Constructing Phylogenetic Networks based on Trinets

James William Oldman

School of Computing Sciences

University of East Anglia

A thesis submitted for the degree of

Doctor of Philosophy

September 2015

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification at this or any other university or other institute of learning.

Acknowledgements

I would like to first acknowledge and thank my supervisors Prof Vincent Moulton, Dr Taoyang Wu and Dr Geoffrey McKeown for their continuous support and collaboration throughout the past four years. I would also like to thank all my past and present colleagues in the Computational Biology Laboratory for making the last four years even more enjoyable. I would like to thank the University of East Anglia for the financial support and funding provided throughout the project, in particular for providing me with the opportunity to attend research conferences. Finally, I would like to thank all my family and friends for their understanding, encouragement and support throughout my PhD.

I would like to dedicate this thesis to my family and close friends...

Statement of Originality

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged.

Abstract

The motivation of phylogenetic analysis is to discover the evolutionary relationships between species, with the broader aim of understanding the origins of life. Our understanding of the molecular characteristics of species through DNA sequencing permanently changed the approach to understanding the evolution of species. Indeed, the advancement of technology has played a major role in the fast sequencing of DNA as well as the use of computers in solving biological problems in general. These evolutionary relationships are often visualised and represented using a phylogenetic tree. As a natural generalisation of phylogenetic trees, phylogenetic networks are used in biology to represent evolutionary histories that contain reticulate, or non-treelike events such as recombination, hybridisation and horizontal gene transfer. The reconstruction of explicit phylogenetic networks from biological data is currently an active area of phylogenetics research. Here we consider the problem of constructing such networks from trinets, that is, phylogenetic networks on three leaves. More specifically, we present the SeqTrinet and TriLoNet methods, which form a supernetwork based approach to constructing level-1 phylogenetic networks directly from multiple sequence alignments.

Contents

List of Figures

Chapter 1

Introduction

Before the second half of the 20th Century, much of the work in the classification and study of evolutionary relationships between species was based on the morphological characteristics of species. However, our understanding of the molecular characteristics of species through DNA sequencing permanently changed this approach to understanding the evolution of species. Indeed, the advancement of technology has played a major role in the fast sequencing of DNA as well as the use of computers in solving biological problems in general. Phylogenetics is one such area that has greatly benefited from the collaboration of researchers in computational, mathematical and biological disciplines. In many ways computer science acts as the bridge between mathematics and biology; the theoretical ideas are developed into algorithms which are then implemented as analysis tools for use by biologists.

The motivation of phylogenetic analysis is to discover the evolutionary relationships between species, with the broader aim of understanding the origins of life. These relationships are often visualised and represented using a phylogenetic tree, with the Tree of Life being an example [Maddison et al., 2007]. Phylogenetic trees are useful for displaying speciation events, shown as branching in the tree, where one species speciates into two or more species. Phylogenetic trees and their construction have been extensively studied, see e.g [Semple and Steel, 2000].

Even though phylogenetic trees have proven to be useful in biology, in recent years there has been significant interest in studying evolution that involves *reticulate*, or non-treelike evolutionary events. While evolution is believed to be primarily a branching process, genes and genomes are not necessarily inherited vertically; they can also be inherited via a lateral process. These events include recombination, horizontal gene transfer (HGT) and hybridisation and result in a network rather than a tree structure. Such reticulate events are believed to occur in organisms such as bacteria, plants, viruses and certain groups of fishes and frogs [Than et al., 2008]. Trees are of more limited use when trying to infer an evolutionary history of a set of taxa which is believed to contain non-treelike evolutionary events.

Phylogenetic networks extend the definition of phylogenetic trees by facilitating the modelling of reticulate evolutionary events. They are used in scenarios for which a tree structure is not sufficient to model a proposed evolutionary history. There currently exist a range of well established algorithms for computing unrooted phylogenetic networks that implicitly represent evolution [Huson et al., 2010]. However, this is not so much the case for rooted phylogenetic networks which aim to explicitly represent evolutionary histories. Hence, there is at present a great interest in creating practical computational methods for inferring rooted phylogenetic networks.

Rooted phylogenetic networks provide both a computationally interesting and biologically relevant area in science with room for much research into the development of efficient algorithms and heuristics. Although recently there have been several algorithms proposed that construct such networks [Jansson and Sung, 2006], [Huber et al., 2011b], [Habib and To, 2011] (for example, from triplets i.e. phylogenetic trees on precisely three leaves), none have become established and reliable enough to become standard tools. As suggested in [Huber and Moulton, 2013], a better understanding of how to reconstruct phylogenetic networks from basic structures may thus be necessary to achieve this.

The work in this thesis is concerned with the construction of rooted phylogenetic networks from trinets, that is, phylogenetic networks with precisely three leaves that we construct from DNA sequence data.

[Huber and Moulton, 2013] presented an algorithm that decides whether a dense set of trinets (i.e. one that contains a trinet on every 3-element subset of a set) can be displayed by a level-1 network or not and, if so, constructs that network. This work extends that of [Huber and Moulton, 2013] as here we present an algorithm called TriLoNet that will, given a dense set of trinets (possibly obtained from sequence data), always construct a binary rooted level-1 rooted phylogenetic network thus addressing some of the problems associated with noisy input data.

Thesis chapter summary

We now briefly summarise the chapters presented in the rest of this thesis:

- In **Chapter 2** we introduce the key background terminology required for the rest of the thesis. This includes some definitions on graph theory, phylogenetic trees, phylogenetic networks and some discussion on various approaches to constructing trees and networks.

- In **Chapter 3** we present an algorithm called SeqTrinet, a new approach to constructing phylogenetic networks on three leaves from multiple sequence alignments. We detail the key steps of the method and present the pseudocode for the algorithm. We also present experiments where we simulate some simple recombinant data sets evolved down phylogenetic networks to help inform parameter choice in the method.

- In **Chapter 4** we describe in detail the key steps of the TriLoNet algorithm along with the necessary proofs and pseudocode. The theoretical proofs and results presented in this chapter were produced in collaboration with Dr Taoyang Wu. The implementation of all the algorithms shown was carried out by myself.

- In **Chapter 5** we present results from three experiments designed to evaluate the performance of TriLoNet. We begin with a comparison study between TriLoNet and Lev1athan, a level-1 triplet-based network construction algorithm [Huber et al., 2011b]. We then artificially simulate recombinant sequence data for some simple scenarios following a similar methodology used in [Holland et al., 2002]. We also consider seven real biological data sets containing known recombination to test the ability of TriLoNet to identify recombinant taxa from sequence data. TriLoNet has been implemented in Java and is freely available.

- Finally, in **Chapter 6** we conclude our findings by summarising our contributions as well as considering some possible future theoretical ideas and technical improvements to the work presented here.

We currently have a manuscript in preparation for publication, entitled TriLoNet: A supernetwork approach to construct level-1 phylogenetic networks. The authors are James Oldman, Taoyang Wu, Leo Van Iersel and Vincent Moulton.

Chapter 2

Background

2.1 Chapter summary

This chapter introduces the relevant background information and terminology required for the rest of the thesis. Section 2.2 outlines the basic graph theory used in this thesis. Section 2.3 introduces the concept of phylogenetic trees, triplets and tree construction methods. Section 2.4 then covers phylogenetic networks, trinets and some current network construction algorithms.

2.2 Definitions and terminology

We begin with some basic definitions and notation. The choice of terminology mainly follows [Huson et al., 2010]. A *graph* $G = (V, E)$ consists of a finite set $V = V(G)$ of vertices and a finite set $E = E(G)$ of *edges*, with each edge $e \in E$ consisting of a pair $e = \{u, v\}$ of distinct vertices in V. Two vertices u and v are *adjacent* if $e = \{u, v\}$ is an edge in E and u and v are called the *endpoints* of e. An *undirected path* $P = u_0, u_1, u_2 \cdots, u_k$ is a sequence of vertices starting at u_0 and ending at u_k (denoted as $u_0 \rightarrow u_k$) connected by a sequence of edges (none of which occur more than once), with u_{i-1} and u_i being adjacent to edge e_i for $1 \leq i \leq k$. The *length* of a path is the number of edges it contains. With $G = (V, E)$ as a graph, $H = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$ where all edges in E' contain only vertices in V'. A *cycle* in a graph is a path for which the first vertex is equal to the last vertex $u_0 = u_k$, with this vertex being the only vertex to occur more than once.

A *digraph*, also called a *directed graph* $N = (V, E)$ consists of a finite set $V = V(N)$ of *vertices* and a finite set $E = E(N)$ of *arcs*. Each arc consists of an ordered pair $a = (u, v)$ of vertices in V in which u is said to be a *parent* of v and v a *child* of u. All digraphs in this thesis contain no loops, that is, no vertex is the child of itself. A *directed path* is a sequence $u_0, u_1, \cdots, u_k$ of vertices such that $(u_{i-1}, u_i) \in E$ holds for $1 \leq i \leq k$. An *acyclic graph* is a digraph that does not contain any directed path starting and ending at the same vertex.

An acyclic digraph N induces a canonical partial order $\prec_N$ on its vertex set V, that is, $v \prec_N u$ if there exists a directed path from u to v. In this case, we shall say that u is *below* v. For simplicity, when the digraph N is clear from the context, $\preceq_N$ will be written as $\preceq$. In addition, we write $v \preceq u$ if $v = u$ or $v \prec u$. A *least common ancestor* of u and v is a lowest vertex w in N such that both $v \preceq w$ and $u \preceq w$ hold.

Suppose that $N = (V, E)$ is a digraph. Arc $a = (u, v)$ in E, is said to be directed from u to v. Vertices u and v in V are *incident* to a. Such an arc a is an *out-edge* of u and an *in-edge* of v. The *in-degree* of vertex u is the number of vertices v in V such that (v, u) is an arc, and the *out-degree* of u is the number of vertices w in V with (u, w) as an arc. The *degree* of a vertex in N is the sum of its in-degrees and out-degrees. A vertex ρ is a *root* of the directed graph (V, E) if ρ has in-degree 0. If an acyclic digraph N contains a unique root, which is usually designated by $\rho = \rho(N)$, then it will be referred to as a rooted acyclic digraph. A *leaf* is a vertex of in-degree 1 and out-degree 0. The set of leaves of N is denoted by $L(N)$. Any vertex in N that is neither a root or leaf is referred to as an *interior vertex*. In addition, an interior vertex is a *tree vertex* if it has in-degree 1, and a *reticulation vertex* if it has in-degree greater than 1.

The definition of a subgraph can also used for digraphs. Suppose that $N = (V, E)$ is a digraph. Let $\underline{N}$ be the undirected graph associated with N which is obtained by discarding the direction of the arcs in N. Then N is *connected* if $\underline{N}$ is connected, that is, there exists an undirected path between every pair of distinct vertices in $\underline{N}$. Note that a rooted acyclic digraph is necessarily connected. Let v be a vertex of N. Then v is a *cut-vertex* of N if the removal of v disconnects

$\underline{N}$. Similarly, a *cut-arc* is an arc of N whose removal disconnects $\underline{N}$. A cut-arc is referred to as *trivial* if it is incident with a leaf. A directed graph is *biconnected* if it contains no cut-vertex. A *biconnected component* of N is a maximal biconnected subgraph, which is called trivial if it contains precisely one arc (which is necessarily a cut-arc), and *non-trivial* otherwise.

Given some graph $N = (V, E)$, to *delete* an edge e means that e is removed from the set of edges E. For the deletion of a vertex v from N, firstly v is removed from vertex set V and secondly, all edges incident to v are deleted. A vertex v in a directed graph is *suppressible* if it has in-degree 1 and out-degree 1 and v is *suppressed* by connecting the parent of the in-edge of v to the child of the out-edge of v by an arc and then deleting v.

2.3 Phylogenetic trees

Throughout this thesis, we assume that X is a non-empty, finite set (which will usually represent a set of species or organisms). Unless explicitly stated otherwise, we shall assume $|X| \geq 3$. A subset Y of X is called a *singleton* if $|Y| = 1$, and *non-singleton* if $|Y| \geq 2$. A tree is a connected graph with no cycles.

A rooted tree $T = (V, E)$ is a connected acyclic digraph with precisely one root vertex ρ, with every arc being directed away from the root. A phylogenetic tree T on X is a rooted tree in which no vertex aside from the root can have degree 2, together with a bijective labelling of the elements in X on to the leaves in T so that in particular every leaf has a unique label. The leaf set of a phylogenetic tree T is denoted by $L(T)$.

Given a vertex v in a phylogenetic tree T, the *subtree* T_v of T is the restriction of T on the set of vertices that are below of v, with v being designated as the root. Moreover, $\{x, y\}$ is called a *cherry* of T, $x, y \in X$, if there exists a subtree of N whose leaf set is precisely $\{x, y\}$.

Phylogenetic trees are often drawn as a cladogram to represent ancestral relationships. The set of extant species X are usually drawn at the bottom. The hypothetical ancestors of these current day species are drawn above, with the root at the top of the diagram. The direction of the arcs is often omitted, see Figure 2.1(b).

Given input data about a given set of current species, which could be molecular or morphological, one of the main aims in phylogenetic analysis is to reconstruct a phylogenetic tree that reflects the evolutionary relationship between this set of species. There are many methods to reconstruct such a tree. One that is particularly related to the theme of this thesis is to reconstruct a tree from a set of triplets, which we describe in Section 2.3.2.

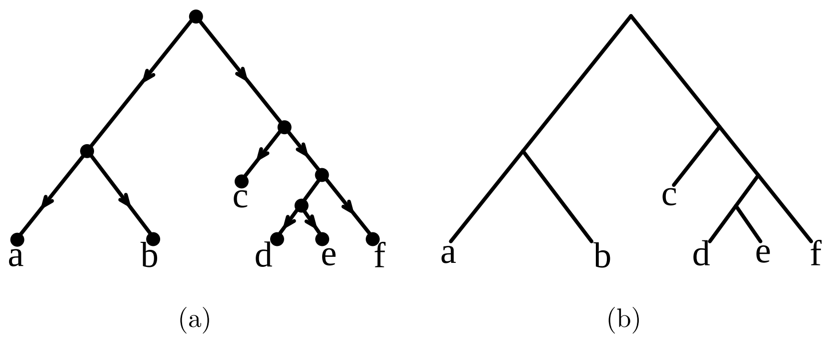


Figure 2.1: An example of rooted phylogenetic trees on the set $X = \{a, b, c, d, e, f\}$. The tree shown in (a) includes the representation of vertices as points and the direction of arcs, however as shown in (b) for simplicity the direction of the arcs and representation of vertices as points can be omitted.

2.3.1 Building phylogenetic trees

A *multiple sequence alignment* (MSA) is an alignment of three or more biological sequences. All MSA's mentioned in this thesis are comprised of DNA sequences on the alphabet $\{A, C, G, T\}$. Phylogenetic trees are often inferred from a multiple sequence alignment of DNA sequences. The two main approaches to solving the phylogenetic tree reconstruction problem are sequence-based methods and distance-based methods. Sequence-based methods attempt to find a phylogenetic tree that best describes a multiple sequence alignment whereas distance-based methods use a distance matrix [Huson et al., 2010]. Sequence-based methods include maximum parsimony, maximum likelihood and Bayesian methods. Neighbor-joining and the unweighted pair group method using arithmetic averages (UPGMA) are two well known distance-based reconstruction methods [Huson et al., 2010].

The Maximum Parsimony and Maximum Likelihood methods provide fast and accurate results for small numbers of taxa [Jin et al., 2007]. Maximum parsimony follows the minimum evolution principle and the idea of Occam's razor; preferring the least complex explanation for an observation [Semple and Steel, 2003]. There are two problems known as the small parsimony problem and the large parsimony problem. The small parsimony problem can be solved in polynomial time through the use of the Fitch algorithm [Fitch, 1971]. With the large parsimony problem, the aim is to find the most parsimonious tree given only a set of sequences. The large parsimony problem is known to be NP-hard so cannot be solved in polynomial time. Many of the problems in computational biology are NP-hard so the development and application of heuristics play an important part in phylogenetic analysis [Jin et al., 2009].

2.3.2 Triplets

A *triplet* is a phylogenetic tree on three leaves. There are three possible triplets on a set of three taxa. For instance, Figure 2.2 depicts the three possible triplets on the set $\{x, y, z\}$. We use the notation $xy|z$ to represent the triplet in which $\{x, y\}$ is a cherry.

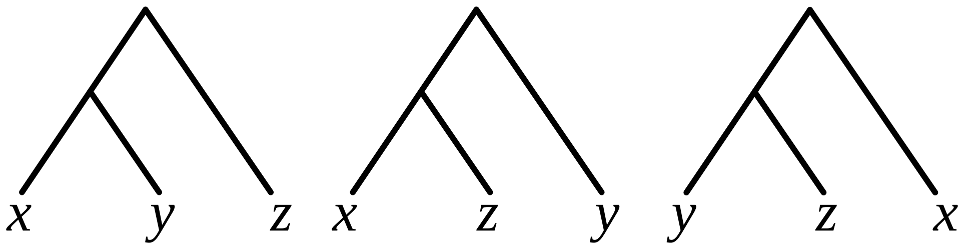


Figure 2.2: The three possible triplets on the set $\{x, y, z\}$, denoted $xy|z$, $xz|y$ and $yz|x$.

Let Tr denote a set of triplets. If Tr is a set of triplets then its leaf set $L(Tr)$ is defined as $\bigcup_{T \in Tr} L(T)$. A set of triplets Tr is called *dense* if for every set of three taxa $\{x, y, z\} \subseteq L(Tr)$ at least one of the triplets of the form $xy|z, xz|y, yz|x$ belongs to the triplet set Tr.

A triplet $xy|z$ is said to be *consistent* with a tree T if the triplet is an embedded subtree of T, i.e. a lowest common ancestor of x and y in T is a proper descendant of a lowest common ancestor of x and z in T [Jansson et al., 2006].

The Aho algorithm [Aho et al., 1981], also known as the BUILD algorithm is one of the first methods to construct a phylogenetic tree from a collection of phylogenetic trees. Such methods are also called supertree methods [Huber et al., 2011b]. Trees can be constructed from a set of triplets Tr using the Aho algorithm if such a tree exists that is consistent with all given input triplets.

The algorithm uses a top-down approach from the root to the leaves. The main idea of the algorithm is to partition the leaf set of a set of triplets Tr into blocks. A block B is dependent on the triplets in Tr. The algorithm outputs a tree with a root vertex whose children are the roots of the trees obtained by recursing on each block. Only the rooted triplets in Tr whose three leaves belong to a block are considered when recursing on B. The base case of the recursion is met when the leaf set consists of a single leaf. Any subset of leaves with size greater than 2 is partitioned into blocks by making use of an *Aho graph*, defined in Algorithm 1. For a more thorough description, see [Jansson et al., 2012].

The Aho algorithm can be unsuccessful if the input is not consistent with a tree. Thus the algorithm is not very useful for noisy data. The Aho algorithm makes use of what is known as an auxiliary or Aho graph. In practice the algorithm usually reports "No tree exists" due to the encountering of only one connected component in an Aho graph. One compromise to deal with finding only one component in the Aho graph as described in [Semple and Steel, 2000] and [Page, 2002], is to identify a minimum set of edges that when deleted from the graph will result in a graph with two connected components [Huson et al., 2010].

2.4 Phylogenetic networks

A phylogenetic network can very generally be described as any graph used to show evolutionary relationships between a set that labels some of its vertices, usually the leaves [Huson et al., 2010]. The two main categories of phylogenetic networks are rooted and unrooted. Rooted phylogenetic networks are used to

Algorithm 1 Aho Algorithm

INPUT: A set of rooted triplets Tr on a leaf set X.
OUTPUT: Minimal rooted tree labelled by X that is consistent with Tr if one exists, otherwise it reports “No tree exists”.

```
1: If $|X| = 1$ then construct a tree $R$ with a single vertex $x$ and return $R$
2: If $|X| = 2$ then construct and return a tree $R$ with a two vertices $x$ and $y$
3: if $|X| >= 3$ then
4:    Construct Aho graph $AG(Tr) = (V, E)$ with $V = X$ and $(y, z)$ is an edge
      in $E$ if and only if there exists $x$ such that the triplet $yz|x \in Tr$.
5:    if $AG$ contains one connected component then
6:       return “No tree exists”
7:    else if $AG$ contains more than one connected component then
8:       for all vertex sets $U$ of each connected component in $AG$ do
9:          Compute set of triplets $Tr|_U$
10:         Recursively compute phylogenetic subtree restricted to $Tr|_U$
11:      end for
12:   end if
13: end if
14: Create root vertex $\rho$
15: Join all subtrees by connecting $\rho$ to the root of each the subtrees
```

visualise reticulate evolutionary events for which a tree based model may not be the best representation tool. In this thesis we will be mainly focusing on rooted phylogenetic networks.

2.4.1 Definitions and terminology

A *phylogenetic network* $N = (V, E)$ on X is a connected acyclic digraph with a unique root ρ and leaves uniquely labelled by the elements in X (that is, there is a bijective mapping between $L(N)$ and X). We will usually just assume $L(N) = X$ in case the labelling is clear from the context. Without loss of generality, throughout this thesis we will also assume that N does not contain any vertex with in-degree 1 and out-degree 1, and all leaves have in-degree 1. The network N is *binary* if each tree vertex, as well as the root, has out-degree 2, and each reticulation vertex has in-degree 2 and out-degree 1.

Two phylogenetic networks $N_1 = (V_1, E_1)$ and $N_2 = (V_2, E_2)$ on X are said to be *isomorphic*, denoted by $N_1 \cong N_2$, if there exists a bijection $f : V_1 \to V_2$ such that $f(x) = x$ for all $x \in X$, and (u, v) is an arc in N_1 if and only if $(f(u), f(v))$ is an arc in N_2.

Given a network N on X and a subset $Y \subseteq X$, a vertex in N is a *stable ancestor* of Y if it is contained in every path from the root to some leaf $x \in Y$. Note that for two stable ancestors u and v of Y, we have either $u \preceq v$ or $v \preceq u$. Therefore, there exists a unique vertex w in N, which will be referred to as the *lowest stable ancestor* of Y in N and denoted by $\textsc{lsa}_N(Y) = \textsc{lsa}(Y)$, such that w is a stable ancestor of Y while no vertex below w is a stable ancestor of Y. Note that if $|Y| \geq 2$, then there exists two elements x and y in Y such that $\textsc{lsa}(Y) = \textsc{lsa}(\{x, y\})$. In addition, we have $\textsc{lsa}(x) = x$ for a leaf x in N.

Suppose N is a phylogenetic on X. A *cluster* is any subset of X, excluding the empty set $\emptyset$. Note that X itself is regarded as a cluster. With u as a vertex in a digraph, let $\textsc{ch}(u)$ be the set of children of u.

If u is a vertex in a network $V(N)$, let the cluster induced by u, denoted by $\mathcal{C}_N(u) = \mathcal{C}(u)$ be defined as the subset of the taxa in $V(N)$ below u. If vertex u is a leaf vertex then $\mathcal{C}(u) = \{u\}$. Let $\mathcal{C}(N) = \{\mathcal{C}(u) : u \in V(N)\}$ denote the set of clusters displayed by N.

The *subnet* or *subnetwork* of N on Y, denoted by $N|_Y$, is defined as the subgraph obtained from N by deleting all vertices that are not on any path from $\textsc{lsa}(Y)$ to elements in Y and subsequently suppressing all in-degree 1 and out-degree 1 vertices and parallel arcs. Note that by definition $N|_X = N$ if and only if $\textsc{lsa}(X) = \rho(N)$, in this case N is referred to as a *recoverable* network. Note that every subnet of N is necessarily recoverable. The process of obtaining the subnet $N|_Y$ with $Y = \{c, e, i\}$ is illustrated in Figure 2.3.

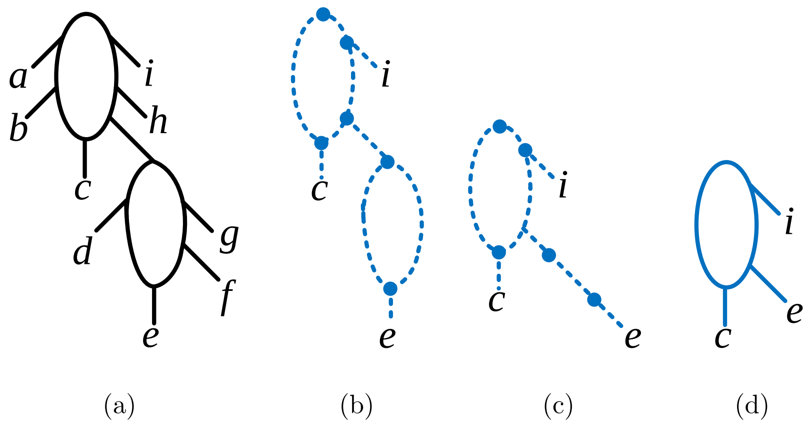


Figure 2.3: (a) A phylogenetic network N on the leaf set $X = \{a, b, c, d, e, f, g, h, i\}$. (b) The highlighted paths to every element in $Y = \{c, e, i\}$. (c) The result of suppressing multiple arcs in (b). (d) Suppressing vertices with in-degree 1 and out-degree 1 in (c) results in the subnet $N|_Y$ with $Y = \{c, e, i\}$.

2.4.2 Level-k networks

Suppose N is a phylogenetic network. Then N is a level-k ($k \geq 0$) network if each of its biconnected components contains at most k reticulation vertices. To some extent, the concept of the level of a network can be regarded as a measure of its 'distance' to being a phylogenetic tree. For instance, N is a level-0 network if and only if it is a phylogenetic tree.

Level-1 networks are relatively tree-like in terms of structure and hence somewhat more tractable to deal with. An example level-1 network is shown in Figure 2.4. It is for this reason that current research is focusing on developing new tools and evaluation methods for level-1 networks. Their relatively simplistic structure provides a good starting point; once these structures have been properly understood the next logical step is to investigate level-k networks, for $k > 1$.

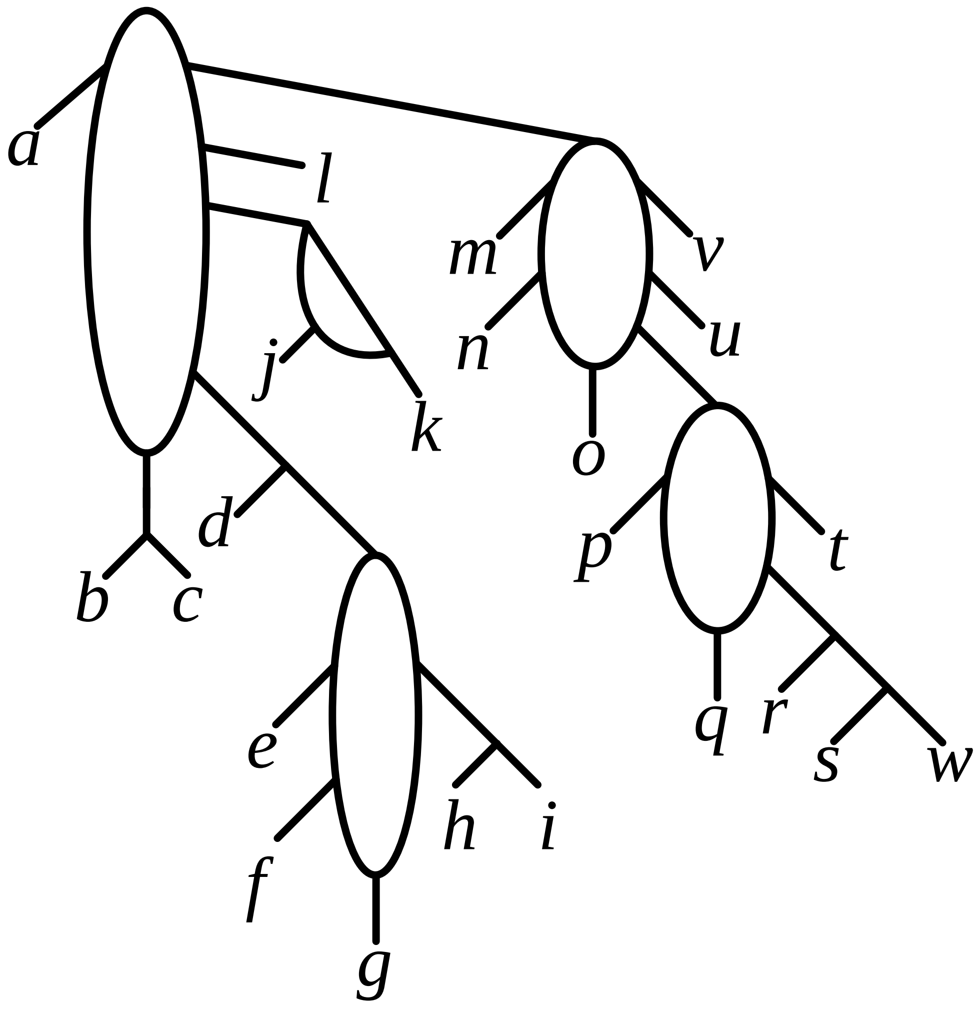


Figure 2.4: A level-1 phylogenetic network on the set $X = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w\}$.

2.4.3 Reconstructing networks from triplets

As with tree reconstruction, there are several ways to reconstruct phylogenetic networks from various types of input data. Such input data includes splits, clusters, sequences, distances, trees, quartets and triplets [Huson et al., 2010]. One method closely related to our work is the construction of networks from triplets.

There exist several methods that construct rooted phylogenetic networks from a set of triplets including those presented in [Jansson and Sung, 2006], [Jansson et al., 2006], [To and Habib, 2009], [Huber et al., 2011b], [Byrka et al., 2010] and [Habib and To, 2011].

A triplet $xy|z$ is said to be *consistent* with a network N if that network contains a *subdivision* of the triplet, i.e. if N contains vertices $u \neq v$ and pairwise internally vertex-disjoint paths $u \to x$, $u \to y$, $v \to u$ and $v \to z$ [van Iersel et al., 2009]. Given a network N, the set $Tr(N)$ denotes the set of all triplets that are consistent with N. A set of triplets is consistent with a network if every triplet in that set is consistent with the network.

