Rearranging Phylogenetic Networks

Rearranging Phylogenetic Networks

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof.dr.ir. T.H.J.J. van der Hagen,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 26 mei 2021 om 12:30 uur

door

Remie Janssen

Master of Science in Mathematische Wetenschappen,
Universiteit Utrecht, Nederland,
geboren te Gendt, Nederland.

Dit proefschrift is goedgekeurd door de promotoren.

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. K.I. Aardal,	Technische Universiteit Delft, promotor
Dr.ir. L.J.J. van Iersel,	Technische Universiteit Delft, copromotor

Onafhankelijke leden:

Prof.dr. M. Fischer	Universität Greifswald, Duitsland
Prof.dr. V. Moulton	University of East Anglia, Verenigd Koninkrijk
Prof.dr.ir. M.J.T. Reinders	Technische Universiteit Delft
Dr. S.M. Kelk	Universiteit Maastricht
Prof.dr.ir. A.W. Heemink	Technische Universiteit Delft, reservelid

Overig lid:

Dr. M.E.L. Jones,	Technische Universiteit Delft

Dit onderzoek is deels gefinancierd door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (Vidi-beurs 639.072.602).

Keywords: Graph theory, Mathematical biology, Phylogenetics, Rearrangement moves

Printed by: GVO drukkers & vormgevers

Front & Back: Photo by P.A.M.E. Janssen,
Design by R. Janssen and S. Janssen

ISBN 978-94-6332-758-9

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Contents

Summary

Evolution plays an important role in biology, to such an extent that one of the best-known quotes about biology is Theodosius Dobzhansky's "Nothing in biology makes sense except in the light of evolution." To study evolution, it is important to have a structured and standardized way to represent hypotheses about evolutionary histories. This is where phylogenetic networks come in. These provide a mathematical and graphical representation of an evolutionary history as a graph.

Finding the most accurate phylogenetic network given some genetic data gives rise to many computationally hard problems. So, one often has to resort to heuristics. An important part of many of these heuristics is a local search through the space of phylogenetic networks; the aim is to find a good phylogenetic network by taking small steps through this space. These steps correspond to small changes made to a network, which are called *rearrangement moves*.

There is currently no standard type of rearrangement move, and each piece of software defines their own set of moves. When such software is published, they often mention the types of rearrangement moves they use. However, they rarely justify their choice of moves, even though this choice can have large consequences for the functionality of the heuristic. For example, to guarantee that an optimal network can be found, each network must be reachable from each other network by taking small steps through the space. In this thesis we study such problems, which are all aimed at answering the following question.

Which rearrangement moves can be used in local search heuristics?

To answer this question, we take a mathematical approach, where we use a graph to represent the space of phylogenetic networks—which are graphs themselves as well. A graph is a collection of nodes (points) connected by edges (lines), and in this graph, each node represents a network, and there is an edge between two networks if there is a rearrangement move that changes the one into the other. The requirement for a good move we mentioned before (each network must be reachable from any other network) can then be stated compactly in graph theoretical language as follows: Is the space of phylogenetic networks connected under a certain rearrangement move?

The main part of this thesis answers this question for a small set of rear-

rangement moves that are quite similar to moves that are used in practice. The general conclusion of this study is that most spaces are connected. And, as a result of the used techniques, we can additionally show that the number of steps between each pair of networks is relatively small compared to the number of networks. This is a nice property for the use of these rearrangement moves in local search heuristics, as it shows that an optimal network can (in principle) be found quickly if the right moves are chosen.

The computational hardness of the reconstruction problems unfortunately implies that choosing the right moves is hard as well. This also holds for another computational problem we study in this thesis: finding the shortest sequence of rearrangement moves between two networks. We show that several versions of this problem are NP-hard. This implies that, given two networks, there is no fast way to find a rearrangement move that modifies one network so that it becomes more like the other network.

Finally, in the discussion, we apply our results to published reconstruction software. As mentioned, most of these publications do not study their search spaces, so it needs to be checked that, at the very least, these search spaces are connected. As the moves used in the software are similar to the moves studied in this thesis, we can apply our results to the search spaces used in the software. Fortunately, we conclude that, with one exception, all these search spaces are connected. This solidifies the theoretical basis of these methods, and justifies their application to biology.

Samenvatting

Evolutie speelt een belangrijke rol in de biologie. Een van de bekendste uitspraken over biologie zegt zelfs dat je evolutie nodig hebt om biologie te kunnen begrijpen: "Nothing in biology makes sense except in the light of evolution" – Theodosius Dobzhansky. Daarom is het belangrijk dat we een gestructureerde en gestandaardiseerde manier hebben om hypotheses over evolutie weer te kunnen geven. Dit is waar fylogenetische netwerken het toneel betreden: deze wiskundige structuren worden gebruikt als (grafische) representatie van mogelijke evolutionaire geschiedenissen.

Het reconstrueren van de echte evolutionaire geschiedenis komt dan neer op het vinden van het fylogenetische netwerk dat het beste bij de (genetische) data past. Dit geeft ons computationele problemen, die doorgaans moeilijk zijn om op te lossen; ze zijn vaak NP-moeilijk. Het is daarom vaak nodig om heuristieken te gebruiken. Een belangrijk onderdeel van deze heuristieken is een lokale zoektocht naar een goed netwerk: hiervoor beschouwen we de zoekruimte (alle mogelijke fylogenetische netwerken) als een graaf genaamd de ruimte van fylogenetische netwerken, en nemen we kleine stappen door deze ruimte. Deze stappen, die we *herschikkingsstappen*(rearrangement moves) noemen, corresponderen met kleine veranderingen in een netwerk.

Er is momenteel geen gestandaardiseerde definitie voor deze herschikkingsstappen. Iedere softwaretool gebruikt zijn eigen definitie. Bij het publiceren van zulke tools wordt doorgaans geen aandacht besteed aan deze keuze, terwijl hij van grote invloed kan zijn op de werking van de geïmplementeerde heuristiek. Het kan bijvoorbeeld onmogelijk zijn om een netwerk in een ander netwerk te veranderen met een gegeven type herschikkingsstap. In dat geval kan het ook onmogelijk zijn om het beste netwerk te vinden gebruik makend van alleen dit type herschikkingstappen. Daarom bestuderen we in dit proefschrift een aantal herschikkingsstappen en de bijbehorende ruimtes van fylogenetische netwerken. We trachten in het bijzonder om de volgende vraag te beantwoorden.

Welke herschikkingsstappen zijn geschikt voor het gebruik in heuristieken?

Om deze vraag te beantwoorden gebruiken we wiskundige technieken uit de grafentheorie. Een ruimte van fylogenetische netwerken is voor ons een graaf, waar iedere knoop een fylogenetisch netwerk voorstelt. De herschikkingstap-

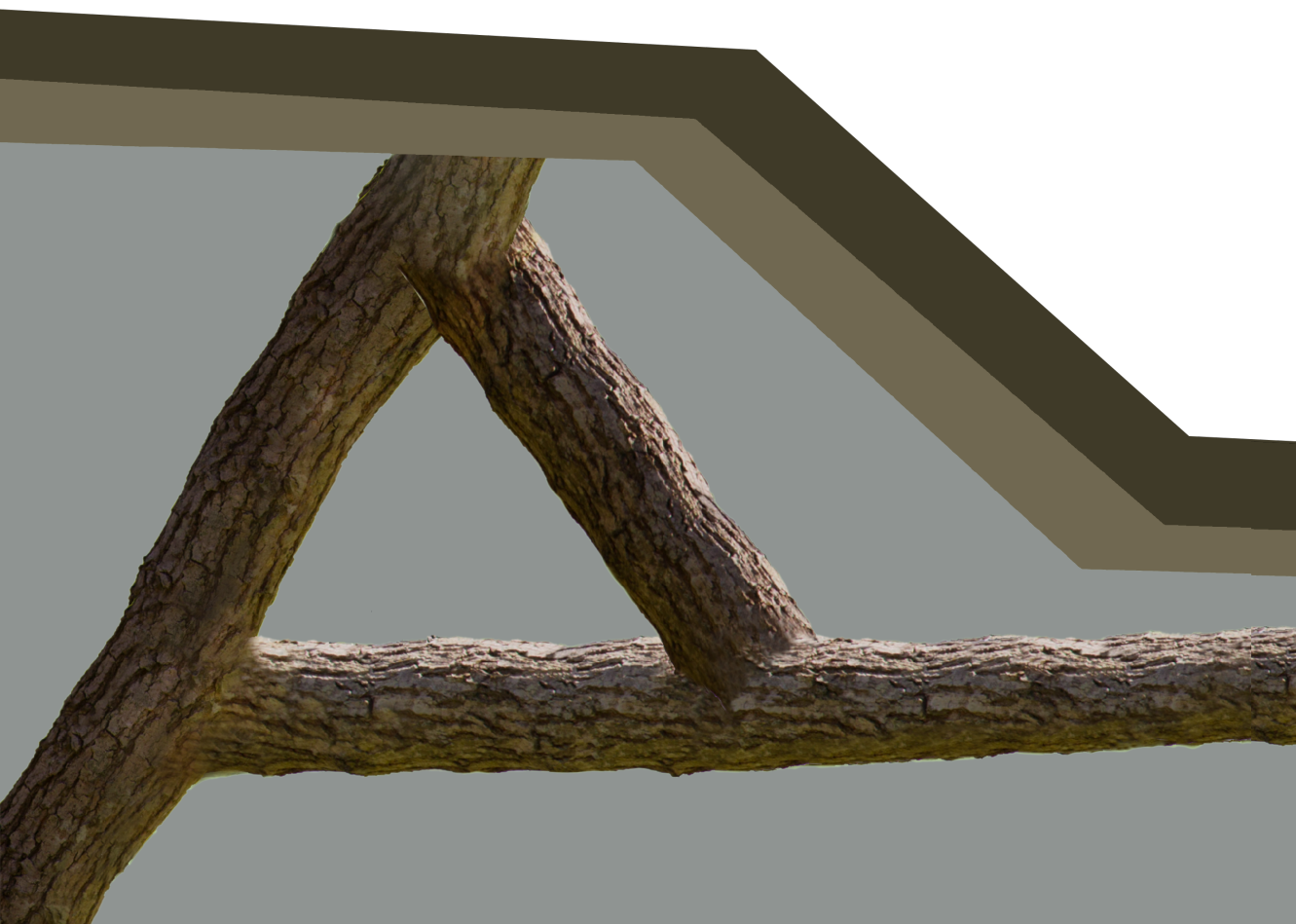
pen worden gecodeerd door de lijnen in deze graaf: er is een lijn tussen twee netwerken precies wanneer het ene netwerk in het andere kan worden veranderd in één herschikkingsstap. Zoals eerder genoemd, is het belangrijk om ons af te vragen of we met een bepaald type herschikkingsstap ieder netwerk in ieder ander netwerk kunnen veranderen. Deze vraag kan compact uitgedrukt worden in de taal van de grafentheorie: Is de ruimte van fylogenetische netwerken verbonden als graaf?

Het grootste deel van dit proefschrift is gericht op het beantwoorden van deze vraag voor verschillende herschikkingsstappen, die erg vergelijkbaar zijn met de herschikkingsstappen die in de praktijk gebruikt worden. Over het algemeen concluderen we dat de ruimtes van fylogenetische netwerken verbonden zijn voor deze herschikkingsstappen. Daarbij zijn de technieken die we gebruiken om dit te bewijzen constructief. Dit betekent dat we daadwerkelijk een reeks herschikkingsstappen kunnen vinden tussen twee gegeven netwerken, en dat we de afstanden tussen netwerken kunnen afschatten. Deze afstanden blijken relatief klein ten opzichte van het aantal fylogenetische netwerken in een gegeven ruimte. Dit is een fijne eigenschap in de praktijk, omdat het betekent dat het beste netwerk in principe altijd in een klein aantal stappen gevonden kan worden.

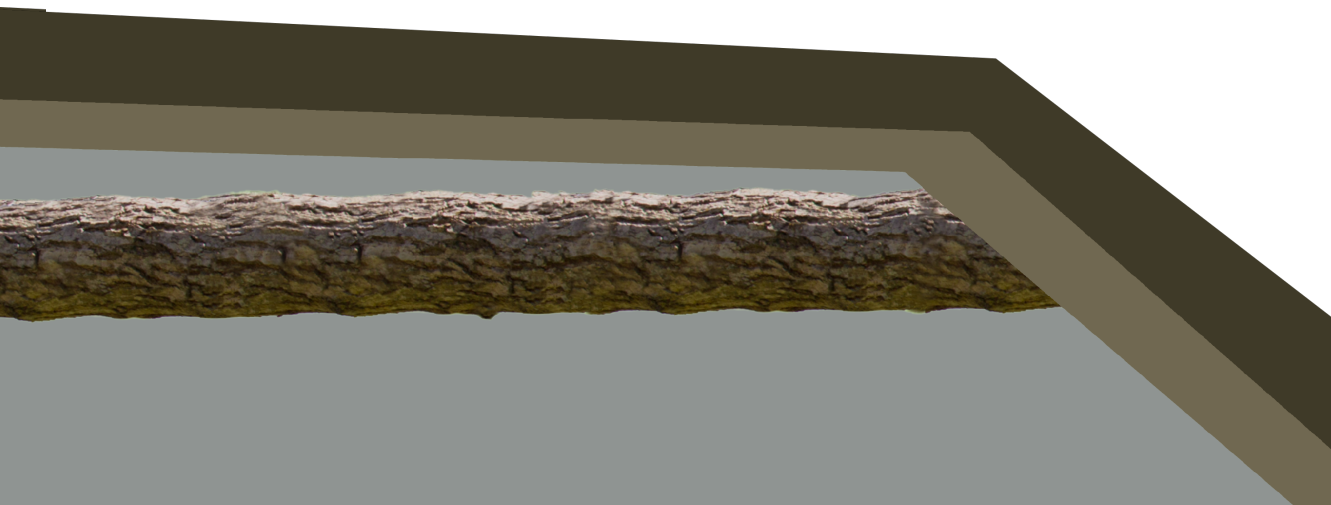
Helaas kunnen we niet makkelijk zo'n korte reeks stappen vinden. Dit is omdat het vinden van het beste netwerk vaak NP-moeilijk is. Een ander NP-moeilijk probleem is het vinden van de korste reeks stappen tussen twee netwerken. We bewijzen in dit proefschrift dat dit probleem daadwerkelijk NP-moeilijk is voor een aantal types herschikkingsstappen. Dit betekent dat, hoewel we een afstand tussen twee netwerken kunnen definiëren als het minimale aantal stappen tussen deze netwerken, deze afstand niet gemakkelijk te berekenen is. We geven, op basis van onze bewijzen van verbondenheid, wel een aantal heuristieken voor het bepalen van deze afstanden. Het zal blijken dat deze heuristieken in veel gevallen een redelijk korte reeks stappen kunnen produceren.

Afsluitend, in de discussie, beschouwen we ruimtes van fylogenetische netwerken die voorkomen in gepubliceerde software tools. We gebruiken daar onze resultaten om te controleren of aan de minimale eis voor een goede zoekruimte voldaan wordt, verbondenheid. Omdat we in dit proefschrift herschikkingstappen bestuderen die erg lijken op de herschikkingsstappen in deze software tools, kunnen we dit gemakkelijk staven. Gelukkig kunnen we concluderen dat de meeste van deze zoekruimtes verbonden zijn, op een enkele na. Dit geeft een extra theoretische verantwoording van het gebruik van deze software. Dit proefschrift versterkt dus de fundering van het biologisch onderzoek dat gebruik maakt van deze heuristieken voor fylogenetische netwerken.

1



Introduction



1.1 What are phylogenetic networks?

Phylogenetic networks are a type of graph used in biology, to represent evolutionary history. The most common shape for these networks is a tree. Trees have a long history in biology. This starts with their use in taxonomy, where they became popular in the eighteenth century [Rag09], but examples from as early as 1592 exist as well [Zal40]. These trees had nothing to do with evolution, taxonomic trees simply represented a classification of (living) things.

