ in $V$ (initially $\rho$),
- node $w$ in $W$ (initially $\kappa$),
- list $L$ of bijections $\psi : V' \to W'$ where $V' \subseteq V$ and $W' \subseteq W$ (initially $[\{\rho \mapsto \kappa\}]$),
Output:
- (possibly empty) list $\Lambda$ of bijections $\phi : V \to W$.

if L is emtpy then
    return L
end
if $v$ is a leaf and $w$ is a leaf then
    return L
end
$c \leftarrow$ direct children of $v$
$d \leftarrow$ direct children of $w$
if $c$ and $d$ do not contain the same amount of elements then
    return empty list
end
$L_{result} \leftarrow$ empty list
B $\leftarrow$ all bijections between $c$ and $d$
for bijection $b$ in $B$ do
    for bijection $\psi$ in $L$ do
        if $b$ in in contradiction with $\psi$ then
            go to next iteration
        end
        $\psi' \leftarrow$ copy of $\psi$
        update $\psi'$ with the bijection b
        $L_{current} \leftarrow [\psi']$
        for $x$ in $c$ do
            $y \leftarrow$ image of $x$ in b
            $L_{current} \leftarrow$ compare_networks($N$, $M$, $x$, $y$, $L_{current}$)
        end
        add $L_{current}$ to $L_{result}$
    end
end
return $L_{result}$
```

6 EXPERIMENTS

It has been shown that the `Tril2Net` algorithm produces the desired output network in case the input trinet set is induced by a recoverable level-2 phylogenetic network. However, the latter will usually not be the case in practice. Hence, it is of interest to analyse what happens to the output network when noise is introduced in the input trinet set.

In this chapter, different types of noise that can exist in a network set are introduced. This makes it possible to categorize different input trinet sets based on the levels and types of noise they contain. Thereafter, several ways that networks and network sets can be compared to each other are defined. Next, it is explained how problem instances are sampled and how they are analysed. Finally, different categories of trinet sets are sampled and put into the `TriL2Net` algorithm and the output is analysed.

6.1. NETWORK SET METRICS

The input trinet set might contain some contradicting or incorrect trinets, or even miss some completely. To quantify this, some metrics about the quality of network sets in general are defined below:

Definition 6.1.1 (Network set metrics)**.** Let $\mathcal{N}$ be a set of networks on n species. Then:

- $\mathcal{N}$ has *redundancy* if there exist two networks $N_1, N_2 \in \mathcal{N}$ such that they describe the same n-set. The *redundancy* of $\mathcal{N}$ is defined as:

$$redundancy(\mathcal{N}) = \frac{volume(\mathcal{N}) - coverage(\mathcal{N})}{coverage(\mathcal{N})}.$$

  Remember that volume is the sum of all the multiplicities of the networks in $\mathcal{N}$ and that coverage is the amount n-sets described by $\mathcal{N}$. Furthermore, $\mathcal{N}$

is called *non-redundant* if its redundancy is zero.

- $\mathcal{N}$ is called *dense* if for every n-set X in $\mathcal{N}_X$ there exists at least one network $N \in \mathcal{N}$ such that $N_X = X$. The *density* of a network set $\mathcal{N}$ is defined as:

$$density(\mathcal{N}) = \frac{coverage(\mathcal{N})}{\binom{|\mathcal{N}_X|}{n}}.$$

  Hence $\mathcal{N}$ is dense if its density is one.

- $\mathcal{N}$ has *a depth-0 inconsistency* if there exist two networks $N_1, N_2 \in \mathcal{N}$ such they describe the same n-set but are not equal. The depth-0 inconsistency of $\mathcal{N}$ is defined as:

$$inconsistency_0(\mathcal{N}) = \frac{size(\mathcal{N}) - coverage(\mathcal{N})}{coverage(\mathcal{N})}.$$

- $\mathcal{N}$ has a *depth-d inconsistency* if there exists two networks $N_1, N_2 \in \mathcal{N}$ such that there is a depth-0 inconsistency in the union of the $(n-d)$-network sets $\mathcal{N}_{n-d}(N_1)$ and $\mathcal{N}_{n-d}(N_2)$: the network sets containing all exhibited networks with $(n-d)$ leaves of N_1 and N_2 respectively. This inconsistency is defined as:

$$inconsistency_d(\mathcal{N}) = \frac{size(\mathcal{N}^{(d)}) - coverage(\mathcal{N}_X^{(d)})}{coverage(\mathcal{N}_X^{(d)})},$$

  where $\mathcal{N}^{(d)}$ is the union of all $(n-d)$-network sets induced by each $N \in \mathcal{N}$:

$$\mathcal{N}^{(d)} = \bigcup_{N \in \mathcal{N}} \mathcal{N}_{n-d}(N).$$

- $\mathcal{N}$ is called *consistent* if $inconsistency_d(\mathcal{N}) = 0$ for $0 \le d \le n-2$.

- $\mathcal{N}$ is called *concise* if it is non-redundant, dense and consistent.

Note that induced network sets are always concise.

6.2. CONSISTENCY METRICS

The objective of the algorithm is to assist phylogenists by creating a larger level-2 phylogenetic network from multiple smaller ones. That is, given a set of networks $\overline{\mathcal{N}}$ the algorithm should return a phylogenetic network $\hat{N}$ that is consistent with many of the networks in $\overline{\mathcal{N}}$.

There are several ways to measure this consistency. As the set of exhibited trinets of a recoverable level-2 phylogenetic network is unique (van Iersel and Moulton 2014), it makes sense to look at the *trinet consistency score* (Oldman et al. 2016):

$$\frac{|\mathcal{N} \cap \mathcal{T}(\hat{N})|}{|\mathcal{N}|}, \qquad \mathcal{N} = \bigcup_{N_i \in \overline{\mathcal{N}}} \mathcal{T}(\overline{N_i}).$$

As the algorithm is based on the trinets, we do not only consider this trinet consistency score but also the *triplet consistency score* (Huber, van Iersel, et al. 2011):

$$\frac{|\dot{\mathcal{N}} \cap \dot{\mathcal{T}}(\hat{\mathrm{N}})|}{|\dot{\mathcal{N}}|}, \qquad \dot{\mathcal{N}} = \bigcup_{\overline{N_i} \in \overline{\mathcal{N}}} \dot{\mathcal{T}}(\overline{N_i}).$$

Remember that $\dot{\mathcal{T}}$ represents the set of exhibited triplets of $\overline{N_i}$. Another interesting metric is the *cluster consistency score*:

$$\frac{|\ddot{\mathcal{N}} \cap \ddot{\mathcal{N}}(\hat{\mathrm{N}})|}{|\ddot{\mathcal{N}}|}, \qquad \ddot{\mathcal{N}} = \bigcup_{\overline{N_i} \in \overline{\mathcal{N}}} \ddot{\mathcal{N}}(\overline{N_i}).$$

Here $\ddot{\mathcal{N}}(\overline{N_i})$ represents the set of exhibited clusters of $\overline{N_i}$. Furthermore, the *cut-arc set consistency score* is of interest:

$$\frac{|CA \cap CA(\hat{\mathrm{N}})|}{|CA|}, \qquad CA = \bigcup_{\overline{N_i} \in \overline{\mathcal{N}}} CA(\overline{N_i}),$$

where $CA(N)$ is the set of non-trivial cut-arc sets of a phylogenetic network N.

Note that the trinet consistency score is the same for every iterating method. This is not the case, however, for the triplet and the cluster consistency scores.

6.3. SAMPLING METHODS

The algorithm is tested on input which `TriLoNet` was tested on and is tested on newly sampled input. This section details how this newly sampled input is generated.