As mentioned above, the Aho algorithm will, given a set of triplets as input, construct a tree that is consistent with all input triplets, if such a tree exists. The work in [Jansson and Sung, 2006] demonstrated that it is possible to reconstruct in polynomial-time a rooted level-1 phylogenetic network from a dense set of triplets if there exists such a network that is consistent with every triplet in the input. However, if the input is not required to be dense then the problem becomes NP-hard. Work by [van Iersel et al., 2009] has further extended the construction of a network from a dense set of triplets to level-2 phylogenetic networks.

In case such networks exist, the polynomial-time algorithms presented in [van Iersel and Kelk, 2011] construct the simplest possible level-1 and level-2 networks, that is the ones containing the minimum number of reticulations that are consistent with the dense triplet input set. In the case where the input set is precisely equal to the set of triplets consistent with some network, the resulting network produced as output is minimised in terms of both level and the overall number of reticulations in the network.

2.5 Trinets

A tree T can always be described by its rooted triplet encoding, i.e. T is the unique phylogenetic tree containing the set of triplets $Tr(T)$ that arises from taking all combinations of three leaves in T [Iersel and Moulton, 2013]. However, a phylogenetic network is not always encoded by the triplets it contains [Gambette and Huber, 2012]. An example from [Iersel and Moulton, 2013] illustrated in

Figure 2.5 shows that three different networks can all contain the same set of triplets.

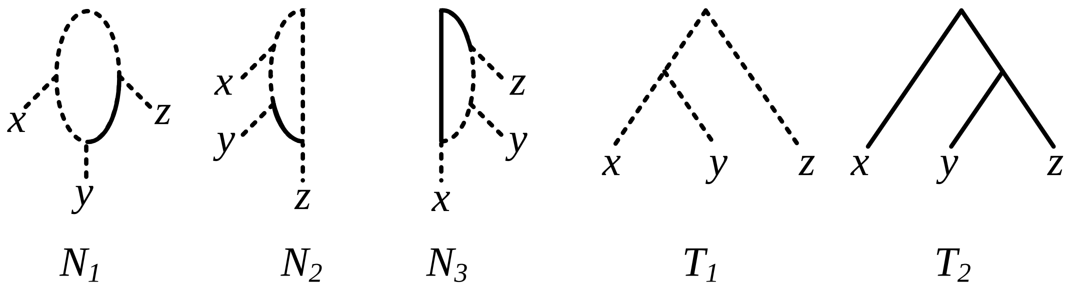


Figure 2.5: An illustration from [Iersel and Moulton, 2013] showing that three level-1 networks can all have the same set of triplets, $Tr(N_1) = Tr(N_2) = Tr(N_3) = \{T_1, T_2\}$. Dotted lines are used to show how T_1 is contained in the networks N_1, N_2 and N_3.

In light of this, [Huber and Moulton, 2013] suggested the possibility of trying to encode networks instead by subnetworks with three leaves. Motivated by this, we define a *trinet* to be a phylogenetic network with precisely three leaves. Given a set of trinets $\mathfrak{T}$, we let $L(\mathfrak{T}) = \cup_{T\in\mathfrak{T}} L(T)$ be the leaf set of $\mathfrak{T}$. A trinet set $\mathfrak{T}$ on X is a non-empty set of trinets with $L(\mathfrak{T}) = X$ and $L(T) \neq L(S)$ for distinct trinets S and T in $\mathfrak{T}$. If $Y \subseteq X$ with $|Y| \geq 3$, we let $\mathfrak{T}_Y$ be the subset of $\mathfrak{T}$ consisting of those trinets T in $\mathfrak{T}$ with $L(T) = Y$. A set $\mathfrak{T}$ of trinets on X is called *dense* on X if for each subset $Y \subseteq X$ with $|Y| = 3$, there exists precisely one trinet T in $\mathfrak{T}$ with $L(T) = Y$, that is, $\mathfrak{T}_Y = \{T\}$. Finally, given a phylogenetic network N on X, let

$$\mathfrak{T}(N) = \{N|_Y : Y \subseteq X \text{ and } |Y| = 3\}$$

be the dense trinet set on X induced by N.

There are fourteen non-isomorphic level-1 trinets on three leaves shown in [Huber and Moulton, 2013]. Note that there are precisely eight binary non-isomorphic level-1 trinets, as depicted in Fig. 2.6.

Another important structure we consider is called a *binet*, which is a phylogenetic network on two leaves. Note that there are two types of binets, as shown in Figure 2.7. The first type of binet, T_0, consists of a vertex v with two leaf

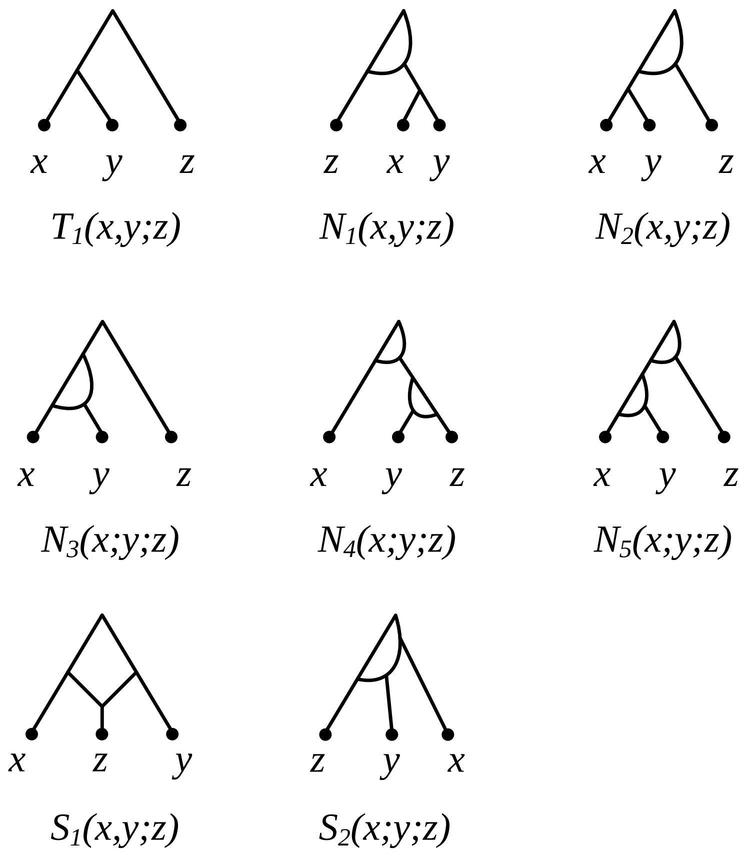


Figure 2.6: The eight binary rooted level-1 trinets on $\{x, y, z\}$.

vertices x and y. This binet is denoted by $T_0(x, y)$.

The second type is S_0 which is a network with cycle of size three that contains two leaves. The binet in Figure 2.7 is denoted by $S_0(x; y)$. Note the ordering of the leaves, leaf x is on the side of the cycle and leaf y is below the reticulation vertex.

Here we will use a notation system that indicates the interchangeability between leaves. For instance, in the trinet with type $T_1(x, y; z)$, leaves x and y are interchangeable, which are separated by a comma, while y and z are distinguishable, which are separated by a semicolon. The same convention applies to types N_1, N_2 and S_1, in which the leaves x and y are interchangeable. For the other four types, the three leaves are distinguishable. For a type T_0 binet, the leaves x and y are interchangeable whereas for a type S_0 binet, the two leaves x and y are distinguishable by a semicolon.

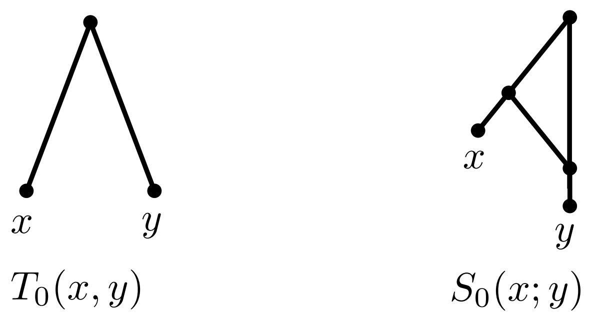


Figure 2.7: The two types of level-1 binets on $\{x, y\}$.

Approaching the problem of reconstructing a level-1 phylogenetic network using trinets instead of triplets may be more beneficial as a set of trinets exactly *encode* (uniquely describe) the level-1 network that is produced [Huber and Moulton, 2013]. In other words, if N and N' are phylogenetic networks with $\mathcal{T}(N) = \mathcal{T}(N')$ then $N \cong N'$.

2.6 Types of reticulation events

Phylogenetic networks extend the phylogenetic tree model with the extra possibility that two branches combine into one new branch at a reticulation vertex. Evolutionary events displayed on a phylogenetic network in which two branches lead to the same vertex are known as *reticulation events.* These evolutionary events imply convergence between objects [Jansson and Sung, 2006]. Any non-treelike event such as recombination, Horizontal Gene Transfer and hybridisation can be modelled by a reticulation [van Iersel, 2009]. The work presented in this thesis is mainly concerned with the reconstruction of phylogenetic networks that model recombination events.

Recombination is a process by which pieces of DNA are broken and then recombined to produce new combinations of alleles, this process creates genetic diversity at the level of genes that reflects differences in the DNA sequences of different organisms [Scitable, 2015]. *Horizontal (Lateral) Gene Transfer* (HGT) involves the direct transfer of genes from one organism to another. HGT is common among bacteria, even amongst bacteria that share a distant relation. It has been shown to be a key factor in the evolution of many organisms. HGT is also believed to be a driving factor of increased drug resistance [Gyles and Boerlin, 2013] when one bacterial cell acquires resistance and quickly transfers the resistance genes to many species. *Hybridisation* is an evolutionary process in which two lineages combine their DNA to form a new lineage. The term hybrid refers to the sexual crossing of two different lineages that go on to produce a new offspring [Morin, 2007].

2.7 Formats for representing trees and networks

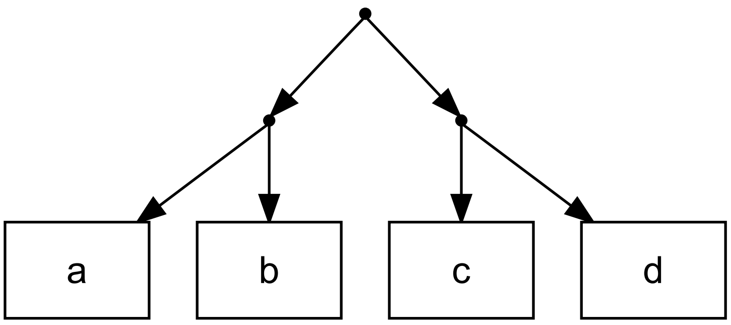
In this section we discuss two file formats required in later chapters. The Newick format is used to represent phylogenetic trees and networks and the DOT format will be used to provide a visual representation of phylogenetic networks.

2.7.1 Newick format

The topology of a rooted phylogenetic tree can be described in a single line using the Newick format, also known as Newick notation. The format itself was adopted in 1986 and has become the standard notation for the representation of trees in computer readable form [Jin et al., 2009]. A pair of parentheses are used to represent internal vertices. Leaves are represented by its taxon label.

The children of an internal vertex are enclosed between the set of parentheses representing that internal vertex. The children of a vertex are separated by commas. Internal vertices can also be labelled by inserting a label directly after the closing parenthesis representing that vertex [Huson et al., 2010]. A semicolon is used to terminate the tree description. Figure 2.8 illustrates the representation of a simple phylogenetic tree using the Newick format.

Unrooted phylogenetic trees can also be represented using the Newick format. It is also possible to represent edge lengths in the description. A semicolon is written after the vertex, followed by the edge length to represent the length of the branch immediately below that vertex.

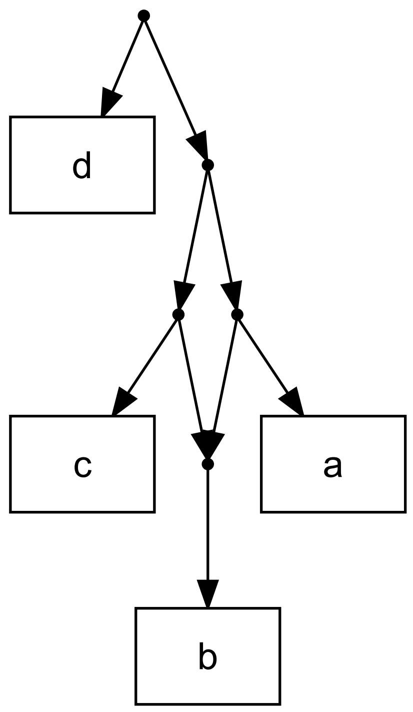


((a,b),(c,d));

Figure 2.8: Representing a phylogenetic tree on the leaf set $X = \{a, b, c, d\}$ using the Newick format.

The extended-Newick (eNewick) format builds upon this representation and applies the same principle to representing phylogenetic networks. In one implementation of the eNewick format, a network is represented by its underlying tree structures. The method presented in [Jin et al., 2009] represents a network as a collection of trees.

Another implementation of the eNewick format proposed in [Morin and Moret, 2006] requires the repeated labelling of a reticulate vertex so as to identify which vertices need to be merged together, we have chosen to follow this convention in this thesis. The network is modified so that all but one incoming edge of a reticulate vertex is cut. The new vertices formed from this modification are then given the same label. The labelling convention here for a reticulate vertex is $\#Hi$ where i is a number. Figure 2.9 presents an example of a level-1 phylogenetic network and the corresponding eNewick string representation.

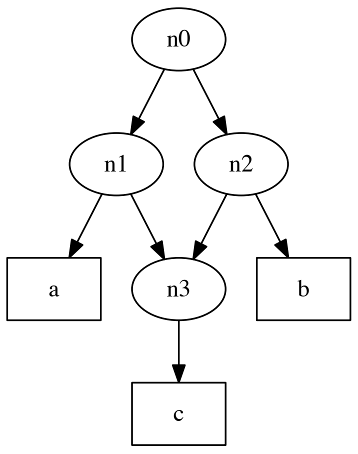


(d,((a,(b)#H1),(c,#H1)));

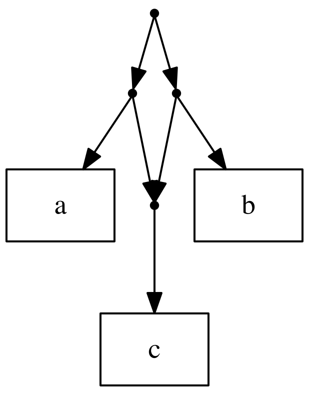
Figure 2.9: Representing a phylogenetic network using the eNewick format.

2.7.2 DOT format

The DOT format is popular for displaying both directed and undirected graphs [Gansner et al., 2006]. The output network resulting from the TriLoNet algorithm presented in Chapter 4 is displayed in DOT format. The internal vertices are labelled arbitrarily. The DOT file contains a list of arcs, for example, the arcs $(n0, n1)$, $(n0, n2)$, $(n1, a)$, $(n1, n3)$, $(n2, b)$, $(n2, n3)$, $(n3, c)$ construct the trinets shown in Figure 2.10.



(a)



(b)

Figure 2.10: Displaying trinets using the DOT format. The trinet $S_1(a, b; c)$ shown in (a) includes labels of all vertices for illustration, however as shown in (b), for simplicity the interior vertices can be represented as points.

2.8 Concluding remarks

We have outlined the necessary background definitions required for the rest of this thesis. In particular, we have introduced the concept of phylogenetic trees and why phylogenetic networks may be a more appropriate to visualise reticulate evolutionary events. The next chapter will present a new approach to constructing a dense collection of trinets from sequence data.

Chapter 3

Sequences to trinets

3.1 Chapter summary

In this chapter we present an approach to constructing a dense set of trinets from a multiple sequence alignment on a set of taxa X. This will allow us to apply our network reconstruction algorithm TriLoNet presented in Chapter 4 to real biological data sets.

3.2 Overview

To date, several methods, including MPNet [Fischer et al., 2015] and PhyloNet [Than et al., 2008], have been proposed to construct phylogenetic networks from sequence data based on some parsimony score. However, a problem in common with all of these methods is that they are unable to distinguish between an S_1 and S_2 trinet (defined in section 2.5), because they are based on the trees in the network and the two trees embedded in each of these two networks are identical.

In this chapter we present a novel approach for constructing trinets from sequence data which allows us to address this problem. This method, called SeqTrinet will take as input a multiple sequence alignment on a set of taxa X and associate a trinet to each subset in $\binom{X}{3}$, the collection on the subsets of X containing three elements. The SeqTrinet algorithm can be roughly divided into the following four parts:

1. Compute a score δ_t on each subset $t = \{x, y, z\}$ in $\binom{X}{3}$, introduced to measure the tree-likeness of the subalignment on the elements in t.

2. Using δ_t and a threshold κ divide the set $\binom{X}{3}$ into a subset Σ that contains those $t \in \binom{X}{3}$ with δ_t greater than or equal to κ and a subset Π that contains those $t \in \binom{X}{3}$ with δ_t less than κ.

3. Assign to every entry in Π a trinet, which is of type S_1 or S_2.

4. Using the binets displayed by the trinets associated with the elements in Π, assign a trinet to every element in Σ.

The details of the SeqTrinet algorithm are presented in Section 3.3, with pseudocode in Section 3.4 and the experiments to determine the κ threshold to be used in Step 2 in Section 3.5.

3.3 Method

In this section, we present a detailed description of our SeqTrinet algorithm. We assume we are given as input a multiple sequence alignment (MSA) $\mathcal{A}$ on a set of taxa X that has been preprocessed to remove sites containing gaps or any other characters other than $\{A, C, G, T\}$.

Step 1

In this first step we compute a score δ_t for each subset $t = \{x, y, z\}$ in $\binom{X}{3}$. Recall that a triplet $xy|z$ is a decomposition of $\{x, y, z\}$ into a pair $\{x, y\}$ and a singleton $\{z\}$. The three possible triplets on $\{x, y, z\}$ are $xy|z, xz|y$ and $yz|x$, which correspond to the three phylogenetic trees on $\{x, y, z\}$, see Figure 3.2.

One of the key concepts used in our approach for this step is the identification of the informative character sites in $\mathcal{A}$, where an informative site is a site for which precisely two of the three sequences are the same character. Non-informative sites, i.e. sites with three identical characters or three different characters are disregarded. For a triplet $xy|z$, we compute a *support weight* $w(xy|z)$ that, over every site in $\mathcal{A}$, counts the sites in $\mathcal{A}$ for which the character in sequence x and sequence y have the same and the character in sequence z is different. Note that for the set $\{x, y, z\}$, $w(xy|z) + w(xz|y) + w(yz|x)$ is therefore equal to the number of informative sites in $\mathcal{A}$.

An example of how informative sites are identified from a $\mathcal{A}$ is shown in Figure 3.1. The support weight for each of the three potential triplets of the three sequences is recorded in the three support weights $w(xy|z), w(xz|y)$ and $w(yz|x)$. Support weight $w(xy|z)$ is equal to 7 because of sites 1, 3, 6, 11, 12, 15, $w(xz|y) = 4$ because of sites 4, 7, 10, 13 and $w(yz|x) = 1$ because of site 8. Sites 2, 5, 9 and 14 are uninformative sites as the characters are either all identical or all different. Then for $t = \{x, y, z\}$, $\delta_t = \frac{6-4}{4-1} = \frac{2}{3}$.

	Site														
Taxa	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x	$\boxed{A}$	A	$\boxed{G}$	$\boxed{G}$	T	$\boxed{C}$	$\boxed{C}$	T	G	$\boxed{T}$	$\boxed{T}$	$\boxed{G}$	$\boxed{A}$	G	$\boxed{C}$
y	$\boxed{A}$	C	$\boxed{G}$	A	T	$\boxed{C}$	T	$\boxed{A}$	G	G	$\boxed{T}$	$\boxed{G}$	T	A	$\boxed{C}$
z	T	T	T	$\boxed{G}$	T	G	$\boxed{C}$	$\boxed{A}$	G	$\boxed{T}$	C	A	$\boxed{A}$	T	G

Figure 3.1: A multiple sequence alignment on $\{x, y, z\}$ depicting the identification of informative sites, which are 1, 3, 4, 6, 7, 8, 10, 11, 12, 13 and 15.

With three support weights computed and assuming without loss of generality $w(xy|z) \geq w(xz|y) \geq w(yz|x)$, the δ score on t denoted by δ_t is defined as the difference between the highest and second highest support weight divided by the difference between the second highest and lowest support weight, that is,

$$\delta_t = \frac{w(xy|z) - w(xz|y)}{w(xz|y) - w(yz|x)}.$$

If the value of the denominator is equal to zero then we follow the convention that $\delta_t = w(xy|z) - w(xz|y)$. Note that a similar number is defined for quartets in [Holland et al., 2002] which is used to determine tree-likeness of a distance matrix.

We have included the optional ability to specify breakpoints in the sequences which introduces some scaling into the trinet calculation. This may be useful in cases where breakpoints are known, and in particular when the lengths of these different regions vary, causing one portion of the MSA to heavily influence the overall weighting. If breakpoints are used, the δ_t score is calculated to give equal weighting to the different regions of the sequence alignment.

For example, given a MSA $\mathcal{A}$ on a set of taxa X with two breakpoints, the MSA can be split into three separate regions. For $i = 1, 2, 3$, let n_i denote the length of that region. We replace the quantity $w(xy|z)$ in the equation for δ_t by

$$\frac{n_1 w_1(xy|z) + n_2 w_2(xy|z) + n_3 w_3(xy|z)}{n_1 + n_2 + n_3}$$

and similarly for $w(xz|y)$ and $w(yz|x)$.

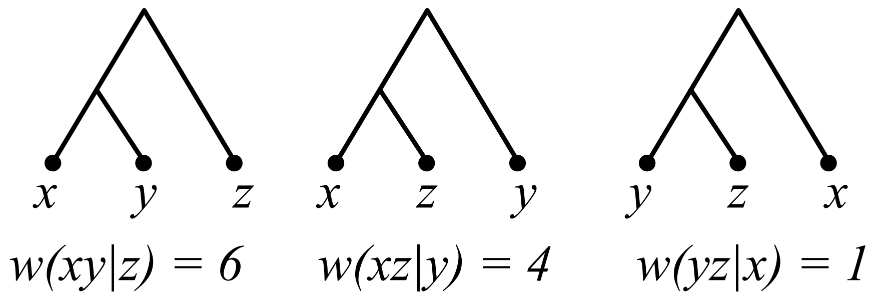


Figure 3.2: Using the informative site support weights to calculate δ_t for $\{x, y, z\}$.

Step 2

We next use a threshold κ to decompose $\binom{X}{3}$ into two disjoint subsets Σ and Π, that will be treated differently in the following steps. Formally, Σ consists of all $t \in \binom{X}{3}$ with δ_t greater than or equal to κ and Π consists of all $t \in \binom{X}{3}$ with δ_t less than κ. Intuitively, the set Σ contains the $t \in \binom{X}{3}$ which give rise to triplet topologies which are highly supported by $\mathcal{A}$, and Π contains those $t \in \binom{X}{3}$ whose triplet topologies are less clear and should probably be represented by an S_1 or S_2 trinet.

After performing some experiments (see Section 3.5) we found that a κ value between 6.0 and 7.0 works well in practice. Again considering the example in Figure 3.1, the value of δ_t suggests the set t does not show high support for any particular triplet over the other two possibilities. Therefore this set t will later be assigned either a S_1 or S_2 trinet.

Step 3

Having decomposed $\binom{X}{3}$ into Σ and Π, in this step we will assign to each $t \in \Pi$ a trinet, which will be of type S_1 or S_2. To do this, we determine for each $t = \{x, y, z\}$ in Π which of the three taxa x, y, z is to be placed under the reticulation vertex. To this end for each $x \in X$ we compute a score r_x. The score r_x counts the number of times leaf x appears in the sets contained in Π. Then, for every $t \in \Pi$, the taxon x in t with the highest score r_x will be designated to the taxon below the reticulation in S_1 or S_2. If two or more of the taxa for a $t \in \Pi$ have an equal maximal r_x score then from these taxa we choose uniformly at random the leaf to be placed under the reticulation.

With the taxon under the reticulation decided from the set $t = \{x, y, z\}$, we next from the three support weights $w(xy|z), w(xz|y)$ and $w(yz|x)$ examine the two highest scoring associated triplets. The cherries from these two triplets are then used to determine whether t will give rise to an S_1 or S_2 trinet.

More specifically, assuming that $xy|z$ and $xz|y$ are the two triplets with the highest support weights such that $w(yz|x) \leq w(xz|y) \leq w(xy|z)$, we associate to $t = \{x, y, z\}$ in Π either an S_1 or S_2 trinet with leaves in t in the following way:

- If $r_x \geq r_y$ and $r_x \geq r_z$ then t is assigned the trinet $S_1(y, z; x)$;
- If $r_y \geq r_x$ and $r_y \geq r_z$ then t is assigned the trinet $S_2(z; x; y)$;
- If $r_z \geq r_x$ and $r_z \geq r_y$ then t is assigned the trinet $S_2(y; x; z)$, (see Figure 3.3).

The idea behind this is to make sure the two triplets with the highest support weights are embedded in the assigned trinet. The set of trinets obtained from this step is denoted by $\mathcal{T}_\Pi$.

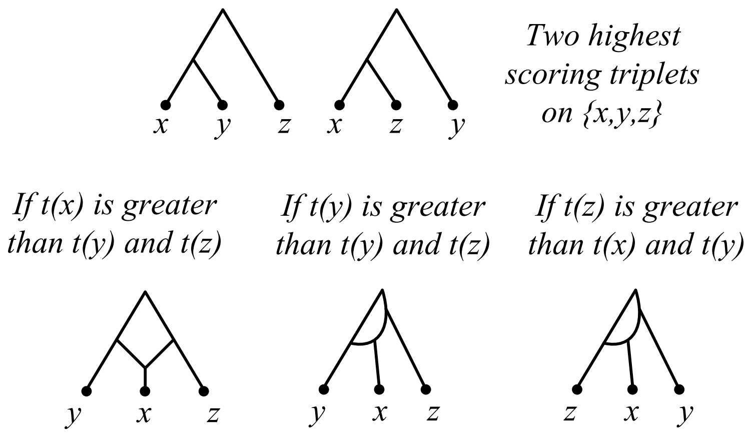

Two highest scoring triplets on {x,y,z}
x
y
z
x
z
y
If t(x) is greater than t(y) and t(z)
If t(y) is greater than t(y) and t(z)
If t(z) is greater than t(x) and t(y)
y
x
z
y
x
z
z
x
y

Figure 3.3: Deciding between a S_1 and S_2 trinet.

Step 4

The last step of SeqTrinet is to associate a trinet to every $t = \{x, y, z\}$ in Σ. For simplicity, we assume that the highest support weight for the three triplets associated with t is $w(xy|z)$.

Recall from Chapter 2 that there are two types of binets, as shown in Figure 3.4. We use the binets displayed by the trinets in $\mathfrak{T}_\Pi$ to associate a trinet to each $t = \{x, y, z\}$ in Σ. The idea behind this step is to associate trinets to the sets in Σ so as to maximise the consistency of the binets displayed by the trinets associated with the elements in Π.

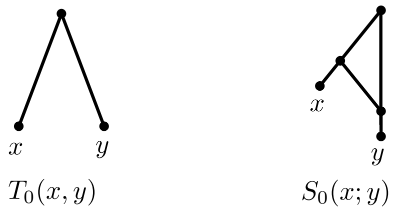


Figure 3.4: The two types of binet.

More specifically, for each $t = \{x, y, z\}$ let a_1, a_2 and a_3 be the number of trinets in $\mathfrak{T}_\Pi$ that display the binets $T_0(x, y)$, $S_0(x; y)$ and $S_0(y; x)$, respectively. These scores determine the binet type and order on the cherry $\{x, y\}$ in $T_1(x, y; z)$. An $S_0(x; y)$ or $S_0(y; x)$ binet is chosen over a $T_0(x, y)$ binet only if a_2 or a_3 is greater than a_1. If a_2 and a_3 are equal and also both greater than a_1 then a_2 is selected as the maximum.

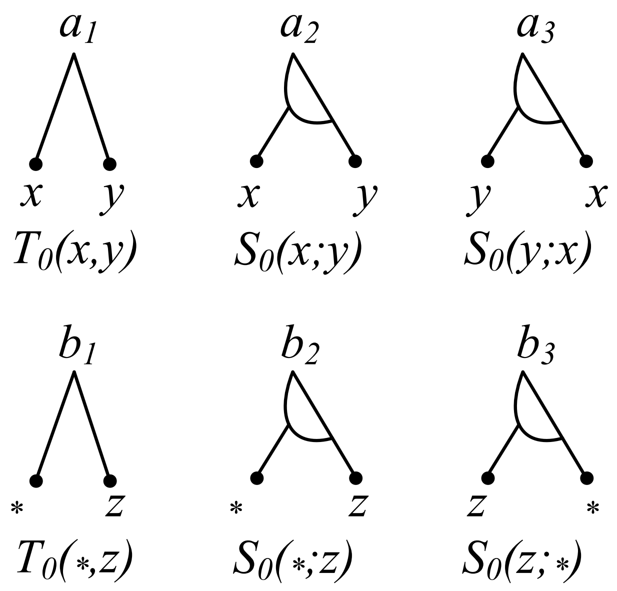


Figure 3.5: Binet Structures.

Next we follow a similar process to determine the structure and position of leaf z. Again, we let b_1, b_2 and b_3 be the number of trinets in $\mathcal{T}_\Pi$ that display the binets $T_0(z,*)$, $S_0(z;*)$ and $S_0(*;z)$ respectively, with $* \in \{x,y\}$. Then, similarly, an $S_0(*;z)$ or $S_0(z;*)$ is chosen over $T_0(z,*)$ only if b_2 or b_3 is greater than b_1. If b_2 and b_3 are equal and also both greater than b_1 then b_2 is selected as the maximum. Using the maximum a_i scores and the maximum b_i scores we use the following table to determine the trinet on $\{x,y,z\}$. For example, if a_3 and b_1 are maximal we assign the trinet $N_3(y,x;z)$.

Table 3.1: The lookup table used in Step 4 of the algorithm.

a_i	b_i	Trinet
a_1	b_1	$T_1(x, y; z)$
a_1	b_2	$N_1(x, y; z)$
a_1	b_3	$N_2(x, y; z)$
a_2	b_1	$N_3(x, y; z)$
a_2	b_2	$N_4(x, y; z)$
a_2	b_3	$N_5(x, y; z)$
a_3	b_1	$N_3(y, x; z)$
a_3	b_2	$N_4(y, x; z)$
a_3	b_3	$N_5(y, x; z)$

3.4 Pseudocode

We now present in pseudocode the SeqTrinet algorithm that takes as input a multiple sequence alignment $\mathcal{A}$ on a set of taxa X and outputs a dense set of trinets $\mathcal{T}$ on X.