One of the first examples of evolutionary trees can be found in the book "Philosophie Zoologique" by Jean-Baptiste Lamarck in 1809. However, the most well-known early examples are by the hand of Charles Darwin, who laid the basis for the currently accepted theory of evolution. For a more complete overview of the history of trees in the representation of evolutionary history, see, for example, [Arc14].

Modern evolutionary trees, also called *phylogenetic trees*, show a branching pattern that corresponds to the branching pattern of evolution caused by speciation. Such trees are often interpreted both as taxonomies and as phylogenies. This dual interpretation of a phylogeny as a taxonomy breaks down when additional *non-vertical processes*, such as *hybridization* [e.g. AAA+13], *horizontal gene transfer* (HGT) [e.g. ZD11, KGDO05, KP08], and *recombination* [e.g. VB15] are involved as well.

With such additions, evolutionary histories become reticulate (i.e., net-like), so they can no longer be represented by trees, but only by *phylogenetic networks*. In such networks, there is no clear hierarchical grouping of the taxa as in a tree. Hence, unlike a phylogenetic tree, a phylogenetic network cannot simply be read as a taxonomy, although some taxonomic information may still be extracted, for example by studying *clusters* [NW05, KNTX08, HRS10, Ste16]. The main use of phylogenetic networks is therefore as a representation of evolutionary history.

Phylogenetic trees and networks represent evolutionary histories by showing the flow of hereditary information. In biological applications, this is most often in the form of genetic information. There are also applications outside of biology such as in linguistics [e.g. Dun15, JL19, LS20] and other anthropological topics like board games [e.g. Kra00, Car14, BSP+19] and archaeology [Pre19], where, for example, the evolution of tools is subjected to phylogenetic analysis [e.g. Hou12, OBB+14, WPR19]. In those cases, it is less clear which flow of information is represented in the network exactly, and these types of information may not behave similar to genetic information, which makes accurate

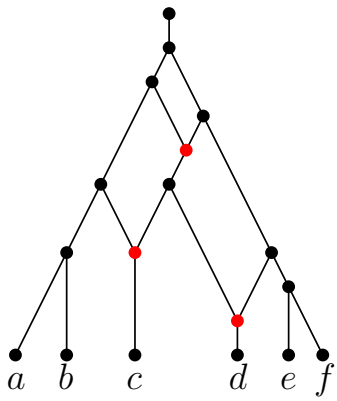


Figure 1.1: A phylogenetic network with six leaves (representing extant taxa) at the bottom, and the root (ancestral taxon) at the top. Edges are directed downwards, showing the passing of time. The red nodes are the three reticulations (i.e., reticulate evolutionary events), which make this network a tier-3 network.

reconstruction of these phylogenies challenging [Mor13, Str19]. Nevertheless, in all these cases, phylogenetic trees or networks are assumed to represent some kind of evolutionary history.

In its broadest mathematical sense, a phylogenetic network can be thought of as a leaf-labelled graph, usually without parallel edges and degree-2 nodes (Figure 1.1) [Mor11, HRS10]. The underlying graphs of the networks may be directed (and acyclic) or undirected. Between these, directed networks have the simplest interpretation as evolutionary histories (Figure 1.2). In a directed tree, the arcs represent periods of descent with modification, and the nodes represent speciation/divergence events. In a directed network, there is a third type of node, a reticulation node. Such a node represents the combination of hereditary information like in hybrid speciation.

Undirected networks often only represent genetic data, but, in some cases, they may be though of as the undirected version of a directed network, in which we simply ignore or are ignorant of the direction. These two types of networks are sometimes confused, leading to controversy: [FFRF20] uses a median joining network (MJN; a data displaying network) and reads it as an evolutionary history, as [SPKPS$^+$20] point out. This paints a sufficient, albeit strongly simplified, picture of the interpretation of phylogenetic networks as evolutionary histories, to which we will get back in the Section 8.2.1 of the Discussion.

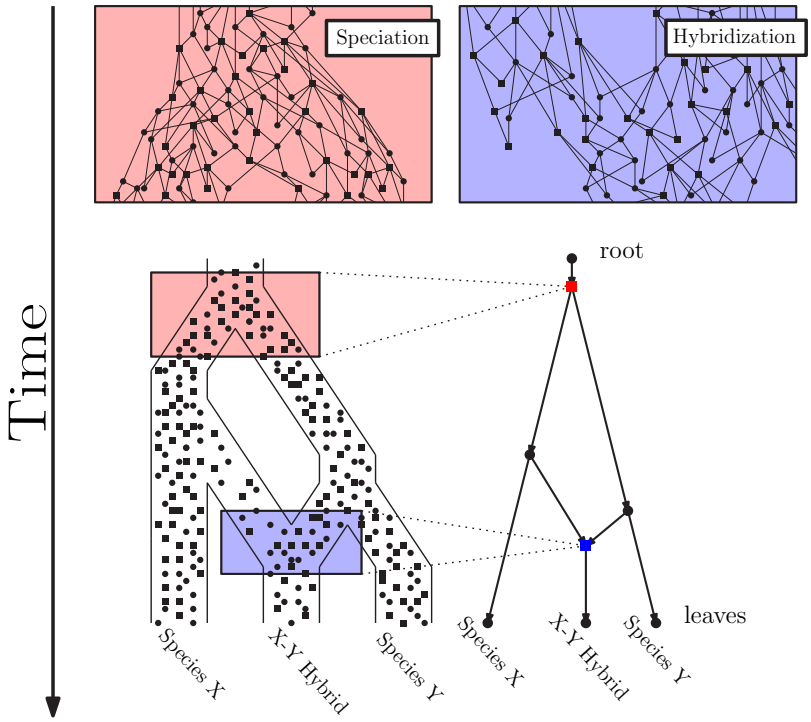


Figure 1.2: Schematic and simplified visualization of how phylogenetic networks represent evolutionary history. The squares and circles on the top and left represent males and females respectively in a sexual species, and the arcs connecting these individuals represent parent-child relations. Red: Speciation, represented by a tree node in the network; Blue: Hybridization, represented by a reticulation node in the network.

1.2 What is their use?

As mentioned in the previous section, phylogenetic trees have a dual use in taxonomy and evolution, which breaks down for networks. So, if networks cannot directly be used for classification, what reasons remain for the construction of phylogenetic networks?

Evolutionary histories are, among other things, needed to find the reservoir or initial infection for some disease [e.g., GBR+99, LCK+16], to construct accurate pictures of what ancestral species looked like (i.e., ancestral state reconstruction) [Fit71, YHBH15, HF17], to make decisions in conservation using phylogenetic diversity indices [e.g. Mag13]— which have been extended to net-

works as well [VMM+14, WF17, WF18]— or to learn about the evolution of genes, giving us insight in how they work [e.g., SKS94, YHBH15, JLM+16, GANA+17, ATD18].

The obvious question is why we should use networks instead of the well-established trees for these applications. The main reason is that evolutionary histories cannot all be represented by trees because of non-vertical processes. Arguments in this direction go as far as to relabel the tree of life as the *tree of one percent* based on the limited amount of vertical inheritance [DM06]. Clearly, if the evolutionary history cannot be represented by a tree, we should not even attempt to construct a tree that represents it.

On the other hand, there are examples where the vertical inheritance is clearly the most important mode of inheritance, for example, when we are interested in the phylogenetic relations between distantly related mammals. In such cases, it could be better to search for a tree representation. To determine whether one should search for a tree or network, [BA20] recommend first using separate methods that detect non-vertical inheritance, such as the ABBA-BABA test [DPRS11] or HyDe [BCWK18]. In the end, the authors argue that neither tree nor network should be rejected in favor of the other, as the analyses can complement each other, for example by detecting ancient reticulations in snakes using network methods, and estimating time-scales and geographical areas using well-established tree methods [BG18].

As mentioned in the previous section, phylogenetic methods are also used in fields other than biology. In these fields, it may also be right to assume networks are needed to represent the evolutionary histories. However, the debate about the use of trees or networks does not seem to have arrived to these fields yet. This is probably because the use of phylogenetic methods in these fields is not as well established as in biology. This is exemplified by [JL19], who argue for the use of phylogenetic methods in linguistics and, at the same time, debate the use of trees versus networks.

1.3 How do we find them?

Phylogenetic networks represent past events, which we cannot observe directly. This means we can only infer evolutionary histories based on present information. For the reconstruction of evolutionary histories, this implies we must primarily use information obtained from extant taxa, in some cases perhaps supplemented by data collected from fossils. Extant taxa provide information in the form of DNA sequences, which are found by sequencing the genome of individuals in a given taxon.

These sequences are typically preprocessed before they are used to find a phylogeny. The first step is nearly always to *align* the sequences, that is, to link the positions of the sequences of different taxa that have the same evolutionary origin. The resulting alignment can then be used directly in some reconstruction tools, whereas others require further preprocessing. These tools may, for example, require pairwise distances between the sampled taxa, which can be calculated by counting the pairwise differences of the sequences in the alignment [e.g. BS16, BHMS18, vIMM20].

Another often used form of preprocessed data are multi-locus sequences. These are obtained by either sampling the genome at different loci, or by partitioning the alignment of a contiguous sequence into blocks that are inherited as a unit (see the introduction of [JGvI+19] for an overview of such methods). These blocks can then be used directly to find a network [e.g. YDLN14, SLA16], or further processed into *gene trees*: a phylogenetic tree for each block.[1] For overviews of methods for tree building, see [KYT20, LSV09, SDG20]; and for a recent overview of network methods, see [EOZN19].

After preprocessing, the data can be used to reconstruct the evolutionary history. This reconstruction can broadly take two forms. It either consists of finding a distribution of networks using a Bayesian approach, where networks that explain the data better have a higher probability; or it uses an optimization problem where the goal is to find a simplest network that explains the data or a network that explains the data best. We now give a few examples of such optimization problems.

If the data consists of distances between pairs of taxa, the optimization problem may ask for the simplest network in which each distance corresponds to the length of a certain type of path between the taxa [e.g. BS16, BHMS18, vIMM20]. For data given as a set of gene trees, it may consist in finding the simplest network which contains all these trees [BGMS05, LS19, WBZ13, vIJJ+19b]. A network being 'simple' can have different meanings, although it often refers to the number of reticulation events, where fewer reticulation events gives a simpler network. Finding a network with as few reticulations as possible may be thought of as a parsimony based method (being parsimonious with respect to the number of non tree-like events), but parsimony methods typically refer to the next kind of reconstruction method.

Parsimony in phylogenetic tree reconstruction refers to a specific problem, our next example, where one wants to find a tree which explains the sequence evolution with as few mutations as possible. This model has recently been ex-

[1] The name gene tree does not accurately reflect what blocks are/should be chosen. Indeed, as a result of recombination or domain reshuffling, only a small part of a gene (like a domain or an exon) inherits as a unit [e.g. KZNL02, VTPL05]. Hence, it is unlikely that a complete gene is always inherited as a unit

tended to phylogenetic networks as well [FvIKS15, VanIJS17]. Although these network methods are based on parsimony, the term parsimony for network reconstruction is increasingly reserved for methods that minimize deep coalescence [YBN13, YCLN20]. These methods are parsimonious with respect to some aspects of embeddings of gene trees in networks. We will use parsimony exclusively for the methods that are parsimonious with respect to mutations, and we will refer to the latter simply as methods that minimize deep coalescence.

The last example is the *maximum likelihood* (ML) problem where one takes an alignment and searches for a network (with arc lengths) that has the highest *likelihood*, i.e., probability of producing the given alignment [e.g. YDLN14, SLA16]. Of course, to define this likelihood, one needs a model of sequence evolution. Hence, such reconstruction problems are also called *model based* methods in the literature.

1.3.1 Heuristics

As most of the optimization problems are computationally hard, it is often infeasible to find an optimal solution [e.g. FG82, CT06, Roc06, BFLS17]. To find a good solution within a reasonable time, one needs to use heuristics. One type of heuristic used for these problems is a *local search* heuristic [trees: Fel04, LVDMH$^+$08, NSvHM14, LCK$^+$16][networks: YDLN14, WYHN16]. Instead of considering all possible solutions as a set, these methods use a space of solutions, where similar solutions are close to each other (Figure 1.3). One then attempts to find a good solution by making small steps through this space.

When an underlying model of sequence evolution is used, one can also opt for a Bayesian approach instead of an optimization approach [see, e.g., Lar20, for an overview]. To use such methods, one needs a stochastic model of (sequence) evolution, an alignment, and a *prior distribution* on the set of phylogenetic networks. The aim of the Bayesian method is to update the prior distribution with the information from the alignment to obtain a posterior distribution. To reach this goal, heuristics are often employed here, too. In contrast to the local search heuristics for optimization problems, these heuristics are not meant to find an optimum, but to sample the *posterior distribution*. The estimate for the posterior distribution is then simply the distribution of the samples.

Sampling is performed by randomly walking through the space of networks using a *Markov Chain Monte Carlo* (MCMC) Metropolis-Hasting algorithm (e.g., [WYN16]). In this algorithm, one performs a random walk through the space of networks. This random walk *proposes* a neighbour of the current network, and accepts this with a probability that depends on the prior distribution and the data. Choosing this probability so that it is proportional to the proba-

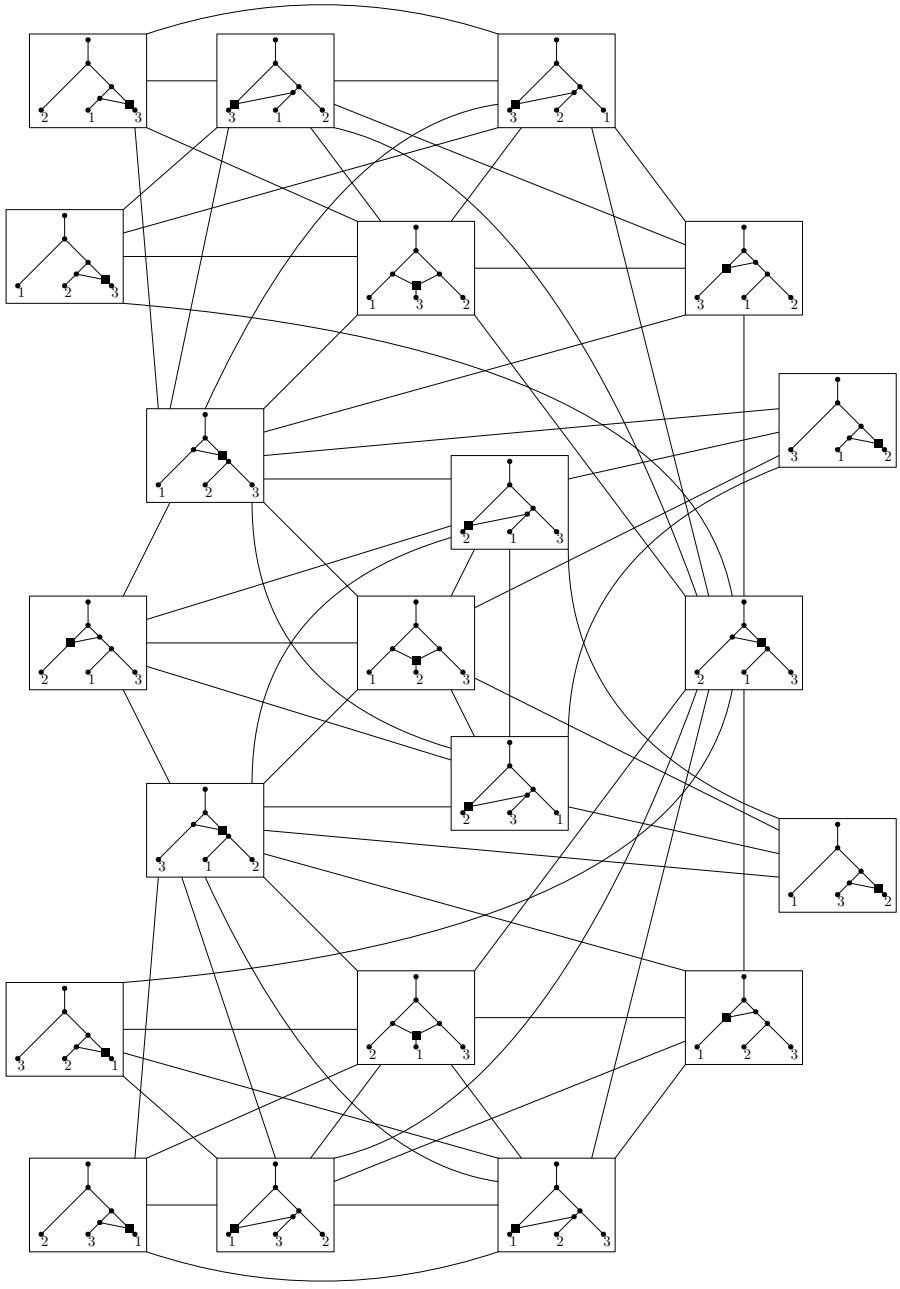


Figure 1.3: The space $\mathcal{N}_{tail}(3,1)$ of networks with three leaves and one reticulation, where there is an edge between two networks if they can be transformed into each other by a small change called a *tail move*.

bility in the posterior distribution, the estimation of the posterior distribution simply consists of the frequencies with which the networks are visited.