6.3.1. NETWORK SAMPLING

To generate a trinet set, first a recoverable level-k phylogenetic network must be generated. This is done as follows. First an ordered set Σ of l biconnected level-k phylogenetic networks is generated. Each of these networks is created by taking a level-k generator and adding m leaves to each of its edge sides and two leaves to each of its reticulation sides. Next, let $N = N_1$ and iteratively replace a random leaf x of N with the network N_i, $i = 2, \ldots, l$. Now the number of leaves of N is reduced to n by randomly terminating leaves.

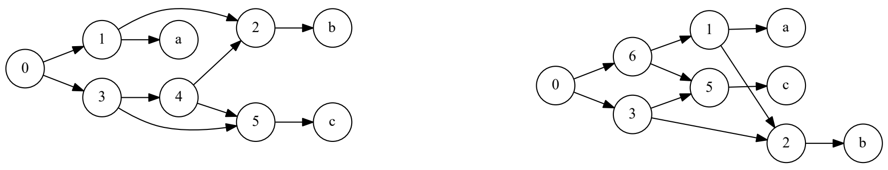
Note that in case reticulation leaves are terminated the number of reticulations is reduced as well. Therefore, if a large percentage of the leaves is terminated it can happen that the sampled network is of a lower level. Moreover, a large percentage of leaves is terminated if l and m are high compared to n. Lastly, if m is high, the probability that a reticulation leaf is terminated is smaller than if m is low. In the remainder of this chapter, level-1 networks are sampled using $m = 4$ and $l = \lceil \frac{3}{14} n \rceil$. For level-2, the networks are sampled using $m = 2$ and $l = \lceil \frac{3}{22} n \rceil$.

6.3.2. TRINET SET SAMPLING

Using this sampled level-1 or level-2 phylogenetic network N, an induced trinet set $\mathcal{T} = \mathcal{T}(N)$ can be computed. In order to induce noise in this set, two distortion filters are introduced.

The first filter is the *uniform replacement filter*, which replaces a trinet $T \in \mathscr{T}$ with another trinet (on the same species) at random. The probability that a trinet is replaced is denoted with Q_u. This is the method that Oldman et al. (2016) used to test the TriLoNet algorithm. In case that the sampled network is generated using level-k generators, the trinets may only be replaced with level-k trinets.

The second filter is the *tail move filter*. Here, a tail move is as in Janssen et al. (2018): moving the tail of an arc to a second arc. An example of a tail move can be seen in Figure 6.1. Note that a tail move preserves the number of reticulations a trinet contains, but can alter its level. This is done in such a way that the trinet's level is at most 2, both for trinet sets induced by a network sampled from level-1 and level-2 generators. The probability that a tail move is performed on a trinet is denoted using Q_{tm}. The arc that is moved and the arc that it is moved to are chosen at random, but in such a way that the trinet is altered into a different trinet with level at most two. This filter is chosen because a tail move changes a trinet relatively little.



(a) The original trinet

(b) The trinet obtained by moving the tail of arc (4,5) to the arc (0, 1).

Figure 6.1: An example of a tail-move.

It follows that the input trinet set is generated from a recoverable level-2 phylogenetic network N using two parameters: Q_u and Q_{tm}. Note that both filters are only used once.

6.4. EXPERIMENT STRUCTURE

In this section, the experiment structure is outlined. First, a level-1 or level-2 network $\overline{N}$ is sampled. From this network, a distorted trinet set is generated using the filters Q_u and Q_{tm}. This network combined with the distorted trinet set can be seen as a problem instance. The way `TriL2Net` solves this instance depends on the settings of the algorithm. These settings define which way of network set iterating should be used for different parts of the algorithm. For example, it defines which method is used when computing p_k of Equation (5.1), the fraction of trinet in a trinet set which are strictly level-k. This iteration method can be different for the computation of $\mathbf{\Omega}$, p_k, p_g, $p_{x,I}$ and $p_{x,y}$.

Hence, the inputs of an experiment are the network sampling parameters, the trinet set sampling parameters and the five iteration methods. The outputs of this experiment are the generated network $\overline{N}$, the distorted trinet set $\mathscr{T}$ and the output network $\hat{N}$.

Next, the consistency scores can be computed from $\overline{\mathscr{N}}$ and $\hat{N}$. Note that $\overline{\mathscr{N}}$ consists of only $\overline{N}$.

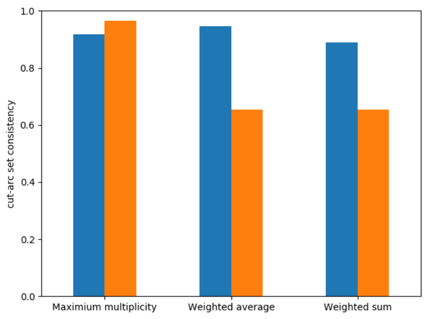
In the coming sections, the relations between the runtime and consistency scores to the redundancy, density and inconsistencies of the input trinet set are analysed for phylogenetic networks with different properties.

6.5. DETERMINING THE BEST SOLVER SETTINGS

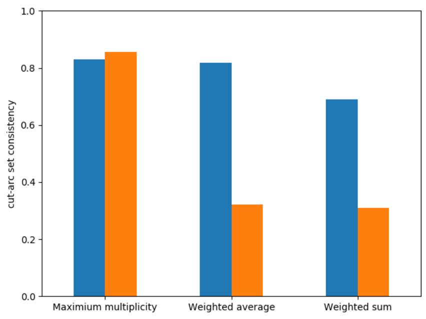
First, it is determined what the impact of different solver settings are on the consistency scores. As the first part of the algorithm determines the cut-arc sets of the algorithm, the impact of different settings on the cut-arc sets consistency is analysed first. Thereafter, the impact of the other settings on the the trinet consistency score is analysed.

6.5.1. OPTIMIZING FOR CUT-ARC CONSISTENCY

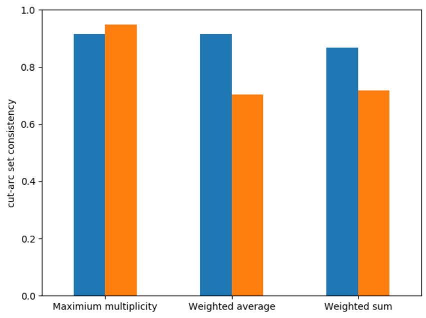
The settings that change the way `TriL2Net` determines the cut-arc sets are the iterator method used when computing the weighted auxiliary graph $\mathbf{\Omega}$ and the method used that determines a set from this graph: Algorithm 1 or Algorithm 2. In the figures below, it can be seen what the average cut-arc consistency is for different settings, levels and different values of Q_u and Q_{tm}.



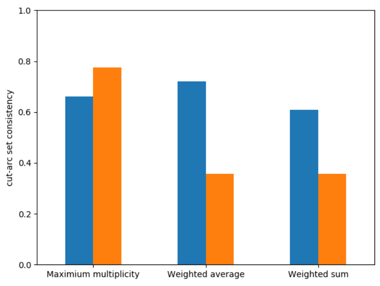
(a) Networks sampled using level-1 generators, $Q_{tm} = Q_u = 0.05$.



(b) Networks sampled using level-1 generators, $Q_{tm} = Q_u = 0.15$.



(c) Networks sampled using level-2 generators, $Q_{tm} = Q_u = 0.05$.



(d) Networks sampled using level-2 generators, $Q_{tm} = Q_u = 0.15$.

Figure 6.2: The average cut-arc set consistency score for six different solver settings. For the blue bars Algorithm 2 is used to compute the minimal cut-arc sets. For the orange bars Algorithm 1 is used instead. The average is taken over nine networks N with $n \in \{20, 25, 30\}$. The trinet sets are then induced by the network N and passed through the noise filters with parameters Q_{tm} and Q_u as defined in the caption. Note that the same trinet sets have been used for different bars in the same graph.

The blue bars are the iterators (specified on the x-axis) that use Algorithm 2 to derive the minimal cut-arc set. The orange bars use Algorithm 1 instead. The input networks were sampled using level-1 generators for the top two graphs and using level-2 generators for the bottom two graphs. The noise levels Q_{tm} and Q_u are 0.05 for the left two graphs and 0.15 for the right two graphs.