Algorithm 2 SeqTrinet Algorithm

INPUT: Multiple sequence alignment $\mathcal{A}$ on set of taxa X and a threshold κ.

OUTPUT: Dense set of trinets $\mathcal{T}$ on X.

```
for Every $t = \{x, y, z\} \in \binom{X}{3}$ do
    Compute $w(xy|z), w(xz|y)$ and $w(yz|x)$ using informative sites
end for
Calculate $\delta_t$ for every $t = \{x, y, z\}$ in $\binom{X}{3}$
Let $\Sigma = \{t \in \binom{X}{3} : \delta_t \geq \kappa\}$ and $\Pi = \{t \in \binom{X}{3} : \delta_t < \kappa\}$.
for Every leaf $a \in X$ do
    $r_a$ = number of $t \in \Pi$ containing $a$
end for
for Every $t = \{x, y, z\} \in \Pi$ do
    Assign $z \in \{x, y, z\}$ with maximum $r_z$ score as the leaf below reticulation
    Select cherries from two highest values from $w(xy|z), w(xz|y)$ and $w(yz|x)$
    if Both cherries contain $z$ then
        Associate t to $S_1(x, y; z)$
    else
        if $z$ is contained in first cherry $\{z, y\}$ then
            Associate t to $S_2(x, y; z)$
        end if
        if $z$ is contained in second cherry $\{z, y\}$ then
            Associate t to $S_2(x, y; z)$
        end if
    end if
end for
$B_\Pi$ = binets displayed by the $S_1$ and $S_2$ trinets associated with every $t$ in $\Pi$
for Every triplet with highest support weight $w(xy|z)$ associated with $t$ in $\Sigma$
do
    Use $B_\Pi$ and lookup table to associate $t$ to trinet of type other than $S_1$ or $S_2$
end for
Output a dense set of trinets on $X$ corresponding to the elements in $\Sigma \cup \Pi$.
```

3.5 κ threshold experiments

When developing the SeqTrinet algorithm we performed some experiments to identify an appropriate value for the κ threshold used in Step 2 of the SeqTrinet algorithm. To this end, we conducted a simulation study using the Seq-Gen [Rambaut and Grass, 1997] software for simulating molecular sequence data.

3.5.1 Generating collections of weighted trinets

We generate a collection $\mathrm{Tr}_{\mathcal{C}}$ of arc-weighted trinets for which the distance from the root of a trinet to each of the three leaves is the same. This is assumed as we are assuming a molecular clock which is required in the Seq-Gen software. For example, given an arc $a = (u, v)$, a weight of 5 would indicate there would be 5 substitutions per site in the time taken for the sequence to evolve from u to v.

The trinets of type T_1 in this collection $\mathrm{Tr}_{\mathcal{C}}$ are generated in the following way. Given trinet $t = T_1(x, y; z)$ in Figure 3.6 as an example, the parameters d_1 and d_2 are varied such that $d_1 = i$ and $d_2 = 100 - i$ for $1 \leq i < 100$. With these parameters we generate a collection of 99 weighted T_1 trinets.

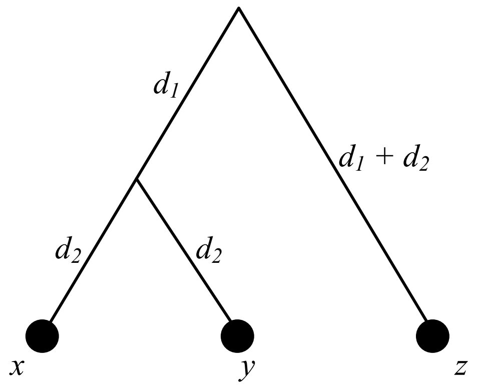


Figure 3.6: Generating arc lengths on a T_1 trinet.

For trinets of type other than T_1 in the collection $\mathrm{Tr}_{\mathcal{C}}$, we generate arc-weighted trinets in the following way. Given the trinet $S_2(y; z; x)$ presented in Figure 3.7 as an example, each of the four parameters d_1, d_2, d_3, d_4 are varied between 1 and 20 while the total height i.e $d_1 + d_2 + d_3 + d_4$ is 20. More precisely, we generate solutions $\{d_1, d_2, d_3, d_4\}$ such that $d_1 + d_2 + d_3 + d_4 = 20$ with $1 \leq d_i \leq 20$ for $1 \leq i \leq 4$. For all trinet types other than T_1 the total height is also set to 20, where the number 20 is chose to set a reasonable size for the collection of weighted trinets and to also keep the computational time reasonable. The use of these parameters resulted in a collection of 153 weighted S_2 trinets.

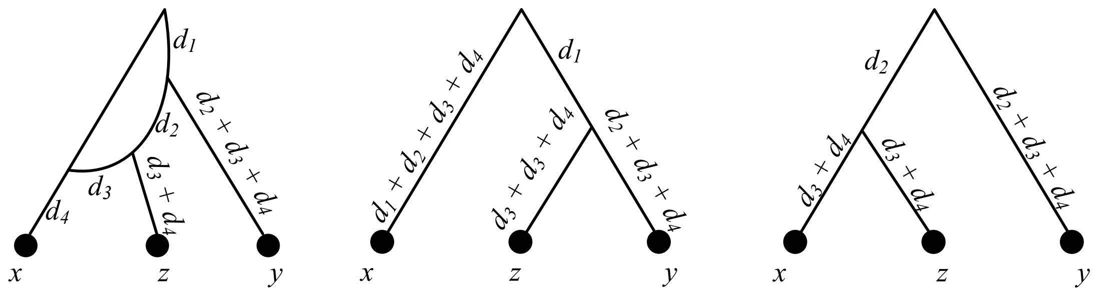


Figure 3.7: Extracting the two trees embedded in an S_2 trinet and computing the arc lengths.

A similar approach was used to obtain collections for the other types of trinets. For trinets of type S_2, four parameters were used to generate the arc weights. The number of parameters required for a particular type of trinet is dependent on the number of arcs in a trinet of that type. The total size of each collection for each trinet type is presented in Table 3.2.

Trinet Type	T_1	N_1	N_2	N_3	N_4	N_5	S_1	S_2
Size of collection	99	171	153	152	815	679	323	153

Table 3.2: Number of trinets generated in each collection for each type of trinet.

3.5.2 Simulation of recombination data sets

Having generated the collection of weighted trinets, we now generate an MSA for each weighted trinet in $\mathrm{Tr}_{\mathcal{C}}$. For the trinets that displayed one or more trees such that the root vertex of that tree and the root vertex of the trinet were not equal, the expected substitution rate for that tree was calculated in proportion to the expected substitution rate for a tree that shared its root vertex with the trinet.

An example of identifying the trees in an S_2 trinet is show in Figure 3.7. The total height of the S_2 trinet is equal to $d_1 + d_2 + d_3 + d_4$. The tree $zy|x$ is also the same height. The height of the other tree $xz|y$ embedded in $S_{(y;z;x)}$ is a proportion of the total height of the trinet equal to $d_2 + d_3 + d_4$.

An expected substitution rate of 0.3 was used in the experiments performed in [Holland et al., 2002], for the following simulation experiments we tested this value as well as some values above and below. For each expected substitution rate $\gamma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and for the tree $zy|x$ the expected number of substitutions from the root to each taxa is calculated as

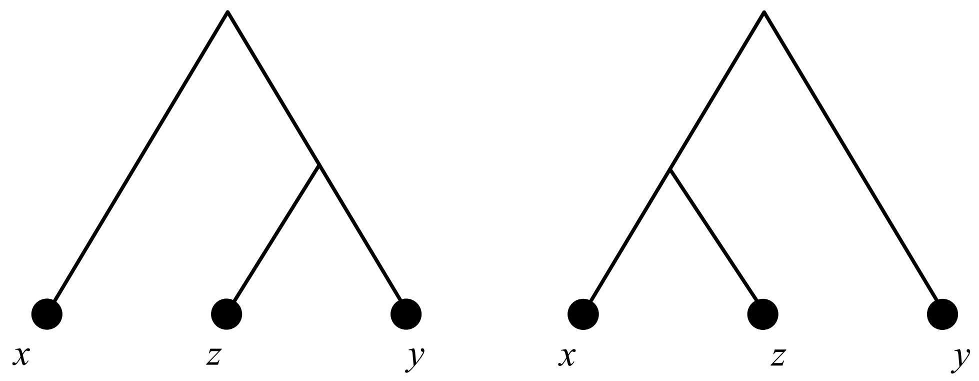
$$\frac{d_1 + d_2 + d_3 + d_4}{d_1 + d_2 + d_3 + d_4} \times \gamma,$$

and for the tree $xy|z$ the expected number of substitutions from the root to each taxa is calculated as

$$\frac{d_2 + d_3 + d_4}{d_1 + d_2 + d_3 + d_4} \times \gamma.$$

A similar approach is taken to compute the expected substitution rate for every tree embedded in each trinet in proportion to the total height of the trinet.

We then used Seq-Gen [Rambaut and Grass, 1997] to simulate recombination sequence data sets. Seq-Gen is able to, given a tree topology, simulate the evolution of sequences along that tree. By concatenating two sequence data sets produced by two different tree topologies such that the first tree underlies the first part of the sequence alignment and the second tree underlies the second portion, we simulate a recombination data set. An example of this sequence concatenation is presented in Figure 3.8, where the first to the fifth characters are simulated from the tree $(yz|x)$ and the sixth to tenth characters are simulated from the tree $xz|y$.



	Site									
Taxa	1	2	3	4	5	6	7	8	9	10
x	G	A	G	C	A	A	C	C	T	A
y	A	C	G	G	T	T	C	G	G	T
z	A	C	G	G	T	A	C	C	T	A

Figure 3.8: An example of the simulation of a recombinant data set on an $S_2(y; z; x)$ trinet. Seq-Gen is used to simulate the evolution of sequences of length 5 on the two trees $yz|x$ and $xz|y$ embedded in $S_2(y; z; x)$. The two data sets are concatenated to give a MSA with sequences of length 10 on three taxa x, y and z.

In Seq-Gen we used the K2P model [Kimura, 1980] of sequence evolution which is selected in Seq-Gen by setting the model parameter to $mHKY$, $t2.0$ (to set the transition-transversion bias to 4) and parameter $l5000$ for the lengths of the sequences to be generated. As discussed in the next section, in the experiments we vary d the scale tree length parameter. All other parameters were left with their default settings.

We used Seq-Gen to simulate the evolution of sequences on all of the trinets. As Seq-Gen takes trees as input, we used the trees embedded in the trinets. The trinet T_1 contains only one embedded tree on three leaves (itself), trinets N_1, N_2 and N_3 each contain two trees. Trinets of type N_4, N_5 both contain four trees, again distinguished by the selection of the root vertex and last stable ancestor of leaves x and y. Trinets S_1, S_2 each contain two different trees.

For each tree embedded in a trinet and for each taxa, sequences of length

5,000 were generated. For the trinet T_1 the same tree was used to generate two sets of sequences which were then concatenated. All sequences produced from the trees inside trinets aside from N_4 and N_5 were 10,000 characters in length. As trinets N_4 and N_5 contained four embedded trees, the sequences produced for these trinets were 20,000 characters in length.

3.5.3 Experiments and results

For each experiment we fix the expected substitution rate for some $\gamma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and vary κ between 4 and 10, with the aim of selecting an appropriate value for κ. For each weighted trinet $t \in \mathrm{Tr}_{\mathcal{C}}$ we generate a multiple sequence alignment $S_{t\gamma}$ on the three taxa of t as described above. We next compute $\delta(S_{t\gamma})$ for every trinet in the collection $\mathrm{Tr}_{\mathcal{C}}$. A weighted trinet is said to be correctly separated by κ if $\delta(S_{t\gamma}) < \kappa$ when t is of type S_1 or S_2 and $\delta(S_{t\gamma}) \geq \kappa$ otherwise.

For each expected substitution rate and each κ value between 4.0 and 10.0 we calculated the average number of correctly separated trinets for the collection of trinets $\mathrm{Tr}_{\mathcal{C}}$ introduced in Section 3.5.1. These results are presented in Table 3.3 and indicate that a κ value of 6.0 or 7.0 would be suitable for separating S_1 and S_2 trinets from the other types of trinet since these values would be a reasonable compromise in the for the identification of T_1 trinets as well as S_1 and S_2 trinets.

	0.1									
	T1	N1	N2	N3	N4	N5	S1	S2		Average
Kappa										
4	95.96	97.08	100.00	99.34	99.88	99.85	81.11	73.86		93.38
5	92.93	97.08	100.00	99.34	99.88	99.85	84.83	77.78		93.96
6	90.91	96.49	99.35	98.68	99.88	99.85	86.38	81.05		94.07
7	85.86	95.91	98.69	96.71	99.63	99.85	88.24	84.31		93.65
8	84.85	94.74	97.39	96.05	99.26	99.71	89.16	84.97		93.27
9	83.84	94.15	96.08	95.39	98.90	99.26	90.40	86.93		93.12
10	81.82	92.40	95.42	94.74	98.40	98.82	91.64	88.24		92.69

Figure 3.9: A table presenting the percentage of correctly separated trinets for each type of trinet with a γ value of 0.1 and the κ threshold varied between 4 and 10.

0.2										
	T1	N1	N2	N3	N4	N5	S1	S2		Average
Kappa										
4	93.94	100.00	99.35	100.00	100.00	100.00	81.11	73.86		93.53
5	88.89	99.42	99.35	99.34	99.88	100.00	85.45	79.74		94.01
6	88.89	99.42	99.35	98.68	99.88	100.00	87.93	81.70		94.48
7	86.87	98.83	99.35	98.68	99.63	99.85	89.78	84.97		94.75
8	84.85	98.25	98.69	98.03	99.63	99.85	90.71	86.93		94.62
9	83.84	98.25	98.69	97.37	99.51	99.85	92.57	88.24		94.79
10	81.82	97.08	98.69	97.37	99.26	99.85	92.57	88.89		94.44

Figure 3.10: A table presenting the percentage of correctly separated trinets for each type of trinet with a γ value of 0.2 and the κ threshold varied between 4 and 10.

0.3										
	T1	N1	N2	N3	N4	N5	S1	S2		Average
Kappa										
4	96.97	100.00	100.00	99.34	100.00	100.00	77.71	73.86		93.48
5	90.91	100.00	99.35	99.34	100.00	99.85	80.19	77.78		93.43
6	87.88	99.42	99.35	98.03	100.00	99.85	85.76	82.35		94.08
7	86.87	99.42	98.69	98.03	99.88	99.71	87.93	84.97		94.43
8	82.83	99.42	98.69	96.71	99.88	99.71	89.47	86.93		94.20
9	81.82	98.25	98.04	95.39	99.63	99.56	92.26	88.89		94.23
10	81.82	97.66	96.73	94.74	99.63	99.41	92.88	90.20		94.13

Figure 3.11: A table presenting the percentage of correctly separated trinets for each type of trinet with a γ value of 0.3 and the κ threshold varied between 4 and 10.

0.4										
	T1	N1	N2	N3	N4	N5	S1	S2		Average
Kappa										
4	91.92	100.00	100.00	100.00	100.00	100.00	77.09	73.86		92.86
5	90.91	98.25	100.00	99.34	100.00	100.00	81.73	77.12		93.42
6	90.91	98.25	100.00	98.03	99.88	100.00	85.76	79.08		93.99
7	90.91	97.66	100.00	97.37	99.75	100.00	87.62	81.70		94.38
8	87.88	97.66	99.35	96.05	99.14	99.85	89.47	84.31		94.21
9	87.88	97.66	98.69	94.74	98.65	99.85	90.40	87.58		94.43
10	87.88	97.66	98.04	92.76	98.28	99.85	92.26	89.54		94.53

Figure 3.12: A table presenting the percentage of correctly separated trinets for each type of trinet with a γ value of 0.4 and the κ threshold varied between 4 and 10.

0.5										
	T1	N1	N2	N3	N4	N5	S1	S2		Average
Kappa										
4	88.89	98.25	99.35	97.37	100.00	100.00	77.09	71.90		91.60
5	88.89	97.66	99.35	96.05	100.00	100.00	79.57	77.78		92.41
6	88.89	97.08	99.35	95.39	99.88	100.00	82.35	80.39		92.92
7	88.89	95.32	97.39	95.39	99.63	100.00	84.52	81.05		92.77
8	86.87	94.74	97.39	94.74	99.39	99.85	85.76	83.66		92.80
9	85.86	94.74	95.42	94.08	99.14	99.85	87.62	86.93		92.95
10	83.84	94.74	95.42	92.76	98.53	99.71	89.47	89.54		93.00

Figure 3.13: A table presenting the percentage of correctly separated trinets for each type of trinet with a γ value of 0.5 and the κ threshold varied between 4 and 10.

	γ				
κ	0.1	0.2	0.3	0.4	0.5
4	93.38	93.53	93.48	92.85	91.60
5	93.96	94.00	93.42	93.41	92.41
6	94.07	94.47	94.07	93.98	92.91
7	93.65	94.74	94.43	94.37	92.77
8	93.26	94.61	94.20	94.21	92.79
9	93.11	94.78	94.22	94.43	92.95
10	92.68	94.44	94.13	94.53	93.00

Table 3.3: Average percentage across all types of trinets correctly separated for κ values between 4 and 10 for expected substitution rate γ between 0.1 and 0.5.

A κ value of 9 or 10 increases the percentage of S_1 and S_2 trinets with a score below κ. However, using this value impacts on the identification of T_1 trinets. Loosely speaking, using a value this high would mean our algorithm would identify T_1 trinets as either S_1 or S_2 trinets. The variation of the κ parameter variation had less impact on the identification of $N_i, 1 \leq i \leq 5$ trinets in comparison to trinets of type T_1, S_1 and S_2.

3.6 Concluding remarks

We have introduced a method with four key steps that will output a dense set of trinets when given as input a multiple sequence alignment. We have also detailed the experiments we performed and found a suitable κ threshold value of 6.5 to be used in Step 2 of the SeqTrinet algorithm. We will now use the output produced by SeqTrinet as input to our main algorithm TriLoNet, presented in Chapter 4.

Chapter 4

Network construction

4.1 Chapter summary

In this chapter, we present the algorithm TriLoNet which constructs level-1 phylogenetic networks from a dense collection of trinets. We begin with some preliminary definitions as well as some useful theoretical results. We then describe the key steps of the TriLoNet algorithm and include some proofs on its complexity and consistency.

4.2 Definitions

We begin by introducing some additional notation that we will need in this chapter. There are three types of subnets that will be particularly important for our approach to reconstructing level-1 networks. The first one is a *cherry* of N. This is a subnet consisting of an interior vertex of N incident with two leaves (see Figure 4.1). We also consider *reticulate cherries*, which are like cherries except they contain a cycle of length 3 (see Figure 4.1).

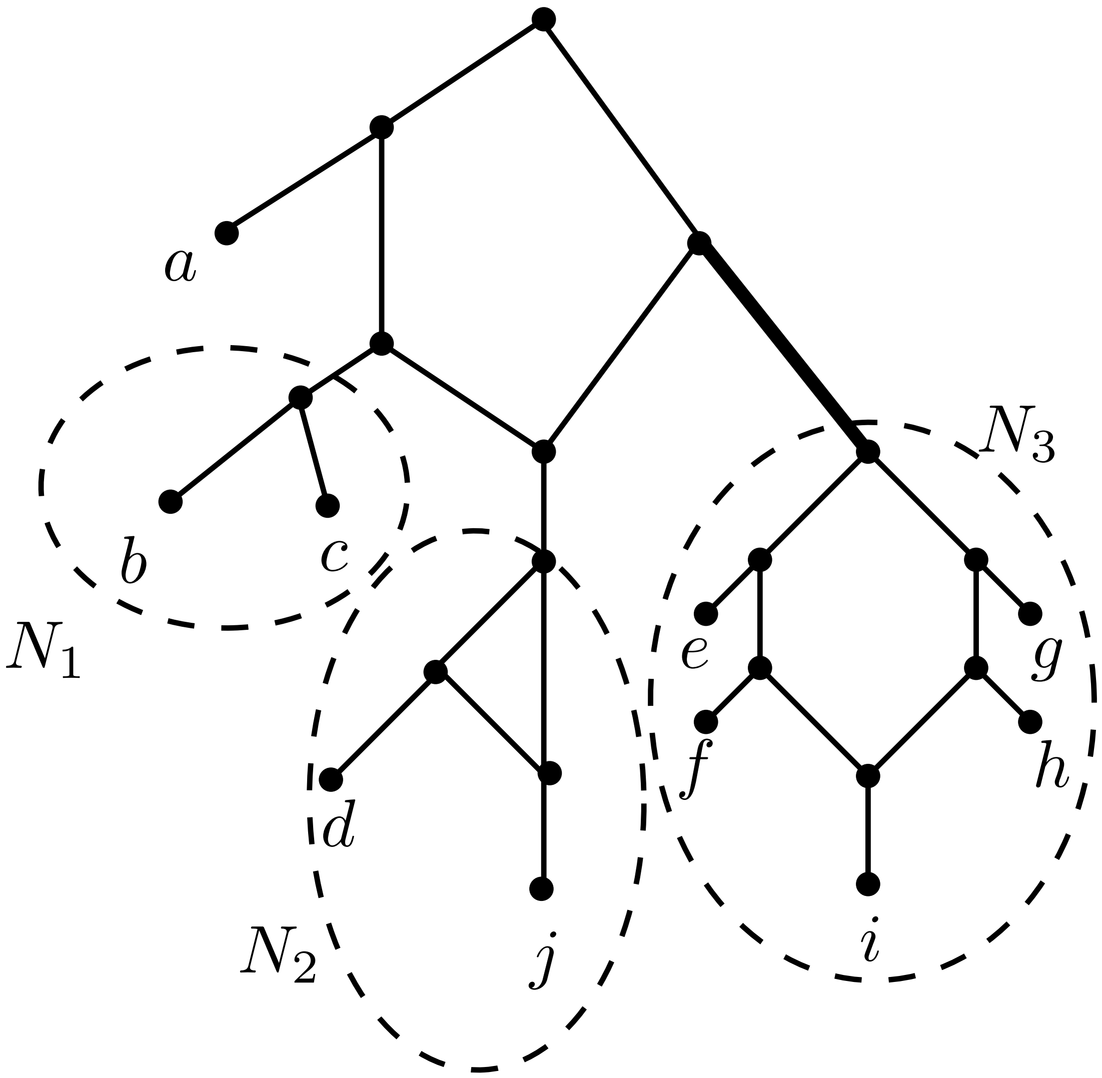


Figure 4.1: Example of a level-1 phylogenetic network. The network contains a cherry N_1, a reticulate-cherry N_2 and a cactus N_3 as indicated by the dotted circles. Here all arcs are directed from the root, and the arc highlighted in bold is a cut-arc.

The other type of subnet of special interest is a *cactus* or a subnet with at least three leaves whose associated undirected graph contains one cycle and each vertex is either contained in this cycle or a child of a vertex of this cycle (see Figure 4.1). More precisely, a *cactus* in a network N on X is the subnet $H = N|_Y$ for some subset $Y = \{a_1, \dots, a_p, b_1, \dots, b_q, z\}$ of X with $p + q \geq 2$ such that $\underline{H}$ contains exactly one cycle and $p + q + 1$ pendant leaves as shown in Figure 4.2. Here

z will be referred to as the *bottom leaf* of H, and its leaf set Y the *support of H*. For brevity, we also say that the tuple $(a_1, a_2, \ldots, a_p : b_1, b_2, \ldots, b_q : z)$, or equivalently $(b_1, b_2, \ldots, b_q : a_1, a_2, \ldots, a_p : z)$, is a cactus.

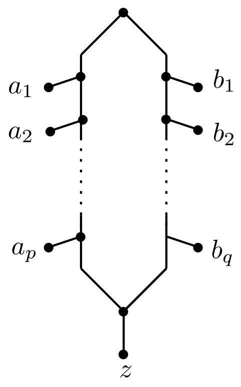


Figure 4.2: An illustration of a cactus. The support of the cactus is $\{a_1, \ldots, a_p, b_1, \ldots, b_q, z\}$.

Following [Iersel and Moulton, 2013], a subset A of X is called a CA-set (Cut-Arc set) of N if there exists a cut-arc (u, v) of N such that $A = \mathcal{C}(v)$. That is, A is a CA-set if there exists a cut-arc (u, v) in the network such that a taxon is contained in A precisely when it is below v. A CA-set contains at least two leaves as here we consider only cut-arcs not incident with a leaf. We call a CA-set *minimal* if no proper subset B of A is a CA-set. Note that a minimal CA-set in a network is necessarily the leaf set of a cherry, reticulate-cherry or a cactus.

Next, we extend the concept of SN-sets, which were introduced for constructing networks from triplets [Jansson and Sung, 2006], [Jansson et al., 2006]. A subset A of taxa is called an *SN-set* of a collection $\mathcal{T}$ of trinets if it is a singleton or, for every trinet T in $\mathcal{T}$ which contains precisely two taxa from A, these two taxa form a CA-set of T. This definition is in agreement with those in [Jansson and Sung, 2006; Jansson et al., 2006] when each trinet in the collection $\mathcal{T}$ is type T_1, that is, it is a triplet. In addition, a subset of taxa is an SN-set of $\mathcal{T}(N)$ if and only if it is a singleton, the set X or a CA-set of N (see, also [Iersel and Moulton, 2013, Theorem 1]).

Finally, a *small SN-set* is a non-singleton SN-set for which none of its proper non-singleton subsets is an SN-set. More precisely, an SN-set A (with $|A| \geq 2$) of a collection of trinets $\mathcal{T}$ is called a small SN-set if $A \subseteq X$, but no proper subset $B \subset A$ with $|B| > 1$ is an SN-set.

4.3 Theoretical results

The first step of our TriLoNet algorithm is to identify small SN-sets. For a set $\mathcal{T}$ of dense trinets on X we construct a digraph on X: (x, y) is an arc in $\Omega_i(\mathcal{T})$ for taxon x to y if and only if there is no taxon $z \in X - \{x, y\}$ such that the set $\{x, z\}$ is a CA-set for the trinet T in $\mathcal{T}$ with leaf set $\{x, y, z\}$. For example, Figure 4.3 is the digraph $\Omega(\mathcal{T}(N))$ for the trinets $\mathcal{T}(N)$ induced by the network N depicted in Figure 4.1.

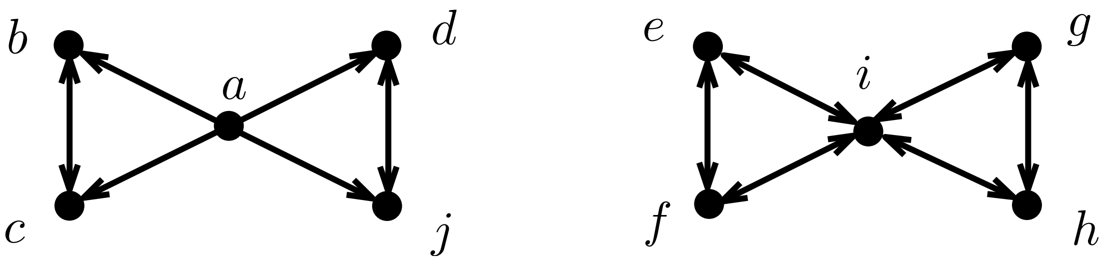


Figure 4.3: Digraph $\Omega(\mathcal{T}(N))$ for the network N shown in Figure 4.1.

In general, a subset A of the vertex set of a digraph is called a *sink set* if there exists no arc (u, v) in the digraph with $u \in A$ and $v \notin A$. In addition, a sink set A in a digraph G is *minimal* if none of its proper subsets is a sink set. We refer to a sink set A as *small* if A is non singleton and none of its proper, non-singleton subsets is a sink set. For the network N depicted in Figure 4.1, the minimal CA-sets are $\{b, c\}$, $\{d, j\}$ and $\{e, f, g, h, i\}$, and these are the same as the minimal sink sets in $\Omega(\mathcal{T}(N))$ (see Figure 4.3). As a generalisation of this observation, the following theorem relates minimal CA-sets, small SN-sets, and minimal sink sets for a level-1 network N.

Theorem 4.3.1. *Suppose that N is a level-1 phylogenetic network on X with $|X| \geq 3$. Then the following assertions are equivalent for subsets $A \subset X$ with $1 < |A| < |X|$:*

(i) *A is a minimal CA-set in N.*

(ii) *A is a small SN-set of $\mathfrak{T}(N)$.*
(iii) *A is a minimal sink set in $\Omega(\mathfrak{T}(N))$*

To establish this theorem, we need the following three lemmas. The first one relates CA-sets and SN-sets, whose proof is omitted as it follows directly from [Iersel and Moulton, 2013, Theorem 1].

Lemma 4.3.2. *Suppose that N is a level-1 network on X with $|X| \geq 3$ and A is a subset of X with $1 < |A| < |X|$. Then A is an SN-set in $\mathfrak{T}(N)$ if and only if A is a CA-set in N.* □

The second lemma relates SN-sets to sink subsets in the digraph Ω.

Lemma 4.3.3. *Suppose that $\mathfrak{T}$ is a dense collection of trinets on X with $|X| \geq 3$. Let A be an SN-set of $\mathfrak{T}$ with $|A| \geq 2$. Then A is a sink set in $\Omega(\mathfrak{T})$.*

Proof. Suppose that (x, y) is an arc in $\Omega(\mathfrak{T})$ and $x \in A$. Then it suffices to show $y \in A$. To this end, fix a taxon $a \in A - \{x\}$. Without loss of generality, we may further assume $a \neq y$ as otherwise we clearly have $y \in A$.

Next, since $\mathfrak{T}$ is dense, there exists a unique trinet T in $\mathfrak{T}$ with leaf set $\{x, y, a\}$. Noting that $(x, y) \in \Omega(\mathfrak{T})$, it follows that the set $\{x, a\}$ is not a CA-set in the trinet T. This implies that $y \in A$ as otherwise $A \cap \{x, y, a\}$ has cardinality two and is not a CA-set in T, a contradiction to the fact that A is an SN-set. □

For a network N, the third lemma relates minimal sink sets in the digraph $\Omega(\mathfrak{T}(N))$ to CA sets in N.