Using Bayes' rule, the posterior probability of a network together with a parameter set is proportional to

$$\mathbb{P}(\text{data}|\text{network, parameters})\mathbb{P}(\text{network, parameters}),$$

where the first probability depends on the chosen evolutionary model, and the second is simply the probability of the network with parameters in the chosen prior distribution. For a more complete explanation of such Bayesian methods for phylogenetics, see for example [Lar05, EOZN19].

1.3.2 Rearrangement moves

In all these heuristics, one needs to define the steps that can be taken through the space. For phylogenetic trees, these steps are called *rearrangement moves*. Several of these moves have long been studied for phylogenetic trees. The most prominent ones are Nearest Neighbour Interchange (NNI), Subtree Prune and Regraft (SPR), and Tree Bisection and Reconnection (TBR) [Fel04, SS03]. All of these moves take one edge, and move one or both endpoints to other locations in the tree. To use local search heuristics for phylogenetic networks, one needs rearrangement moves that work for networks as well.

A handful of rearrangement heuristics for phylogenetic networks have been published recently, and each of them uses its own set of rearrangement moves. For example, the PhyloNet method InferNetwork_ML [YDLN14] uses the "Relocating the source of an edge" and "Relocating the destination of a reticulation edge" moves, and the BEAST 2.5 add-on SpeciesNetwork [ZODS18] uses the "Branch relocator" operation. These sets of moves are often quite similar; for example, the above-cited moves from [YDLN14] respectively move the head and the tail of an arc of the network, and the SpeciesNetwork move is a combination of these moves where the moving arc is allowed to be redirected as well.

Papers introducing these heuristics typically do not study the properties of the moves they use, even though it is important to check some properties of the corresponding spaces of networks. For example, for the heuristics to reach an optimum or posterior distribution, the corresponding spaces need to be connected. And for the heuristics to work efficiently, it is important to choose a set of moves that guarantees each network can be reached using a small number of moves. Hence, researchers have become interested in defining basic rearrangement moves for phylogenetic networks and studying their properties [BLS17, FHMW17, GvIJ+17a, HMW16].

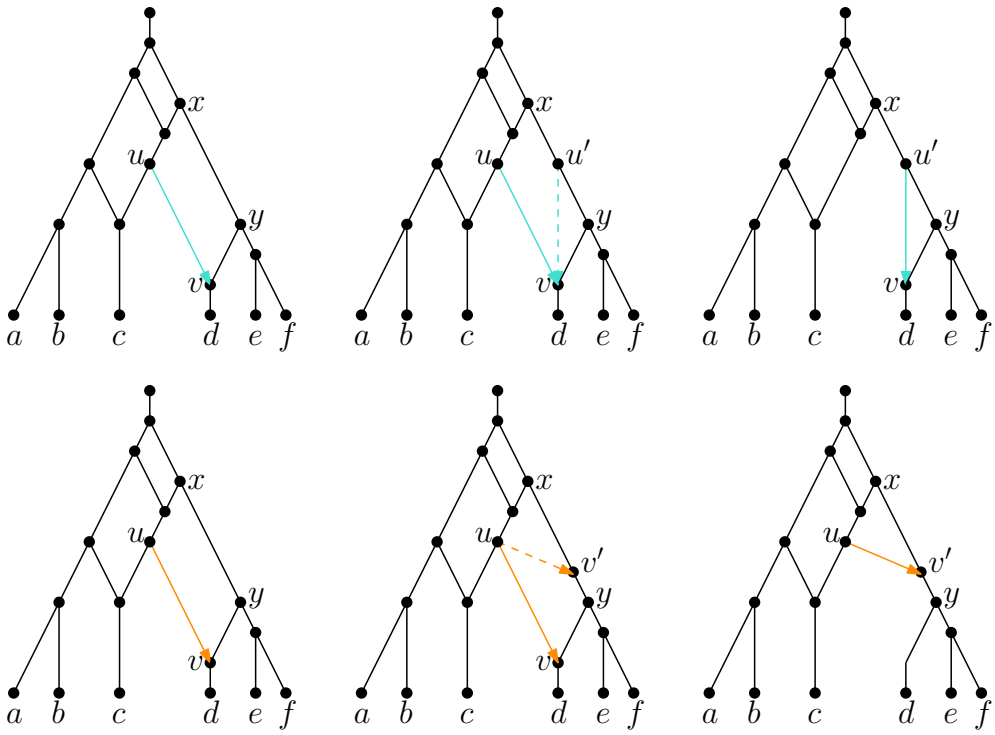


Figure 1.4: Top: the tail move (u, v) to (x, y); Bottom: the head move (u, v) to (x, y). On the left, the starting networks in which the moving edges are coloured. The right networks are the resulting networks after the moves, with the moved edge coloured differently. The middle graph is a combination of the left and the right network, with the moving edge coloured differently. The solid coloured edge is the moving edge of the network before the move, the dashed coloured edge is the moving edge of the network after the move. We distinguish the moves with edge colours: blue is a tail move, orange is a head move.

Huber et al. [HMW16] generalized NNI moves to undirected phylogenetic networks, and showed the connectivity under these moves of the *tiers* of phylogenetic-network space, i.e., phylogenetic networks having the same number of reticulations. Other generalisations of tree moves that have been proposed include tail moves and head moves, which are moves that relocate the tail or head of an arc (Figure 1.4). For example, one *rSPR move* [GvIJ+17a] on a network consists of one head move or one tail move, and one *rNNI move* consists of one head move or tail move that relocates an arc to an adjacent arc. *SNPR moves* [BLS17] are a variation on this theme: they are defined as a

tail move or the deletion/addition of an arc on networks where parallel edges are allowed. Because of this deletion/addition of an arc, SNPR moves can change the reticulation number. Moves that do this are called *vertical moves*, as they allow us to move up or down a tier; moves that do not change the number of reticulations are called *horizontal moves*. In this thesis, we study such horizontal moves.

1.3.3 Internal labels

Previous studies of the properties of rearrangement moves have primarily focused on networks in which only the leaves of the network are labeled. In this thesis, we consider networks where internal nodes can be labeled as well—this allows for the placement and labeling of ancestral taxa in the network. The data for these taxa could, for example, be obtained from fossils [e.g., SGF15, GWSD14], or from data simply sampled through time such as for pathogens in ongoing epidemics [e.g., transmission trees in YvBW13, GWSD14] or for cancer development within one patient [e.g., JVD$^+$14].

Internal labels for trees have recently been introduced in different mathematical phylogenetic contexts, like clusters [BDEM$^+$20, JBZ20] and the combination of trees [FBL20]. Bernardini et al. [BBDVP19] also considered fully labelled phylogenetic trees. Their application was to cancer research, where, sampling occurs throughout the tree, instead of only at the leaves. Most notable about this paper is that it uses rearrangement moves on internally labeled trees. To our knowledge, no research has been done to investigate rearrangement moves on internally labeled phylogenetic networks.

1.4 Thesis scope

In this thesis, we study rearrangement moves and the corresponding spaces of phylogenetic networks. The existing literature studying these topics gives a fragmented view of the connectedness and diameters of phylogenetic network spaces. For example, before the papers used for this thesis, there were only diameter bounds for a few types of moves. Here, we aim to give a complete characterization of the connectedness and diameters of these spaces, where we restrict to networks without parallel edges. We also consider networks with internal labels. We focus on spaces defined by horizontal moves only, and we do not consider *classes* of phylogenetic networks (e.g., tree-child, tree-based, or reticulation visible networks). These are all major differences with the recently published thesis by Jonathan Klawitter [Kla20b], wish also studies spaces of phylogenetic networks.

Our results contain complete characterizations of the connected components of the spaces of (internally labeled) phylogenetic networks for tail moves, head moves, rSPR moves, distance-1 tail moves, rNNI moves, SPR moves, and NNI moves. For all these spaces, we also prove diameter bounds, some of which are asymptotically tight.

The connectedness results in this thesis are all constructive, in the sense that they provide a method to find a sequence of moves between any pair of networks in the same component, so each of these proofs gives an upper bound on some diameter. Hence, this thesis is essentially a compendium of diameter (upper) bounds for phylogenetic network spaces, and the proofs form the main part of this thesis.

1.4.1 Structure of the thesis

To structure these results, we study each type of rearrangement move together with its local variants in a separate chapter. This structure deviates from the structures of the papers this thesis is based on [JJE+18, Jan21, JK19]. The results for each move were scattered across the papers, which did not exclusively concern one type of move. To make the original proofs easy to find, each previously published result is accompanied by a reference to the original. Many results do not have such an accompanying reference; those results are new. We will now present the structure of this thesis, with indication of the origin of different parts of the thesis.

Chapter 2, Preliminaries This chapter contains all relevant definitions and specific notation. It starts with basic notation for graphs, and builds up complexity by continuing with the definition of phylogenetic networks, then rearrangement moves, and then spaces of networks. These definitions are the same as the commonly used definitions, although the notation may be different. For a complete list of notation, see the Symbol Index at the end of this thesis.

Chapter 3, Tail Moves The first move type we consider is the tail move. All connectedness results for networks without internal labels are taken from [JJE+18]. The notation in these proofs has been updated, and some proofs have been rewritten to improve the exposition. This led to an improved upper bound for the space of networks under tail moves. All other results in this chapter are new (i.e., lower bounds, and connectedness of spaces of internally labeled networks).

Chapter 4, Head Moves The second type of move we consider is the head move. All results about networks without internal labels are based

on [Jan21]. For these networks, only the upper bound for local head moves is new, as the paper did not calculate this bound based on the constructive proof. Like for the tail move results, the notation has been modified, and proofs have been rewritten to increase readability and to improve the bounds. Also like in the tail move chapter, all other results pertaining to internally labeled networks are new.

Chapter 5, rSPR and rNNI Moves Next, we consider the union of tail and head moves, i.e., the rSPR move. This chapter starts with a section that studies the relation between tail and head moves in networks with at least two leaves, taken from [Jan21]. Then, it continues with constructive connectedness proofs for rSPR spaces, based on [JJE+18], and rNNI spaces, based on [JK19]. Like for the previous two chapters, all these results are updated, and the results about internally labeled networks are new.

Chapter 6, SPR and NNI Moves In this chapter, we study spaces of undirected networks. We heavily leverage the results for directed networks. Hence, the chapter starts with a section about the relation between spaces of directed and undirected networks. This section is partly based on results from [JJE+18] and partly new. Then, we turn to the connectedness question. The connectedness results for SPR moves are taken from [JJE+18], and the results for NNI moves are from [JK19]. The results concerning internal labels are new.

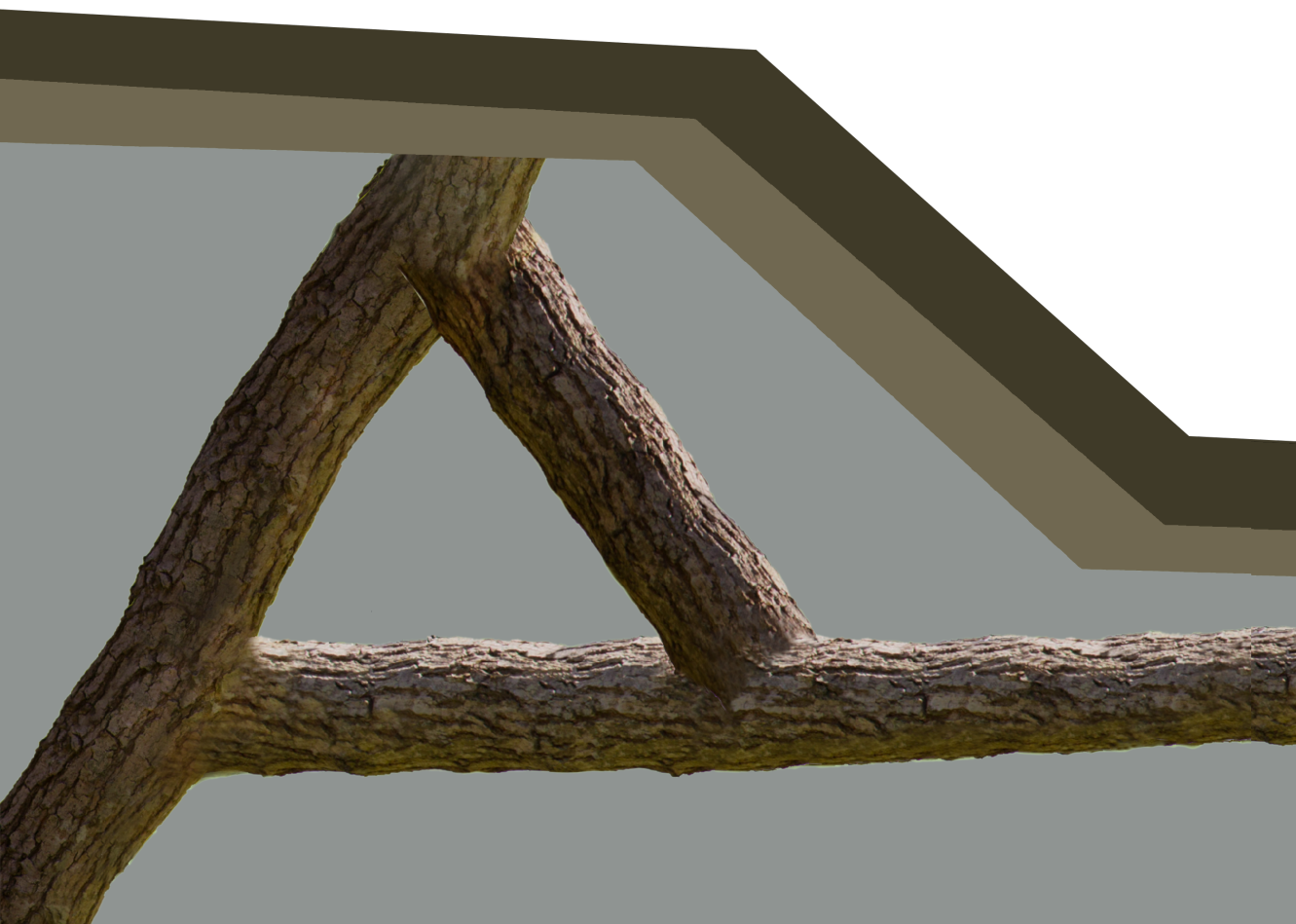
Chapter 7, Computing Sequences All constructive proofs can, in principle, be converted into algorithms that compute a sequence of moves between a given pair of networks. In this chapter, we show that computing the shortest such sequence is NP-hard for each move. Then, we explicitly convert some of our proofs into algorithms and we test the quality and practical running time of python implementations of these algorithms. The complexity results are existing results, mostly based on [JJE+18] and [Jan21], but all algorithmic and experimental results are new.