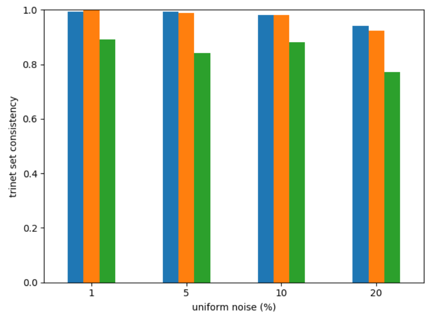
It can be seen that the maximum multiplicity iterator and Algorithm 2 perform better for all combinations of the level of the network and the noise percentage. However, for the maximum multiplicity iterator, Algorithm 1 performs better. It follows that using the maximum multiplicity iterator in combination with Algorithm 1 gives the best result. Hence, in the rest of this chapter these settings are used. Note that it is possible that the other methods work better for higher levels of noise, as it is then less likely that the maximum multiplicity method yields the original data.

Moreover, it can be seen that the cut-arc set consistency decreases when the noise is increased, as would be expected. Furthermore, the cut-arc set consistency is also lower for level-2 than for level-1, especially for higher levels of noise tested. This could be due to a different number and size of the cut-arc sets present in the generated level-1 compared to the level-2 networks. Another potential cause could be, that for level-2 networks, the uniform filter can replace a trinet with a trinet from a larger set of trinets. Lastly, it is possible that tail-moves have more impact on the cut-arc sets of level-2 trinets than on the cut-arc sets of level-1 trinets.

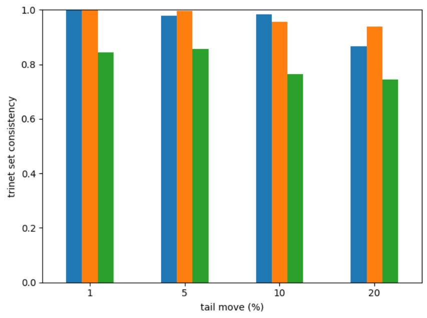
6.5.2. OPTIMIZING THE OTHERS SETTINGS FOR TRINET CONSISTENCY

Next, the other settings of the algorithm are optimized for trinet consistency. Here, three alternatives are juxtaposed. Each alternative uses one of the three iteration methods for the computation of each of the values p_k, p_g, $p_{x,I}$ and $p_{x,y}$. The result can be seen in Figure 6.3.

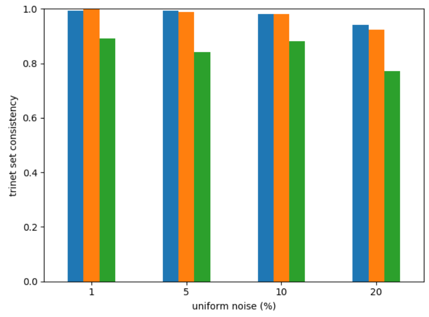
The blue, orange and green bars represent the maximum multiplicity, weighted average and the weighted sum iterator respectively. The trinet sets of the upper two graphs are generated using level-1 generators, the sets of the bottom two graphs using level-2 generators. For the left two graphs uniform noise was used to distort the trinet set, for the right two graphs this was tail move noise.



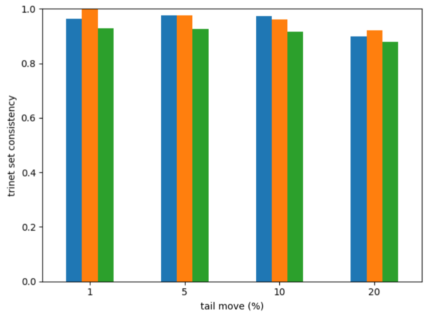
(a) Networks sampled using level-1 generators, $Q_{tm} = 0$.



(b) Networks sampled using level-1 generators, $Q_u = 0$.



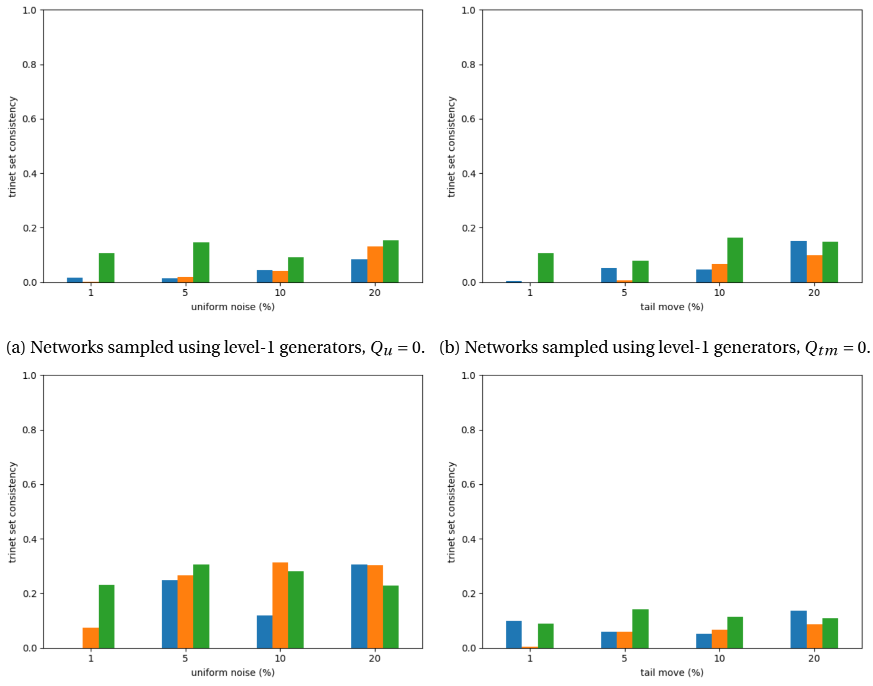
(c) Networks sampled using level-2 generators, $Q_{tm} = 0$.



(d) Networks sampled using level-2 generators, $Q_u = 0$.

Figure 6.3: The average trinet consistency score for three different solver settings and four different noise levels. The blue bars, orange bars and green bars represent that the maximum multiplicity, weighted average and the weighted sum iterators have been used to compute p_k, p_g, $p_{x,I}$ and $p_{x,y}$ respectively. The average is taken over 18 networks with $n \in \{15, 20, 25\}$. For level-1, the networks are sampled using $m = 4$ and $l = \lceil \frac{3}{14} n \rceil$. For level-2, the networks are sampled using $m = 2$ and $l = \lceil \frac{3}{14} n \rceil$. The trinet sets are then induced by N and passed through the noise filters with parameters Q_u and Q_{tm} as defined on the x-axis. Note that different trinet sets have been used for the different bars in the same graph.

Note that the consistency scores should not increase if the noise is increased, but that this does happen for the weighted sum and the maximum multiplicity iterator. By looking Figure 6.4, which contains the standard deviation of the data sets in Figure 6.4, it can be concluded that this is the result of too small sample sizes.



(a) Networks sampled using level-1 generators, $Q_u = 0$. (b) Networks sampled using level-1 generators, $Q_{tm} = 0$.

(c) Networks sampled using level-2 generators, $Q_u = 0$. (d) Networks sampled using level-2 generators, $Q_{tm} = 0$.

Figure 6.4: The respective standard deviation of the data sets of Figure 6.3

Nonetheless, it can be seen that the average trinet consistency score remains close to 1 for uniform noise levels of $< 10\%$ for level-1, and for tail move levels of $< 10\%$ for level-1 *and* level-2. Moreover, if the high value of the average trinet set consistency for $Q_u = 20\%$ and the weighted sum iterator is attributed to the small sample size, it can be concluded that the maximum multiplicity and the weighted average iterator perform best. Again, note that this result does not need to translate to higher values of noise. Lastly, it can be seen that the average trinet consistency scores are slightly lower for the level-2 networks. This might be the result of the small data set. If not, one of the possible causes might be that there are more level-2 networks, hence any two random level-2 networks are less likely to be alike.