Lemma 4.3.4. *Suppose that N is a binary level-1 phylogenetic network on X with $|X| \geq 3$, and A is a minimal sink set in $\Omega(\mathfrak{T}(N))$. Then $|A| \geq 2$ and hence A is a small sink set. In addition, A is either the set X or a CA-set in N.*

Proof. For simplicity, let $\mathfrak{T} = \mathfrak{T}(N)$. Suppose that A is a minimal sink set in $\Omega(\mathfrak{T})$. We may assume that $|A| < |X|$ as otherwise the lemma clearly holds.

Fix a taxon $x \in A$ and let $p(x)$ be the parent of x in N. Note that for each $y \in \mathcal{C}(p(x)) - \{x\}$ and a trinet T in $\mathfrak{T}$ that contains both x and y, leaf y is below the parent of x in T. This implies that there exists no taxon z in $X - \{x, y\}$ such that the subset $\{x, z\}$ is a CA-set for the trinet in $\mathfrak{T}$ with leaf set $\{x, y, z\}$. Hence

(x, y) is an arc in $\Omega(\mathcal{T})$. Therefore, we have $\mathcal{C}(p(x)) \subseteq A$ because A is a sink set in $\Omega(\mathcal{T})$.

Note that we may further assume that $p(x)$ is not the root of N because otherwise we have $X = \mathcal{C}(p(x)) = A$, a contradiction. Denoting the parent of $p(x)$ by x^*, we divide the remainder of our proof into the following two cases:

Case i: The arc $(x^*, p(x))$ is a cut-arc in N. Then $\mathcal{C}(p(x))$ is a CA-set of N with $2 \leq |\mathcal{C}(p(x))| < |X|$. By Lemma 4.3.4 and Lemma 4.3.3 it follows that $\mathcal{C}(p(x))$ is a sink set in $\Omega(\mathcal{T})$. Using the minimality of A, we must have $\mathcal{C}(p(x)) = A$, from which we can conclude that $|A| \geq 2$ and A is a CA-set in N, as required.

Case ii: The arc $(x^*, p(x))$ is not a cut-arc in N. Let H be the cycle that contains $p(x)$. In addition, let h^* be the reticulate vertex contained in H, and let B denote the subset of X consisting of elements $y \in X$ that are below some vertex in H. Then $|B| \geq 2$. In addition, the set B is either X or a CA-set in N. Hence B is necessarily a sink set in $\Omega(\mathcal{T})$. Now we have the following two subcases.

The first subcase is $h^* = p(x)$. Then x is the only child of h^*. In addition, for each element $y \in B - \{x\}$ and a trinet T in $\mathcal{T}$ that contains both x and y, taxon x is the only child of a reticulate vertex v in T and y is below some vertex in the biconnected component of T containing v. Hence (x, y) is an arc in $\Omega(\mathcal{T})$. This implies $B \subseteq A$ because A is a sink set in $\Omega(\mathcal{T})$. In addition, it follows that $|B| < |X|$ and B is a CA-set in N. Finally, by the minimality of A, we have $A = B$, and thus $|A| \geq 2$ and A is a CA-set in N, as required.

The other subcase is that $h^* \neq p(x)$. Then h^* is below $p(x)$ in N, and hence $\mathcal{C}(h^*) \subset \mathcal{C}(p(x)) \subseteq A$. Note that we must have $|\mathcal{C}(h^*)| = 1$ as otherwise $\mathcal{C}(h^*)$ is a CA-set of N and hence also a sink set in $\Omega(\mathcal{T})$ by Lemma 4.3.4 and Lemma 4.3.3, a contradiction to the minimality of A. In addition, $|A| \geq 2$. Denoting the leaf below h^* by h, then (x, h) is an arc in $\Omega(\mathcal{T})$. By an argument similar to that in the first subcase, it follows that (h, y) is an arc in $\Omega(\mathcal{T})$ for all $y \in B - \{h\}$. Therefore, B is a subset of A and hence $|B| < |X|$. This implies that B is a CA-set of N. On the other hand, by the minimality of A we have $A = B$. Hence A is a CA-set in N, which completes the proof of the theorem. □

With the last three lemmas, we are in a position to prove the main result of this section.

Proof of Theorem 4.3.1: To simplify notation, let $\mathcal{T} = \mathcal{T}(N)$. The equivalence between (i) and (ii) follows directly from Lemma 4.3.4.

We will first show that (ii) implies (iii). To this end, suppose that A is a small SN-set of $\mathcal{T}$. Then by Lemma 4.3.3 we know that A is a sink set in $\Omega(\mathcal{T})$. Note that if A were not a minimal sink set in $\Omega(\mathcal{T})$, then there would exist a subset $A' \subset A$ so that A' is a minimal sink set in $\Omega(\mathcal{T})$. By Lemma 4.3.4, we have $|A'| > 1$ and A' is a CA-set of N. By Lemma 4.3.4, it follows that A' is an SN-set of $\mathcal{T}$, a contradiction to the assumption that A is a small SN-set of $\mathcal{T}$. Therefore we can conclude that A is a minimal sink set in $\Omega(\mathcal{T})$, as required.

It remains to show (iii) implies (ii). To this end, assume that A is a minimal sink set in $\Omega(\mathcal{T})$. Then by Lemma 4.3.4 and Lemma 4.3.4 it follows that A is an SN-set of $\mathcal{T}$. Note that if A were not a small SN-set of $\mathcal{T}$, then there would exist a subset $A' \subset A$ such that $|A'| > 1$ and A' is an SN-set of $\mathcal{T}$. By Lemma 4.3.3, A' is a sink set in $\Omega(\mathcal{T})$, a contradiction to the assumption that A is a minimal sink set in $\Omega(\mathcal{T})$. Therefore we can conclude that A is a small SN-set of $\mathcal{T}$, as required. □

4.4 The TriLoNet algorithm

In this section, we present the main algorithm of TriLoNet to reconstruct level-1 networks from a dense set of level-1 trinets. We adopt a bottom-up approach, a feature shared with the Neighbor-Joining algorithm used for the inference of phylogenetic trees [Saitou and Nei, 1987]. Loosely speaking, given a dense trinet set $\mathcal{T}$ on X, we first identify an appropriate subset Y of X and construct a cherry, reticulate cherry or cactus N_Y on Y using the restriction of $\mathcal{T}$ on Y. Next, we compute the trinet set $\mathcal{T}^*$ induced by $\mathcal{T}$ on the set X^* formed by replacing the subset Y in X with a new element y^* and then we obtain a level-1 network N^* from $\mathcal{T}^*$ recursively. Finally, we combine the two level-1 networks N_Y and N^* to form a level-1 network on X.

4.4.1 Finding small SN-sets

Here, we introduce the FindSmallSNSet$(X, \mathcal{T})$ algorithm to identify small SN-sets Y of X for a dense trinet set on X. It works as follows.

We first identify the the smallest i such that $\varphi(x, y) > |X| - 2 - i$ holds for some pair x, y in X. We construct a digraph Ω with $V(D) = X$ and (x, y) is an arc in D if and only if $\varphi(x, y) > |X| - 2 - i$. If we are unable to identify a graph containing at least one edge then we output $Y = X$. Otherwise, we construct the condensed digraph $\Omega^*(\mathcal{T})$ of Ω using Tarjan's algorithm [Tarjan, 1972] which identifies the strongly connected components of a graph. Essentially, the connected components of Ω form the vertex set of $\Omega^*(\mathcal{T})$ and (π_1, π_2) is an arc in $\Omega^*(\mathcal{T})$ if and only if for some vertex there exists an arc from some vertex in $L(\pi_1)$ to some vertex in $L(\pi_2)$.

For each vertex u in $\Omega^*(\mathcal{T})$, let π_u be equal to the set of vertices in Ω contained in the strongly connected component corresponding to u. The set A^* is equal to the set of leaves contained in $\Omega^*(\mathcal{T})$. If there is a leaf in A^* such that $|\pi_u|$ is greater than or equal to two then we return π_a for some $a \in A^*$ such that the size of $|\pi_a|$ is greater than one and $|\pi_a| \leq |\pi_u|$ for all $u \in A^*$ with $|\pi_u| \geq 2$.

If this condition is not met then we consider the set B^*, the set of all parents of leaves contained in $\Omega^*(\mathcal{T})$. For every $b \in B^*$, let π_b^* be the union of π_b and π_u over all descendants u of b in $\Omega^*(\mathcal{T})$. Finally, we return the set π_b^* for some $b \in B^*$ such that $|\pi_b^*| \leq |\pi_u^*|$ for all $u \in B^*$ with $\pi_u^* \geq 2$.

Lemma 4.4.1. *Given a dense set of binary level-1 trinets $\mathcal{T}$ on X with $|X| \geq 3$, algorithm FindSmallSNSet$(X, \mathcal{T})$ outputs a subset of X of size at least two in $O(|X|^3)$ time. In addition, if $\mathcal{T} = \mathcal{T}(N)$ holds for a binary level-1 network N, then FindSmallSNSet$(X, \mathcal{T})$ returns a subset of X that is a small SN-set of $\mathcal{T}$.*

Proof. Let $n = |X|$. Note first by line 15 and line 19 the algorithm will output a subset of X with size at least two.

Next, we show that the running time of the algorithm is $O(n^3)$. Since $\mathcal{T}$ contains precisely $n(n-1)(n-2)/6$ trinets, the first for loop (lines 2 - 9) in the algorithm has run-time $O(n^3)$. Based on the values of φ, the digraph Ω can be constructed in time $O(n^2)$. In addition, computing the condensed digraph Ω^* of Ω from the digraph Ω_i has run-time $O(n^2)$ using the well-known Tarjan's

Algorithm 3 FindSmallSNSet($X, \mathfrak{T}$)

INPUT: A dense set of binary level-1 trinets $\mathfrak{T}$ on a leaf set X with $|X| \geq 3$.
OUTPUT: A subset of X containing at least two elements (that is a small SN-set of $\mathfrak{T}$ when $\mathfrak{T} = \mathfrak{T}(N)$ holds for a binary level-1 network N on X).

```
1: let $\varphi : X \times X \to \mathbb{Z}_{\geq 0}$ be defined as $\varphi((x, y)) = 0$ for all $(x, y) \in X \times X$
2: for all $T \in \mathfrak{T}$ do
3:    if $T = T_1(x, y; z)$ or $T = N_i(x, y; z)$ for $1 \leq i \leq 2$ or $T = N_j(x; y; z)$ for
      $3 \leq j \leq 5$, then
4:       $\varphi(a, b) = \varphi(a, b) + 1$ for all $a \in \{x, y, z\}$ and $b \in \{x, y\}$
5:    end if
6:    if $N = S_1(x, y; z)$ or $T = S_2(x; y; z)$ then
7:       $\varphi(a, b) = \varphi(a, b) + 1$ for all $a \in \{x, y, z\}$ and $b \in \{x, y, z\}$
8:    end if
9: end for
10: find the smallest $i$ such that $\varphi(x, y) > |X| - 2 - i$ holds for some $x, y$ in $X$
11: construct the digraph $\Omega = \Omega_i(\mathfrak{T})$ on $X$ in which $(x, y)$ is an arc if and only if
    $\varphi(x, y) > |X| - 2 - i$
12: construct the condensed digraph $\Omega^*(\mathfrak{T})$ of $\Omega$; for each vertex $u$ in $\Omega^*(\mathfrak{T})$, let
    $\pi_u$ be the set of vertices in $\Omega$ contained in the strongly connected component
    corresponding to $u$
13: let $A^*$ be the set of leaves in $\Omega^*(\mathfrak{T})$
14: if $|\pi_u| \geq 2$ holds for some $u \in A^*$ then
15:    return $\pi_a$ for some $a \in A^*$ such that $|\pi_a| > 1$ and $|\pi_a| \leq |\pi_u|$ for all
       $u \in A^*$ with $|\pi_u| \geq 2$
16: end if
17: let $B^*$ be the set consisting of all parents of leaves in $\Omega^*(\mathfrak{T})$
18: for each $b \in B^*$, let $\pi^*_b$ be the union of $\pi_b$ and $\pi_u$ over all descendant $u$ of $b$
    in $\Omega^*(\mathfrak{T})$
19: return $\pi^*_b$ for some $b \in B^*$ such that $|\pi^*_b| \leq |\pi^*_u|$ for all $u \in B^*$ with $|\pi^*_u| \geq 2$
```

algorithm Tarjan [1972]. Moreover vertices in Ω_i^* with out-degree 0 can be checked in time $O(n)$. Therefore, sets A^* and B^* can be constructed in time $O(n)$. Because the set π_b^* for each $b \in B^*$ can be constructed in $O(n^2)$ by a breadth-first search, we conclude that the running time of the algorithm is $O(n^3)$.

Finally, we shall establish the correctness of the algorithm. By lines 13 - 14, the output $\pi \subseteq X$ is a strongly connected component of $\Omega(\mathcal{T})$ containing at least two elements. Therefore it remains to prove the last statement of the lemma, that is, if $\mathcal{T} = \mathcal{T}(N)$ for a binary level-1 network N on X, then the output of the algorithm is a small SN-set of $\mathcal{T}$. We have the following two cases to consider.

Case i: The network N contains no cut-arcs. Then N is necessarily a cactus as $|X| \geq 3$ and there exists only one small SN-set of $\mathcal{T}$, that is, the set X itself. On the other hand, let v be the reticulate vertex of $\mathcal{T}$ and $z \in X$ be the only child of v in $\mathcal{T}$. Then for each element $x \in X - \{z\}$, both arcs (x, z) and (z, x) are contained in Ω. Hence, $\Omega = \Omega_1$ and it contains only one strongly connected component, i.e., X itself. Therefore, the algorithm terminates at line 15 and outputs X, as required.

Case ii: The network contains some cut-arcs. Then let π be a strongly connected component in Ω such that it has out-degree 0 in Ω^*.

We shall first show that π must form a sink set in $\Omega(\mathcal{T})$. Indeed, if π is not a sink set, then there exists $x \in \pi$ and $y \in X - \pi$ such that (x, y) is an arc in $\Omega(\mathcal{T})$, and hence the out degree of π in the condensed digraph of $\Omega(\mathcal{T})$ is at least one, a contradiction.

Next, since each sink set of a digraph must be the union of the vertex sets of a set of strongly connected components, we know π must be a minimal sink set.

Finally, since N contains cut-arcs and all elements below a cut-arc in N must be a sink set in Ω, we know that π does not contain all elements of X. Hence by Lemma 4.3.4 it follows that $1 < |\pi| < n$ and π is a CA-set in N containing at least two taxa. By Theorem 4.3.1, it follows that π is a small SN-set of $\mathcal{T}$. This implies that A^* is not empty and that $|\pi_a| \geq 2$ holds for some $a \in A^*$. Therefore, the algorithm terminates at line 15 and the output is a small SN-set of $\mathcal{T}$, as required. □

4.4.2 Constructing binets

Our next step is to construct cherries, reticulate-cherries and cacti, on a subset Y. In this step we will associate a network to Y given as output from Algorithm 3. This step consists of Algorithm 4 and 5, according to whether the size of the subset is two or more. We first deal with the case when the size is two. See Section 4.4.3 for how to deal with subsets of size three or more.

If the set Y contains precisely two elements then the network N_Y associated to Y will be a binet which is obtained using the Binet$(\mathfrak{T}, Y)$ algorithm (see Algorithm 4 for the pseudocode). For example, given $Y = \{x, y\}$, there are three possible binets on Y; namely $T_0(x, y)$, $S_0(x; y)$ and $S_0(y; x)$. One of these three binets is selected and returned as the binet on Y by using a majority rule using the number of trinets in $\mathfrak{T}$ displaying N_Y. We now show that Binet$(\mathfrak{T}, Y)$ is correct and has running time $O(|X|)$.

Lemma 4.4.2. *Given a dense set $\mathfrak{T}$ of binary level-1 trinets on X with $|X| \geq 3$ and a subset $Y \subset X$ with size two, Binet($\mathfrak{T}, Y$) outputs a binet on Y in time $O(|X|)$. In addition, if $\mathfrak{T} = \mathfrak{T}(N)$ holds for a level-1 network N, then Binet($\mathfrak{T}, Y$) outputs $N|_Y$.*

Proof. Noting that the for loop (lines 2 - 7) in the algorithm will terminate in time $O(|X|)$ and N_i $(1 \leq i \leq 3)$ constructed in the algorithm is a binet on Y, we can conclude that the algorithm will output a binet on Y in time $O(|X|)$.

If $\mathfrak{T} = \mathfrak{T}(N)$ holds for a level-1 network N, then there exists $k \in \{1, 2, 3\}$ so that $N|_Y = N_k$ holds. Thus after the for loop, we have $t_i = |X| - 2$ if $i = k$, and $t_i = 0$ for $i \in \{1, 2, 3\} - \{k\}$. In other words, the output of Binet$(\mathfrak{T}, Y)$ is N_k, as required. □

4.4.3 Constructing a cactus

Having presented the algorithm to deal with the blocks with two leaves (cherries and reticulate cherries), we next deal with the case where the subset Y obtained from Algorithm 3 contains more than two leaves. Here we present the subroutine to construct a cactus from a dense trinet set (see Algorithm 5 for the pseudocode).

Algorithm 4 Binet($\mathcal{T}, Y$)

INPUT: A dense set of binary level-1 trinets $\mathcal{T}$ on X and a subset $Y = \{x, y\} \subsetneq X$.
OUTPUT: A binet on Y (which equals $N|_Y$ if $\mathcal{T}$ is induced by a binary level-1 network N).

```
1: let $t_i = 0$ for $1 \leq i \leq 3$
2: let $N_1 = T_0(x, y)$, $N_2 = S_0(x; y)$, and $N_3 = S_0(y; x)$
3: for all $z \in X - \{x, y\}$ do
4:     find the trinet $T \in \mathcal{T}$ with leaf set $\{x, y, z\}$
5:     find the index $i \in \{1, 2, 3\}$ so that $T|_Y$ is isomorphic to $N_i$
6:     $t_i = t_i + 1$
7: end for
8: find the smallest index $j \in \{1, 2, 3\}$ with $t_j \geq \max\{t_1, t_2, t_3\}$
9: return the network $N_j$
```

In more detail, we obtain a cactus $H = (A : B : g)$ from the subset Y in the following way. Firstly, we identify one leaf $g \in Y$ as the bottom leaf of H selected by a majority rule using the S_1 and S_2 trinets in $\mathcal{T}$ (lines 2 - 7). Next we construct a digraph D, with the vertex set equal to $Y - g$ and with the arc set constructed using trinets of type S_2 in $\mathcal{T}$. Let A and B be two empty lists. From the digraph D, we select and place into A the vertex with minimum in-degree and maximum out-degree over all vertices in $V(D)$. The other vertices are then sorted into the two lists and removed from Y until Y is empty lines (12 - 22). The cactus $H = (A : B : g)$ is then returned.

We now show that CactusFitting($\mathcal{T}$) is correct and runs in polynomial time.

Lemma 4.4.3. *Given a dense set $\mathcal{T}$ of binary level-1 trinets on X with $|X| \geq 3$, CactusFitting($\mathcal{T}$) outputs a cactus in $O(|X|^3)$ time. In addition, if $\mathcal{T} = \mathcal{T}(H)$ holds for a cactus H, then CactusFitting($\mathcal{T}$) returns H.*

Proof. Since the for loop (lines 2 - 6) in the algorithm will terminate in $O(|X|^3)$ iterations, and the while loop (lines 11 - 23) will terminate in $O(|X|^2)$ time, the algorithm will output a cactus with run-time $O(|X|^3)$.

Now assume that $\mathcal{T} = \mathcal{T}(H)$ holds for a cactus H with support X. Denote the bottom leaf of H by z', and let A' and B' be the two lists so that $H = (A' : B' : z')$.

Let p and q denote the number of elements contained in A' and B', respectively. Then we have $p+q = |X|-1$. In addition, list the elements in A' as $(a_1, a_2, \dots, a_p)$, and those in B' as $(b_1, b_2, \dots, b_q)$.

For each trinet $T \in \mathcal{T}$, we have $T = S_1(x, y; z)$ for some $x, y, z \in X$ if and only if $z = z'$, and either $x \in A', y \in B'$ or $x \in B', y \in A'$. On the other hand, $T = S_2(x; y; z)$ or $T = S_2(y; x; z)$ holds precisely when $z = z'$, and $x, y \in A'$ or $x, y \in B'$. Therefore, for the function φ defined in the algorithm, we have $\varphi(x) = 0$ for all $x \neq z'$ and

$$\varphi(z') = \binom{p}{2} + \binom{q}{2} + pq > 0.$$

It follows that $\varphi(z') \geq \varphi(x)$ for all $x \in X$, and hence the element g constructed in line 7 is z'. In addition, the digraph D constructed in line 8 has vertex set $X - \{z'\}$ and arc set

$$\{(a_i, a_j) \,:\, 1 \leq i < j \leq p\} \cup \{(b_i, b_j) \,:\, 1 \leq i < j \leq q\}.$$

This implies that the vertex in D that has the maximum out-degree is either a_1 (with out-degree p) or b_1 (with out-degree q) in D. Thus, for the two lists A and B constructed in the algorithm, we have $\{A, B\} = \{A', B'\}$. It follows that algorithm outputs a cactus $(A : B : g)$ or $(B : A : g)$. Since in both cases the output is H, this completes the proof of the lemma. $\square$

4.4.4 Main TriLoNet algorithm

With the subroutines described in Algorithms 3, 4 and 5, we now present in Algorithm 6 the main TriLoNet algorithm. It works as follows.

If the subset Y obtained from Algorithm 3 is equal to X itself, the algorithm stops and outputs the network N_Y (lines 7 - 8) . Otherwise, this step continues recursively as follows: (i) we compute the trinet set $\mathcal{T}^*$ induced by $\mathcal{T}$ on the set X^* formed by replacing the subset Y in X with a new taxon y^* (line 10), (ii) we obtain a level-1 network N^* for T^* recursively (line 13), and (iii) we combine the two networks N_y and N^* to form a level-1 network on X by replacing the taxon

Algorithm 5 CactusFitting($\mathcal{T}, X$)

INPUT: A dense set of binary level-1 trinets $\mathcal{T}$ on a leaf set X with $|X| \geq 3$.
OUTPUT: A cactus with support X (which equals to H if $\mathcal{T} = \mathcal{T}(H)$ holds for a cactus H on X).

```
1: let $\varphi(x) = 0$ for all $x \in X$
2: for all $T \in \mathcal{T}$ do
3:    if $T = S_1(x, y; z)$ or $T = S_2(x; y; z)$ for some $x, y, z \in X$ then
4:       $\varphi(z) = \varphi(z) + 1$
5:    end if
6: end for
7: find one taxon $g \in X$ with $\varphi(g) \geq \varphi(x)$ for all $x \in X$
8: construct the digraph $D$ on $X - \{g\}$ in which $(x, y)$ is an arc if $\mathcal{T}|_{\{g,x,y\}} =$
   $\{S_2(x; y; g)\}$
9: let $A$ and $B$ be two empty lists
10: choose a vertex $a_1$ in $D$ that has the maximum out-degree, and let $A = (a_1)$
    and put $Y = X - \{g, a_1\}$
11: while $Y \neq \emptyset$ do
12:    choose a vertex $x$ in $Y$ that has the maximum out-degree over all vertices
       in $Y$
13:    let $t_a$ (resp. $t_b$) be the number of vertices $u$ in $A$ (resp. $u$ in $B$) such that
       $|\text{CH}(u) \cap \text{CH}(x)| \geq 0.5|\text{CH}(x)|$ holds in $D$
14:    let $t'_a = t_a + |B| - |t_b|$ and $t'_b = t_b + |A| - |t_a|$
15:    if $|B| > 0$ then
16:       let $t'_a = t'_a/|A|$ and $t'_b = t'_b/|B|$
17:    end if
18:    if $t'_a \geq t'_b$ then
19:       put $A = (A, x)$
20:       else put $B = (B, x)$
21:    end if
22:    put $Y = Y - \{x\}$
23: end while
24: return the cactus $(A : B : g)$
```

y^* in N^* with N_Y (line 15).

We now show that TriLoNet is consistent, meaning that the algorithm will, given a dense set of trinets obtained from a rooted level-1 phylogenetic network N as input, output N.

Theorem 4.4.4. *Given a dense set $\mathfrak{T}$ of binary level-1 trinets on X with $|X| \geq 3$, TriLoNet($\mathfrak{T}$) outputs a level-1 network on X in time $O(|X|^4)$. In addition, if $\mathfrak{T} = \mathfrak{T}(N)$ holds for a level-1 network N, then TriLoNet($\mathfrak{T}$) outputs N.*

Proof. Let $n = |X|$. First, assuming that the worse case running time of this algorithm is $f(n)$, we shall show that $f(n) = O(n^4)$. For simplicity, let m denote the size of the subset Y constructed in line 2. Then we have $2 \leq m \leq n$. By Lemma 4.4.1, line 2 has run time $O(n^3)$. By Lemma 4.4.3 and Lemma 4.4.2, the network N_1 in the algorithm is constructed in $O(m^3)$. If $m = n$, then the algorithm terminates by line 8 and the run time is $O(n^3)$. Therefore we may assume that $m < n$. This implies the set X^* constructed in line 10 has size $n - m + 1$. If $m = n - 1$, then line 12 is executed and we have $f(n) = O(n^3)$ by Lemma 4.4.2. Otherwise, we have $f(n) = f(n - m + 1) + O(n^3)$, and solving the recursion on $f(n)$ shows that $f(n) = O(n^4)$, as required.

We now establish the claim that TriLoNet outputs a level-1 network on X which is N if $\mathfrak{T} = \mathfrak{T}(N)$ holds for a level-1 network N. We use induction on n. The base case $n = 3$ is trivial as in this case $\mathfrak{T}$ contains exactly one trinet T, which is returned by line 2. Now assume that there exists $n_0 > 3$ such that the claim holds for all n with $3 \leq n < n_0$, and we shall establish the induction step by showing that it also holds for $n = n_0$.

We begin with showing that TriLoNet outputs a level-1 network on X. By the induction assumption, Lemma 4.4.3 and Lemma 4.4.2, we know that both N_1 and N_2 are level-1 networks, and hence we can conclude that the output of the algorithm must be a level-1 network on X.

Finally, we need to show that the algorithm returns N. To begin with, note that the subset Y of X obtained in line 2 is a small SN-set of $\mathfrak{T}$ by Lemma 4.4.1, and hence also a minimal CA-set in N in view of Theorem 4.3.1. Therefore, it follows that $N|_Y$ must be a cherry, reticulate-cherry or cactus. Together with Lemma 4.4.3 and Lemma 4.4.2, this implies that $N|_Y = N_1$ holds for the network

N_1 constructed in the first if loop (lines 3 - 6). If $X = Y$, then the algorithm terminates by line 8 and returns $N_1 = N$, as required. Therefore we may assume $X \neq Y$.

Now let y^* be the element in Y that is chosen by the algorithm in line 10. Note that different choices of y^* lead to the same output of the algorithm. Using Lemma 4.4.2 and the induction assumption, we know $N_2 = N|_{X^*}$. Since replacing the leaf y^* in $N|_{X^*}$ with $N|_Y$ results in the network N, we can conclude that the output of the algorithm is N, as required. □

Algorithm 6 The main algorithm: TriLoNet($\mathcal{T}$)

INPUT: A dense set of binary level-1 trinets $\mathcal{T}$ on X with $|X| \geq 3$.
OUTPUT: A binary level-1 network on X (which is N if $\mathcal{T} = \mathcal{T}(N)$ holds for a binary level-1 network N).

```
1: if $|X| = 3$, return the unique network in $\mathcal{T}$
2: identify a subset $Y \subseteq X$ with $|Y| \geq 2$ by calling FindSmallSNSet($X, \mathcal{T}$)
3: if $|Y| = 2$ then
4:     construct a network $N_1$ on $Y$ by calling Binet($\mathcal{T}, Y$)
5:     else construct a level-1 network $N_1$ on $Y$ by calling CactusFitting($\mathcal{T}|_Y, Y$)
6: end if
7: if $X = Y$ then
8:     return $N_1$
9: end if
10: choose an element $y^*$ in $Y$, and let $X^* = (X - Y) \cup \{y^*\}$
11: if $|X^*| = 2$ then
12:     construct a network $N_2$ on $X^*$ by calling Binet($\mathcal{T}, X^*$)
13:     else recursively call TriLoNet($\mathcal{T}|_{X^*}$) to obtain a level-1 network $N_2$ on $X^*$
14: end if
15: return the level-1 network obtained from $N_2$ by replacing the leaf $y^*$ with
    the network $N_1$
```

4.5 Concluding remarks

We have presented some theoretical results on the equivalence of CA-sets in networks, SN-sets in collections of trinets and minimal sink sets in digraphs. We have presented the TriLoNet algorithm and shown that it is consistent, meaning that, given the dense set of trinets of a level-1 network N as input, it will always output N. We have also given pseudocode for TriLoNet. In the next chapter we evaluate the performance of the TriLoNet algorithm, on both artificial and real biological data sets.

Chapter 5

Simulations and real data sets

5.1 Chapter summary

In this chapter we present sections detailing various simulations that we carried out to test SeqTrinet and TriLoNet. In Section 5.2.1 we outline a new approach to extracting trinets from phylogenetic networks, a step required for the comparison of two networks that we use in Section 5.2. In the Section 5.2 we present and discuss a noise simulation experiment which includes a comparison of TriLoNet with Lev1athan [Huber et al., 2011b], a triplet based network reconstruction approach mentioned in Chapter 2. In Section 5.3, we then present some results following a similar methodology used in [Holland et al., 2002] to test the SeqTrinet algorithm presented in Chapter 3. Finally, in Section 5.4, to test the usability of our network approach we apply TriLoNet to some real biological sequence data sets.

5.2 Noise simulation experiments

The experiments and methodology presented in this section follow a similar approach to those presented in [Huber et al., 2011b]. The aim of the experiments in this section is to test the robustness and reconstructive power of TriLoNet when the input data had been subjected to varying amounts of noise. We also compare our results to Lev1athan. We used the same random level-1 network generator that was used in [Huber et al., 2011b] Lev1athan.