Chapter 8, Discussion In this last chapter, we discuss all our results and place them in a more applied context. First, we give a short summary of the results, including a tabulated overview of all diameter bounds. Then, we reconsider the use of a network as a representation of (biological) evolutionary histories, and we discuss our assumptions used to define a network. Then, we apply our results to several software tools, showing that for most of these tools our results are sufficient and necessary to show that their search spaces are connected.

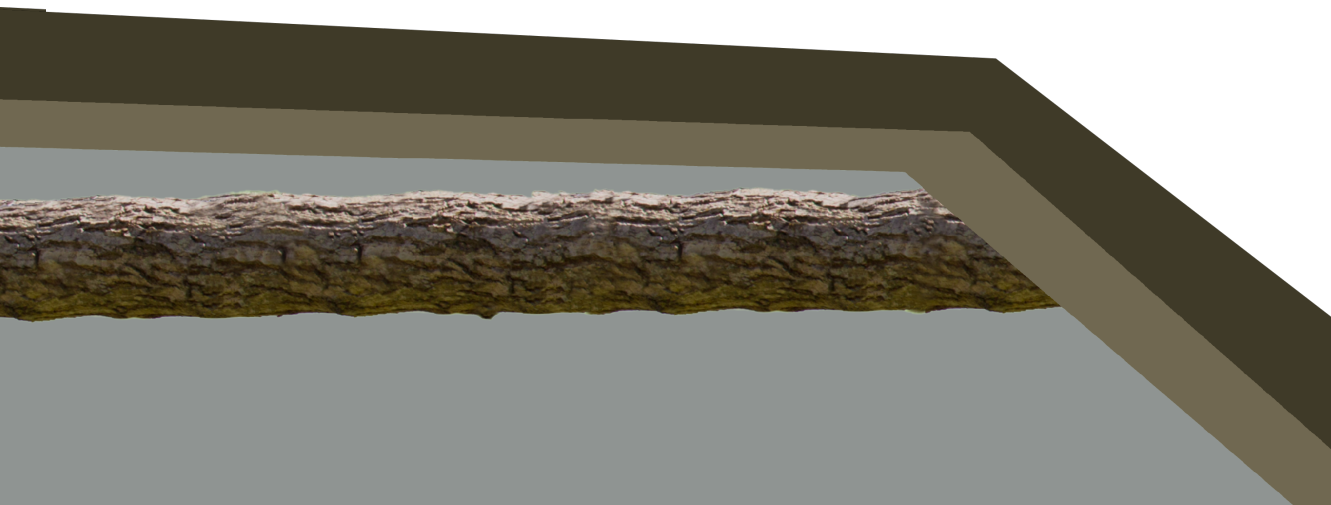
Chapter A, Rearrangement Moves in Software This appendix contains the actual survey of the literature for software that uses rearrangement moves mentioned in Chapter 8. We select several software tools and compare the rearrangement moves used in these tools to the moves used in this thesis. In particular, we check for each of these tools whether the search space is connected, and, if so, whether they are still connected when restricted to networks with the same number of reticulations.

Chapter B, Open Problems Although the discussion (Chapter 8) already contains several open problems, they are scattered throughout this discussion. In this appendix, we give a more condensed and comprehensive list of open problems related to this thesis.

2



Preliminaries



This chapter provides an overview of the specific notation and definitions used throughout this thesis. It also includes some relevant basic results from mathematical phylogenetics. For standard notation, e.g., writing $\mathbb{N}$ for the natural numbers, refer to the Symbol Index at the end of this thesis.

We assume a basic knowledge of some mathematics such as complexity theory, including NP-hardness and big-O notation. However, we will repeat some graph theory basics, as graph theoretical notation is pivotal for the definition of phylogenetic networks.

To define the main topic of this thesis, we start with a general introduction for graphs. This is followed by an increasingly specialized introduction for phylogenetic networks and their properties, culminating in the introduction of rearrangement moves and the corresponding spaces of networks.

2.1 Graphs

This thesis concerns specific types of graphs, so we start with a general introduction of graph theoretical concepts. We mostly follow standard notation for graphs, such as can be found in [Die12].

2.1.1 Undirected graphs

An *undirected graph* $G = (V, E)$ consists of a set, V, of *vertices* (or *nodes*), and a set $E \subseteq \binom{V}{2}$ of *edges*. Note that this definition precludes the existence of *parallel edges*: two edges are *parallel* if they share both endpoints.

We write $V(G)$ for the nodes of a graph G, and $E(G)$ for its set of edges. Two nodes x and y are *neighbours* in G if G contains the edge $\{x, y\}$, and we say that x (resp. y) and $\{x, y\}$ are *incident*. The *degree* of a node is the number of edges it is incident to.

A $\{1, 3\}$-*graph* is a simple graph with nodes of degree 1 and degree 3. Similarly a $\{1, 2, 3\}$-*graph* is a simple graph with nodes of degree 1, degree 2, and degree 3. A graph with nodes of degree-3 only is called a *cubic graph*.

A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Let G, G' be two graphs. Then an *isomorphism between G and G'* is a bijection $\phi : V(G) \to V(G')$ such that there is an edge $\{u, v\} \in E(G)$ if and only if there is an edge $\{\phi(u), \phi(v)\} \in E(G')$. We say G and G' are *isomorphic* if there exists an isomorphism between G and G', and we write $G \simeq G'$, or, when we want to specify the isomorphism, $G \overset{\phi}{\simeq} G'$.

Connectedness and connectivity

An *(undirected) path of length* $n \in \mathbb{N}$ in G is a sequence of nodes $(x_0, \ldots, x_n)$ such that x_i is a neighbour of x_{i-1} for all $i \in [n]$. If there is such a path, we say the nodes x_0 and x_n are connected by a path.

A graph is *connected* if each pair of nodes is connected by a path. If a graph is not connected, we call it *disconnected* A path of length at least 1 that starts and ends at the same node is called a *cycle*. A graph without cycles is a *forest*, and a connected forest is a *tree*.

A *spanning tree* of a graph G is a connected subgraph T of G such that $V(G) = V(T)$ and T is a tree. A graph has a spanning tree iff it is connected.

A *shortest path* is a path of minimal length, and the *distance* $d_G(x, y)$ is the length of a shortest path between x and y. If there is no path between x and y, we write $d_G(x, y) = \infty$. For a connected graph G, the *diameter* $\operatorname{diam}(G)$ of G is the maximal length of a shortest path in G, i.e. $\operatorname{diam}(G) = \min_{x,y \in V(G)} d_G(x, y)$.

A graph is k-connected if one needs to remove at least k edges from G to make it disconnected.[1] The *connectedness* of a graph is the property of being connected, whereas the *connectivity* of a graph G is the maximal number k such that G is k-connected.

A graph is *biconnected* if it is 2-connected. A *biconnected component* of a graph G is a maximal subgraph of G that is biconnected

Multi-graphs

Although we generally assume graphs do not contain self-loops or parallel edges, these do show up on several occasions. Graphs that do allow for these structures are called multigraphs, and are defined as follows. A *multi-graph* $G = (V, E, \epsilon)$ consists of a set of nodes V, a set of edges E, and an endpoint mapping $\epsilon : E \to V \times V$. For simplicity, we refer to an edge e by its image $\epsilon(e) = \{x, y\}$ under ϵ, so that notation for graphs and multigraphs becomes the same.

Modifying graphs

Let $\{u, v\}$ be an edge in a graph. *Subdividing* $\{u, v\}$ consists of deleting it from the graph and adding a new node x and edges $\{u, x\}$ and $\{x, v\}$. A *subdivision* of a graph G is any graph obtained from G by repeatedly subdividing edges.

The reverse operation is *suppressing* a degree-2 node. Let x be such a node with adjacent edges $\{u, x\}$ and $\{x, v\}$, then suppressing x consists of removing

[1] This is generally referred to as k-edge-connected, to distinguish it from k-vertex-connected. However, in this thesis, we will only use edge-connectedness, so we simply call this k-connected.

the edges $\{u, x\}$ and $\{x, v\}$ and the node x, and then adding an edge $\{u, v\}$. If the graph already contains the edge $\{u, v\}$, this results in a multigraph with two copies of the edge $\{u, v\}$.

Let $G = (V, E)$ be a graph with nodes $x, y \in V$, then *identifying* the nodes x and y results in the following new graph $G' = (V', E')$. In this graph $V' = V \setminus \{y\}$, and $E' = \{\{u, v\} \in E : u, v \neq y\} \cup \{\{x, u\} : u \neq x, \{y, u\} \in E\}$.

Any graph G' obtained from G by repeatedly identifying pairs of nodes is called a *quotient* of G. If $\sim$ is an equivalence relation on $V(G)$, then $G/\sim$ is the quotient of G obtained by identifying each pair of nodes in the same equivalence class of $\sim$.

2.1.2 Directed graphs

A *directed graph* (or *digraph*) $D = (V, A)$ consists of a set, V, of *vertices* or *nodes*, and a set $A \subseteq V \times V$ of *arcs*. We write $V(D)$ for the set of nodes, and $A(D)$ for the set of arcs of a digraph D. Like undirected graphs, directed graphs do not have *parallel arcs*, where two arcs (u, v) and (w, z) are said to be *parallel* if $u = w$ and $v = z$. Let $e = (u, v)$ be the arc from u to v of a digraph, then we say that u is the *tail* of e, and v is the *head* of e.

Let D, D' be two digraphs. Then an *isomorphism between D and D'* is a bijection $\phi : V(D) \to V(D')$ such that there is an arc $(u, v) \in A(D)$ if and only if there is an arc $(\phi(u), \phi(v)) \in A(D')$. We say D and D' are *isomorphic* if there exists an isomorphism between D and D', and we write $D \simeq D'$, or, when we want to specify the isomorphism, $D \overset{\phi}{\simeq} D'$.

A *subgraph* of a digraph $D = (V, A)$ is a graph $D' = (V', A')$ such that $V' \subseteq V$ and $A' \subseteq A$. Let $D = (V, A)$ be a digraph, and let $Y \subseteq V$ be a subset of the nodes, we will write $D[Y]$ to denote the subgraph of D induced by Y, i.e., the graph $(Y, A \cap (Y \times Y))$ and we will write $N \setminus Y$ to denote $N[V \setminus Y]$.

Connectedness and connectivity

Each directed graph $D = (V, A)$ has an *underlying undirected multi-graph* $U(D) = (V, E)$ where $E = \{\{x, y\} : (x, y) \in A\}$. If for all pairs of vertices $x, y \in V$ at most one of the arcs (x, y) and (y, x) is in A and $(x, x) \notin A$ for all $x \in V$, then $U(D)$ is a graph.

A *(directed) path of length* $n \in \mathbb{N}$ in D is a sequence of nodes $(x_0, \ldots, x_n)$ such that $(x_{i-1}, x_i) \in A(D)$ for all $i \in [n]$. An *up-down path* in a digraph D is a sequence of nodes $(x^1_{n_1}, \ldots, x^1_0, t, x^2_0, \ldots, x^2_{n_2})$ such that $(t, x^i_0, \ldots, x^i_{n_i})$ is a path in D for both $i \in [2]$. A *(directed) cycle* is a path of length at least one starting and ending at the same node.

A digraph D is *connected* if $U(D)$ is connected, it is *strongly connected* if there is a directed path from each vertex to any other vertex. A subgraph D' of D is a *biconnected component* if $U(D')$ is a biconnected component of $U(D)$.

Modifying graphs

Suppression of nodes and subdivision of arcs are defined similar for digraphs as for graphs, where the new arcs inherit the direction of the old arcs. Let (u, v) be an arc in a digraph. *Subdividing* (u, v) consists of deleting it from the graph and adding a node x and arcs (u, x) and (x, v). A *subdivision* of a digraph G is any graph obtained from G by repeatedly subdividing arcs.

The reverse operation is *suppressing* an indegree-1, outdegree-1 node. Let x be such a node with in-arc (u, x) and out-arc (x, v), then suppressing x consists of removing the arcs (u, x) and (x, v) and the node x, and then adding an arc (u, v).

Digraph quotients are defined analogous to graph quotients: identifying two nodes $x, y \in V(D)$ results in a new digraph $D' = (V', A')$ with $V' = V \setminus \{y\}$ and $A' = \{(u, v) \in E : u, v \neq y\} \cup \{(x, v) : (y, v) \in E, v \neq x, y\} \cup \{(u, x) : (u, x) \in E, u \neq x, y\}$.

DAGs

A *directed acyclic graph* (DAG) is a digraph without directed cycles. The absence of directed cycles implies there is a natural partial order on the vertices of a digraph.

This order relation for DAGs diverts from the notation for rooted trees in [Die12], as we imagine the arcs of a DAG to be pointed downwards, instead of upwards as in [Die12]. A node u is *above* a node v if there is a directed path from u to v, we also say that v is *below* u. Note that for each node u, there is a directed path (u) from u to u, so each node is below and above itself. A node u is *strictly* above (resp. below) v if u is above (resp. below) v and $u \neq v$. Moreover, an arc (x, y) is above a node u if y is above u, and it is below u if x is below u.

Using terminology derived from phylogenetics, we alternatively use the following terms to indicate that u above v. In that case, we say that u is an *ancestor* of v and v is a *descendant* of u. Similarly, if $(u, v) \in A(D)$, we say u is a *parent* of v and v is a *child* of u. A child of a node u is often denoted $c(u)$, and we write $p(u)$ for a parent of u. Alternatively, we say u is *directly above* v, and v is *directly below* u. A *lowest node* in a DAG is a node that has no descendants.

A *last common ancestor* (or *lowest common ancestor*) of two nodes u and v is an ancestor w of u and v such that no node below w is an ancestor of both u and v. The set of lowest common ancestors of u and v is denoted $\mathrm{LCA}(u, v)$.

A subset $Y \subseteq V(D)$ of the vertices of a DAG D is *down-closed* if there is no arc (y, z) in D with $y \in Y$ and $z \notin Y$. Similarly, a subset $Y \subseteq V(D)$ of the vertices of a DAG D is *up-closed* if there is no arc (z, y) in D with $y \in Y$ and $z \notin Y$.

2.1.3 Labeled graphs

A *labeled (di)graph* is a (di)graph G together with a bijective labeling $l : W \to X$ of a subset of the nodes $W \subseteq V(G)$ with labels X.

Definition 2.1. Let G_1, G_2 be two (di)graphs labeled by maps $l_1 : W_1 \to X_1$ and $l_2 : W_2 \to X_2$. If $G_1 \overset{\phi}{\simeq} G_2$ such that for any label $x \in Y \subseteq X_1 \cap X_2$, the node $\phi(l_1^{-1}(x)) = l_2^{-1}(x)$, then ϕ is a *labeled isomorphism* between G_1 and G_2 with respect to Y. We say G_1 and G_2 are *labelled isomorphic with respect to* Y if there exists a labeled isomorphism between G_1 and G_2 with respect to Y, and we write $G_1 \overset{\phi}{\simeq}_Y G_2$.

2.2 Directed networks

In this section, we define directed phylogenetic networks, the explicit representation of an evolutionary history. After this definition, we introduce several properties such a network may have. Then, we consider relevant structures that may be found in a directed phylogenetic network, and a few examples of (classes of) networks. Lastly, we introduce a special type of containment of networks within other networks.