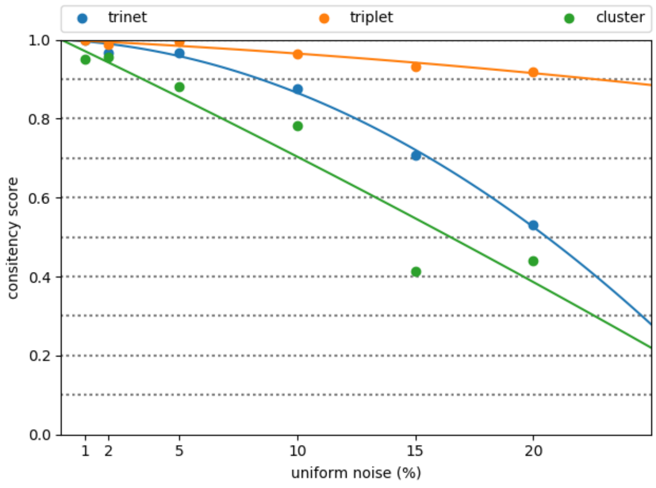
In the remainder of the experiments of this chapter, the weighted average iterator will be used to compute p_k, p_g, $p_{x,I}$ and $p_{x,y}$.

6.5.3. PERFORMANCE AND COMPARISON STUDY

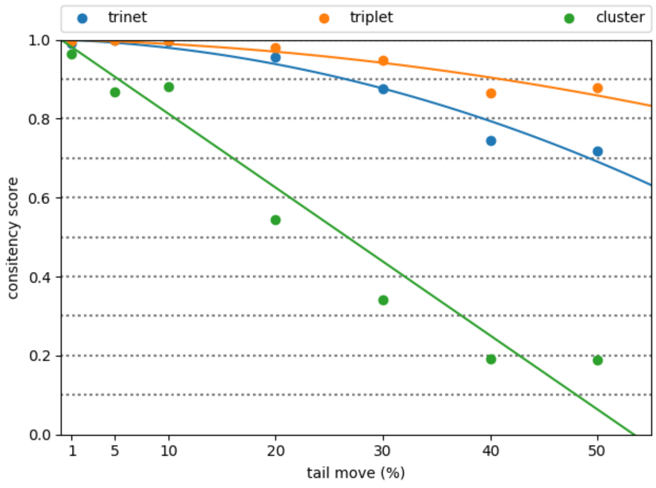
Using these optimal settings found in the previous subsections, we will now analyse the performance of `TriL2Net`. In Figure 6.5, the average trinet, triplet and cluster consistency scores are plotted against the level of noise. Furthermore, a polynomial in the form of:

$$1 - a_1 x - a_2 x^2 - a_3 x^3 - a_4 x^4,$$

with $a_i \geq 0$ is fitted for each score. In the upper figure, the noise is generated using the uniform noise filter. In the lower figure this is done using tail moves. The blue, orange and green line denote the trinet, triplet and cluster consistency respectively. Note that the maximum noise percentage in Figure 6.5a is much higher than in Figure 6.5b. We chose to increase this range, as the scores were still close to 1 for tail move levels of up to 20%.



(a) Networks sampled using level-2 generators, $Q_{tm} = 0$.



(b) Networks sampled using level-2 generators, $Q_u = 0$.

Figure 6.5: The average consistency scores of `TriL2Net` for different levels and types of noise. The averages are taken over 18 level-2 networks N with $n \in \{15, 20, 25\}$. The trinet sets are induced by N and passed through the noise filters with parameters Q_u and Q_{tm} as defined on the x-axis.

It can be seen that the average triplet consistency is stable and remains above 85% for all plotted noise levels. The average trinet consistency score remains above 85% for uniform noise levels up to 10% and for tail move noise levels up to around 30%. The cluster consistency score declines rapidly for both uniform and tail move noise. This might be explained by the fact that a small increase in the number of biconnected components or reticulation leaves has a large effect on the exhibited clusters of a network. Lastly, it can be seen that the algorithm performs better for noise induced by tail moves than for noise induced by the uniform filter. This is due to the fact that a tail-move is a less radical change than replacing a trinet with a random other trinet. The difference in the impact of these filters is analysed in more depth in Section 6.7.

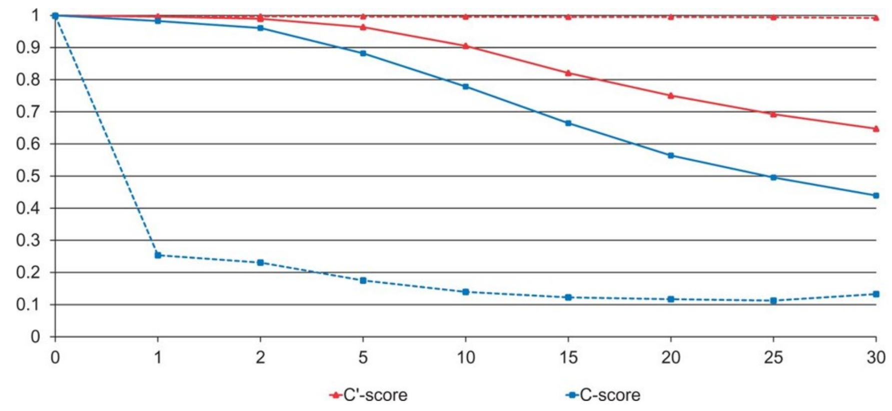


Figure 6.6: The performance of the TriLoNet and the Lev1athan algorithm on level-1 phylogenetic networks. The averages are taken over 100 level-1 networks with $20 \leq n \leq 100$ for each percentage of uniform noise. The solid line represents TriLoNet and the dashed line represents Lev1athan. The lines with triangles are the triplet consistency scores, the lines with squares the trinet consistency scores. More information on those the sampled networks can be found in Oldman et al. (2016).

In Figure 6.6, the average trinet and triplet consistency scores of TriLoNet and Lev1athan are plotted against the percentage of uniform noise (Oldman et al. 2016). The networks used to create this graph are all level-1 and are sampled in a similar, but not equivalent, fashion as is done in this paper. The uniform noise filter that is used is exactly the same. The lines with triangles denote the triplet consistency score, and the lines with squares denote the trinet consistency score. The dashed lines are Lev1athan's and the solid lines are TriLoNet's.

Lev1athan works with triplets (Huber, van Iersel, et al. 2011), whereas TriLoNet works with level-1 trinets. This can be recognized by the fact that Lev1athan's triplet score is higher but its trinet score is lower than TriLoNets's. As the trinets exhibited by a level-1 phylogenetic network N encode the network N, Oldman et al. (2016) conclude that the networks produced by TriLoNet are topologically more similar to the input networks.

When comparing the scores of TriL2Net to the ones of TriLoNet, it can be seen that Tril2Net performs better for the triplet consistency score and about the same for the trinet consistency score. There are two possible explanations for this. The scores in

Figure 6.6 are computed using a large sample of different size level-1 phylogenetic networks, whereas the scores in Figure 6.5 are computed using a smaller sample of small size level-2 phylogenetic networks. Hence, on the one hand, it might be that it is more difficult to achieve high consistency scores for larger networks. On the other hand, it might be more difficult to achieve a high consistency score for level-1 networks than for level-2 networks. Whether either of these two is the case is unknown. We expect that it is indeed possible that larger networks are more difficult to reconstruct. However, we do not expect that level-2 networks are easier to reconstruct then level-1 networks, as their topology is more complex. Both possibilities need further investigation.