5.2.1 Extracting trinets from phylogenetic networks

In this section, we consider two different approaches to extracting the set of trinets from a level-1 phylogenetic network. We start by outlining an intuitive approach in Section 5.2.1.1; see [Huber and Moulton, 2013] for a more thorough description. We then present an alternative and more efficient approach that we found to work better in practice in Section 5.2.1.2. This is a key step needed to compute the metric based on trinets that we introduce in Section 5.2.3 where we extend a measure described in [Huber et al., 2011b] that used triplets to compare two networks.

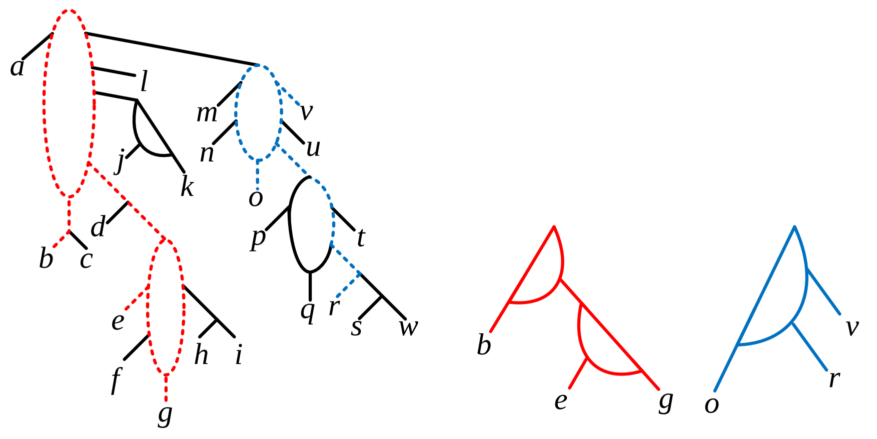
5.2.1.1 Initial trinet extraction approach

In our preliminary experiments we implemented and used the definition first described by [Huber and Moulton, 2013] to identify and extract the trinets induced by a level-1 phylogenetic network. Given a binary level-1 network N on a leaf set X as input, the output of the algorithm is a dense collection of trinets displayed by N. The general outline of the trinet extraction algorithm is as follows.

- For every subset of three leaves in X, identify their lowest stable ancestor (LSA);
- From this vertex, we then highlight all paths to the three leaves and combine these paths into a single digraph;
- Repeatedly remove vertices from this digraph with in-degree 0 out-degree 1, vertices with in-degree 1 out-degree 1 and any parallel arcs until none remain

In more detail, the first step is to for every subset of three $Y = \{x, y, z\}$ leaves in X, identify the vertex $\text{LSA}(Y)$. From this vertex, we then highlight all paths to the leaves $\{x, y, z\}$ and combine these paths into a single digraph. We use a Depth First Search approach to identify the paths from $\text{LSA}(Y)$ to each of the three leaf vertices. The next step is to repeatedly remove vertices from the digraph with in-degree 0 out-degree 1, vertices with in-degree 1 out-degree 1 and any parallel arcs until none remain. The resulting digraph is isomorphic to one of the eight binary level-1 trinets.

Figure 5.1 illustrates the extraction process for two example trinets. Suppose N is a phylogenetic network on the leaf set $X = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w\}$. Figure 5.1(a) highlights the paths in N used in the extraction of two example trinets shown in Figure 5.1(b), $N_4(g; e; b)$ and $S_2(o; v; r)$.



(a) (b)

Figure 5.1: (a) A level-1 phylogenetic network N on the set $X = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w\}$. (b) The two example trinets $N_4(g; e; b)$ and $S_2(o; v; r)$ extracted from N.

We implemented this algorithm and found it to be too slow for practical use with networks containing above 70 taxa. We thus developed a more complex but also more efficient algorithm called TriExtract to obtain the dense collection of trinets from a level-1 phylogenetic network, which we now describe.

5.2.1.2 TriExtract algorithm

The TriExtract algorithm takes as input a rooted binary level-1 phylogenetic network N with leaf set X and returns the dense collection of trinets on X displayed by N. The high level idea of the algorithm is to first find all the binets $\mathcal{B}(N)$ displayed by N and then use this information to obtain the dense collection of trinets $\mathcal{T}(N)$. We now describe in more detail the key steps of the algorithm and then present it as pseudocode.

Step 1

We begin by performing a topological sort on the vertices in the network N [Kahn, 1962]. For every arc $\{u, v\}$ from vertex u to vertex v, u comes before v in the ordering. This gives us $v_1, ..., v_m$ where v_1 is the root and the descendants of v_i have subscripts greater than i. This is a necessary step for the identification of the Lowest Stable Ancestor (LSA) for every binet. Once we have an ordering of the vertices in the network, we next identify for every vertex $v \in V(N)$ the cluster $\mathcal{C}(v)$ by using the Depth First Search algorithm. Recall from Chapter 2 that the cluster $\mathcal{C}(v)$ is the set of taxa in N reachable on some path from v.

Step 2

The next step is to obtain the collection of binets $\mathcal{B}(N)$ from the network N. For each binet we need to determine the binet type, the LSA and the binet ordering (for type S_0 binets). Here we construct the LSA table, a data structure that maps each distinct pair of leaves x, y to its lowest stable ancestor in N.

Recall that there are two types of binet; T_0 and S_0, which we refer to as a cherry and reticulate-cherry respectively. For example, the binets $T_0(x, y)$ and $S_0(x; y)$ are shown in Figure 3.4. Note the ordering of the leaves in type S_0 binets. In this step we consider only interior tree vertices; leaf vertices and reticulation vertices are ignored.

For every interior tree vertex v in N, we construct sets A and B in X. Given that we only consider vertices with out-degree 2, for a vertex v with children w_l and w_r we let $A = \mathcal{C}(w_l)$ and $B = \mathcal{C}(w_r)$.

We also consider the set differences $A \setminus B$ and $B \setminus A$. The pair of elements in the sets $(A \setminus B) \times A$ and $(B \setminus A) \times B$ correspond to the binets in $\mathcal{B}$ for which v is the LSA. The LSA for a given binet is recorded once and is not overwritten.

For two leaves $\{x, y\}$, if x or y is contained in $A \cap B$ then the corresponding binet will be a reticulate-cherry, otherwise the binet on $\{x, y\}$ will be $T_0(x, y)$. If x is contained in $A \cap B$ then the corresponding binet is $S_0(x; y)$, otherwise it is

$S_0(y; x)$.

Step 3

Using the collection of binets $\mathcal{B}$ in N and the LSA table obtained from Step 2, we compute the collection of trinets $\mathcal{T}(N)$. For each subset of three leaves $\{x, y, z\}$ in X we consider the binets on $\{x, y\}$, $\{x, z\}$ and $\{y, z\}$. We then consider the following sub-steps depending on the types of the three binets:

- If the three binets on $\{x, y\}$, $\{x, z\}$ and $\{y, z\}$ are all cherries then the trinet on $\{x, y, z\}$ will be of type T_1, see Figure 2.6 for an illustration of the different types of trinets. The binet with a different LSA to the other two is isomorphic to the cherry in the trinet. For example, if $\text{LSA}(x, z) = \text{LSA}(y, z)$ then the corresponding trinet will be $T_1(x, y; z)$.

- If two of the binets are of type T_0 then the corresponding trinet on $\{x, y, z\}$ will be of type N_3. The type S_0 binet is isomorphic with the reticulate-cherry in the corresponding trinet. For example, given the binets $S_0(x; y)$, $T_0(x, z)$ and $T_0(y, z)$ the corresponding trinet is $N_3(x; y; z)$.

- If none of the three binets are of type T_0 then the trinet on $\{x, y, z\}$ is either of type N_4 or N_5. Consider the two S_0 binets with the same leaf under the reticulation vertex. If the LSA for both of these binets is identical then the corresponding trinet is of type N_4, otherwise it is of type N_5.

  - If the trinet is of type N_4 then the binet with a different LSA is isomorphic with the reticulate-cherry contained in the trinet. For example, given the binets $S_0(x; y)$, $S_0(z; x)$ and $S_0(z; y)$ with $\text{LSA}(x, y) = u$, $\text{LSA}(x, z) = v$ and $\text{LSA}(y, z) = v$, the corresponding trinet is $N_4(x; y; z)$.
  - If the trinet is of type N_5 then the binet with a different LSA from the other two is isomorphic with the reticulate-cherry contained in the trinet. For example, given the binets $S_0(x; y)$, $S_0(x; z)$ and $S_0(y; z)$ with $\text{LSA}(x, y) = u$, $\text{LSA}(x, z) = v$ and $\text{LSA}(y, z) = v$, the corresponding trinet is $N_5(x; y; z)$.

- If one of the binets is of type T_0 then the trinet on $\{x, y, z\}$ will be of type N_2, S_1, N_1 or S_2. Consider the two S_0 binets. If the leaf below the reticulation in these binets is not identical then the corresponding trinet will be of type N_2 and the cherry in this trinet is isomorphic to the T_0 binet. Given the binets $T_0(x, y)$, $S_0(x; z)$ and $S_0(y; z)$, the corresponding trinet is $N_3(x, y; z)$.

  - If the LSA for each of the three binets is identical then the corresponding trinet will be of type S_1. The leaf below the reticulation vertices in the type S_0 binets is set as the leaf below the reticulation vertex in the S_1 trinet. For example. given the binets $T_0(x, y)$, $S_0(x; z)$ and $S_0(y; z)$, the corresponding trinet is $S_1(x, y; z)$.
  - To distinguish between a type S_2 and N_1 trinet we have to consider $\mathcal{C}(\text{LSA}(x, y))$, with x and y being the two leaves from the type T_0 binet. Given three binets $T_0(x, y)$, $S_0(x; z)$ and $S_0(y; z)$, if $z \in \mathcal{C}(\text{LSA}(x, y))$ then the corresponding trinet is of type S_2, otherwise it is of type N_1.
  - If the trinet is of type S_2, the leaf z below the reticulation vertex is equal to the leaf below the reticulation vertices in the two S_0 binets. The child of the LSA of the type T_0 binet in N that is a leaf vertex is set as x for the corresponding trinet $S_2(x; y; z)$.
  - If the trinet is of type N_1, the type T_0 binet is isomorphic to the cherry in the trinet. Given the binets $T_0(x, y)$, $S_0(x; z)$ and $S_0(y; z)$, the corresponding trinet is $N_1(x, y; z)$.

Pseudocode

In summary, we present the pseudocode for the TriExtract algorithm in Algorithm 7.

Algorithm 7 TriExtract(N)

INPUT: A binary rooted level-1 phylogenetic network N on the set of taxa X.
OUTPUT: The dense collection of trinets $\mathcal{T}(N)$ on X.

```
1: Topologically sort vertices of $N$
2: for all $v \in V(N)$ do
3:     Compute $\mathcal{C}(v)$
4:     With $w_l$ and $w_r$ as children of $v$, compute $C_l = \mathcal{C}(w_l)$ and $C_r = \mathcal{C}(w_r)$
5: end for
6: Let $\mathcal{B} = \emptyset$ and $\mathcal{T} = \emptyset$
7: Fill LSA table and compute binet type for every binet on $\{x, y\} \in \mathcal{B}$ (see Step
   2)
8: for all subsets $\{x, y, z\} \in \binom{X}{3}$ do
9:     Compute trinet $T$ on $\{x, y, z\}$ using binets on $\{x, y\}, \{x, z\}$ and $\{y, z\} \in \mathcal{B}$
       and add $T$ to $\mathcal{T}$ (see Step 3)
10: end for
11: Return $\mathcal{T}$
```

5.2.2 Comparing TriLoNet to Lev1athan

We generate random level-1 phylogenetic networks by using the Lev1Generator program [Huber et al., 2011a]. The random level-1 network generator is guaranteed to output a level-1 phylogenetic network M. From this network we extract the set of triplets $Tr(M)$ and trinets $\mathcal{T}(M)$ displayed by M for use as input to Lev1athan and TriLoNet, respectively.

We use a parameter ϵ to control the amount of noise in the input data. To introduce noise to a set of trinets, we uniformly at random select a specified percentage of trinets and change their types as well as the leaf orderings. Our noise generator algorithm takes a set of trinets $\mathcal{T}(M)$ and ϵ as input parameters and outputs $\mathcal{T}_\epsilon(M)$ with ϵ% of trinets changed. Using our approach to extract a set of triplets from a set of trinets we also compute the set of triplets $Tr_\epsilon(M)$ contained in the trinets in $\mathcal{T}_\epsilon(M)$ to use as input to Lev1athan. Note that changing the trinet type does not necessarily change the triplets contained inside that trinet.

In total 7200 phylogenetic networks were randomly generated and used in the noise simulation experiments. All of the randomly generated networks used in our simulation had no vertices of out-degree 3 or higher. We obtain a collection $\mathcal{M}$ of random level-1 networks that contains 100 networks with leaf sizes in the range $1 + (10 \times j)$ to $10 \times (j + 1)$ for each $2 \leq j \leq 9$.

Our network construction algorithm TriLoNet constructs a network $\mathcal{N}_1$ and Lev1athan constructs a network $\mathcal{N}_2$. The set of trinets $\mathcal{T}(\mathcal{N}_1)$ is extracted from $\mathcal{N}_1$, from which $Tr(\mathcal{N}_1)$ is obtained. The set of trinets $\mathcal{T}(\mathcal{N}_2)$ is extracted from $\mathcal{N}_2$, from which $Tr(\mathcal{N}_2)$ is obtained. Here we compare the trinets and triplets displayed in the networks constructed by TriLoNet ($\mathcal{T}(\mathcal{N}_1)$ and $Tr(\mathcal{N}_1)$) and Lev1athan ($\mathcal{T}(\mathcal{N}_2)$ and $Tr(\mathcal{N}_2)$) firstly to the trinet and triplet sets from the original randomly generated networks ($\mathcal{T}(M)$ and $Tr(M)$) and secondly to the trinet and triplet sets subjected to noise ($\mathcal{T}_\epsilon(M)$ and $Tr_\epsilon(M)$).

5.2.3 Measures

The first measure we considered which was also used by [Huber et al., 2011b] was the *triplet-network triplet consistency measure* C' where

$$C'(N, M) = \frac{|Tr(M) \cap Tr(N)|}{|Tr(M)|}. \tag{5.1}$$

This measure indicates the fraction of triplets in $Tr(N)$ consistent with $Tr(M)$. We extended this measure to trinets to determine the fraction of trinets in a given trinet set that is consistent with a network, viz

$$C(N, M) = \frac{|\mathcal{T}(M) \cap \mathcal{T}(N)|}{|\mathcal{T}(M)|}. \tag{5.2}$$

The aim of the experiments is to measure the similarity between a randomly generated level-1 network M and the network $\mathcal{N}_1$ that TriLoNet outputs and the network $\mathcal{N}_2$ that Lev1athan outputs when the trinets extracted from M have been subjected to noise. Note that $C(N, M) = 1$ implies that N is equal to M, although this does not necessarily hold for the C'-score.

In the experiments we recorded four consistency measures for TriLoNet and four consistency measures for Lev1athan. For the triplet consistency score C', for TriLoNet we recorded the following (the same measures hold for Lev1athan by replacing $\mathcal{N}_1$ with $\mathcal{N}_2$.):

$$C'(N, M) = \frac{|Tr(M) \cap Tr(\mathcal{N}_1)|}{|Tr(M)|}. \tag{5.3}$$

$$C'(N, M) = \frac{|Tr_\epsilon(M) \cap Tr(\mathcal{N}_1)|}{|Tr_\epsilon(M)|}. \tag{5.4}$$

and for the trinet consistency score C we recorded

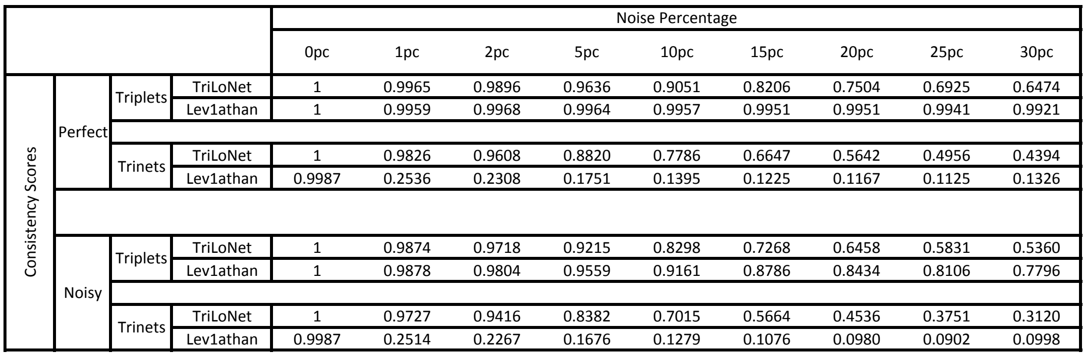
$$C(N, M) = \frac{|\mathcal{T}(M) \cap \mathcal{T}(\mathcal{N}_1)|}{|\mathcal{T}(M)|}. \tag{5.5}$$

$$C(N, M) = \frac{|\mathcal{T}_\epsilon(M) \cap \mathcal{T}(\mathcal{N}_1)|}{|\mathcal{T}_\epsilon(M)|}. \tag{5.6}$$

We also considered using as a measure the number of reticulation vertices in the original input network in comparison with the networks generated by Lev1athan and TriLoNet. This measure was useful in determining the number of reticulation vertices incorrectly created or removed by Lev1athan and TriLoNet.

5.2.4 Simulation results

In our noise simulation experiments the parameter ϵ is used to control the amount of noise in the data used as input. The values of ϵ we used in our experiments were $\epsilon = 0$, 1, 2, 5, 10, 15, 20, 25, 30. The results of the experiments are presented in Table 5.2.



				Noise Percentage								
				0pc	1pc	2pc	5pc	10pc	15pc	20pc	25pc	30pc
Consistency Scores	Perfect	Triplets	TriLoNet	1	0.9965	0.9896	0.9636	0.9051	0.8206	0.7504	0.6925	0.6474
			Lev1athan	1	0.9959	0.9968	0.9964	0.9957	0.9951	0.9951	0.9941	0.9921
		Trinets	TriLoNet	1	0.9826	0.9608	0.8820	0.7786	0.6647	0.5642	0.4956	0.4394
			Lev1athan	0.9987	0.2536	0.2308	0.1751	0.1395	0.1225	0.1167	0.1125	0.1326
	Noisy	Triplets	TriLoNet	1	0.9874	0.9718	0.9215	0.8298	0.7268	0.6458	0.5831	0.5360
			Lev1athan	1	0.9878	0.9804	0.9559	0.9161	0.8786	0.8434	0.8106	0.7796
		Trinets	TriLoNet	1	0.9727	0.9416	0.8382	0.7015	0.5664	0.4536	0.3751	0.3120
			Lev1athan	0.9987	0.2514	0.2267	0.1676	0.1279	0.1076	0.0980	0.0902	0.0998

Figure 5.2: Summary of the results from the comparison study for TriLoNet Lev1athan.

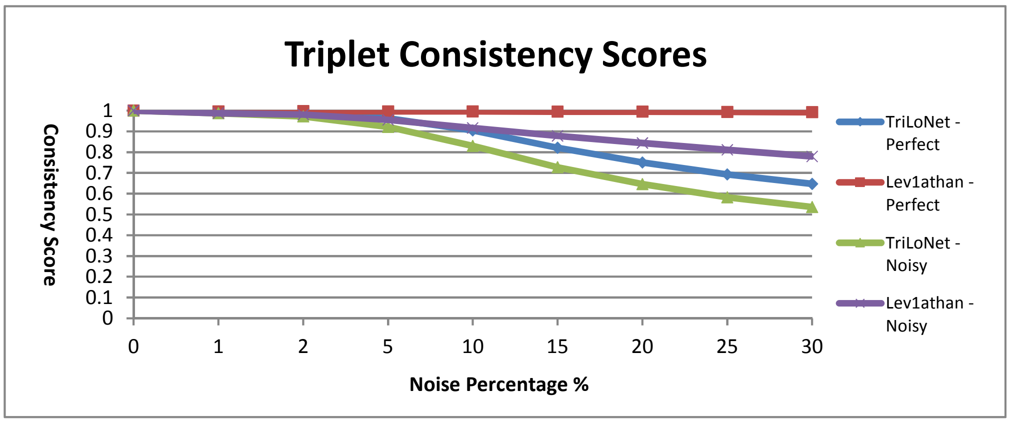


Figure 5.3: Summary of the triplet consistency results comparing TriLoNet against Lev1athan.

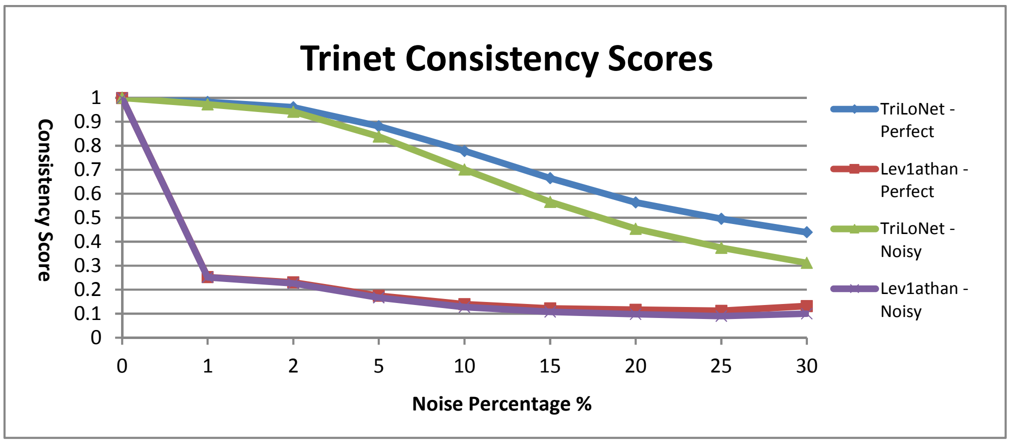


Figure 5.4: Summary of the trinet consistency results comparing TriLoNet against Lev1athan.

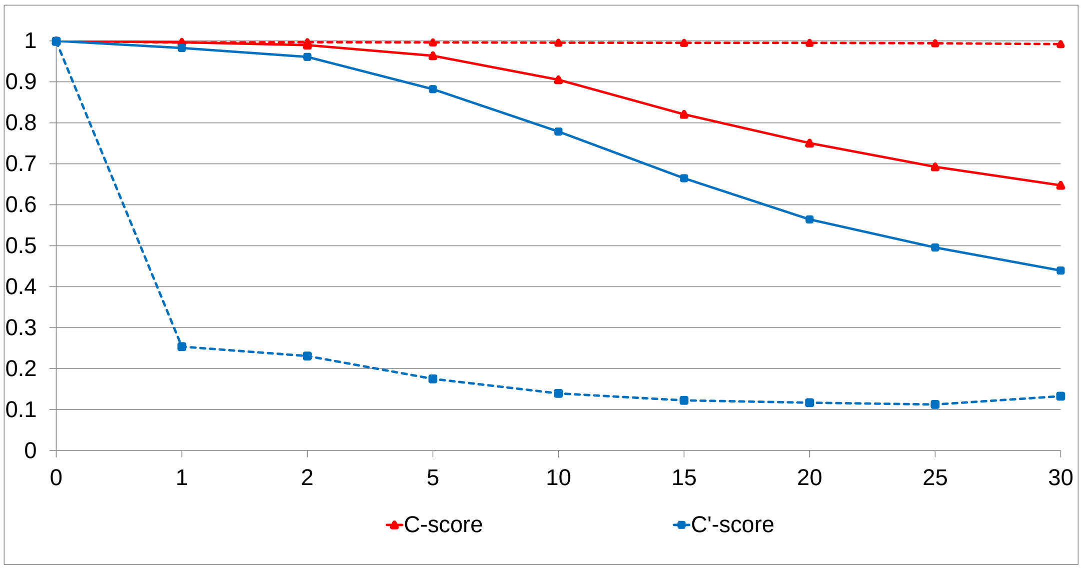


Figure 5.5: The plot of ϵ against the average triplet (C', triangle markers) and trinet (C, square markers) consistency scores . The x-axis is labelled by ϵ and the y-axis is labelled by average triplet and trinet consistency. The solid lines correspond to the average triplet and trinet consistency for the networks constructed by TriLoNet. The dotted lines correspond to the average triplet and trinet consistency for the networks constructed by Lev1athan.

We first tested both algorithms by using as input $Tr_\epsilon(M)$ and $\mathcal{T}_\epsilon(M)$, with ϵ having a value of 0, this represents perfect data. Our algorithm was always able to reconstruct back the original network when ϵ was equal to 0. Interestingly there were some instances where Lev1athan was unable to correctly reconstruct back the correct network even when ϵ was equal to 0. It was especially interesting to observe that for low levels of noise (1, 2,5), our algorithm was in some cases able to correct all of the trinets in the input $\mathcal{T}_\epsilon(M)$ that had been affected by noise so that the set of trinets $\mathcal{N}_1$ was equal to $\mathcal{T}(M)$.

As expected, there was a general decrease in the consistency scores as the value of ϵ was increased for both the trinet and triplet measures. The triplet consistency scores for Lev1athan were above 99% even up to ϵ with a value of 30. For the same measure our scores were similar at low values of ϵ (0.9965 with $\epsilon = 1$ and 0.9896 with $\epsilon = 2$) however TriLoNet's triplet consistency score decreased linearly for the other values of ϵ.

There was a considerable difference when comparing the trinet consistency scores of our algorithm and Lev1athan. With a ϵ value of 1 Lev1athan was able to correctly construct back an average of 25% of trinets contained in the original input network. For the same ϵ value our algorithm was able to correctly construct back an average of 98% of the trinets contained in the original network.

For all tested values of ϵ our algorithm outperformed Lev1athan in both trinet consistency measures. As the value of ϵ increased, the difference in performance between our algorithm and Lev1athan decreased.

5.2.5 Reticulation difference experiments

To further investigate the difference in the average triplet consistency scores between TriLoNet and Lev1athan, we computed the difference in the number of reticulations between the original input networks and the networks constructed by the two algorithms.

We first considered the phylogenetic networks constructed from input with $\epsilon = 1$. Interestingly, as the number of taxa in the networks increased, the difference in the number of reticulations between the networks constructed by TriLoNet and Lev1athan compared to the original networks also increased. This can be seen in Figures 5.6, 5.7 and 5.8.

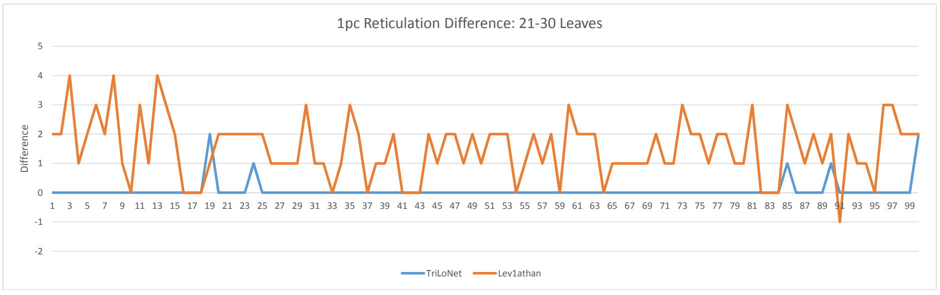


Figure 5.6: Comparing the difference in the number of reticulation vertices from the original input networks to the networks constructed by TriLoNet and Lev1athan for the 100 networks containing between 21-30 leaves.

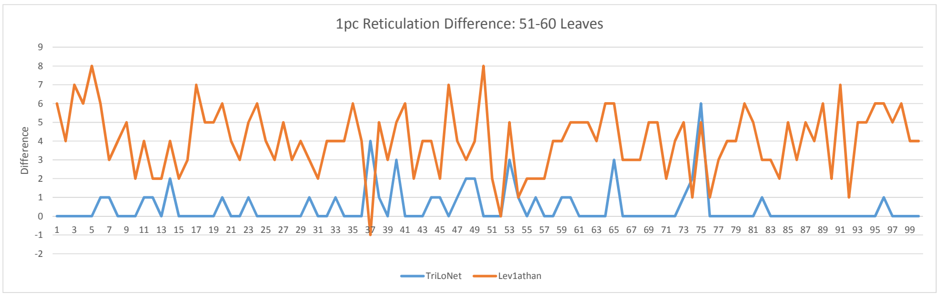


Figure 5.7: Comparing the difference in the number of reticulation vertices from the original input networks to the networks constructed by TriLoNet and Lev1athan for the 100 networks containing between 51-60 leaves.

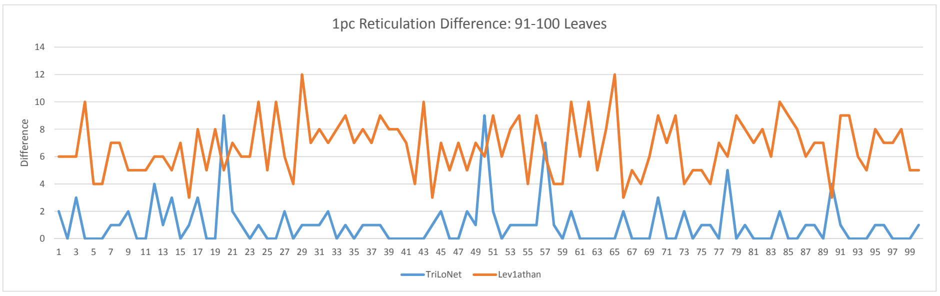


Figure 5.8: Comparing the difference in the number of reticulation vertices from the original input networks to the networks constructed by TriLoNet and Lev1athan for the 100 networks containing between 91-100 leaves.

For three values of ϵ (1,5,10), Figure 5.9 presents the difference in the median number of reticulations in the networks outputted by the two algorithms compared to the original input networks. As the value of ϵ and size of the networks increase, Lev1athan on average introduces more reticulation vertices than TriLoNet. This could explain why Lev1athan is able to correctly infer a higher percentage of triplets but not trinets.