Definition 2.2. A *directed phylogenetic network* $N = (V, A, l)$ on a set of *taxa* X is a DAG (V, A) labeled with $l : L(N) \to X$, where $L(N)$ denotes the set of leaves of N, with nodes of the following types.

Root Indegree-0, outdegree-1 node, it has exactly one of these;

Tree nodes Indegree-1, outdegree-k node, with $k \geq 2$;

Reticulation Indegree-k, outdegree-1 node, with $k \geq 2$;

Leaf Indegree-1, outdegree-0 node.

In a *subdivided phylogenetic network*, a fifth type of node is allowed.

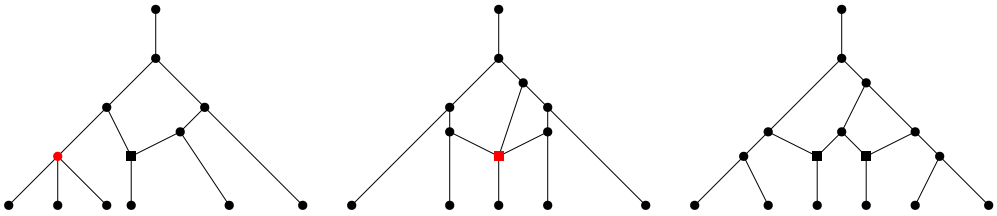


Figure 2.1: Two non-binary networks and a binary network (labels are not shown). Square nodes indicate reticulations, and red nodes indicate non-binary nodes. The network on the right is binary because it has no non-binary nodes.

Degree-2 Node Indegree-1, outdegree-1 node.

Incoming arcs of reticulation nodes are called *reticulation arcs* and incoming arcs of leaves are called *leaf arcs*. The root of a network is often denoted as ρ.

Definition 2.3. An *internally labeled (subdivided) directed network* is a (subdivided) directed network $\dot{N} = (V, A, l)$ ($\ddot{N} = (V, A, l)$) in which the labeling is a bijective map $l : V \to X$. The set of labels X can be partitioned into sets that correspond bijectively to the types of nodes. The set of leaf labels is X^l, the set of tree node labels is X^t, the set of reticulation labels is X^r, the set of root labels is $\{x^\rho\}$, and the set of degree-2 node labels is $X^{(2)}$.

Because each leaf has a unique label in a network, we will often interchange $l(v)$ and v freely for leaves in networks. For internally labeled networks, we do this for all nodes.

Definition 2.4. Let $\ddot{N}$ be a subdivided directed network, then the suppression of $\ddot{N}$, $S(\ddot{N})$, is the multi-digraph obtained from $\ddot{N}$ by suppressing all indegree-1 outdegree-1 nodes. If $S(\ddot{N})$ is a directed network (i.e., if it is a digraph without parallel arcs or self-loops), then we say $\ddot{N}$ is *suppressable*.

Definition 2.5. A network is *binary* if all its nodes are binary, i.e., if all tree nodes have out-degree equal to two and all reticulations have in-degree equal to two (Figure 2.1).

In this thesis, we will only consider binary directed phylogenetic networks, so, henceforth, we will drop phylogenetic and binary whenever we refer to a binary phylogenetic network. For example, we will say "directed network" instead of "directed binary phylogenetic network". When it is clear from context that the network is directed, we will simply refer to it as a network.

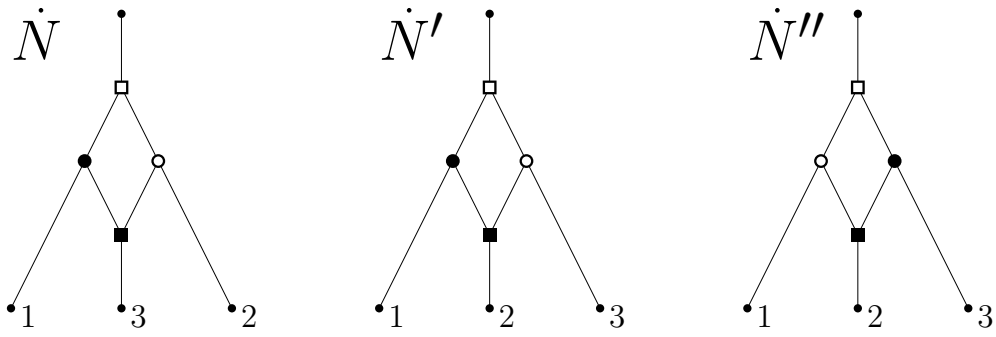


Figure 2.2: Three internally labeled networks. All three networks are isomorphic, but only $\dot{N}'$ and $\dot{N}''$ are leaf isomorphic. None of these networks are labeled isomorphic with respect to the full label set.

Definition 2.6. The *reticulation number* of a (non-binary) phylogenetic network N is

$$\sum_{v \in R(N)} \deg^-(v) - 1,$$

where $R(N)$ denotes the set of reticulation nodes of N.

Observation 2.7. Let $N = (V, A, l)$ be a binary network with n leaves, m degree-2 nodes, and reticulation number k, then N has $2n + 3k + m - 1$ arcs and $2n + 2k + m$ vertices, of which n are leaves, 1 is the root, k are reticulation nodes, m are degree-2 nodes, and $n + k - 1$ are tree nodes.

Definition 2.8. Two networks are *(labeled) isomorphic* if they are (labeled) isomorphic as labeled graphs, and the notation is the same as for labeled graphs. We say two networks are *leaf-isomorphic* if they are labeled isomorphic with respect to their leaf labels X^l (Figure 2.2).

2.2.1 Substructures

For a network N and an internal node v, $N \downarrow v$ is the (subdivided) subnetwork of N induced by v and all nodes below v. If v is a tree node, then a new root-arc is added to the resulting digraph. A subgraph S of a network N is a *pendant subnetwork* if $S = N \downarrow v$ for some $v \in V(N)$ and all degree-2 nodes of S are also degree-2 nodes in N. If S is a tree, it is called a *pendant subtree*, and if S has exactly two leaves l and l' it is called a *cherry* (l, l') on l and l'.

A network N has a *chain* $(x_1, \ldots, x_l)$ of length l if x_i are leaves of N, and N contains the subgraph consisting of the arcs $\{(p_i, x_i) : i \in [l]\} \cup \{(p_i, p_i + 1) : i \in [l - 1]\}$, and p_l is a tree node.

An important substructure in a network is the triangle. This structure has a large impact on the rearrangements we can do in a network.

Definition 2.9. If a network N has arcs (x, y), (x, z), and (y, z), then we say x, y and z form a *triangle* in N, or $\langle x, y, z\rangle \in N$. If y is a tree node, we write $\langle x, y, z\rangle_t$, and if y is a reticulation, we write $\langle x, y, z\rangle_r$.

We call x the *top* of the triangle, y the *side* of the triangle, and the reticulation z the *bottom* of the triangle. The arc (x, z) is called the *long arc* and (y, z) is called the *bottom arc* of the triangle.

For a subdivided network $\ddot{N}$, we will often need to consider substructures of $S(\ddot{N})$ and the corresponding parts of $\ddot{N}$. To simplify the notation in these cases, we introduce some terms for these structures, where we simply replace arc (in $S(\ddot{N})$) with path (in $\ddot{N}$).

The *root path* of $\ddot{N}$ is the path corresponding to the root arc of $S(\ddot{N})$. Similarly, the *leaf path* of a leaf x in $\ddot{N}$ is the path that corresponds to the incoming arc of x in $S(\ddot{N})$.

If $S(\ddot{N})$ has a pair of parallel arcs from x to y, then the corresponding pair of paths in $\ddot{N}$ are called *parallel paths*.

Lastly, if $S(\ddot{N})$ has a triangle $\langle x, y, z\rangle$, then we say $\ddot{N}$ contains the subdivided triangle $\langle \cdot x, y, z\cdot\rangle$ which consists of the paths corresponding to the three arcs of $\langle x, y, z\rangle$.

Definition 2.10. A *blob* of a (subdivided) phylogenetic network $N = (V, A, l)$, is the network formed by a biconnected component of the underlying DAG $D = (V, A)$ of N, together with its out-going arcs, where each leaf l is labelled with the set of leaves of N below the node corresponding to l in N.

Definition 2.11. The *level* of a phylogenetic network N is the maximum reticulation number in a blob of N.

For an example of blobs of a network, see Figure 2.3. The network in this figure has three blobs, one of which is level-1 and the other two are level-2 blobs.

Note that any level-2 blob with outdegree-1 (in any network) is isomorphic to the level-2 blob with one outgoing arc in Figure 2.3.

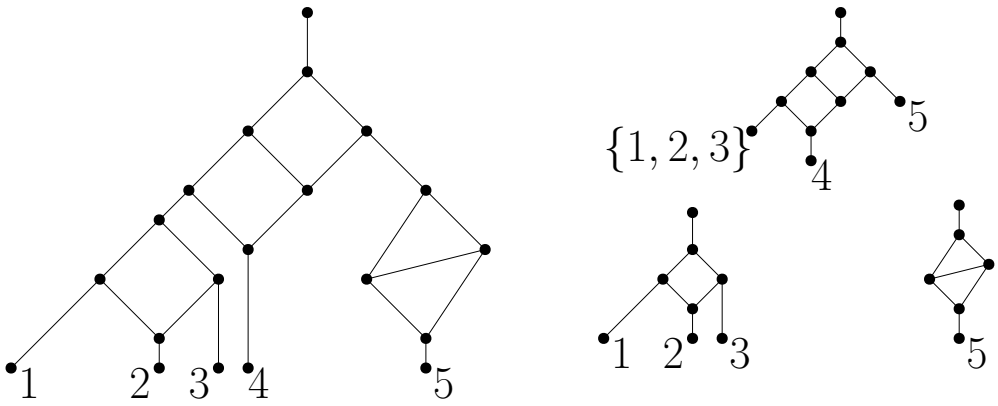


Figure 2.3: A level-2 network (left) with its three blobs (right).

2.2.2 Recurring examples of networks

Definition 2.12. A *directed phylogenetic tree* is a phylogenetic network without reticulations.

A (nonbinary) phylogenetic tree can be defined using a *Newick string*, which represents this tree as a nested set of leaves followed by a semicolon. It is defined recursively as follows: Let $n_1, \ldots, n_l$ be Newick strings without the semicolon, with corresponding trees $T_1, \ldots, T_l$, then $(n_1, \ldots, n_l)$; is the Newick string for the tree obtained by identifying the roots of the trees $T_1, \ldots, T_l$ and adding a new root arc; if x is a leaf, the newick string x; represents the tree with the one leaf x.

Definition 2.13. Let $X = \{x_1, \ldots, x_n\}$ be an ordered set of labels. The *caterpillar* $C(X)$ is the tree defined by the Newick string

$$(\cdots(x_1, x_2), x_3)\cdots, x_n);$$

Definition 2.14. Let $X = \{x_1, \ldots, x_{2^i}\}$ be an ordered set of 2^i labels. The *balanced tree* $B(X)$ is defined recursively as the tree obtained from $B(x_1, \ldots, x_{2^{i-1}})$ and $B(x_{2^{i-1}+1}, \ldots, x_{2^i})$ by identifying their roots and adding a new root arc.

Up to isomorphism, there is one networks with two leaves, one reticulation, and no degree-2 nodes. Such a network has the shape of the capital letter A. Hence, for any such network, we say $N \simeq N_A$. These networks will make many appearances in the chapter about tail moves, as only the space of networks with two leaves and one reticulations is not connected under tail moves.

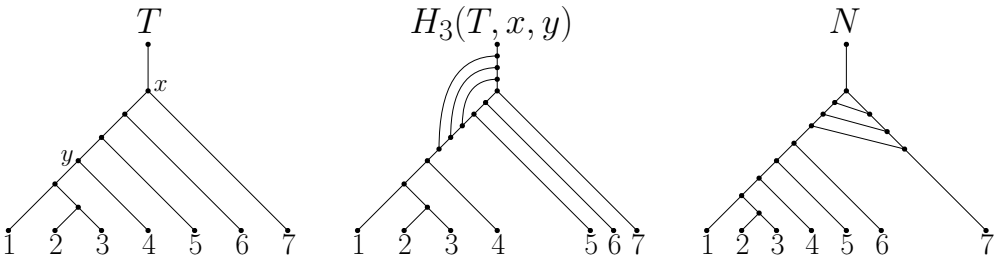


Figure 2.4: A tree T, a handcuffed tree $H_3(T, x, y)$ constructed from T by adding handcuffs (arcs) between the incoming arcs of x and y, and a ladder tree $N = H_3(T, p(6), 7)$ constructed from T. If 2 were in a cherry with leaf 1 instead of with leaf 3 in all these networks, then T would be a caterpillar, and N a ladder caterpillar.

Definition 2.15. For any tree T with n leaves, denote with T^+ the tree with $n+1$ leaves constructed from T by *attaching* the $(n+1)$-th leaf to the root arc, i.e., subdividing the root arc with a node x and adding the arc $(x, n+1)$

Definition 2.16. Let T be a tree and $x, y \in V(T)$, then $H_k(T, x, y)$ is the *handcuffed tree* constructed from the tree T by subdividing the incoming arcs of x and y with k degree-2 nodes each, and attaching a new arc (a *handcuff*) from the i-th degree-2 nodes on the incoming path of x to the i-th degree-2 node on the incoming path of y for each $i = 1, \ldots, k$ (Figure 2.4).

Definition 2.17. A network has k reticulations *at the top* if it has the following structure:

1) the node c: the child of the root;

2) nodes a_i and b_i and an arc (a_i, b_i) for each $i \in \{1, \ldots, k\}$;

3) the arcs (c, a_1) and (c, b_1);

4) for each $i \in \{1, \ldots, k-1\}$ there are arcs (a_i, a_{i+1}) and (b_i, b_{i+1}) or arcs (a_i, b_{i+1}) and (b_i, a_{i+1}).

We say the there are k reticulations *neatly at the top* if they are all directed to the same arc, i.e. we replace point 4) with

4') for each $i \in \{1, \ldots, k-1\}$ there are arcs (a_i, a_{i+1}) and (b_i, b_{i+1}).

Examples are shown in Figure 2.5.

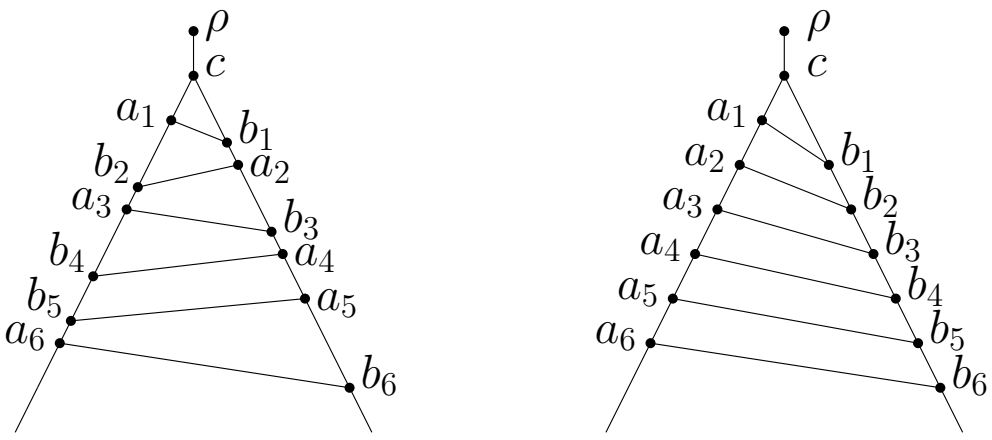


Figure 2.5: The top parts of two networks with 6 reticulations at the top. On the right, the reticulations are neatly at the top.

Definition 2.18. A *ladder tree* is a network with all reticulations neatly at the top (Figure 2.6). The non-trivial biconnected component of a ladder tree is called the *ladder*.