The average runtime of the algorithm for non-redundant dense level-2 trinet sets on n leaves can be seen in Figure 6.7. The theoretical upper bound for worst-case runtime follows from Theorem 5: $O(n \cdot t^2 + n^5)$. As $t = O(n^3)$ for non-redundant dense trinet sets, it follows that the average runtime should be $O(n^7)$. After fitting polynomials of different orders, it was found that the following fifth order polynomial fits the points well:

$$1.54n + 1.64 \cdot 10^{-2} n^4 + 6.70 \cdot 10^{-6} n^5.$$

The fact that this is lower than the theoretical runtime can be attributed to the crude bounds used in the lemmas concerning the restricting and collapsing of the trinet sets: Lemma 5.2.1 and Lemma 5.3.1 respectively. As in practise there are very little contradicting trinets, this restricting and collapsing is much faster than the given bound $O(t^2) = O(n^6)$. If the term $n \cdot t^2$ is neglected, the worst-case runtime of the algorithm becomes $O(n^5)$. Note that one of these n's comes from the maximum number of biconnected components a phylogenetic network can have. As this number is much lower for the sampled level-2 phylogenetic networks, it makes sense that the coefficient of the fifth degree is small compared to the coefficient of the fourth degree.

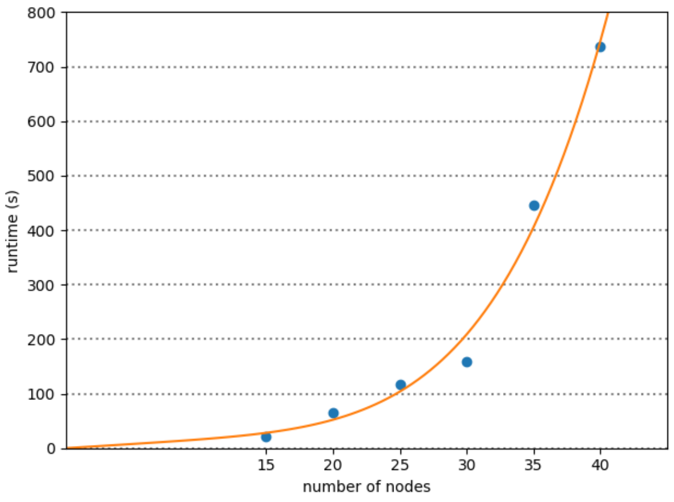


Figure 6.7: The average runtime of the `TriL2Net` algorithm. The averages are taken over four level-2 phylogenetic networks per $n \in \{15, 20, 25, 30, 35, 40\}$. The fitted curve is approximately $1.54n + 1.64 \cdot 10^{-2} n^4 + 6.70 \cdot 10^{-6} n^5$.

6.6. BIOLOGICAL DATA

In this section, we apply `TriL2Net` to biological data, using the optimal settings found in Subsection 6.5.1. Unfortunately, no algorithm that computes level-2 trinets from sequence data exists yet, and designing such an algorithm is non-trivial. Therefore, the two level-1 trinet sets used to illustrate `TriLoNet` are used instead. These sets concern six Giardia subtypes and 25 HBV subtypes respectively.[1] The depth-1 inconsistency of both these sets is analysed and a tail move filter value that corresponds with this inconsistency is deduced from Figure 6.10b. The tail move filter is chosen as we believe that the noise this filter induces is more consistent with the noise in the biological data. Next, both sets are solved for using `Tril2Net` and the results are compared to `TriLoNet`'s results.

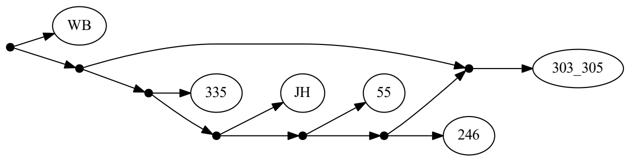
6.6.1. GIARDIA

The Giardia trinet set $\mathscr{T}_{Giardia}$ on six subtypes generated using `TriLoNet` consists of 20 trinets all on different leaf sets. Hence this set is dense, non-redundant and depth-0 consistent. It does, however, have a depth-1 inconsistency of 60.0%. The network derived by `TriLoNet` can be seen in Figure 6.8a (Oldman et al. 2016). The trinet and triplet consistencies of this network are 55.0% and 80.0% respectively.

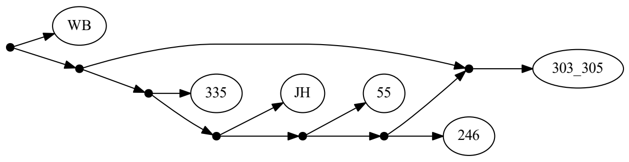
The two networks derived by `Tril2Net` can be seen in Figures 6.8b and 6.8b. It derives two networks due to the fact that some choices are made at random in case of ties.

[1]The sequence data for these can be found in the zip-file on https://www.uea.ac.uk/computing/TriLoNet, the files used here are Cooper and Bollyky respectively. This zip-file also contains the algorithm used to create the trinets. More details on the data and the algorithm used to construct these sets can be found in Oldman et al. (2016)

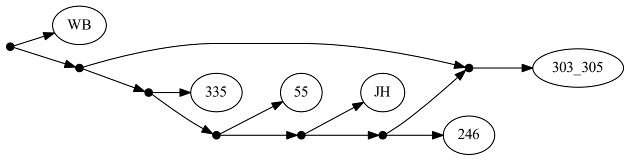
Figure 6.8b is the same as the one `TriLoNet` produces. Figure 6.8c is slightly different: the subtypes JH and 55 are switched. This switch does not change the trinet consistency, in fact both networks cover the same trinets that are in $\mathscr{T}_{Giardia}$. The switch does lower the triplet consistencies slightly to 77.5%. Note that the tail-move distance conform (Janssen et al. 2018) is one. This difference comes from the fact that these two algorithms order the leaves in a slightly different way and the fact that the trinet set $\mathscr{T}_{Giardia}$ is inconsistent in the ordering of the leaves JH and 55: an equal number place JH above 55 as the other way around.



(a) The network derived by `TriLoNet`. This network has a 55.0% trinet and a 80.0% triplet consistency score.



(b) The first of the two networks derived by `TriL2Net`. This network has a 55.0% trinet and a 80.0% triplet consistency score.



(c) The second of the two networks derived by `TriL2Net`. This network has a 55.0% trinet and a 77.5% triplet consistency score.

Figure 6.8: The networks derived by `TriLoNet` and `TriL2Net` for the Giardia trinet set $\mathscr{T}_{Giardia}$.

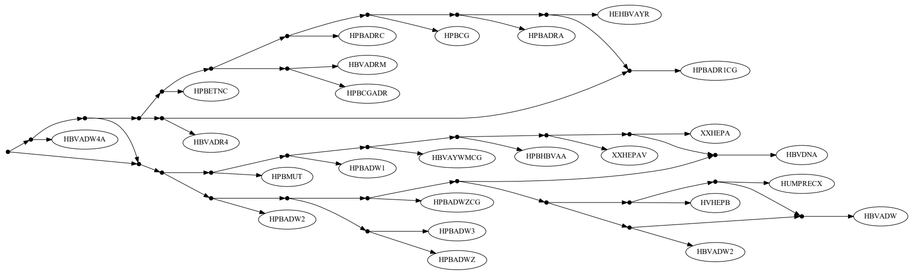
6.6.2. HBV

The HBV trinet set $\mathscr{T}_{HBV}$ on 25 subtypes generated using `TriLoNet` consists of 2300 trinets all on different leaf sets. Hence this set is dense, non-redundant and depth-0 consistent. It does, however, have a depth-1 inconsistency of 91.0%. The network derived by `TriLoNet` can be seen in Figure 6.9a (Oldman et al. 2016). The trinet and triplet consistencies of this network are 15.5% and 70.0% respectively. In Figure 6.9b, the network generated by `TriL2Net` can be seen. Note that `TriL2Net` always creates the same network for this data set. This network's trinet and triplet consistency score are slightly higher: 17.0% and 72.1% respectively.