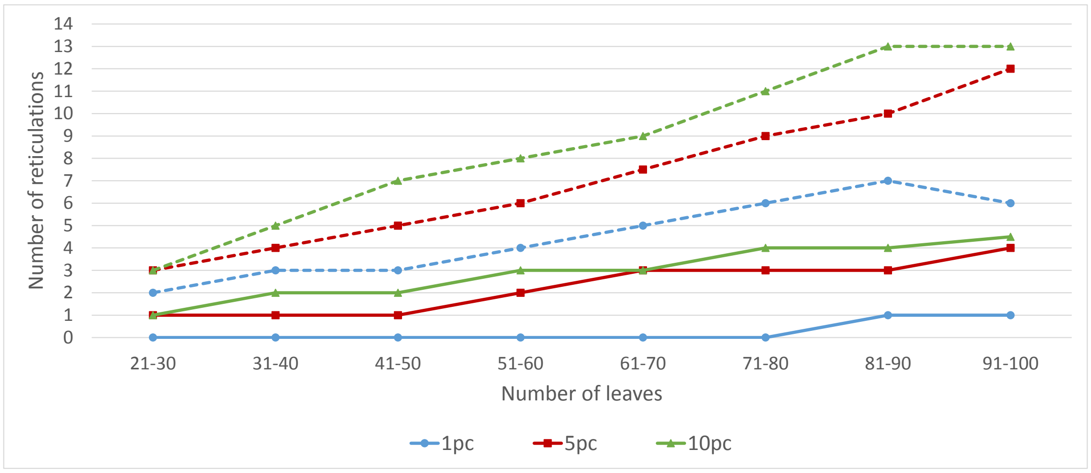


Figure 5.9: The figure displays the median difference of the number of reticulations in the phylogenetic networks constructed by TriLoNet and Lev1athan compared to the original randomly generated networks. The x-axis is labelled by the number of leaves and the y-axis is labelled by the number of reticulations.

The results of the reticulation number comparison measure are summarised in Table 5.1. This table shows that in most cases (all cases when the input has not been subjected to noise), Lev1athan constructs and outputs networks with a higher number of reticulations than TriLoNet.

ϵ	0	1	2	5	10	15	20	25	30
%	0	1.5	2.5	3.125	3.125	6.0	8.75	9.625	12.875

Table 5.1: The percentage of networks where the difference in the number of reticulations in the network constructed by TriLoNet is higher than the number of reticulations in the network constructed by Lev1athan for $\epsilon \in \{0, 1, 2, 5, 10, 15, 20, 25, 30\}$

.

5.2.6 Triplet and trinet noise difference

One motivation in subjecting a set of trinets to noise in the experiments in this section was to attempt to provide TriLoNet and Lev1athan with input that was as similar as possible. To do this, we subjected the set of trinets extracted from a randomly generated level-1 network to noise; this set of trinets was given to TriLoNet as input. The set of triplets contained in each of the trinets was given to Lev1athan. It should be noted that subjecting a set of trinets to a certain amount of noise does not necessarily cause the same amount of noise to be found in the triplets contained in the trinets. For example, consider the trinet $N_3(x; y; z)$. Changing this trinet to $N_1(y, x; z)$ does not change the triplet contained within both of these networks. With this in mind, we compared the difference in noise between the sets of trinets and triplets. These results are summarised in Table 5.2. It is interesting to note that the method we chose to obtain input to the two algorithms causes the input to Lev1athan to contain less noise in comparison with the input to TriLoNet.

	ϵ							
Number of taxa	1	2	5	10	15	20	25	30
21-30	0.67	1.31	3.27	6.49	9.84	13.12	16.40	19.57
31-40	0.65	1.31	3.28	6.53	9.78	13	16.35	19.63
41-50	0.66	1.31	3.26	6.54	9.83	13.11	16.37	19.63
51-60	0.66	1.31	3.28	6.54	9.85	13.08	16.32	19.61
61-70	0.65	1.32	3.27	6.55	9.82	13.06	16.37	19.63
71-80	0.65	1.31	3.28	6.55	9.83	13.1	16.34	19.67
81-90	0.65	1.32	3.27	6.55	9.83	13.1	16.38	19.64
91-100	0.66	1.31	3.28	6.54	9.83	13.1	16.37	19.64

Table 5.2: A table summarising the average amount of noise in the triplet sets given to Lev1athan as input compared to the amount of noise in the set of trinets used as input to TriLoNet.

5.3 Artificial data simulation experiments

To test the performance of TriLoNet on sequence data, we performed some simulation experiments. To do this we followed a similar approach to [Holland et al., 2002]. In particular, we simulated the evolution of artificial sequence data on six phylogenetic networks presented in Figure 5.10 containing recombinant taxa using Seq-Gen [Rambaut and Grass, 1997]. We then used this sequence data as input to SeqTrinet to create a dense set of trinets to take as input to TriLoNet. We varied parameters including the sequence length, the κ threshold and the left-right sequence contribution percentages.

Seq-Gen [Rambaut and Grass, 1997] when given a phylogenetic tree as input outputs a sequence alignment obtained by simulating evolution of nucleotide sequences along the tree, with the option of selecting from a number of models for the substitution process. To obtain a simulated sequence alignment containing a recombinant taxon we generated and then concatenated sequence alignments from two trees with the same leaf set but different topology. Each alignment was obtained by generating and concatenating alignments from the 2 underlying trees in each network in Figure 5.10. Initially, in the concatenated sequence alignments the first 50% of sites are from the first tree and the second 50% are from the second tree.

We used the following parameters in Seq-Gen: -mHKY, which when used in conjunction with -f0.25,0.25,0.25,0.25, to set nucleotide frequencies to use the K2P Model; -t2.0, to give a transition-transversion bias of $\kappa = 4$ and -d0.3, to scale the tree lengths. The expected number of substitutions from the root to each leaf was 0.3. The sequence length parameter was initially set to -l 25,000 to give a total sequence length of 50,000 for the alignments generated for each of the networks. A threshold parameter of $\kappa = 6.5$ was initially used for these experiments.

The six networks presented are denoted here as $AN1$, $AN2$, $AN3$, $AN4$, $AN5$ and $AN6$. Each contains 9 taxa with 1 recombinant in each network.

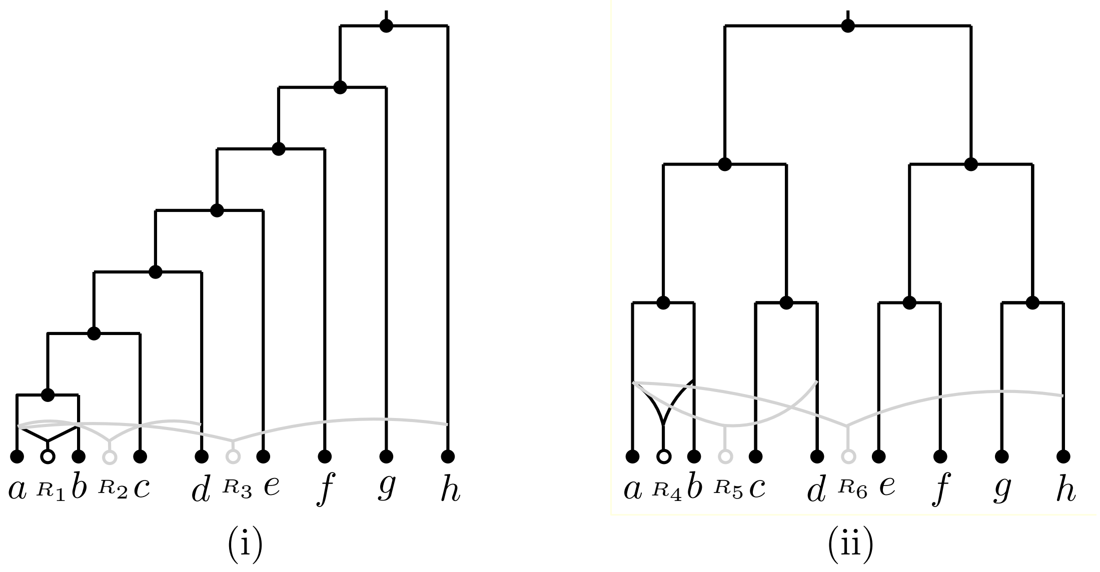


Figure 5.10: The 6 phylogenetic networks presented in [Holland et al., 2002]. The networks in (i) have an unbalanced topology, with the network containing R_1 as the most unbalanced. The positioning of the reticulations in relation to their parents is close, intermediate and divergent for R_1, R_2 and R_3 respectively. The networks in (ii) have a balanced topology. Similarly, the positioning of the reticulations in relation to their parents is close, intermediate and divergent for R_4, R_5 and R_6 respectively.

For each of the six networks we generated 100 multiple sequence alignments. We used these sequence alignments as input to SeqTrinet and generated 100 dense sets of trinets. These trinet sets were used as input into TriLoNet. TriLoNet completed 100 runs on average for each of the six networks in 46 seconds. We used the measures defined in Section 5.2.3 to examine the ability of TriLoNet to construct phylogenetic networks from sequence data. The results are summarised in Table 5.3.

	AN1	AN2	AN3	AN4	AN5	AN6
Average Triplet Consistency Score	100	100	87.57	100	100	100
Average Trinet Consistency Score	97.26	97.73	83.07	100	100	100

Table 5.3: Average triplet and trinet consistency scores for the six phylogenetic networks presented in 5.10 constructed from sequence alignments with length 50,000.

Artificial Network 1 (AN1) is the most unbalanced of the six networks, with the recombinant parent taxa a and b positioned closely with taxon R_1. For the experiments on phylogenetic network AN1, in 94 out of 100 runs the network constructed by TriLoNet exactly matched the network that was used to generate the alignment. In some of the cases where a trinet consistency score of 100 was not achieved, taxon a was placed under the reticulation in place of taxon R_1. In the other network an extra reticulation vertex was inserted, which created a gall with taxa g and h on either side and the rest of the network below this extra reticulation vertex.

Artificial Network 2 (AN2) has an unbalanced topology, and the parent taxa a and d of the recombinant taxon R_2 are at an intermediate distance from R_2. Similarly for AN2, 96 out of 100 runs resulted in the correct network being constructed by TriLoNet. For one of the incorrectly constructed networks, an extra reticulation was inserted, creating a gall similar to the one described above for the three incorrect networks in the AN1 experiments. In the other case, an extra reticulation was inserted that placed taxon g below a reticulation, with taxon h on one side of the gall and the rest of the network on the other side.

Artificial Network 3 (AN3) has an unbalanced topology, and recombinant taxon R_3 has taxa a and h as divergent parents. For the experiments on AN3, in several of the 100 runs the same network was constructed with taxa a and b in a cherry, whereas in the original network taxa a and b are located under separate arcs. Taxon g is located under a separate arc in the networks constructed by TriLoNet, whereas in the original network g is positioned in the same cactus as all other taxa. Interestingly, increasing the κ threshold value increased the number networks correctly constructed by TriLoNet. A κ value of 15.0 resulted in all networks constructed by TriLoNet matching the original network.

For the more balanced phylogenetic networks AN4, AN5 and AN6, TriLoNet was able to exactly reconstruct the original network on which Seq-Gen evolved the sequences down. Artificial Network 4 (AN4) is the most balanced of the 6 networks, with the recombinant parent taxa a and b positioned closely with taxon R_4. Artificial Network 5 (AN5) has an balanced topology, the parent taxa a and d of the recombinant taxon R_5 are at an intermediate distance from R_5. Artificial Network 6 (AN6) has a balanced topology, recombinant taxon R_6 has divergent parents, taxa a and h.

We repeated the above experiments and changed the total sequence length from 50,000 to 1,000, 10,000 and 100,000. The results are presented in Table 5.5 , Table 5.4 and Table 5.6.

	AN1	AN2	AN3	AN4	AN5	AN6
Average C'-score	91.94	91.47	89.87	96.21	95.71	96.45
Average C-score	31.67	33.64	80.74	39.70	43.08	94.33

Table 5.4: Average triplet and trinet consistency scores for the six phylogenetic networks presented in 5.10 constructed from sequence alignments with length 1,000.

	AN1	AN2	AN3	AN4	AN5	AN6
Average C'-score	99.93	99.57	89.74	100	100	100
Average C-score	66.44	71.71	80.96	95.51	98.05	100

Table 5.5: Average triplet and trinet consistency scores for the six phylogenetic networks presented in 5.10 constructed from sequence alignments with length 10,000.

	AN1	AN2	AN3	AN4	AN5	AN6
Average C'-score	100	100	87.50	100	100	100
Average C-score	99.77	100	83.35	100	100	100

Table 5.6: Average triplet and trinet consistency scores for the six phylogenetic networks presented in 5.10 constructed from sequence alignments with length 100,000.

The same general trends can be seen when compared to the experiments where the multiple sequence alignments had length 50,000 and 100,000, although the shortening of the sequence length decreased the number of phylogenetic networks that exactly match the input networks. In particular, the average trinet consistency scores for the more unbalanced networks (AN1, AN2, AN3) decreased, with AN1 and AN2 being more difficult to accurately reconstruct from the sequence alignments.

Decreasing the κ value from 6.5 to 2.0 for the experiments with AN1 and sequence length 10,000 resulted in TriLoNet constructing the correct network for each of the 100 runs. However, with network AN3, increasing κ to 20.0 resulted in TriLoNet correctly constructing each network. As suggested by the experiments on the HIV data set subalignments in Section 5.4.1, increasing κ tends to highlight the more network-like features of a data set. This increase seems to force the taxa in the cherry $\{a, b\}$ found in several of the networks constructed with a $\kappa = 6.5$ to separate and be placed under separate arcs.

We next varied the κ threshold on the alignments with sequences of length 50,000 to see what impact this would have on the topologies of the networks, the results are summarised in Table 5.7. The results here suggest that a lower κ threshold causes TriLoNet to correctly construct AN1, an unbalanced network with close parents. With networks AN3 and AN6 (divergent parents), a higher κ threshold increases the average triplet and trinet consistency scores.

Kappa Value		AN1	AN2	AN3	AN4	AN5	AN6
1.0	Average C'-score	100	98.88	60.14	100	100	72.98
	Average C-score	100	98.81	46.86	100	100	63.44
2.0	Average C'-score	100	98.88	72.32	100	100	64.07
	Average C-score	100	98.81	63.1	100	100	50.86
3.0	Average C'-score	100	99.98	74.38	100	100	100
	Average C-score	100	99.98	65.83	100	100	100
4.0	Average C'-score	100	100	87.36	100	100	100
	Average C-score	99.77	100	83.15	100	100	100
5.0	Average C'-score	100	100	87.5	100	100	100
	Average C-score	99.41	99.41	83.15	100	100	100
6.0	Average C'-score	100	100	87.5	100	100	100
	Average C-score	98.73	99.08	83.15	100	100	100
7.0	Average C'-score	100	100	88.16	100	100	100
	Average C-score	96.19	96.82	83.86	100	100	100
8.0	Average C'-score	100	100	91.93	100	100	100
	Average C-score	93.5	92.15	88.71	100	100	100
9.0	Average C'-score	100	100	96.51	100	100	100
	Average C-score	90.71	87.13	95.18	100	100	100
10.0	Average C'-score	100	100	98.84	100	100	100
	Average C-score	87.6	80.59	98.1	100	100	100

Table 5.7: The average triplet and trinet consistency scores for the six networks presented in 5.10. The κ threshold used in SeqTrinet has been varied between 1.0 and 10.0.

In all of the experiments on the six networks above, the first 50% of the sequence alignment is generated using the tree with the recombinant attached to its left parent and the second 50% is generated using the tree with the recombinant attached to its right parent. We also tried varying this contribution to 75%-25% with a $\kappa = 6.5$. The results are summarised in Table 5.8. This has some impact on the average triplet and trinet consistency scores, in particular for network AN3. There is also a slight decrease in the scores for AN1, AN2 and AN6.

	AN1	AN2	AN3	AN4	AN5	AN6
Average C'-score	100	100	61.21	100	100	96.12
Average C-score	94.29	94.06	48.27	100	100	94.77

Table 5.8: Average triplet and trinet consistency scores for the six phylogenetic networks presented in 5.10 constructed from sequence alignments with length 33,333, $\kappa = 6.5$ and a 75%-25%. contribution

As remarked in [Holland et al., 2002] and supported by our results, we found it more difficult to accurately identify recombinant taxa in networks with unbalanced topologies. One possible reason for this would be the combination of long and short branch lengths. The length of the sequences also impacts the ability of TriLoNet to reconstruct phylogenetic networks; it being easier to correctly construct networks from longer sequence alignments.

5.4 Application to real data sets

In this section we apply TriLoNet to seven real biological data sets. The data sets presented in this section are multiple sequence alignments that we preprocessed to remove any sites containing gaps and any characters other than the four nucleotides $\{A, C, G, T\}$. Unless stated otherwise, the trinet sets were constructed using a κ value of 6.5. For the discussion in choosing this value see Chapter 3. All network images in this section were produced using the GraphViz [Gansner et al., 2006] software package; see Chapter 2 for details.

The first data set containing HIV-1 virus sequences that we tested was also studied in [Huber et al., 2011b]. This offers further comparison between the TriLoNet and Lev1athan network construction algorithms. To further test and apply TriLoNet we also examined six data sets presented in [Morrison, 2015]. This resource hosts data sets containing known recombinants and our aim was to see if they could be detected in the networks constructed by TriLoNet. Each of the data sets chosen has been previously studied, so we can use this biological background to help interpret the networks constructed by TriLoNet. The six data sets we considered include sequences of freshwater eels [Aoyama et al., 2001], Hepatitis B Virus (HBV) [Bollyky et al., 1996], Giardia parasite [Cooper et al., 2007], a fungus causing wheat scab [O'Donnell et al., 2000], North American Dryopteris (a group of ferns) [Sessa et al., 2012] and partial sequences from sedge and rush plants [Starr et al., 2007].

5.4.1 HIV data

To test our network construction algorithm and compare the results to those produced in [Huber et al., 2011b] we tested a HIV-1 virus data set first analysed in [Salemi and Vandamme, 2003]. The Human Immunodeficiency Virus (HIV) causes the Acquired Immunodeficiency Syndrome (AIDS) condition. The virus attacks and weakens the human immune system which provides further opportunity for other viruses and germs to thrive. Inferences made from phylogenetic approaches has aided in the understanding in the evolution of HIV [Castro-Nallar et al., 2012].

The multiple sequence alignment contains one known recombinant sequence ($KAL{-}153$), sequences A, B, D, F, G, H, J and two already identified breakpoints at sites 2700 and 8926. These two breakpoints induce three subalignments 1-2699, 2700-8925 and 8926-9953. A break point is the position in a sequence where it is believed that foreign genetic material is integrated into the existing sequence. The $KAL - 153$ strain is believed to be a recombinant of taxa A and B [Salemi and Vandamme, 2003].

As remarked in [Huber et al., 2011b], Lev1athan, Cluster networks and Galled networks (two methods implemented in Dendroscope) all had issues with this data set. Rather than using the Lev1athan algorithm, [Huber et al., 2011b] used an optimal simple level-1 network construction algorithm (Algorithm 2, [Huber et al., 2011b]) which by definition constructs a network with only one reticulation vertex. The study in in [Huber et al., 2011b] tested the Cluster networks and Galled networks approaches and the results postulated between two and four recombination events for this data set.

The network constructed by TriLoNet on the entire alignment is shown in Figure 5.11. The leaf H was identified as a recombinant which is not correct. Interestingly, the parent of taxon H is unresolved in the phylogenetic tree constructed on the sub-alignment 2700-8925, as shown in Figure 9 [Huber et al., 2011b].

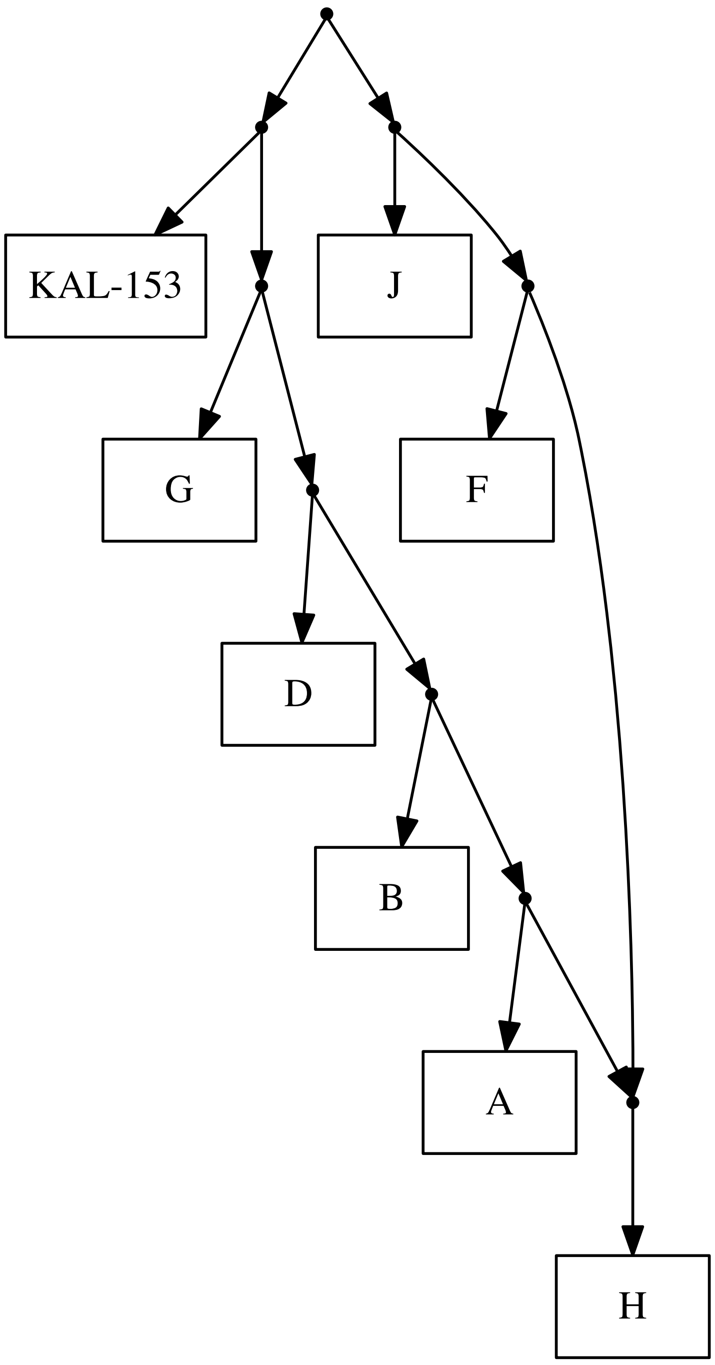


Figure 5.11: Network constructed on entire HIV sequence alignment.

Based on this information, we removed H from the input and ran TriLoNet again on the entire alignment; the resulting network is shown in Figure 5.12. Our algorithm correctly identified leaf $KAL-153$ as a recombinant of taxa A and B.

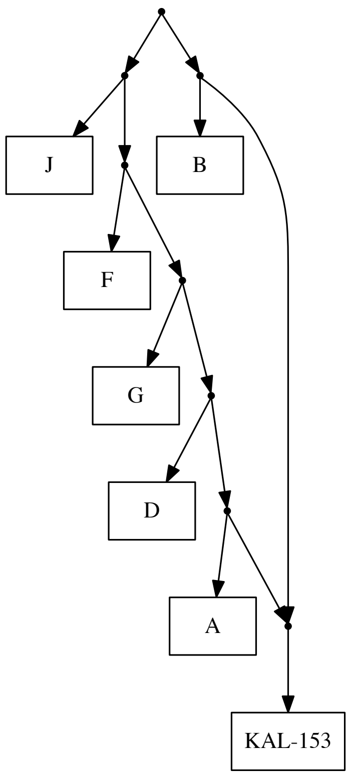


Figure 5.12: Network on HIV entire alignment (1-9953) with H removed.

To better understand why taxon H was selected and positioned under a reticulation vertex, we examined the set of trinets containing taxon H constructed by SeqTrinet from the input sequence data. All but one of these trinets were of type S_1 or S_2 with H as the leaf below the reticulation vertex. Similarly, we examined the set of trinets containing taxon K and found that several of these trinets were of type S_1 or S_2 with K being placed below a reticulation vertex. However, in every trinet of type S_1 or S_2 containing both H and K, it was taxon H that was placed below a reticulation vertex. This would be a possible reason for causing taxon H instead of K to be selected as a leaf below a reticulation vertex in the network constructed by TriLoNet.

We also investigated what would happen when TriLoNet was run on the subalignments mentioned above. Note that a phylogenetic tree is constructed for each subalignment in [Huber et al., 2011b], which we used for comparison with the networks constructed by TriLoNet.

For the first subalignment taking $\kappa = 6.5$ resulted in TriLoNet constructing a cactus with taxon H below the reticulation vertex (not shown). We then varied κ to determine the impact this parameter had on the topology of the networks constructed by TriLoNet. The network constructed by TriLoNet on this subalignment, with a $\kappa = 3.0$ is shown in Figure 5.13. This could indicate that lower κ threshold values brings out the more tree-like features of a data set. In this network taxon H is still placed under a reticulation vertex. Reassuringly, taxa B and D are siblings and taxa A and K are siblings, a similarity shared with the corresponding phylogenetic tree for this subalignment presented in [Huber et al., 2011b].

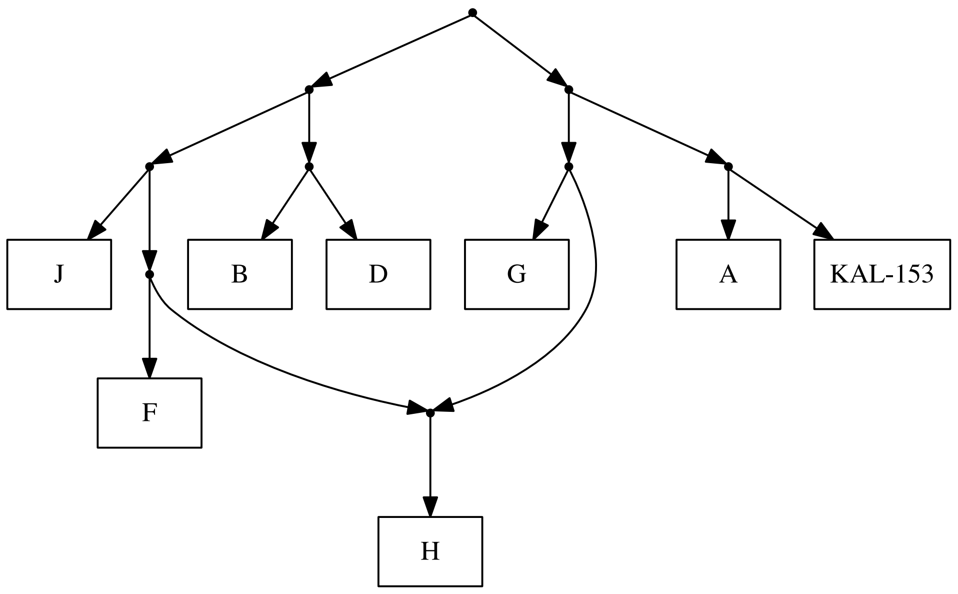


Figure 5.13: Network constructed on the first HIV subalignment covering character sites 1 - 2699, with a κ value of 3.0.

The second subalignment on this data set consists of sites 2700 - 8925 and is the largest of the three subalignments. The network constructed by TriLoNet using a κ value of 6.5 was also a cactus with taxon H below a reticulation vertex (not shown). The network constructed by TriLoNet on this subalignment, with a κ value of 1.0 is shown in Figure 5.14. Lowering the κ parameter from 6.5 to 1.0 increased the number of reticulation vertices in the network constructed by TriLoNet while also grouping taxa B, D, K as well as taxa A, G in a similar way to the second phylogenetic tree in Figure 9 [Huber et al., 2011b].

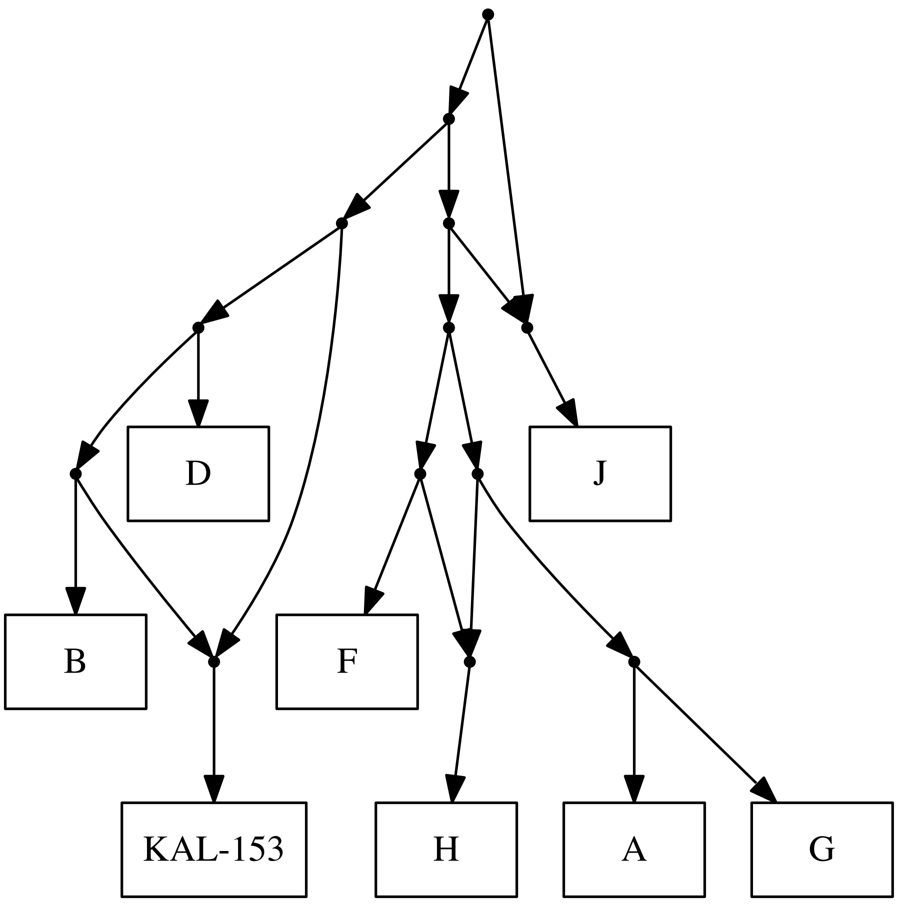


Figure 5.14: Network constructed on the second subalignment of the HIV data set covering character sites 2700 - 8925, with a κ value of 1.0.