If $N \in \mathcal{N}(n,k)$ is a ladder tree, then we can obtain a (unique) tree T from N by removing all arcs (a_i, b_i) and suppressing the resulting degree-2 nodes. Hence, for each network N with all k reticulations neatly at the top, there is a tree T so that $N \simeq_X H_k(T, x, y)$ or $N \simeq_X H_k(T, y, x)$, where x and y are the grandchildren of the root of T.

Definition 2.19. A *ladder caterpillar* is one of the following.[2]

- The one-edge network with one leaf and one root.
- A network with one leaf, one root, and one biconnected component consisting of a path of reticulations $k_1, \ldots, k_l$, a path of tree nodes $s_0, \ldots, s_{l-1}$, an edge from s_i to k_i for all $i = 1, \ldots, l-1$, and the two edges (s_0, k_1) and (s_{l-1}, k_l).
- A ladder tree $N = H_k(T, x, y)$ constructed from a caterpillar T and the grandchildren of the root x and y, where y is a leaf.

Lemma 2.20. *Let $n \in \mathbb{Z}_{\geq 1}$ and $k \in \mathbb{N}$. Then there exists a ladder caterpillar with n leaves and k reticulations, except when $n = k = 1$.*

[2] In [Jan21] a ladder caterpillar was called a ladder network.

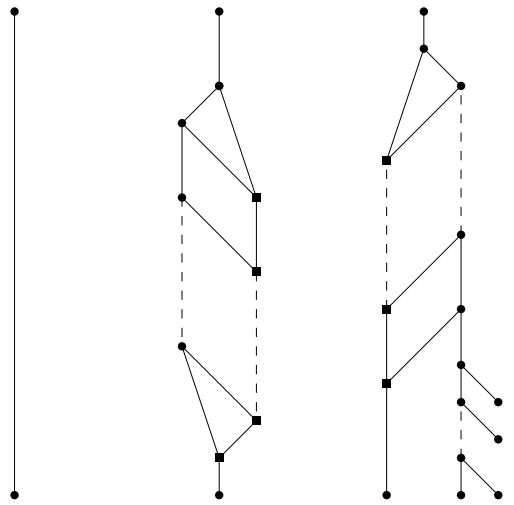


Figure 2.6: Ladder caterpillars as defined in Definition 2.19. Dashed lines indicate repeated structure, i.e., an extension of the ladder or an extension of the caterpillar.

Proof. First note that if $n = k = 1$, then there is no network with n leaves and k reticulations. In particular, there is no such ladder caterpillar.

If $n = 1$ and $k = 0$, then the tree with one leaf is a ladder caterpillar with n leaves and k reticulations; if $k > 1$, then we take the network with one leaf and one biconnected component consisting of a path of reticulations $k_1, \ldots, k_l$, a path of tree nodes $t_0, \ldots, t_{l-1}$, an arc from t_i to k_i for all $i = 1, \ldots, l-1$, and the two arcs (t_0, k_1) and (t_{l-1}, k_l) (Figure 2.6, second network).

Now we may assume $n > 1$. We build a ladder caterpillar starting from a caterpillar on n leaves, of which the leaf l is a child of the top tree node t of the caterpillar, by repeatedly (k times) adding an arc between the two outgoing arcs of the top tree node such that the new reticulation is on the arc which used to be part of (t, l) in the original caterpillar. This results in a ladder caterpillar with n leaves and k reticulations, as the number of leaves does not increase in this process, and the number of retciulations increases by one for each added arc. □

Lemma 2.21. *Let N be a connected network with n leaves and k reticulations, then there exists a ladder caterpillar with the same number of leaves and reticulations as N.*

Proof. Because N is a network, it has at least one leaf. As there is no network with one leaf and one reticulation, the result follows directly from Lemma 2.20. □

2.2.3 Displaying trees and networks

Definition 2.22. Let T be a phylogenetic tree and let $Y \subseteq X$ be a subset of the labels. Then $T|_Y$ is the *subtree of T induced by Y*; that is, $T|_Y$ is the union of all (shortest) undirected paths between nodes of Y.

Definition 2.23. Let $N = (V, A, l)$ and $N' = (V', A', l')$ be two networks with $X' \subseteq X$. Suppose there exists an injective mapping of the nodes $i_V : V' \to V$ and a mapping of the arcs $i_A : A' \to P(N)$ to non-empty paths in N such that:

- $i_A(e)$ and $i_A(f)$ are internal-node disjoint for all $e \neq f$,
- and $i_A((u, v))$ has $i_V(u)$ as starting point and $i_V(v)$ as endpoint for all $(u, v) \in E'$.

Such a mapping is called an *embedding* of N' into N, and if it exists, we say N *displays* N', or that N' is a *subnetwork* of N. When it is clear whether a is a node or an arc, we simply write $i(a)$ instead of $i_V(a)$ or $i_A(a)$. The union

$$i(N') = \bigcup_{a \in A'} i_A(a)$$

is also called the (image of the) embedding of N' into N.

The set of embedded trees of a network N is denoted $\mathcal{T}(N)$. If there exists a $T \in \mathcal{T}(N)$ such that $U(T)$ is a spanning tree of $U(N)$, then N is *tree-based* [FS15].

2.3 Undirected networks

In this section we introduce the undirected version of a phylogenetic network. In most respects, the definitions of undirected networks are analogous to those of directed networks.

Definition 2.24. An *undirected phylogenetic network* $U = (V, E, l)$ on a set of at least 2 taxa X is an undirected, connected, finite, simple graph (V, E) with two types of nodes:

Leaf Degree-1 node;

Internal node Degree-k node, with $k \geq 3$;

labeled with bijective map $l : L(U) \to X$, where $L(U)$ denotes the set of leaves of U. In a *subdivided undirected phylogenetic network*, a third type of node is allowed.

Degree-2 node Degree-2 node.

Edges incident to leaves are called *leaf edges*.

Definition 2.25. An *internally labeled (subdivided) undirected network* is a (subdivided) undirected network $\dot{U} = (V, E, l)$ ($\ddot{U} = (V, E, l)$) in which the labeling is a bijective map $l : V \to X$. The set of labels X can be partitioned into sets that correspond bijectively to the types of nodes. The set of leaf labels is X^l, the set of internal node labels is X^i, and the set of degree-2 node labels is $X^{(2)}$.

Definition 2.26. Let $\ddot{U}$ be a subdivided undirected network, then the *suppression* of $\ddot{U}$, $S(\ddot{U})$, is the multi-graph obtained from $\ddot{U}$ by suppressing all degree-2 nodes. If $S(\ddot{U})$ is an undirected network (i.e., if it is a graph without parallel edges or self-loops), then we say $\ddot{U}$ is *suppressable*.

Definition 2.27. An undirected network is *binary* if all internal nodes have degree three.

Like for directed networks, we will only consider (internally labeled resp. subdivided) binary undirected phylogenetic networks, so, henceforth, we will drop phylogenetic and binary whenever we refer to a (internally labeled resp. subdivided) binary undirected phylogenetic network. For example, we will say "subdivided undirected network" instead of "subdivided binary undirected phylogenetic network". When it is clear from context that the network is undirected, we will additionally drop the adjective undirected, and simply refer to it as a (internally labeled resp. subdivided) network.

Definition 2.28. The *reticulation number* of a (non-binary) network U is $|E| - |V| + 1$.

Observation 2.29. Let $U = (V, E, l)$ be a binary network with n leaves, m degree-2 nodes, and reticulation number k, then U has $2n + 3k + m - 3$ edges and $2n + 2k + m - 2$ vertices, of which n are leaves, $n + 2k - 2$ are internal nodes, and m are degree-2 nodes.

Definition 2.30. Two undirected networks are *(labeled) isomorphic* if they are (labeled) isomorphic as labeled graphs, and the notation is the same as for labeled graphs. We say two networks are *leaf-isomorphic* if they are labeled isomorphic with respect to their leaf labels X^l.

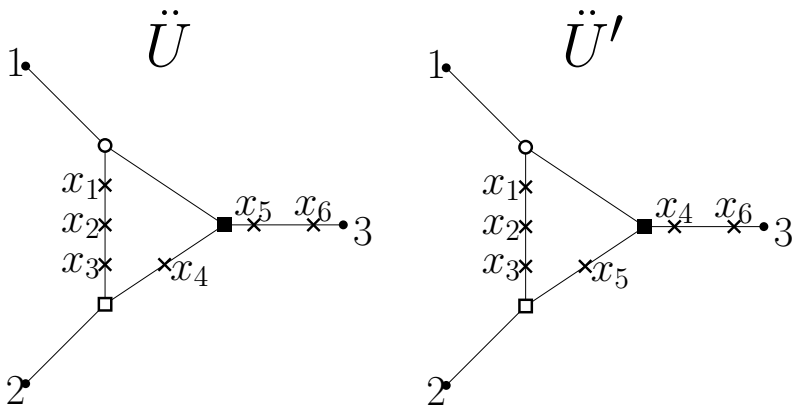


Figure 2.7: Two subdivided undirected networks $\ddot{U}$ and $\ddot{U}'$. The labels of the leaves and degree-2 nodes are shown in symbols, whereas the labels of the degree-3 nodes are represented by node shapes. Note that both networks contain a subdivided triangle, and that the leaf path of leaf 3 contains two degree-2 nodes in both networks. Furthermore, $\ddot{U} \simeq_{X^l \cup X^i} \ddot{U}'$ but the networks are not isomorphic w.r.t. the full label set X, $\ddot{U}$ and $\ddot{U}'$ are suppressable, and $S(\ddot{U}) \simeq_X S(\ddot{U}')$.

2.3.1 Substructures

Definition 2.31. Let U be a network and let x, y, z be nodes of U. We write $\langle x, y, z\rangle \in U$, or say x, y, and z form a *triangle* if there are edges $\{x, y\}, \{y, z\}$ and $\{x, z\}$ in U.

For a subdivided network $\ddot{U}$, we will often need to consider substructures of $S(\ddot{U})$ and the corresponding parts of $\ddot{U}$. To simplify the notation in these cases, we introduce some terms for these structures, where we simply replace edge (in $S(\ddot{U})$) with path (in $\ddot{U}$) (Figure 2.7).

A *leaf path* of a leaf x in $\ddot{U}$ is the path that corresponds to the edge incident to x in $S(\ddot{U})$.

If $S(\ddot{U})$ has a pair of parallel edges from x to y, then the corresponding pair of paths in $\ddot{U}$ are called *parallel paths*.

Lastly, if $S(\ddot{U})$ has a triangle $\langle x, y, z\rangle$, then we say $\ddot{N}$ contains the subdivided triangle $\langle \cdot x, y, z \cdot\rangle$ which consists of the paths corresponding to the three arcs of $\langle x, y, z\rangle$.

Definition 2.32. A *blob* of a (subdivided) network $U = (V, E, l)$, is the network formed by a biconnected component B of the underlying graph $G = (V, E)$ of U, together with its incident edges, where each degree-1 node l is labelled with

the set of leaves of U separated from B by removing the edge incident to l from U.

Definition 2.33. The *level* of a network U is the maximum reticulation number in a blob of U.

2.3.2 Recurring examples of networks

Definition 2.34. An *undirected phylogenetic tree* is an undirected phylogenetic network with reticulation number 0. Equivalently, it is an undirected phylogenetic network whose underlying graph is a tree.

Definition 2.35. An *undirected caterpillar* $C_u(X)$ is an undirected phylogenetic tree with leaf labels X whose internal vertices are all adjacent to a leaf. Equivalently, it is the undirected phylogenetic tree on $X \cup \{\rho\}$ obtained by forgetting the orientation of the arcs in the directed caterpillar $C(X)$.

The following type of network was introduced by Francis et al. [FHMW17].

Definition 2.36. An *Echidna network* with n leaves and k reticulations is any network constructed as follows (Figure 2.8). Start with the graph consisting of the cycle $\{p_1, \ldots, p_{n+k-1}, x_{k-1}, \ldots, x_1, p_1\}$; attach the leaves $\{l_i\}_{i=1}^n$ to the cycle with arcs $\{l_i, p_{j_i}\}$ such that $j_1 = 1$ and $j_n = n+k-1$ and $j_i \neq j_{i'}$ if $i \neq j$; lastly, add an edge $\{x_i, p_{j_i'}\}$ for each $i \in \{1, \ldots, k-1\}$ such that the degree of all nodes becomes either one or three.

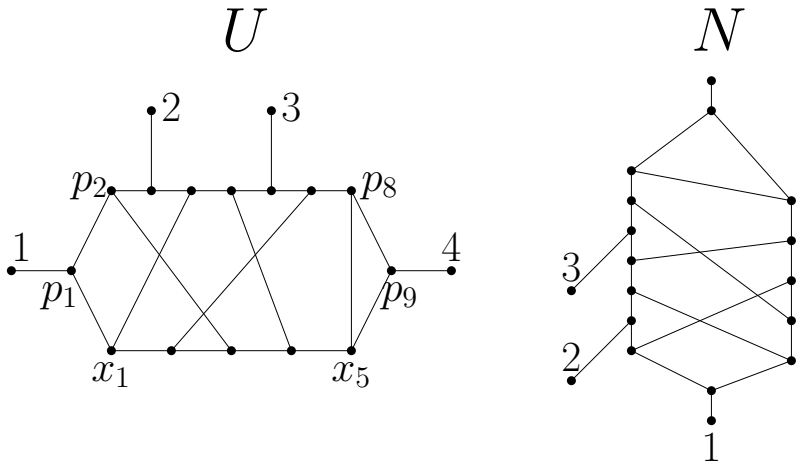


Figure 2.8: An Echidna network U and a directed network N such that $U(N) = U$.

2.3.3 Displaying networks

Definition 2.37. Let T be an undirected phylogenetic tree and let $Y \subseteq X$ be a subset of the labels. Then $T|_Y$ is the *subtree of T induced by Y*; that is, $T|_Y$ is the union of all (shortest) paths between nodes of Y.

Definition 2.38. Let $U = (V, E, l)$ and $U' = (V', E', l')$ be two networks with $X' \subseteq X$. Suppose there exists an injective mapping of the nodes $i_V : V' \to V$ and a mapping of the edges $i_E : E' \to P(N)$ to non-empty paths in N such that:

- $i_E(e)$ and $i_E(f)$ are internal-node disjoint for all $e \neq f$,
- and $i_E((u, v))$ has $i_V(u)$ as starting point and $i_V(v)$ as endpoint for all $(u, v) \in E'$.

Such a mapping is called an *embedding* of U' into U, and if it exists, we say U *displays* U', or that U' is a *subnetwork* of U. The union

$$i(U') = \bigcup_{e \in E'} i_E(e)$$

is also called the (image of the) embedding of U' into U.

The set of embedded trees of a network U is denoted $\mathcal{T}(U)$. If there exists a $T \in \mathcal{T}(U)$ such that T is a spanning tree of U, then U is *tree-based*.

2.4 Rearrangement moves

In this section, we introduce rearrangement moves for phylogenetic networks. These moves provide a way to slightly modify a phylogenetic networks. We will define several of these moves, and present a few basic properties of these moves.

2.4.1 Directed networks

Definition 2.39 (Tail move)**.** Let N be a network containing the distinct arcs (p, u), (u, c), (u, v), and (p', c'). Let D be the (multi-)digraph obtained from N by

- *pruning* (u, v) at u, i.e., replace (p, u) and (u, c) with a new arc (p, c);
- and *reattach* (u, v) at (p', c'), i.e., replace (p', c') by (p', u) and (u, c').

If D is a phylogenetic network (i.e., a DAG without parallel arcs), then we say D is the result of the *tail move* of (u, v) from (p, c) to (p', c') in N, which we also denote $(p, u, c) \xrightarrow{(u,v)} (p', c')$ or $u \xrightarrow{(u,v)} (p', c')$ if the from-arc (p, c) is not relevant. All nodes retain their labels if they were labelled (Figure 2.9).