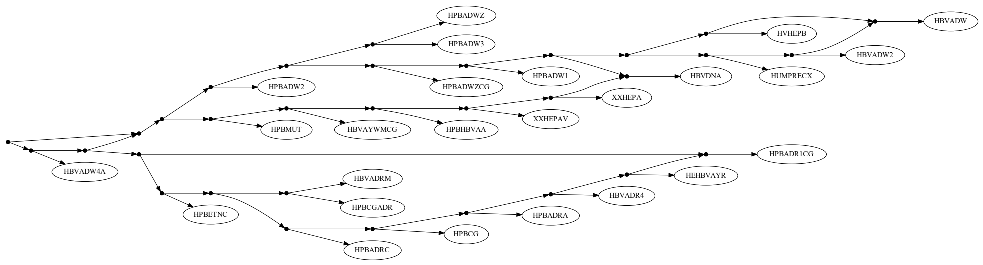
Both algorithms find the same cut-arc sets and reticulation leaves. However, `TriL2Net` has placed the following subtypes differently:

- HUMPRECX is placed on a different side, above HBVADW2 instead of below HVHEPB;
- HPBADW1 is placed on a different side, above HPBADWZCG instead of between HPBHBVAA and HPBMUT;
- HBVADR4 is placed on a different side, between HEHBVAYR and HPBADRA instead of alone on the other side.

These differences can be attributed to the fact that `TriL2Net` uses a different way to assign the leaves to different sides.



(a) The network derived by `TriLoNet`. This network has a 15.5% trinet and a 70.0% triplet consistency score.

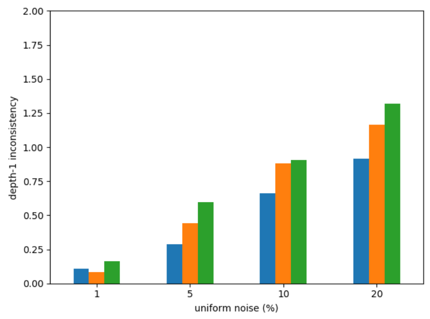


(b) The network derived by `TriL2Net`. This network has a 17.0% trinet and a 72.1% triplet consistency score.

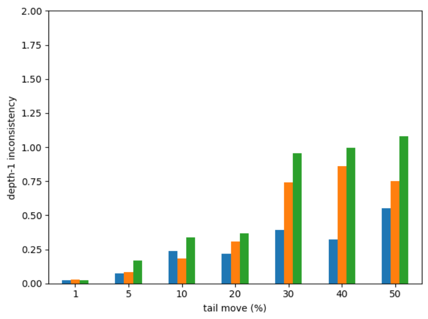
Figure 6.9: The networks derived by `TriLoNet` and `TriL2Net` for the HBV trinet set $\mathscr{T}_{HBV}$. Larger versions of these figure can be found in Appendix A

6.7. IMPACT OF DISTORTION FILTERS ON THE DEPTH-1 INCONSISTENCY

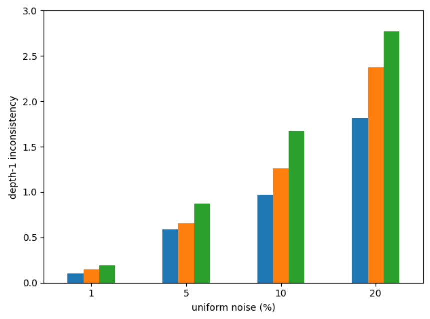
The previous section shows that both `TriLoNet` and `TriL2Net` have difficulty creating a network N with a high trinet consistency score for trinet sets generated from biological data. These generated trinet sets are dense, non-redundant and depth-0 consistent but do have a high depth-1 inconsistency. This section investigates which values of Q_{tm} and Q_u create trinet sets with similar depth-1 inconsistency levels.



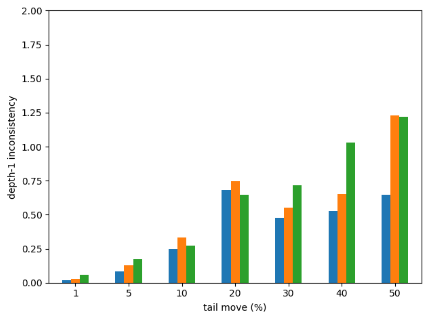
(a) Networks sampled using level-1 generators, $Q_{tm} = 0$



(b) Networks sampled using level-1 generators, $Q_u = 0$



(c) Networks sampled using level-2 generators, $Q_{tm} = 0$



(d) Networks sampled using level-2 generators, $Q_u = 0$

Figure 6.10: The average depth-1 inconsistency for different networks sizes and noise types. The blue bars, orange bars and green bars represent networks containing 15, 20 and 25 leaves respectively. The average is taken over five networks. For level-1, the networks are sampled using $m = 4$ and $l = \lceil \frac{3}{14} n \rceil$. For level-2, the networks are sampled using $m = 2$ and $l = \lceil \frac{3}{14} n \rceil$. The trinet sets are then induced by N and passed through the noise filters with parameters Q_{tm} and Q_u as defined on the x-axis.

The figures above show the depth-1 inconsistency for distorted trinet sets sampled using different generators and noise levels. The sets in the upper two graphs are sampled using level-1 generators and the sets in the lower two using level-2 generators. The sets in the left two graphs are distorted using a uniform filter and the sets in the right two graphs using a tail move filter. The blue bars, orange bars and green bars represent networks containing 15, 20 and 25 leaves respectively. Note that the bottom left graph has a larger y-range.

It can be seen that, as expected, the depth-1 inconsistency increases as the noise increases. Moreover, trinet sets on more species also have a higher depth-1 inconsistency for the same level of noise. Moreover, the uniform filter generates more depth-1 inconsistency than the tail move filter does. This is expected, as a tail-move results in a trinet more consistent with the original than a random trinet. Hence the binets are also relatively consistent with the originals. The depth-1 inconsistency is also larger for the level-2 networks. This is probably due the fact that there are more possible level-2 binets and trinets. Therefore, it makes sense that *TriL2Net* has lower consistency scores than `TriLoNet` for the same noise levels, as was found in Section 6.5.3.

Next, these results are translated to the biological data sets of the previous section. The HBV trinet set contains 25 species and has a depth-1 inconsistency of 91%. This coincides with either a uniform noise level of 10% or a tail move level of 30%. The average trinet consistency scores for these levels of noise follow from Figure 6.5: around 85% and 88% for uniform noise and tail move noise respectively. This is much higher than the trinet consistencies found for the biological data sets. It could be that these two biological data sets are outliers. However, it is more likely that this depth-1 inconsistency on its own is not a reliable measure for the amount of noise in a trinet set.

7 Conclusion and further research directions

We have presented an algorithm that constructs a level-2 phylogenetic network on n species from a set of recoverable level-2 trinets $\mathscr{T}$. The theoretical worst case runtime of this algorithm is $O(n|\mathscr{T}|^2 + n^5)$, but its average runtime for non-redundant dense trinet sets reduces to $O(n^5)$. Moreover, we have proven that, in case the set of trinets is induced by a recoverable level-2 network, this algorithm in fact reconstructs this network. To be able to do this, we have proven that the minimal cut-arc sets of such level-2 networks are precisely the minimal sink-sets of their respective auxiliary graphs. Moreover, we have conjectured that this result can be extended to level-k. We think this can be proven by finding a directed graph on the reticulation leaves such that a trinet containing two adjacent reticulation leaves is biconnected. Similar results have been proven using hierarchies by Huber, Moulton, and Wu (2019) for 2-terminal phylogenetic networks: networks containing at most two reticulation leaves per biconnected component.

Moreover, it was analysed which heuristics work best in various scenarios. It was found that the best way to determine the minimal cut-arc sets is to use the maximum multiplicity iterating method to compute the weighted auxiliary graph. Similarly, Algorithm 1 must be used to find a minimal sink-set in one of the augmented auxiliary graphs. Note that this algorithm is the same as the one used in `TriLoNet`, but that the used iterating method is different. Furthermore, it was found that it is best to use the weighted average iterating method when calculating the other properties of the network. Note that these results follow from preliminary experiments on trinet sets on roughly 20 species. Hence, further experiments on more and on larger networks must be performed to substantiate and extend these results.