The third subalignment on this data set consists of sites 8926 - 9953. TriLoNet had more difficulty in constructing a network similar to the third phylogenetic tree presented in Figure 9 [Huber et al., 2011b]. One reason for this could the that this subalignment is much shorter than the other two subalignments in this data set. With a κ value of 6.5, TriLoNet constructed a cactus with taxon F under a reticulation vertex (not shown). Changing κ to 1.0 resulting in a phylogenetic network shown in Figure 5.15, with taxa A and K and taxa B and D as cherries, a feature shared with the third phylogenetic tree presented in Figure 9 [Huber et al., 2011b].

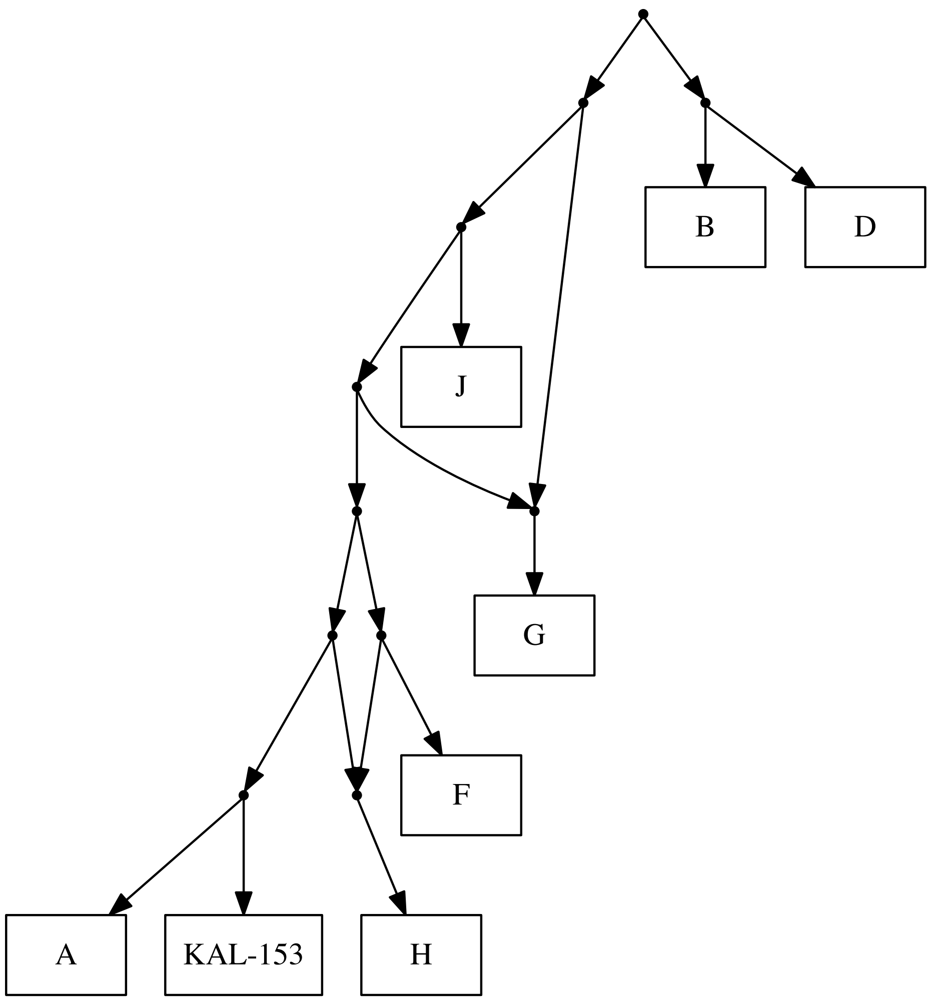


Figure 5.15: Network constructed on the third subalignment of the HIV data set covering character sites 8926 - 9953, with a κ value of 1.0.

We also varied the κ value parameter in SeqTrinet to see what impact this would have on the networks constructed by TriLoNet. This allows for the restriction or relaxation of the condition in determining if a trinet should be of type S_1

or S_2 if κ is above a certain threshold, or any other type otherwise. Lowering this parameter from 6.5 to 1.0 causes more trinets to be considered as having a more tree-like topology. Interestingly, with a κ value of 1.0 the network constructed by TriLoNet (Figure 5.16) on the entire sequence alignment shares similarities with the topology of the phylogenetic tree shown in Figure 9 [Huber et al., 2011b] constructed on the second sub-alignment. Taxa A and G are siblings and taxa B, D and K are in close proximity, with K placed under a reticulation vertex.

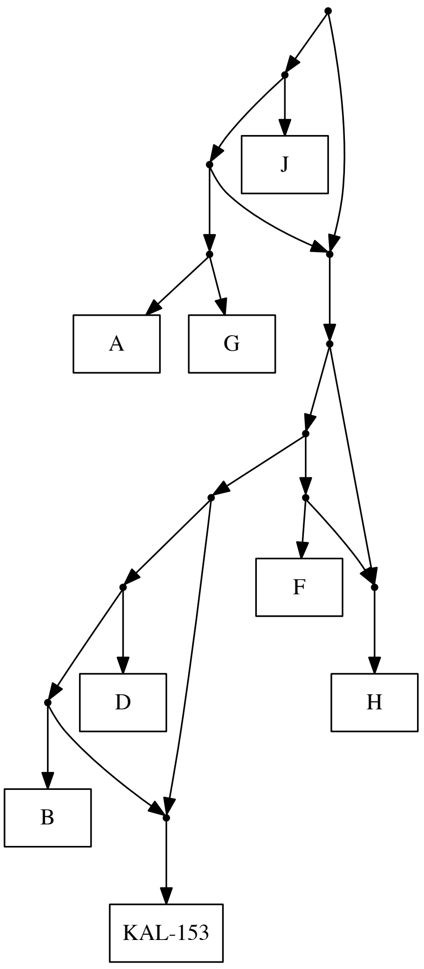


Figure 5.16: The phylogenetic network constructed on the HIV data set with a κ value of 1.0.

5.4.2 Eel data

The multiple sequence alignment presented in [Aoyama et al., 2001] contained 39 Anguilla eel partial gene sequences of length 1140 before preprocessing. The sequences come from a family of catadromous eels that spend their lives in freshwater environments and, with the exception of Anguilla anguilla, are mostly found along the eastern margins of the Australian, Eurasian, African and American continents [Aoyama et al., 2001]. The study aimed to understand why these eels are absent from the South American eastern coastline and the South Atlantic as well as why the two Atlantic species became separated from those in the Indo-Pacific region. Figure 8 [Aoyama et al., 2001] depicts the suggested molecular phylogenetic relationships and dispersion of the genus Anguilla. Phylogenetic analysis was used to examine the geographic distribution and dispersal of the species.

Supplementary notes from [Morrison, 2015] suggest that AB021780 A.bicolor bicolor is a recombinant between A. bicolor (AF006708, AF006709, AF006710, AB021774) (from position 477) and A.mossambica (AF074864, AF074865, AB021782) (up to position 456). We were unable to identify A. bicolor bicolor as a recombinant leaf although we did identify several clades/ species groupings in the network constructed by TriLoNet (see Figure 5.17). These groupings include A. mossambica, A. anguilla, A. rostrata, A. mamorata, A. japonica, A. reinhardti, A. bicolor pacifica and A. australis australis. In the network constructed by TriLoNet the species AB021771_A._megastoma was located below a large gall.

There are some similarities between the network we constructed and the phylogenetic tree presented in Figure 3 [Aoyama et al., 2001]. The A. australis australis and A. australis schmidti taxa from the Oceania species lineage were closely grouped in a non-trivial gall in the network constructed by TriLoNet. The A. celebesensis and A. megastoma taxa from the Tropical Pacific species lineage were also positioned closely in the network. Similarities with Figure 8 [Aoyama et al., 2001] include the close proximity of the A.rostrata, A.anguilla and A.mossambica species.

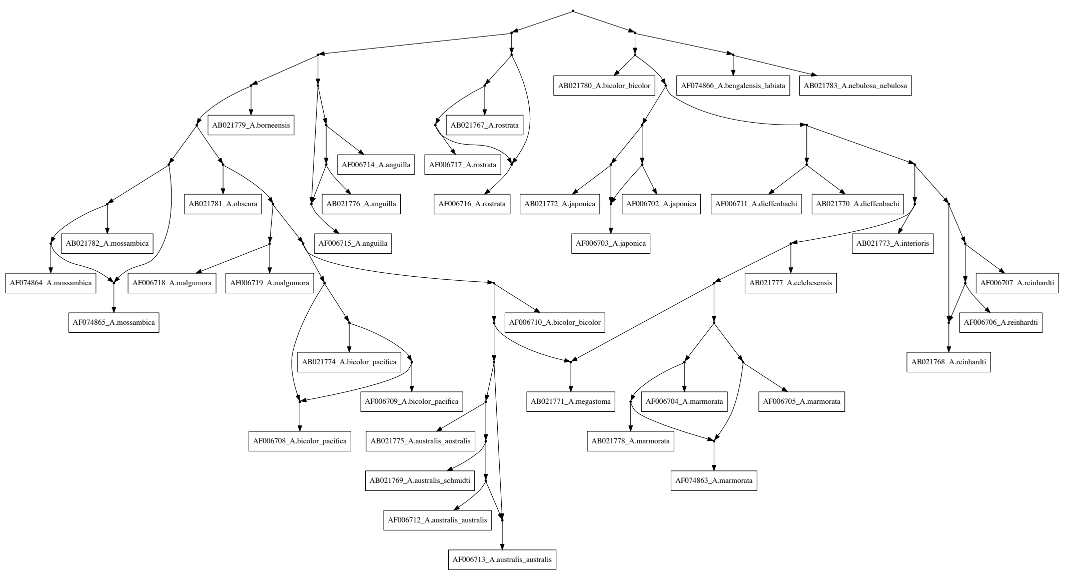


Figure 5.17: Network constructed on the data set presented in [Aoyama et al., 2001].

5.4.3 Hepatitis B Virus data

The multiple sequence alignment presented in [Bollyky et al., 1996] is a comparison of 25 hepatitis B virus (HBV) isolates with complete genome sequences of length 3229. HBV is an infectious virus that attacks the liver. In several areas of the world HBV causes significant mortality and morbidity, in particular tropical Africa and East Asia. In 1993 approximately 10 to 15% of the population in these areas were chronic HBV carriers [Merican et al., 1993]. High rates of chronic infections are also found in the Amazon and the southern areas of eastern and central Europe. Today, approximately 240 million people are chronically affected by HBV and an estimated 780,000 people die each year [Organisation, 2015b]. The work in [Bollyky et al., 1996] used phylogenetic analysis to investigate if recombination was a factor in the genetic diversity in the 25 genomes.

In [Bollyky et al., 1996], two of the isolates were positioned differently in the three phylogenetic trees reconstructed from different open reading frames, a result of recombination between viruses of different genomic and antigentic types. The phylogenetic network constructed by TriLoNet for the data set is presented in

Figure 5.18. Table 1 in [Bollyky et al., 1996] details the genotypes A, B, C, D and F of each taxa in the phylogenetic network. The genotypes are colour-coded in Figure 5.18 with genotype A in grey, genotype B in blue, genotype C in orange, genotype D in green and genotype F in pink. We anticipated that taxa HBVDNA and HPBADW1 might be selected as recombinants in the network constructed by TriLoNet. TriLoNet constructed a network placing taxon HBVDNA under a reticulation vertex. However, this was not the case for HPBADW1, instead, taxa HBVADW and HPBADR1CG were placed under non-trivial reticulation vertices.

The two viruses HBVDNA and HPBADW1 clustered with different genotypes in different open reading frames. These two taxa were located in different viral genotype groups in the trees presented in Figure 1 of [Bollyky et al., 1996], this suggests that these taxa are recombinants. In the first two trees taxon HPBADW1 is in genotype B however in the third tree it is in genotype A. The taxon HBVDNA is in genotype D in the first tree and genotype A in the second and third trees.

Table 2 in [Bollyky et al., 1996] summarises the localisation of recombination events in the HBV sequences. Isolate HBVDNA has parental lineage from virus XXHEPA from genotype D and virus HUMPREX from genotype A. These taxa and genotype groups are positioned closely to HBVDNA in the network constructed by TriLoNet. Similarly, isolate HPBADW1 has parental lineage from virus HPBAD2 from genotype B and virus HPBADWZCG from genotype A.

The positions of the taxa in Figure 5.18 clearly form distinct groups corresponding to their genotype. However, this data set may highlight a limitation of a level-1 network construction approach; we believe that this data set could be more appropriately represented using a level-2 phylogenetic network which could better represent a more complex pattern of evolution for the recombinant taxa HBVDNA and HPBADW1.

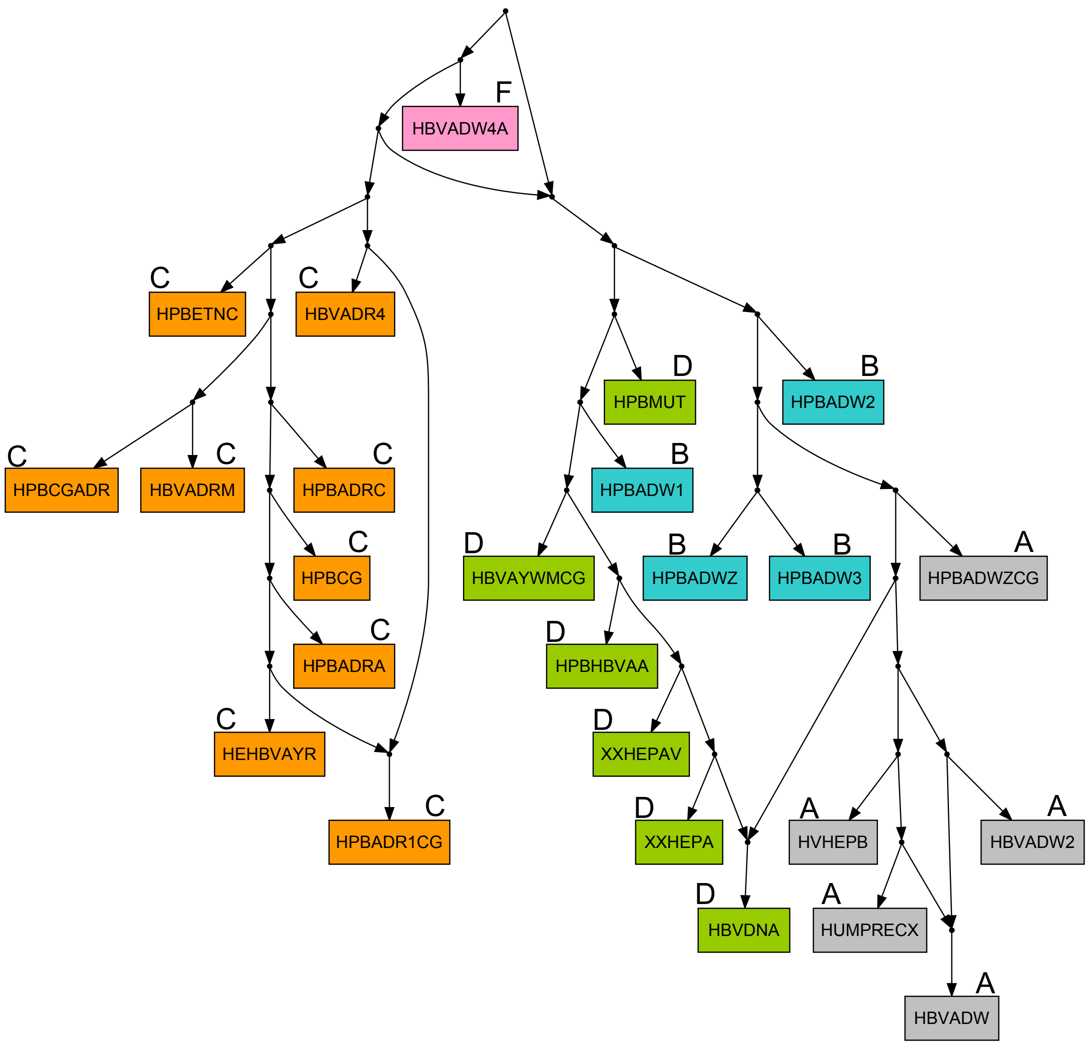


Figure 5.18: Network constructed by TriLoNet on the data set presented in [Bollyky et al., 1996].

We then tried removing taxon HBVNDA from the input sequence alignment and constructed another network on this data set to determine if taxon HBVNDA had any affect on not identifying the other recombinant taxon HPBADW1 in the previously constructed network. This resulted in a similar network, in which the taxon HPBADW1 was now correctly located below a reticulation vertex. This network is shown in Figure 5.19, indicating that the parental lineage of HPBADW1 originates from genotypes A and B.

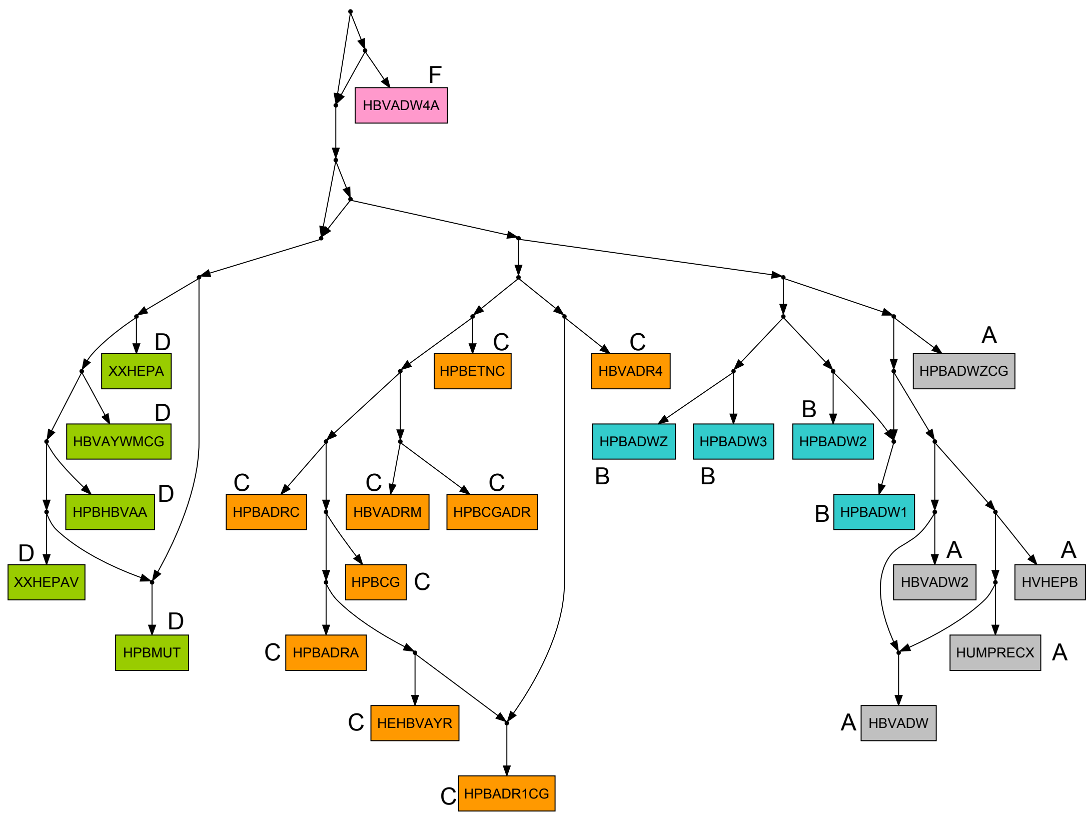


Figure 5.19: Network constructed by TriLoNet on the data set presented in [Bollyky et al., 1996] with taxon HBVDNA removed.

5.4.4 Giardia parasite data

Giardia is a parasite that causes giardiasis through the infection of the intestines. Infection from this single-celled organism with two nuclei often occurs from the consumption of contaminated water or food [Organisation, 2015a]. In this paper, Giardia human faecal isolates were obtained from a population in Lima, Peru, an area that is highly endemic for giardiasis in humans. The study in [Cooper et al., 2007] raised questions on using evidence from population genetics to suggest that recombination is present in Giardia, with results suggesting distinctly different histories between the three examined chromosome loci.

The data set presented in [Cooper et al., 2007] contained 7 sequences of length 17010 for three partial chromosome sequences. Chromosome 3 represented character sites 1 - 5979, Chromosome 4 represented sites 5980 - 7444 and Chromosome

5 represented sites 7445 - 17010.

TriLoNet placed taxon G. intestinalis isolate 335 below a reticulation vertex, as shown in Figure 5.20, as well as placing isolate G. lamblia ATCC 50803 WB as the outgroup and isolates 303 and 305 as siblings. Figure 3 in [Cooper et al., 2007] presents three maximum likelihood trees on the three chromosomes. In the second tree isolate 335 is a sibling of isolate 55 and in the third tree the parent vertex of isolates 335, 55 and JH is unresolved. The topology of the network constructed by TriLoNet shares most resemblance to the maximum likelihood tree on chromosome 5 presented in Figure 3 [Cooper et al., 2007].

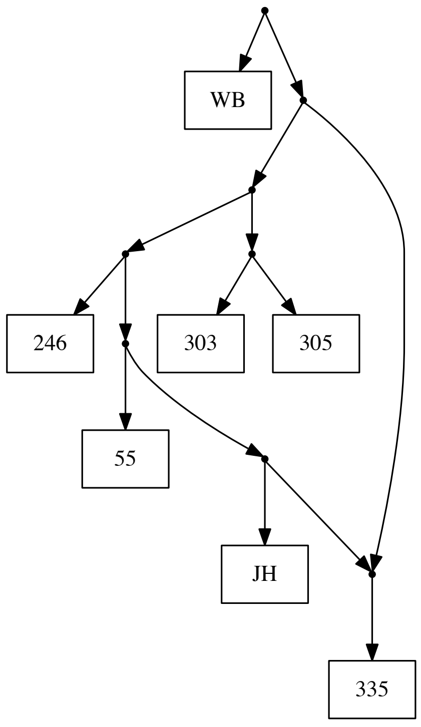


Figure 5.20: Network constructed by TriLoNet on the eel data set presented in [Cooper et al., 2007].

If the breakpoints are known, in Step 1 of Section 3.3 we explain how we can adapt TriLoNet to use this information to calculate the κ value for each individual subalignment such that each subalignment contributes equally to the total κ value. This may be useful if there is a large variation in the number of character sites in particular subalignments. For the data set presented in [Cooper et al., 2007], Chromosome 4 is represented by 1,465 sites whereas Chromosome 5 is represented by 9,566 sites. The phylogenetic network presented in Figure 5.21 was constructed by TriLoNet using the two breakpoints as input parameters. The trinet $S_2(JH; 55; 335)$ displayed by this network which is consistent with the three trees in Figure 3 [Cooper et al., 2007].

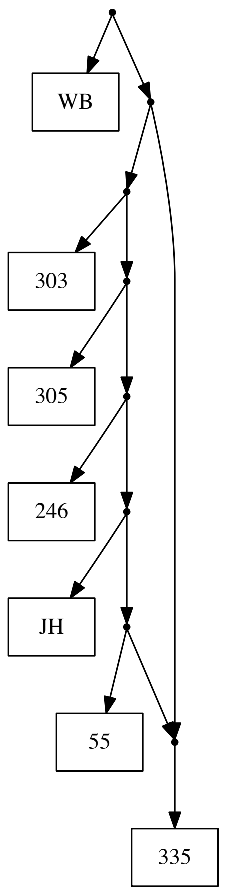


Figure 5.21: Network constructed by TriLoNet on the eel data set presented in [Cooper et al., 2007] using breakpoints to weight the subalignments.

5.4.5 Fungus data

Fusarium graminearum, also known as Gibberella zeae is a plant pathogen which causes a fungal disease called fusarium head blight on both wheat and barley. Worldwide, this disease causes billions of dollars in economic losses annually [De Wolf et al., 2003]. This disease has become an epidemic problem, not only because of the economic impact from decreased seed yield and quality, but also because seeds infected with fusarium graminearum are also often contaminated with mycotoxins that cause harm to animals [O'Donnell et al., 2000]. Results from the study in [O'Donnell et al., 2000] include a phylogenetic tree presented in Figure 3, [O'Donnell et al., 2000] which highlights seven lineages of the Fusarium graminearum clade, indicating possible geographic origin.

The data set presented in [O'Donnell et al., 2000] contained 37 sequences of length 4146 with taxa 28338 and 28721 identified as recombinants. The network constructed by TriLoNet on this data set pictured in Figure 5.22 shares strong topological similarities with the phylogenetic tree presented in Figure 3 [O'Donnell et al., 2000], in particular the grouping of the seven lineages. The colour-coding in Figure 5.22 correspond to the Pan-Northern Hemisphere (grey), Asian (yellow), African (purple), South-Central American (green), African (pink), African (blue) and South American (orange) in Figure 3 [O'Donnell et al., 2000].

In [O'Donnell et al., 2000] it is reported that strain 28721 is a hybrid strain containing alleles from lineages 2 and 6 of the Fusarium graminearum clade. Our findings support this; taxon 28721 is a recombinant leaf in the network constructed by TriLoNet, which is closely positioned to 28436, 28723, 29010 (African lineages) as well as 6101, 13818, 26156 and 28720 (Asian lineages).

The taxon 28338 was not identified as a recombinant and was not a focus of the study by [O'Donnell et al., 2000], although it was closely grouped with 28062, 28065 and 28334. This group of taxa from sequences of F.pseudograminerum were used as an outgroup to root the tree presented in Figure 3 [O'Donnell et al., 2000]. This is supported by TriLoNet as this group of taxa is under a different arc from all other taxa in the network.

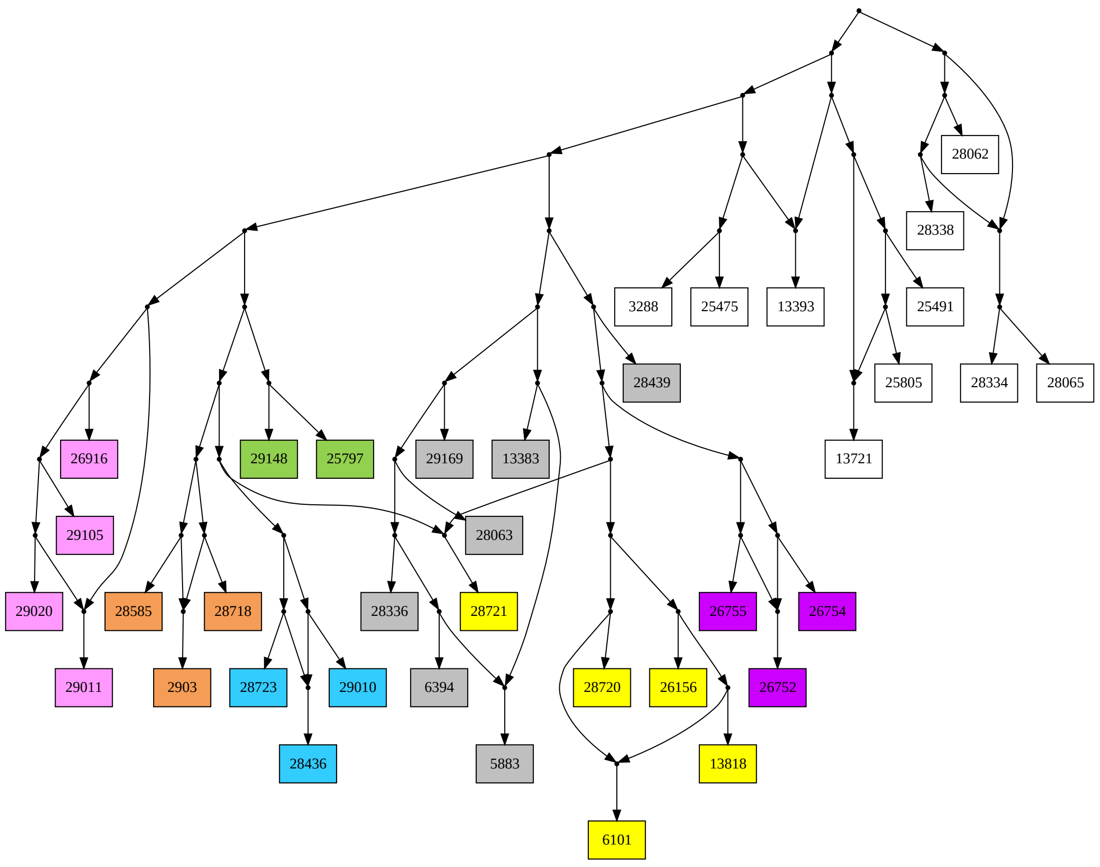


Figure 5.22: Network constructed by TriLoNet on the data set presented in [O'Donnell et al., 2000].

5.4.6 Dryopteris fern data

The study in [Sessa et al., 2012] aimed to explore the reticulate evolution in North American Dryopteris, Dryopteridaceae woodferns. It is believed that recombination has been a part of the evolutionary history of Dryopteris. This group of ferns has been widely studied, in particular in North America due to the suspected extensiveness of reticulate evolution via allopolyploid hybridisation [Sessa et al., 2012].

The data set presented in [Sessa et al., 2012] contained 27 taxa with eight partial gene sequences of length 6042 from Dryopteris ferns with the sequence D. celsa EBS27 as a recombinant taxon, diverging from its paternal parent, D. goldiana, and from its maternal parent, D. ludoviciana.

One hypothesis represented in Figure 1 [Sessa et al., 2012] postulates D. goldiana and D. clintoniana as parents of D. celsa. The maximum likelihood tree in Figure 2 [Sessa et al., 2012] places D. celsa closely with D. ludoviciana and D. goldiana. The network constructed by TriLoNet presented in Figure 5.23 supports this, even though it does not identify D. celsa as a recombinant taxon. Figure 5 [Sessa et al., 2012] presents a reticulation network showing hypothesised polyploidisation events, again suggesting the parentage of taxa D. ludoviciana and D. goldiana to D. celsa.