For simplicity, we also allow moves $(p, u, c) \xrightarrow{(u,v)} (p', c')$ where $(p', c') \in \{(p, u), (u, c)\}$. For these moves the resulting network is per definition equal to the original network.

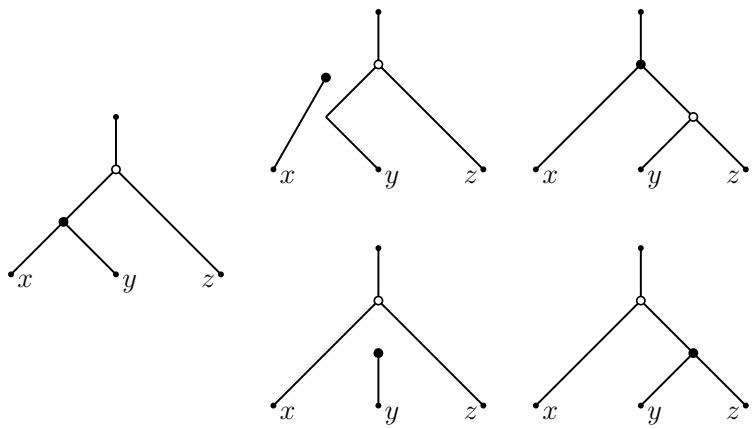


Figure 2.9: Two tail moves on the network on the left. The moves result in two networks that are leaf-isomorphic, but not labeled isomorphic with respect to the full label set. The labels of the internal nodes are represented by the node shapes, the labels of the leaves are shown as letters.

The graph G in the definition above is not necessarily a phylogenetic network, as it may be cyclic or contain parallel arcs.

A head move is defined similarly, with the following small change. Instead of the tail, the head of an arc is pruned and reattached in a head move.

Definition 2.40 (Head move)**.** Let N be a network containing the distinct arcs (p, v), (v, c), (u, v), and (p', c'). Let D be the (multi-)digraph obtained from N by

- *pruning* (u, v) at v, i.e., replace (p, v) and (u, v) with a new arc (p, c);
- and *reattach* (u, v) at (p', c'), i.e., replace (p', c') by (p', v) and (v, c').

If D is a phylogenetic network (i.e., a DAG without parallel arcs), then we say D is the result of the *head move* of (u, v) from (p, c) to (p', c') in N, which we also denote $(p, v, c) \xrightarrow{(u,v)} (p', c')$ or $v \xrightarrow{(u,v)} (p', c')$ if the from-arc (p, c) is not relevant. All nodes retain their labels if they were labelled (Figure 2.9).

In the arrow notation for rearrangement moves, we always need notation for the receiving arc. Sometimes, it is convenient to simply define the receiving arc as the (unique) incoming arc of a tree-node or a degree-2 node, or as the (unique) outgoing arc of a reticulation or a degree-2 node. In such cases, we simply replace the other endpoint of this arc with a centered dot, e.g., we write $u \xrightarrow{(u,v)} (\cdot, x)$ for the move that moves the u-endpoint of (u, v) to the incoming arc of x.

Note that head moves and tail moves cannot change the number of leaves, tree nodes, or reticulations of a network.

Head moves and tail moves together are called *rSPR* moves. The term rSPR was coined as an acronym for "rooted Subtree Prune and Regraft" when used for phylogenetic trees. Now, it is also used for directed phylogenetic networks, even though it does not actually prune off a subtree or subnetwork. Note that each (valid) rSPR move is *reversible*, as applying $(p', u, c') \xrightarrow{e} (p, c)$ after $(p, u, c) \xrightarrow{e} (p', c')$ results in the original network.

Definition 2.41. An rSPR move of (u, v) from (p, c) to (p', c') is a *distance-d* move, or a rSPR_d move, if p' or c' is at distance less than d from p or c in the graph resulting from pruning (u, v) from (p, c).

Similarly, we have tail_d and head_d moves. An rSPR_1 move is also called an *rNNI* move.

Note that a distance-d move cannot necessarily be simulated with a sequence of d distance-1 moves. The path of length-d defining the distance of the move might not be a path over which we can move the endpoint (Figure 2.10).

Comparing optimization scores across networks with different reticulation numbers may be tricky, since a network generally allows a better fit with the data when its reticulation number is higher. For this reason, it is increasingly conventional [e.g. in GvIJ$^+$17a] to make a distinction between horizontal and vertical moves.

Moves that do not change the reticulation number are called *horizontal moves*; this is to contrast them with moves that change the reticulation number, which we call *vertical moves*. Note that all moves discussed above are horizontal. SNPR moves, however, may be vertical as well: an *SNPR move* is either a tail move, or a vertical moveof one of two types. A vertical move can add an arc by subdividing two distinct arcs and attaching a new arc between the two new nodes, or it can remove an arc and suppresses its endpoints [BLS17].

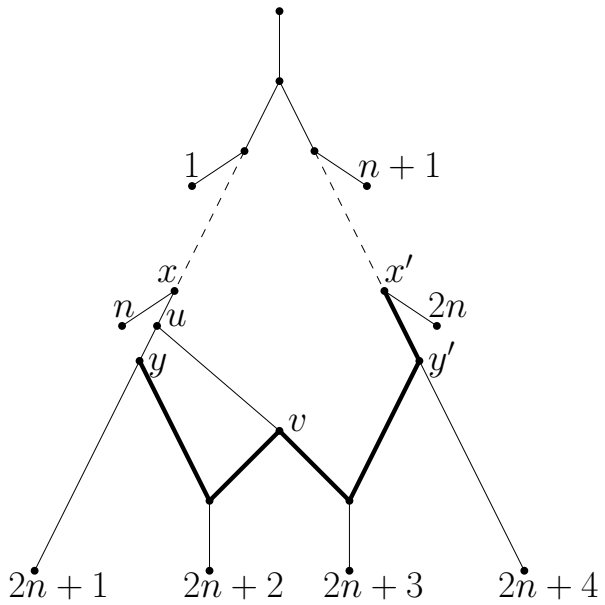


Figure 2.10: A distance-5 tail move $(x, u, y) \xrightarrow{(u,v)} (x', y')$ that cannot be replaced by a sequence of five distance-1 tail moves. The path that determines the distance of the move (thick arcs) contains arcs that u cannot be moved to, as they are below the head of the arc (u, v).

Validity of moves

As we want to use rearrangement moves to traverse network space, each move must result in a phylogenetic network. The definitions of tail moves, head moves, and rSPR moves enforce that this always happens. In this thesis, we often propose a sequence of moves by stating: move the tail of arc e to arc f, then move the head of arc e' to f' and so forth. We then check whether these moves are *valid*, or *allowed*; that is, whether applying the steps in their respective definitions produces a phylogenetic network. A necessary condition for a rearrangement move to be valid, is that the moving arc is *movable*, which ensures that pruning the arc does not create parallel arcs.

Definition 2.42. Let (u, v) be an arc in a network N, then (u, v) is *tail-movable* if u is a tree node with parent p and other child c, and there is no arc (p, c) in N. An arc that is not tail-movable is called *tail-immovable*.

This is equivalent to saying that an arc with tail u is tail-movable if u is a tree node and u is not the side of a triangle $\langle p, u, c\rangle$. We give a similar definition for head moves.

Definition 2.43. Let (u, v) be an arc in a network N, then (u, v) is *head-movable* if v is a reticulation node with other parent p and child c, and there is no arc (p, c) in N. An arc that is not head-movable is called *head-immovable*.

When the type of move is clear from context, we will simply use the terms *movable* and *immovable*. Using the concept of movability, we can now succinctly give sufficient conditions for a move to be valid. Besides movability, we need additional conditions to make sure that reattaching the arc does not create parallel arcs, and that the resulting network has no cycles. These correspond to the second and third conditions in the following lemma.

Lemma 2.44. *A tail move* $u \xrightarrow{(u,v)} (s, t)$ *is valid if all of the following hold:*

- (u, v) *is tail-movable;*
- $v \neq t$*;*
- v *is not above* s*.*

Proof. Because (u, v) is tail-movable, pruning (u, v) at u does not create parallel arcs. Because $v \neq t$, reattaching (u, v) at (s, t) does not create parallel arcs either. Hence, the resulting digraph N' of the tail move contains no parallel arcs.

Now suppose N' has a cycle. As each path that does not use (u, v) in N' corresponds to a path in N, the cycle must use (u, v). This means that there is a path from v to u in N'. Because u is a tree node with parent s, there must also be a path from v to s in N'. This implies there was also a path from v to s in N, but this contradicts the third condition: v is not above s. We conclude that N' is a DAG. As all labelled nodes remain unchanged by the tail move, N' is a phylogenetic network and the tail move is valid. □

The proof of the corresponding lemma for head moves is completely analogous.

Lemma 2.45. *A head move* $v \xrightarrow{(u,v)} (s, t)$ *is valid if all of the following hold:*

- (u, v) *is head-movable;*
- $u \neq s$*;*
- t *is not above* u*.*

We will very frequently use the following corollary of this lemma, which makes it very easy to check whether some moves are valid.

Definition 2.46. Let (u, v) be an arc in a network N. A tail move $u \xrightarrow{(u,v)} (w, z)$ is said to *move a tail up* if z is strictly above u. Similarly, a head move $v \xrightarrow{(u,v)} (w, z)$ is said to *move a head down up* if w is strictly below v.

Corollary 2.47. *Let (u, v) be a tail-movable arc, then moving the tail of (u, v) up is allowed. Similarly, moving the head of a head-movable arc down is allowed.*

Note that, despite the previous corollary, movability of an arc a does not imply that there exists a valid tail move for a: it only ensures that we can remove the tail without creating a clear violation of the definition of a network; it does not ensure that we can reattach it anywhere else. A tail can always be moved to the root arc. However, this may result in an isomorphic network, if, for example, the tail is the child of the root.

2.4.2 Undirected networks

For undirected networks, the two endpoints of an edge cannot be distinguished as being a head or a tail. Hence, there are far fewer types of rearrangement moves on undirected networks. Tail, head, and rSPR moves all have only one analogue in undirected networks, the SPR move. Similarly, tail_1, head_1, and rNNI moves are analogous to the undirected NNI move. Like their directed counterparts, these moves prune an endpoint of an edge, and reattach it somewhere else so that the resulting graph is a network again.

Definition 2.48. Let U be a network containing the distinct edges $\{x, u\}$, $\{u, y\}$, $\{u, v\}$, and $\{x', y'\}$. Let G be the (multi-)graph obtained from U by

- *pruning* $\{u, v\}$ at u, i.e., replace $\{x, u\}$ and $\{u, y\}$ with a new edge $\{x, y\}$;
- and *reattach* $\{u, v\}$ at $\{x, y\}$, i.e., replace $\{x', y'\}$ by $\{x', u\}$ and $\{u, y'\}$.

If G is a phylogenetic network (i.e., a connected graph without parallel edges or self-loops), then we say G is the result of the *SPR move* moving the u-enpoint of $\{u, v\}$ from $\{x, y\}$ to $\{x', y'\}$ in U, which we also denote $\{x, u, y\} \xrightarrow{\{u,v\}} \{x', y'\}$ or $u \xrightarrow{\{u,v\}} (x', y')$ if the from-edge is not relevant. All nodes retain their labels if they were labelled.

Definition 2.49. An SPR move $\{x, u, y\} \xrightarrow{\{u,v\}} \{x', y'\}$ is an *NNI move* if $\{x, y\} \cap \{x', y'\} \neq \emptyset$.

For the definition of NNI moves on internally labeled networks, we need to point out that we made a choice in our definition. While moving the label with a moving endpoint seems natural using the definition of (r)SPR moves. It may be

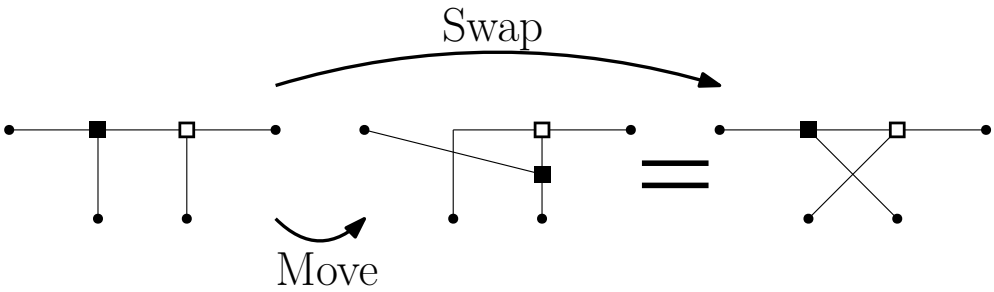


Figure 2.11: Two definitions for labelled NNI moves. This shows how different moves resulting in the same unlabelled network may have the same or different labellings.

less intuitive when one is used to NNI moves interpreted as actual neighbour interchanges (or *swaps*): replacing $\{x, y\}$ and $\{z, w\}$ with $\{x, z\}$ and $\{y, w\}$, where y and z are neighbours.[3] When internal labels are involved, one has to be careful with this equivalence, because the resulting labelling depends on the ways the moves are performed. Fortunately, these moves are equivalent, even when internal labels are introduced, as can be seen in Figure 2.11. However, if there are degree-2 nodes as well, the neighbour interchange definition is more flexible: two adjacent degree-2 nodes can be swapped using one move according to this definition, but not according to our definition as SPR_1 moves.

The distance of an SPR move is defined similar to the distance of an rSPR move.

Definition 2.50. An SPR move of $\{u, v\}$ from $\{x, y\}$ to $\{x', y'\}$ is a *distance-d* move, or an SPR_d move, if x' or y' is at distance less than d from x or y in the graph resulting from pruning $\{u, v\}$ from $\{p, c\}$.

With this definition, we can equivalently define an NNI move as an SPR_1 move.

Definition 2.51. The move $\{x, u, y\} \xrightarrow{\{u,v\}} \{x', y'\}$ in a network U is *valid* (or *allowed*) if the resulting graph is a network.

Definition 2.52. Let $\{u, v\}$ be an edge in a network U, then the u-end of $\{u, v\}$ is *movable* if u is a degree-3 node, and there is no triangle $\langle x, y, u\rangle$ in U with $v \neq x, y$.

Lemma 2.53. *An SPR move $u \xrightarrow{\{u,v\}} \{s, t\}$ is valid if all of the following hold:*

[3] Gambette et al. have proven the equivalence of the swap definition and the definition we use for rNNI moves [GvIJ+17b, Theorem 4].

- *the u-end of* $\{u, v\}$ *is movable;*
- *and* $v \notin \{s, t\}$.

If $u \in \{s, t\}$, then the definition for rSPR moves cannot be applied directly. However, like for moves in directed networks, we do allow for these moves, and we simply define them as moves that result in an isomorphic network.

2.5 Network spaces

In this section, we introduce the main structures we will study in this thesis: the spaces of phylogenetic networks. These spaces will be defined as graphs where the node sets are sets of networks, and there is an edge between two networks if there is a rearrangement move turning the one into the other. As these spaces are graphs, we also introduce several properties of graphs that we will study for these spaces.

Definition 2.54. Let n be a number of leaves, and $k \geq 0$ a number of reticulations. The set of all directed networks with n leaves and k reticulations is denoted $\mathcal{N}(n, k)$. The label set $X = X^l = \{x_1^l, \ldots, x_n^l\}$ of all these networks is arbitrarily fixed.

If, additionally, m is the number of degree-2 nodes, then $\dot{\mathcal{N}}(n, k, m)$ is the set of internally labeled subdivided directed networks with n leaves, k reticulations, and m degree-2 nodes. Again, the label sets are fixed arbitrarily for all these networks to $X^l = \{x_1^l, \ldots, x_n^l\}$, $X^r = \{x_1^r, \ldots, x_k^r\}$, $X^t = \{x_1^t, \ldots, x_{n+k-1}^t\}$, and $X^{(2)} = \{x_1^{(2)}, \ldots, x_m^{(2)}\}$.