When comparing the consistency scores of `TriL2Net` for level-2 networks to the consistency scores of `TriLoNet` for level-1 networks, we found that `TriL2Net` performs slightly better. This difference is most notable in the trinet consistency score. This difference might be attributed to the different ways the data is sampled. The number and size of the biconnected components might be of importance, for example. To better establish the difference in performance of these two algorithms, both algorithms should be run on the same data sets. Furthermore, it would be interesting to analyse the impact of the number, size and type of the biconnected components on the performance, too.

Two biological datasets have been studied using both `TriLoNet` and `TriL2Net`. These datasets concern Giardia subtypes and HBV subtypes. We found that both algorithms create a phylogenetic network on the Giardia dataset whose exhibited trinets are 55% consistent with the input. `TriL2Net` can produce different networks for the same data set in different runs, which is the case for this dataset. One of the two networks it creates is the same as `TriLoNet`'s. The other network has switched the order of two leaves, decreasing the consistency with the triplets induced by the input trinets slightly. That these two leaves can be swapped can be explained by looking at the input trinet set: the amount of trinets that describe each of these options is the same.

The phylogenetic networks created by `TriLoNet` and `TriL2Net` from the other trinet set, the one concerning HBV subtypes, are slightly different. Both algorithms deduce the same cut-arc sets and reticulation leaves. However, in three of the biconnected components, `TriL2Net` places a taxon on another side of the component the `TriLoNet` does. This difference is in favour of `TriL2Net`, whose network is 17% consistent with the input trinets; slightly higher than the 15.5% achieved by `TriLoNet`. We concluded that this is a consequence of the difference in heuristics used. Whether `TriL2Net` always performs better on level-1 trinet sets must be investigated.

We found these trinet consistencies of 55% and 17% are rather low. Therefore, we analysed the depth-1 inconsistency of the biological data sets. This depth-1 inconsistency is a measure for the amount of contradicting binets induced by the input trinet set. We found that this inconsistency is 60% for the Giardia data set and 91% for the HVB data set. Next, we analysed how trinet sets with similar depth-1 inconsistencies can be generated. The result of this analysis shows that replacing a trinet with a random other trinet increases this inconsistency a lot, much more than performing a tail move on this trinet. Moreover, we found that the depth-1 inconsistency of the HBV trinet set can be replicated by either replacing 15% of a trinet set's trinets with random other trinets, or performing a tail move on 30% of the trinet set's trinets. However, the average trinet consistency scores for sampled data with these levels of noise are significantly higher. For example, the average trinet consistency score of the sampled data experiments is around 70% for a uniform noise level of 15%. Therefore, we conclude that this depth-1 inconsistency on its own is not a reliable measure for the amount of noise contained in a trinet set. To obtain better predictors for this noise, and hence predictors for the trinet set consistency of the output, more research needs to be performed.

As currently no algorithm exists that creates level-2 trinets from biological data, it has not been possible to test the algorithm on level-2 trinet sets derived from biological data. Unfortunately, constructing such an algorithm is non-trivial. There are, however, other ways this algorithm can be tested on level-2 trinets based on biological data. For example, if there are multiple level-2 phylogenetic networks, each on a set of species such that these sets overlap, it is possible to construct a level-2 phylogenetic network on the union of all these sets of species. This can be done by applying the algorithm to the union of the trinet sets induced by each of the original networks. The union of these trinet sets will probably contain many duplicate and contradicting trinets. Moreover, there will be many sets of three species for which it does not contain any trinet. In order to deal with these 'holes' the current heuristics might need to be adjusted.

The algorithm is currently restricted to binary level-2 phylogenetic trinets and networks. Possible generalizations can be an increase in degree or level. Another way would be to not limit the number of reticulation nodes per biconnected component, but the number of reticulation *leaves* per biconnected component. Such networks are called k-terminal networks (Huber, Moulton, and Wu 2019). Such generalizations will give rise to several complications.

First of all, it is currently unknown if the phylogenetic networks in these larger classes are encoded by their trinets. Therefore, it is possible that different networks induce the same trinet set. It could also be that networks are not encoded by their trinets, but by their induced n-net set for $n > 3$. In this case a similar approach remains possible. For the class of binary level-2 phylogenetic networks, it has been proven that the phylogenetic networks in this class are encoded by their trinets (van Iersel and Moulton 2014). In this proof, every binary level-1 and level-2 generator is considered. In order to extend this proof to binary level-k networks using the same method, every binary level-k generator must be considered. As there are at least 2^{k-1} such generators (Gambette et al. 2009), this is quite cumbersome.

Another result that is needed is whether the minimal sink-sets of the auxiliary graph of a trinet set are precisely the minimal cut-arc sets of the networks in these larger classes. In this thesis, it has been shown that this is the case for binary level-2 phylogenetic networks (see Corollary 4.2.4). Moreover, we have conjectured that this in fact the case for any binary phylogenetic network. Similarly, Huber, Moulton, and Wu (2019) have shown that this is also the case for, not necessarily binary, 2-terminal phylogenetic networks.

The next complication is the amount of trinets that exist in these larger classes. As there are only 210 level-2 trinets, finding the generator of a level-2 trinet using an exhaustive search is very quick. However, as k increases, the number of level-k generators, and therefore the number of level-k trinets, rises exponentially. Hence, this approach is not ideal when generalizing to higher levels. For k-terminal networks, the amount of generators is already infinite for $k = 1$. This can be seen by repeating the pattern of generator $\mathscr{G}^{2a}$ from Figure 3.2. Thus, when generalizing to k-terminal networks, such an exhaustive approach is impossible.

Another difficulty with the amount of generators is determining their symmetries. This is currently done by hand and hard-coded in the algorithm, which is easily done for a few generators. Finding these symmetries for arbitrary generators should, by definition, be possible through the generators' isomorphisms. However, for a generator containing n nodes there exist at most $n!$ possible isomorphisms. Furthermore, for binary networks a polynomial-time algorithm exists that checks whether two such networks are isomorphic, but whether one exists for arbitrary degree networks is unknown.

Moreover, it is possible that the networks in these larger classes encompass generators that the trinets in that class do not. This is the case, for example, for level$-k$ networks with $k \geq 3$ and for k-terminal networks with $k \geq 1$. In such cases, simply picking the generator that occurs most in a certain set of trinets does not work. For level-k, there are various options to tackle this. First of all, it might be possible to derive higher level generators from a set of lower level generators. A method for this might be deduced from Gambette et al. (2009), which gives an algorithm for constructing the set of all level$-k+1$ generators from the set of all level$-k$ generators. Secondly, it is possible to use networks containing k leaves instead of trinets. This option might be more easily implemented, but as k increases the number of k-nets increases rapidly too. Lastly, Murakami et al. (2019) have introduced an algorithm that constructs a level-k tree-child network N (a subclass of level-k networks) from its reticulate-edge-deleted subnetworks, which are subnetworks obtained by deleting a single reticulation edge. This deletion of reticulation edges reduces the level of a network. Hence, it might be possible to first deduce sets of level-2 trinets from the original level-k trinet set. Then, use `TriL2Net` to construct a level-2 network on each of these level-2 trinet sets. Finally, use the algorithm of Murakami et al. (2019) to combine these level-2 networks in a level-k network.

Bibliography

Aho, A. V., Sagiv, Y., Szymanski, T. G., & Ullman, J. D. (1981). Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM J Comput*, *10*(3), 405–421.

Dress, A., Huber, K. T., Koolen, J., Moulton, V., & Spillner, A. (2012). *Basic phylogenetic combinatorics*. Cambridge: Cambridge University Press.