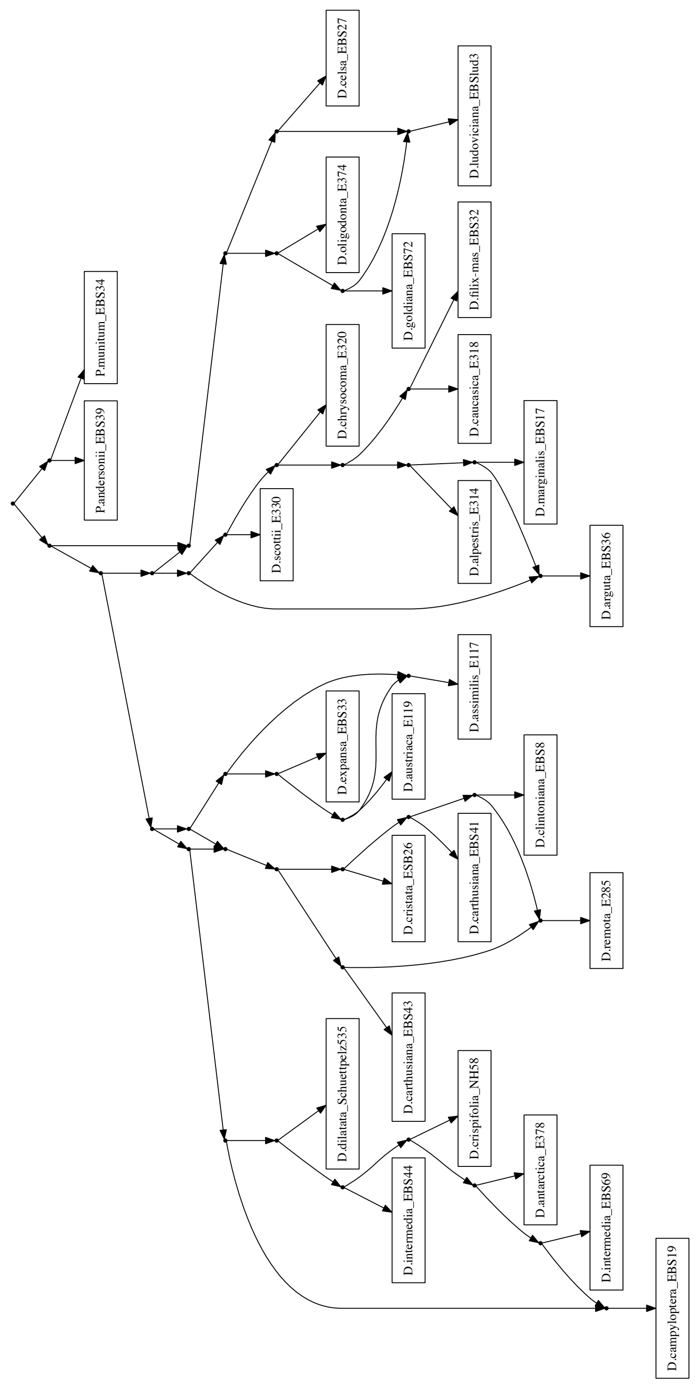


Figure 5.23: Network constructed by TriLoNet on the data set presented in [Sessa et al., 2012] with a κ value of 6.5.

Interestingly, using a κ parameter value of 7.0 in the SeqTrinet algorithm resulted in the identification of D. celsa EBS27 as a recombinant taxon as it swapped positions with D. ludoviciana EBSlud3. This network is shown in Figure 5.24. Both networks constructed by TriLoNet to an extent support the findings of [Sessa et al., 2012] as the three taxa D. ludoviciana, D. goldiana and D. celsa were placed within the same gall but with slightly different positions.

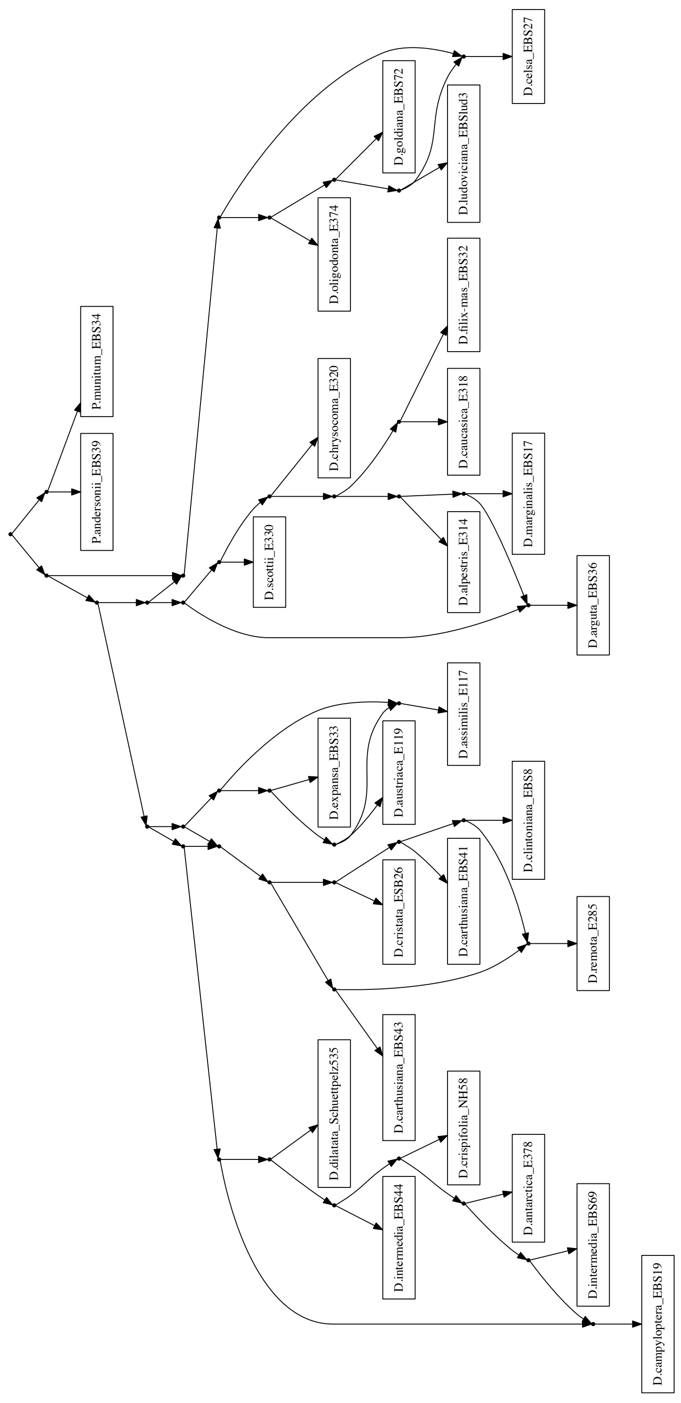


Figure 5.24: Network constructed by TriLoNet on the data set presented in [Sessa et al., 2012] with a κ value of 7.0.

5.4.7 Sedge and rush plant data

The multiple sequence alignment presented in [Starr et al., 2007] contains 22 sequences of length 1399 with partial gene sequences from Cyperaceae Juss (sedge) and selected Juncaceae (rush) plants. Sedge plants are found in a wide range of habitats and are widely distributed. Uses include construction material, paper manufacture and medicines [Starr et al., 2007].

The study aimed to investigate the evolutionary relations within Cyperaceae using molecular characteristics, given that morphological features alone are not able to do so. These relationships are not well known, partly due to the large genetic diversity of the Cyperaceae family. The study also suggests that the position of the recombinant Oxychloe andina (Juncaceae) in previous analyses as either a sister or as nested within Cyperaceae is because it is a Juncaceae/Cyperaceae chimera [Starr et al., 2007]. A chimeric sequence contains DNA from two or more parents. Figure 3 [Starr et al., 2007] presents some phylogenetic trees resulting from their analyses in meeting their objective of determining the phylogenetic position of Oxychloe andina [Starr et al., 2007]. The results from [Starr et al., 2007] suggest that the first half of the sequence probably represents a correct Oxychloe andina sequence, whereas the second half is derived from a Cyperaceae contamination during DNA extraction or PCR amplification.

The network constructed by TriLoNet by taking these sequences as input placed taxon Oxychloe andina below a reticulation vertex as shown in Figure 5.25. The Cyperus taxa are placed on the opposite side of the gall to the Juncus taxa. The phylogenetic trees in Figure 3 [Starr et al., 2007] also suggest two distinct regions Juncaceae and Cyperaceae with Prionium_serratum as an outgroup. The outgroup taxa Prionium_serratum is a sister species to the others and is separated from the other taxa in the network constructed by TriLoNet. Taxon Oxychloe Y12978 in Figure 3(c) [Starr et al., 2007] is a sibling of Distichia_acicularis, this taxon is positioned closely with the recombinant, in the network constructed by TriLoNet. Similarly, taxa Scirpus_polystachyus, Carex_monostachya are positioned closely to the recombinant. The taxa Luzula_nivea, Luzula_purpureosplendens, Luzula_multiflora and Luzula_novaecambriae form a separate cactus structure, with Luzula_nivea placed under a reticulation vertex. The Luzula taxa are located within the Juncaceae region in the phylogenetic trees presented in Figure 3 [Starr et al., 2007], a feature shared with the network constructed by TriLoNet.

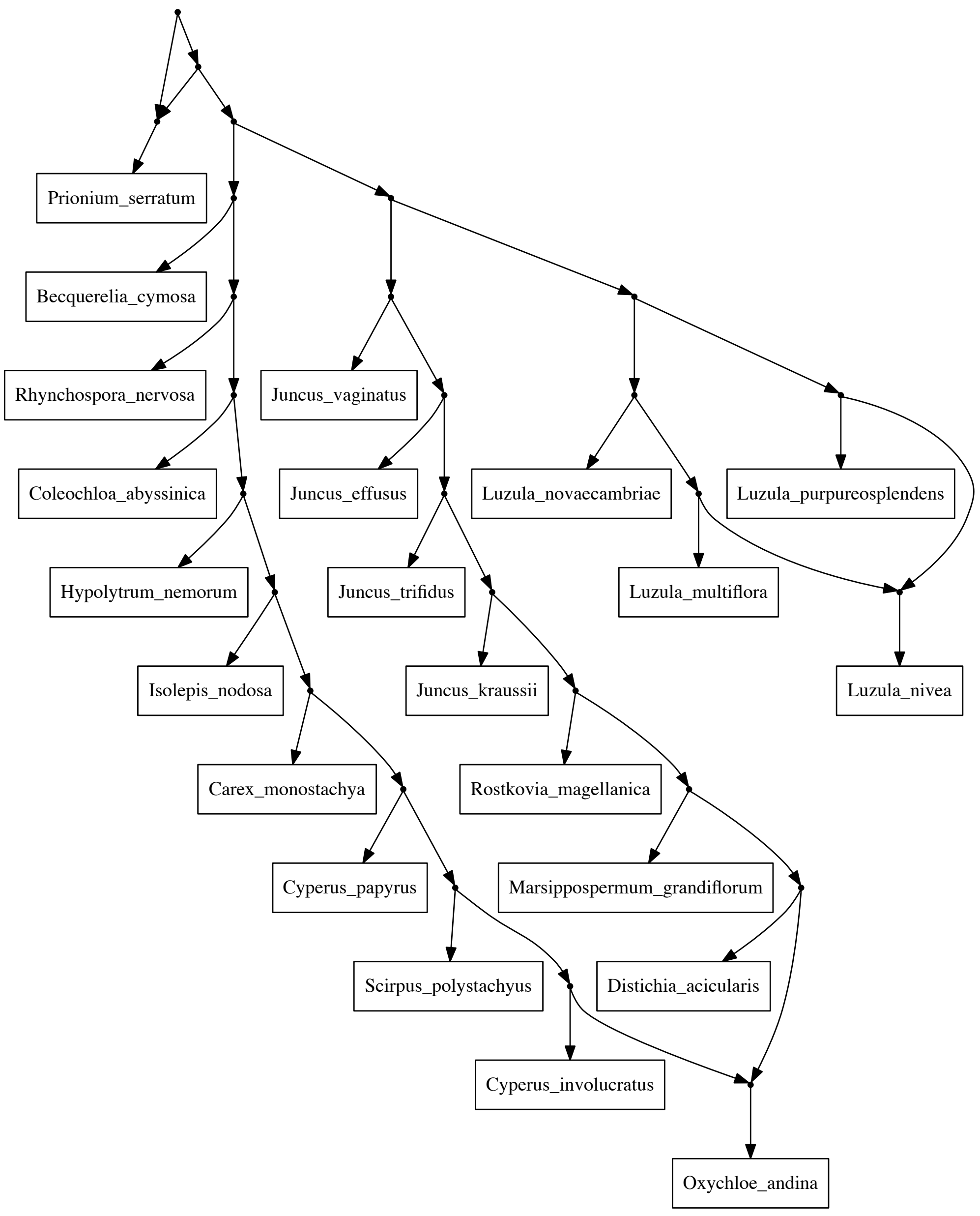


Figure 5.25: Network constructed by TriLoNet on the data set presented in [Starr et al., 2007].

5.5 Concluding remarks

In this chapter we have presented a series of experiments and real data sets to evaluate our approach to phylogenetic network construction. We also compared TriLoNet with Lev1athan, a triplet based reconstruction algorithm. TriLoNet is able to always correctly reconstruct a phylogenetic network from a dense set of trinets not containing noise whereas this is not always the case with Lev1athan. Lev1athan on average has a higher triplet consistency score in comparison to TriLoNet, however, the phylogenetic networks constructed by Lev1athan tend to contain many more reticulation vertices. We found that TriLoNet achieved a higher trinet consistency score than Lev1athan, with a considerable difference in the experiments with lower levels of noise.

We simulated artificial sequence alignments with recombinant taxa on six networks presented in [Holland et al., 2002] and varied the sequence length, κ threshold and left-right sequence contribution. The results support the proposal in [Holland et al., 2002] that is is more difficult to identify recombination in data sets generated with unbalanced topologies. We also found it to be easier to correctly identify recombination in longer rather than shorter multiple sequence alignments.

We ran TriLoNet on several real biological data sets with suggested recombination. We have shown that TriLoNet can in some cases identify recombination taxa. In particular, we were able to correctly identify the recombinant taxa in each data set aside from the eel data set and fungus data set where only one of the two recombinants present was selected. The network constructed from the data set presented in [Bollyky et al., 1996] may indicate that a level-2 network construction approach would be able to better represent the relationships inferred from sequences with more complex patterns of recombination.

TriLoNet is a positive step towards constructing networks directly from sequence data.

Chapter 6

Conclusions

6.1 Conclusions

The rapid sequencing of DNA has fuelled the growth of the area of phylogenetics in recent years. Much of the focus until recently has been on the construction of phylogenetic trees. However, as not all data may be best represented by tree-like structures, more emphasis is now being placed on the use of phylogenetic networks to represent more complex patterns of evolution. Although not yet widely adopted by biologists as much of the work in phylogenetic network construction is relatively new, approaches using phylogenetic networks are becoming increasingly popular.

The key scientific contribution of this thesis is the introduction and implementation of the SeqTrinet and TriLoNet methods, which form a supernetwork based approach to constructing level-1 phylogenetic networks directly from multiple sequence alignments. TriLoNet is the first method to use a supernetwork approach to construct level-1 networks from noisy data by puzzling together information contained in smaller networks. TriLoNet also allows the use of breakpoints in a sequence alignment although they are not required. TriLoNet has been implemented in Java and accepts input from NEXUS [Maddison et al., 1997] and FASTA [Zhang Lab, 2015] files, two of the popular formats used in phylogenetics for representing sequence data.

In more detail, in Chapter 3 we introduced a new approach called SeqTrinet to constructing phylogenetic networks on three leaves from DNA sequence data.

In Chapter 4 we then described the TriLoNet algorithm. In Chapter 5 we presented the results of three simulation and comparison studies. We showed that, particularly when the input contains low levels of noise, TriLoNet compares well with Lev1athan, a triplet based level-1 network construction approach. We also introduced a novel approach called TriExtract for extracting trinets from level-1 phylogenetic networks, which we then used as a measure for the comparison of two phylogenetic networks since the trinets displayed by a level-1 network uniquely determine that network. We also used artificially generated sequence data containing suggested recombination as input to TriLoNet. The results indicate that TriLoNet is able to detect simple recombinant events from sequence data without the use of breakpoints. We then used several real biological data sets containing known recombinant data and found in most cases that TriLoNet was able to identify these recombinant taxa.

6.2 Future work

Here we consider some extensions as well as some possible future directions that could be taken. These include theoretical research ideas as well as some technical extensions to TriLoNet.

6.2.1 Level-2 or higher networks

An obvious although non-trivial direction from here would be to develop and implement a level-2 or higher network construction algorithm. Level-1 networks are a good starting point for the representation of data sets that may not be best represented by a phylogenetic tree. However, as indicated by the HBV data set in Chapter 5, the use of higher level networks may be more appropriate for data sets containing more complex evolutionary events. The work in [Iersel and Moulton, 2013] has shown that for level-2 networks, the trinets uniquely determine the network. This indicates it should be possible to develop an algorithm to construct level-2 phylogenetic networks from trinets. It should be noted that there are many more level-2 trinets than level-1 trinets. Particularly for noisy data, this increase in the number of possibilities would provide even more of a challenge to puzzle these pieces together into a network. It would also be interesting to investigate the computation of level-2 trinets from sequence data. As we have seen, the detection of evolutionary patterns from sequence data is a non-trivial problem for even level-1 trinets so mapping even more complex events from sequence data would present a considerable challenge.

6.2.2 Non-dense input

TriLoNet currently works on the assumption that any input given is dense. Dense triplet sets have been used in many of the current triplet based network construction techniques due to them leading to a more structured network. In reality triplet sets are not always dense Huson et al. [2010], so relaxing this condition and allowing the input set of trinets to be non-dense would widen the appeal and usage of TriLoNet. It has been shown in [Huber et al., 2014a] that it is NP-hard to construct a binary level-1 network from a non-dense set of trinets. They also present a non-polynomial time algorithm for this problem that could be implemented. Alternatively, it would also be interesting to investigate developing a heuristic to address noisy data.

6.2.3 Quarnets

Here we have constructed phylogenetic networks from networks on three leaves. It would be interesting to investigate how an approach using quarnets (phylogenetic networks on four leaves) or even larger networks might be developed. It is not clear how using these more complex building blocks would impact on the networks constructed from this input. Something to consider here would be that the number of quarnets on a set of taxa quickly increases in comparison the number of trinets on the same set of taxa ($O(n^4)$ vs $O(n^3)$, with n the number of taxa). The additional information contained in quarnets could potentially improve the quality of networks constructed as quarnets should provide more topological insight. However, even though quarnets (or larger networks) provide more information that trinets, it would be much more complex to interpret this and develop an algorithm to fit them together in a meaningful way. Also, it is known that even if all subnetworks of a network are given, these do not necessarily determine the network [Huber et al., 2014b], and so it is not clear when quarnets (or larger networks) will uniquely determine networks.

6.2.4 Constructing networks from sequences

We spent a considerable amount of time developing an approach to obtain a set of trinets from a multiple sequence alignment. One of the main challenges of constructing a set of trinets as opposed to a set of triplets from this data has been to identify the detailed evolutionary events displayed by trinets but not triplets. A key example of this is the difficultly in identifying the difference between an S_1 and S_2 trinet for three taxa in a multiple sequence alignment. Several current methods of mapping sequences to networks such as [Jin et al., 2007] and [Fischer et al., 2015] use the trees embedded inside the network to compute the parsimony score of a network. However, these methods have limitations when applied to trinets since a trinet is not always encoded by the triplets it contains. This is a problem that would benefit from further investigation and a situation where moving from tree-based to network-based thinking could be beneficial.

Another possible avenue for investigation would be a maximum likelihood approach for the construction of networks from sequence data. As with the current parsimony approaches, methods that construct networks using a likelihood approach compute the likelihood on the trees embedded in the network [Yu et al., 2014]. The issue again with this approach is that these trees do not necessarily determine the network, and so we would probably need to develop new models to address this problem.

6.2.5 Extension and improvements to program

TriLoNet constructs trinets from DNA sequence data and currently only considers $\{A, C, G, T\}$. It would be useful to extend this part of the program to process character sites with gaps as well as the other nucleic acid codes. Currently TriLoNet will accept noisy input data, it would be interesting to modify

TriLoNet to accept input where a portion of the data is missing. A graphical user interface would make TriLoNet more accessible and provide a better user experience. One feature that would be useful could be to allow the user to select and highlight trinets contained in a network and compare this to the corresponding trinet obtained from a sequence alignment. Also, it would not be too difficult to enable file formats other than NEXUS and FASTA to be accepted as input to TriLoNet. Currently, TriLoNet can read in a single file in NEXUS, FASTA and TNETS format and will output the results to a text file as well as a eNewick string that can be viewed in Dendroscope. It would be useful to modify this to allow the batch processing of multiple input files.

6.3 Final words

In this thesis we developed and implemented an approach to constructing level-1 phylogenetic networks directly from DNA sequence data. We hope the contribution of our supernetwork approach is a positive step toward the development of new network construction methods that can represent complex evolutionary scenarios such as recombination and lateral gene transfer.

References

Aho, A., Sagiv, Y., Szymanski, T., and Ullman, J. (1981). Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421. 10

Aoyama, J., Nishida, M., and Tsukamoto, K. (2001). Molecular phylogeny and evolution of the freshwater eel, genus anguilla. *Molecular Phylogenetics and Evolution*, 20(3):450–459. xii, 78, 87, 88

Bollyky, P. L., Rambaut, A., Harvey, P. H., and Holmes, E. C. (1996). Recombination between sequences of hepatitis b virus from different genotypes. *Journal of Molecular Evolution*, 42(2):97–102. xiii, 78, 88, 89, 90, 91, 102

Byrka, J., Guillemot, S., and Jansson, J. (2010). New results on optimizing rooted triplets consistency. *Discrete Applied Mathematics*, 158(11):1136–1147. 15

Castro-Nallar, E., Pérez-Losada, M., Burton, G. F., and Crandall, K. A. (2012). The evolution of hiv: inferences using phylogenetics. *Molecular phylogenetics and evolution*, 62(2):777–792. 78

Cooper, M. A., Adam, R. D., Worobey, M., and Sterling, C. R. (2007). Population genetics provides evidence for recombination in giardia. *Current Biology*, 17(22):1984–1988. xiii, 78, 91, 92, 93

De Wolf, E., Madden, L., and Lipps, P. (2003). Risk assessment models for wheat fusarium head blight epidemics based on within-season weather data. *Phytopathology*, 93(4):428–435. 94

Fischer, M., Van Iersel, L., Kelk, S., and Scornavacca, C. (2015). On computing the maximum parsimony score of a phylogenetic network. *SIAM Journal on Discrete Mathematics*, 29(1):559–585. 23, 106

Fitch, W. M. (1971). Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic zoology*, pages 406–416. 9

Gambette, P. and Huber, K. (2012). On encodings of phylogenetic networks of bounded level. *Journal of Mathematical Biology*, 65(1):157–180. 15

Gansner, E., Koutsofios, E., and North, S. (2006). Drawing graphs with dot. 21, 77

Gyles, C. and Boerlin, P. (2013). Horizontally transferred genetic elements and their role in pathogenesis of bacterial disease. *Veterinary Pathology Online*, page 0300985813511131. 19

Habib, M. and To, T.-H. (2011). Constructing a Minimum-Level Phylogenetic Network from a Dense Triplet Set in Polynomial Time. *ArXiv e-prints*. 2, 15

Holland, B. R., Huber, K. T., Dress, A., and Moulton, V. (2002). plots: A tool for analyzing phylogenetic distance data. *Molecular Biology and Evolution*, 19(12):2051–2059. xii, 3, 25, 34, 57, 71, 73, 77, 102

Huber, K. and Moulton, V. (2013). Encoding and constructing 1-nested phylogenetic networks with trinets. *Algorithmica*, 66(3):714–738. 2, 16, 18, 58

Huber, K., van Iersel, L., Kelk, S., and Suchecki, R. (2011a). Lev1generator. `http://www.uea.ac.uk/~x3002128/lev1generator/`. [Online; accessed 12-May-2015]. 63

Huber, K., van Iersel, L., Kelk, S., and Suchecki, R. (2011b). A practical algorithm for reconstructing level-1 phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):635–649. 2, 3, 10, 15, 57, 58, 64, 77, 78, 79, 82, 83, 84, 85, 86

Huber, K., van Iersel, L., Moulton, V., Scornavacca, C., and Wu, T. (2014a). Reconstructing phylogenetic level-1 networks from nondense binet and trinet sets. *arXiv preprint arXiv:1411.6804*. 105

Huber, K. T., Van Iersel, L., Moulton, V., and Wu, T. (2014b). How much information is needed to infer reticulate evolutionary histories? *Systematic biology*, page syu076. 106

Huson, D., Rupp, R., and Scornavacca, C. (2010). *Phylogenetic Networks. Concepts, Algorithms and Applications.* Cambridge University Press. 2, 5, 8, 10, 14, 19, 105

Iersel, L. and Moulton, V. (2013). Trinets encode tree-child and level-2 phylogenetic networks. *Journal of Mathematical Biology*, pages 1–23. ix, 15, 16, 42, 44, 104

Jansson, J., Lemence, R. S., and Lingas, A. (2012). The complexity of inferring a minimally resolved phylogenetic supertree. *SIAM Journal on Computing*, 41(1):272–291. 10

Jansson, J., Nguyen, N., and Sung, W. (2006). Algorithms for combining rooted triplets into a galled phylogenetic network. *SIAM Journal on Computing*, 35(5):1098–1121. 10, 15, 42

Jansson, J. and Sung, W.-K. (2006). Inferring a level-1 phylogenetic network from a dense set of rooted triplets. *Theor. Comput. Sci.*, 363(1):60–68. 2, 15, 18, 42

Jin, G., Nakhleh, L., Snir, S., and Tuller, T. (2007). Efficient parsimony-based methods for phylogenetic network reconstruction. *Bioinformatics*, 23(2):e123–e128. 9, 106

Jin, G., Nakhleh, L., Snir, S., and Tuller, T. (2009). Parsimony score of phylogenetic networks: Hardness results and a linear-time heuristic. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 6(3):495 –505. 9, 19, 20

Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562. 60

Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16(2):111–120. 35

Maddison, D. R., Schulz, K.-S., and Maddison, W. P. (2007). The tree of life web project. *Zootaxa*, 1668(Linnaeus Tercentenary: Progress in Invertebrate Taxonomy):19–40. 1

Maddison, D. R., Swofford, D. L., and Maddison, W. P. (1997). Nexus: an extensible file format for systematic information. *Systematic Biology*, 46(4):590–621. 103

Merican, I., Sherlock, S., McIntyre, N., and Dusheiko, G. (1993). Clinical, biochemical and histological features in 102 patients with chronic hepatitis c virus infection. *QJM*, 86(2):119–125. 88

Morin, M. M. (2007). *Phylogenetic Networks: Simulation, Characterization, and Reconstruction*. PhD thesis, University of New Mexico, Department of Computing Science. 19

Morin, M. M. and Moret, B. M. (2006). Netgen: generating phylogenetic networks with diploid hybrids. *Bioinformatics*, 22(15):1921–1923. 20

Morrison, D. (2015). The Genealogical World of Phylogenetic Networks. `http://phylonetworks.blogspot.co.uk/p/datasets.html`. [Online; accessed 7-May-2015]. 78, 87

O'Donnell, K., Kistler, H. C., Tacke, B. K., and Casper, H. H. (2000). Gene genealogies reveal global phylogeographic structure and reproductive isolation among lineages of fusarium graminearum, the fungus causing wheat scab. *Proceedings of the National Academy of Sciences*, 97(14):7905–7910. xiii, 78, 94, 95

Organisation, W. H. (2015a). Giardiasis Disease Information. `http://www.who.int/ith/diseases/giardiasis/en/`. [Online; accessed 12-May-2015]. 91

Organisation, W. H. (2015b). Hepatitis B Factsheet No 204. http://www.who.int/mediacentre/factsheets/fs204/en/. [Online; accessed 12-May-2015]. 88

Page, R. (2002). Modified mincut supertrees. In Guig, R. and Gusfield, D., editors, *Algorithms in Bioinformatics*, volume 2452 of *Lecture Notes in Computer Science*, pages 537–551. Springer Berlin Heidelberg. 10

Rambaut, A. and Grass, N. C. (1997). Seq-gen: an application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Computer applications in the biosciences : CABIOS*, 13(3):235–238. 32, 34, 72

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425. 46

Salemi, M. and Vandamme, A.-M. (2003). *The phylogenetic handbook: a practical approach to DNA and protein phylogeny*. Cambridge University Press. 78

Scitable, N. E. (2015). Recombination. http://www.nature.com/scitable/definition/recombination-226. [Online; accessed 10-Sep-2015]. 18

Semple, C. and Steel, M. (2000). A supertree method for rooted trees. *Discrete Applied Mathematics*, 105:147–158. 1, 10

Semple, C. and Steel, M. (2003). *Phylogenetics*. Oxford lecture series in mathematics and its applications. Oxford University Press. 9

Sessa, E. B., Zimmer, E. A., and Givnish, T. J. (2012). Unraveling reticulate evolution in north american dryopteris (dryopteridaceae). *BMC evolutionary biology*, 12(1):104. xiii, 78, 95, 96, 97, 98, 99

Starr, J. R., Gravel, G., Bruneau, A., and Muasya, A. M. (2007). Phylogenetic implications of a unique 5.8 s nrdna insertion in cyperaceae. *Aliso: A Journal of Systematic and Evolutionary Botany*, 23(1):84–98. xiii, 78, 100, 101

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160. 47, 49

Than, C., Ruths, D., and Nakhleh, L. (2008). Phylonet: a software package for analyzing and reconstructing reticulate evolutionary relationships. *BMC Bioinformatics*, 9(1):322. 2, 23

To, T.-H. and Habib, M. (2009). Level-k phylogenetic network can be constructed from a dense triplet set in polynomial time. *ArXiv e-prints*. 15

van Iersel, L. (2009). *Algorithms, Haplotypes and Phylogenetic Networks*. PhD thesis, Eindhoven University of Technology. 18

van Iersel, L., Keijsper, J., Kelk, S., Stougie, L., Hagen, F., and Boekhout, T. (2009). Constructing level-2 phylogenetic networks from triplets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(4):667 –681. 15

van Iersel, L. and Kelk, S. (2011). Constructing the simplest possible phylogenetic network from triplets. *Algorithmica*, 60:207–235. 10.1007/s00453-009-9333-0. 15

Yu, Y., Dong, J., Liu, K. J., and Nakhleh, L. (2014). Maximum likelihood inference of reticulate evolutionary histories. *Proceedings of the National Academy of Sciences*, 111(46):16448–16453. 106

Zhang Lab, U. o. M. (2015). What is FASTA format? `http://zhanglab.ccmb.med.umich.edu/FASTA/`. [Online; accessed 10-Sep-2015]. 103