Gambette, P., Berry, V., & Paul, C. (2009). The structure of level-k phylogenetic networks. *Combinatorial Pattern Matching*, 289–300.

Huber, K. T., & Moulton, V. (2013). Encoding and constructing 1-nested phylogenetic networks with trinets. *Algorithmica*, *66*(3), 714–738.

Huber, K. T., Moulton, V., & Wu, T. (2019). Hierarchies from lowest stable ancestors in nonbinary phylogenetic networks. *Journal of Classification*, *36*(2), 200–231.

Huber, K. T., van Iersel, L., Kelk, S., & Suchecki, R. (2011). A practical algorithm for reconstructing level-1 phylogenetic networks. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, *8*(3), 635–649.

Huson, D. H., Rupp, R., & Scornavacca, C. (2010). *Phylogenetic networks, concepts, algorithms and applications*. Cambridge University Press.

Huson, D. H., & Scornavacca, C. (2012). Dendroscope 3: An interactive tool for rooted phylogenetic trees and networks. *Syst Biol*, *61*(6), 1061–1067.

Janssen, R., Jones, M., Erdõs, P. L., van Iersel, L., & Scornavacca, C. (2018). Exploring the tiers of rooted phylogenetic network space using tail moves. *Bulletin of Mathematical Biology*, *80*(8), 2177–2208.

Kole, S. (2020). Tril2net. https://github.com/KSjors/TriL2Net. GitHub.

Lot, M., Spillner, A., Huber, K. T., & Moulton, V. (2009). Padre: A package for analyzing and displaying reticulate evolution. *Bioinformatics*, *25*(9), 1199–1200.

Morrison, D. A. (2011). *Introduction to phylogenetic networks*. RJR Productions.

Murakami, Y., van Iersel, L., Janssen, R., Jones, M., & Moulton, V. (2019). Reconstructing tree-child networks from reticulate-edge-deleted subnetworks. *Bulletin of Mathematical Biology*, *81*(10), 3823–3863.

Musser, D. R. (1997). Introspective sorting and selection algorithms. *Software: Practice and Experience*, *27*(8), 983–993.

Oldman, J., Taoyang, T. W., van Iersel, L., & Moulton, V. (2016). TriLoNet: Piecing Together Small Networks to Reconstruct Reticulate Evolutionary Histories. *Molecular Biology and Evolution*, *33*(8), 2151–2162.

Pomerol, J. C., & Barba-Romero, S. (2000). *Multicriterion decision in management: Principles and practive*. Springer.

Poormohammadi, H., Eslahchi, C., & Tusserkani, R. (2014). Tripnet: A method for constructing rooted phylogenetic networks from rooted triplets. *PLoS One*, *9*(9).

Ranwez, V., Berry, V., Criscuolo, A., Fabre, P. H., Guillemot, S., Scornavacca, C., & Douzery, E. J. P. (2007). PhySIC: A veto supertree method with desirable properties. *Syst Biol*, *56*(5), 798–817.

Saitou, N., & Nei, M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol Biol Evol*, *4*(4), 406–425.

Scornavacca, C., Berry, V., Lefort, V., Douzery, E. J. P., & Ranwez, V. (2008). PhySIC_IST: Cleaning source trees to infer informative supertrees. *BMC Bioinformatics*, *9*(1), 413.

Semple, C., & Steel, M. (2003). *Phylogenetics*. Oxford University Press.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM J on Comput*, *1*(2), 146–150.

Than, C., Ruths, D., & Nakhleh, L. (2008). Phylonet: A software package for analyzing and reconstructing reticulate evolutionary relationships. *BMC Bioinformatics*, *9*.

Timme, R. E., Simpson, B. B., & Linder, C. R. (2007). High-resolution phylogeny for helianthus (asteraceae) using the 18s-26s ribosomal dna external transcribed spacer. *American journal of botany*, *94*(11), 1837–1852.

van Iersel, L., & Moulton, V. (2014). Trinets encode tree-child and level-2 phylogenetic networks. *Journal of Computational Biology*, *68*(7), 1707–1729.

A

HBV Networks

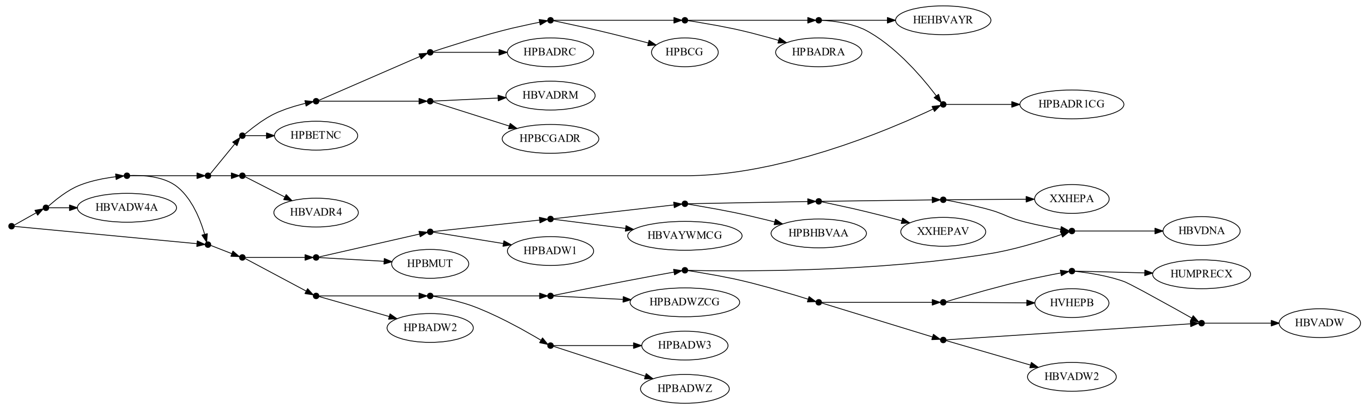


Figure A.1: A larger version of Figure 6.9a: the phylogenetic network derived by `TriLoNet` for the HBV trinet set.

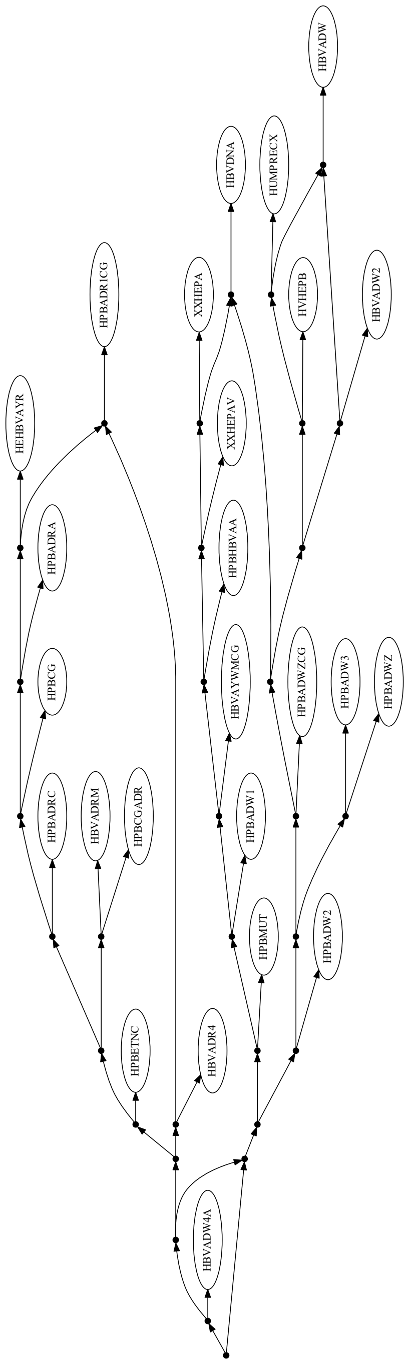


Figure A.2: A larger version of Figure 6.9b: the phylogenetic network derived by `TriL2Net` for the HBV trinet set.