The set of internally labeled networks without degree-2 nodes is simply denoted $\dot{\mathcal{N}}(n, k) = \dot{\mathcal{N}}(n, k, 0)$.

The sets $\mathcal{U}(n, k)$, $\dot{\mathcal{U}}(n, k)$ and $\dot{\mathcal{U}}(n, k, m)$ are defined similarly, but for the (subdivided internally labeled) sets of undirected networks.

Definition 2.55. The *space of phylogenetic networks* defined by the directed rearrangement moves of type M with n leaves and k reticulations, is the graph $\mathcal{N}_M(n, k)$ whose set of nodes is the set of phylogenetic networks $\mathcal{N}(n, k)$ with n leaves and k reticulations, and there is an edge between two networks if there is an M-move that transforms the one into the other (Figure 1.3).

The spaces $\dot{\mathcal{N}}_M(n, k)$, $\dot{\mathcal{N}}_M(n, k, m)$ are defined similarly (Figure 2.12). Moreover, if M is an undirected rearrangement move, then we also define the spaces of undirected networks $\mathcal{U}_M(n, k)$, $\dot{\mathcal{U}}_M(n, k)$, and $\dot{\mathcal{U}}_M(n, k, m)$.

The *k-th tier* [FHMW17, HMW16] of directed network space consists of all networks with k reticulations. As the number of leaves or reticulations

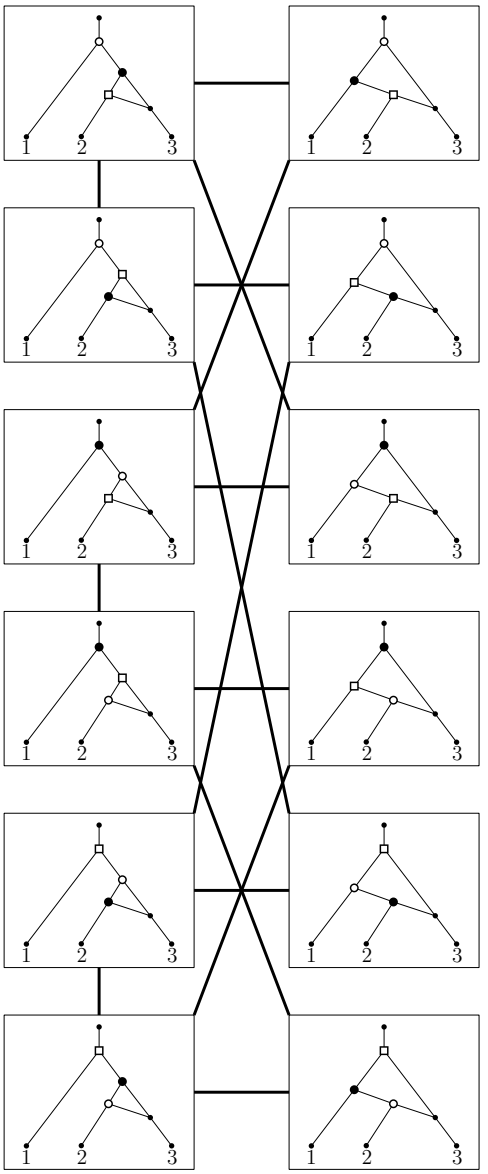


Figure 2.12: A part of $\dot{\mathcal{N}}_{\text{tail}}(3,1)$, corresponding to all labelings of two neighbours in $\mathcal{N}_{\text{tail}}(3,1)$. Note that the connectedness of this subgraph together with the connectedness of $\mathcal{N}_{\text{tail}}(3,1)$ implies the connectedness of $\dot{\mathcal{N}}_{\text{tail}}(3,1)$, as it shows that all permutations of the internal labels can be achieved using tail moves.

cannot be changed by a rearrangement move, we say the k-th tier is connected by a (directed) *move type* M if all $\mathcal{N}_M(n,k)$ are connected, and the space of phylogenetic networks is connected by a move type M if $\mathcal{N}_M(n,k)$ is connected for all $n, k \geq 0$.

Note that spaces that take the shape of a connected graph come with a metric: the distance between two nodes. Hence, a natural follow up question is to ask about the distances between pairs of networks.

Definition 2.56. Let N and N' be phylogenetic networks with the same leaf set in the same tier. We denote by $d_M(N, N')$ the *distance* between phylogenetic networks N and N' using rearrangement moves of type M. That is, $d_M(N, N')$ is the minimum number of M-moves needed to change N into N'.

Definition 2.57. Let $k \in \mathbb{N}$ be the number of reticulations, $n \in \mathbb{Z}_{\geq 2}$ be the number of leaves and M a type of rearrangement move. We denote with $\mathrm{diam}_M(n,k) := \mathrm{diam}(\mathcal{N}_M(n,k))$ (or $\mathrm{diam}(\mathcal{U}_M(n,k))$ for undirected networks) the *diameter* of tier-k of phylogenetic network space with n leaves using moves of type M.

Similarly, we define $\mathrm{diam}_M(n,k,0) = \mathrm{diam}(\dot{\mathcal{N}}_M(n,k))$ and $\mathrm{diam}_M(n,k,m) = \mathrm{diam}(\dot{\mathcal{N}}_M(n,k,m))$, where we use $\dot{\mathcal{U}}$ instead of $\dot{\mathcal{N}}$ if M is an undirected move.

This leads to the computational problems M Distance, where M has to be replaced with a specific move type. The input to this problem consists of two directed or undirected networks N, N', and the output is the distance $d_M(N, N')$. For example, for rSPR we have the following problem.

rSPR Distance
Input: Two directed networks N and N'.
Output: The distance $d_{\mathrm{rSPR}}(N, N')$.

As we will see in Chapter 7, it will be more interesting for some move types to restrict the input to networks from a fixed tier. Hence, for each move type M and tier k, we have the problem M Distance Tier-k.

Note that M Distance Tier-0 is simply the problem of computing rearrangement distances between pairs of trees. We also write M Distance Trees for these problems. Note that M Distance Trees is NP-hard for M equal to SPR [HDRCB08], rSPR [BS05], NNI [JLTZ00], and rNNI.[4]

Definition 2.58. If each move of a type M is also of type M', then we write $M \subseteq M'$.

[4]Follows very easily from [JLTZ00] and Lemma 2.64, but it is unclear whether this lemma was known (See Section 2.5.1).

The following diagram shows some inclusions of move types that follow immediately from the definitions.

$$\begin{array}{ccccc} & & \text{tail} & & \\ & \swarrow & & \searrow & \\ \text{tail}_1 & & \subseteq & & \\ & \searrow & & \swarrow & \\ & & \text{rNNI} & \subseteq & \text{rSPR} \\ & \swarrow & & \searrow & \\ \text{head}_1 & & \subseteq & & \\ & \searrow & & \swarrow & \\ & & \text{head} & & \end{array}$$

The next observation will be useful for obtaining bounds for the diameters of tail, head, and rSPR moves, and their local counterparts.

Observation 2.59. Let N and N' be networks with the same number of leaves, reticulations, and degree-2 nodes, and let $M \subseteq M'$, then $d_M(N, N') \geq d_{M'}(N, N')$.

This observation together with the diagram above gives the trivial bounds for diameters of spaces and distances between two networks for different types of moves shown in the following two diagrams.

$$\begin{array}{ccccc} & & d_{\text{tail}}(N, N') & & \\ & \swarrow & & \searrow & \\ d_{\text{tail}_1}(N, N') & & \leq & & \\ & \searrow & & \swarrow & \\ & & d_{\text{rNNI}}(N, N') & \leq & d_{\text{rSPR}}(N, N') \\ & \swarrow & & \searrow & \\ d_{\text{head}_1}(N, N') & & \leq & & \\ & \searrow & & \swarrow & \\ & & d_{\text{head}}(N, N') & & \end{array}$$

$$\begin{array}{ccccc} & & \text{diam}_{\text{tail}}(n, k) & & \\ & \swarrow & & \searrow & \\ \text{diam}_{\text{tail}_1}(n, k) & & \leq & & \\ & \searrow & & \swarrow & \\ & & \text{diam}_{\text{rNNI}}(n, k) & \leq & \text{diam}_{\text{rSPR}}(n, k) \\ & \swarrow & & \searrow & \\ \text{diam}_{\text{head}_1}(n, k) & & \leq & & \\ & \searrow & & \swarrow & \\ & & \text{diam}_{\text{head}}(n, k) & & \end{array}$$

2.5.1 Tree space diameter bounds

Diameters of tree spaces are well studied. Tree spaces can simply be viewed as the 0-th tier of network spaces, and the question of diameter bounds naturally extends to higher tiers. In this thesis, we mainly investigate this question, and we extend the diameter bounds to higher tiers of network space.

To establish and discuss these bounds, we compare them to diameter bounds for tree spaces. A few of these are the following.

Theorem 2.60 ([DGH11] Corollary 3.5)**.** *Let $n \geq 1$ be a number of leaves, then the space of trees with n leaves under rSPR moves has diameter of order* $\operatorname{diam}_{\mathrm{rSPR}}(n,0) = n - \Theta(\sqrt{n})$*, and, in particular,* $\operatorname{diam}_{\mathrm{rSPR}}(n,0) \leq n$*.*

Theorem 2.61 ([DGH11] Theorem 1.1)**.** *Let $n \geq 2$ be a number of leaves, then the space of undirected trees with n leaves under SPR has diameter bounded by* $n - 2\lceil\sqrt{n}\rceil + 1 \leq \operatorname{diam}_{\mathrm{SPR}}(n,0) \leq n - 3 - \lfloor\frac{\sqrt{n-2}-1}{2}\rfloor$*.*

Theorem 2.62 ([LTZ96] Lemmas 1 and 2)**.** *Let $n \geq 2$ be a number of leaves, then the space of undirected trees with n leaves under NNI has diameter bounded by* $\frac{n-2}{4}\log\left(\frac{2\sqrt{2}}{3e}(n-2)\right) \leq \operatorname{diam}_{\mathrm{NNI}}(n,0) \leq n\log(n) + 2n$*.*

The proof of this theorem follows from the following proposition, which bounds the distance between undirected caterpillars. Note that the intermediate networks are not necessarily caterpillars as well.

Proposition 2.63 ([LTZ96] Lemma 2)**.** *Let C and C' be two undirected caterpillars with $n \geq 2$ leaves, then* $d_{\mathrm{NNI}}(C, C') \leq n\log(n)$*.*

The diameter of tree space for rNNI moves is rarely mentioned explicitly. It is unclear why this is so, but it may be because it is rather trivial to show that the spaces $\mathcal{U}_{\mathrm{NNI}}(n+1)$ and $\mathcal{N}_{\mathrm{rNNI}}(n,0)$ are isomorphic. This can be shown by simply noting that the vertex sets can be mapped bijectively by rooting each undirected tree at the $(n+1)$-th leaf, and that each NNI move can be simulated with an rNNI move in the corresponding directed tree (Figure 2.13). Furthermore, this result easily extends to internally labeled trees.

Lemma 2.64. *Let $T, T' \in \dot{\mathcal{N}}(n,0,0)$, then $d_{\mathrm{rNNI}}(\dot{T}, \dot{T}') = 1$ if and only if $d_{\mathrm{NNI}}(U(\dot{T}), U(\dot{T}')) = 1$.*

Proof. The direction from rNNI to NNI is trivial, as each rNNI move on the rooted trees is an NNI move un their undirected version.

For the other direction, let the NNI move from $U(\dot{T})$ to $U(\dot{T})$ be the one that moves the u-end of $\{u,v\}$ from $\{x,y\}$ to $\{y,z\}$. First suppose that, in $\dot{T}$, the corresponding moving arc is (u,v). In that case, $\dot{T}'$ can be reached with

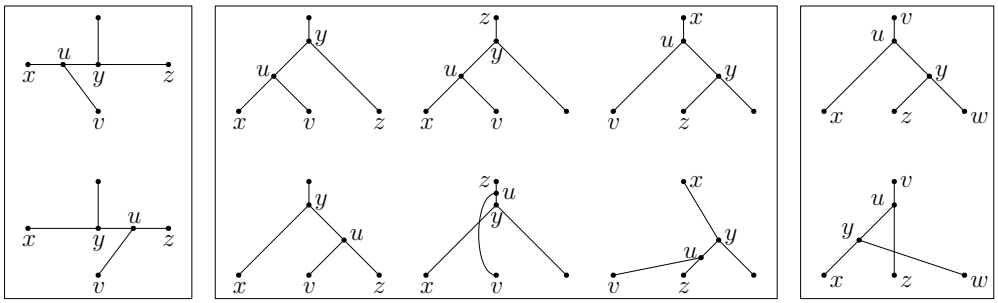


Figure 2.13: An illustration of the proof of Lemma 2.64. The NNI move on the undirected tree (left panel) can be replaced by the corresponding rNNI move if $\{u, v\}$ is directed as (u, v) (middle panel), and by the rNNI move $y \xrightarrow{(y,w)} (u, x)$ if it is directed as (v, u) (right panel).

the rNNI move (u, v) from the directed version of $\{x, y\}$ to the directed version of $\{y, z\}$.

Now suppose that the directed version of $\{u, v\}$ is directed as (v, u) in $\dot{T}$. In that case, the move cannot be performed as before, but we can still reach $\dot{T}'$ with one rNNI move. To see this, note that x and y are the children of u, and z is a child of y. Let $w \neq z$ be the other child of y, then $\dot{T}'$ can be reached from $\dot{T}$ with the rNNI move (y, w) from (u, z) to (u, x) (Figure 2.13). □

The following propositions follow immediately.

Proposition 2.65. *Let $C, C' \in \mathcal{N}(n, k)$ be two directed caterpillars with $n \geq 2$ leaves, then $d_{\mathrm{rNNI}}(C, C') \leq (n+1)\log(n+1)$.*

Proposition 2.66. *For all $n > 0$, the rNNI and NNI tree spaces are isomorphic, i.e., $\mathcal{N}_{\mathrm{rNNI}}(n, 0) \simeq \mathcal{U}_{\mathrm{NNI}}(n+1, 0)$ and $\mathcal{N}_{\mathrm{rNNI}}(n, 0, 0) \simeq \mathcal{U}_{\mathrm{NNI}}(n+1, 0, 0)$.*

Compare this with the following result from Atkins and McDiarmid's about rSPR and SPR moves, which results from the fact that an SPR move can reroot a rooted tree, but that achieving the same rearrangement without re-orienting arcs is less efficient.

Lemma 2.67 ([AM19], Example 2.3)**.** *For each $n > 0$, there are $T, T' \in \mathcal{N}(n-1, 0)$ such that $d_{\mathrm{rSPR}}(T, T') \geq (n-3)/2$ but $d_{\mathrm{SPR}}(U(T), U(T')) = 1$. The caterpillars $T = C(1, 2, \ldots, n-1)$ and $T' = C(n-1, \ldots, 2, 1)$ suffice.*

In their paper, Atkins and McDiarmid do not mention that this result does not hold for rNNI and NNI moves. Similarly, there is a related remark in the review article [SJ17], where they state that "SPR can differ depending on

whether the underlying trees are rooted or not", which is also written in [Ste16]. However, neither of these works relate this to NNI and rNNI moves. It is unclear whether this is because they deem the result trivial or well-known, or because they are not aware of the result at all. Whidden and Matsen do mention this fact for NNI and rNNI moves, but they also do so without proof [WMI18].

2.6 Orienting networks

Although directed and undirected networks are fundamentally different, as one is directed, and the other is not, there is a strong connection between the two. In this section, we will focus on this connection.

The underlying undirected graph of a directed network $N \in \mathcal{N}(n,k)$ is a $\{1,3\}$-graph with at least two degree-1 nodes (the root and a leaf). Hence, forgetting the direction of each arc and labeling the root with the label x_{n+1}^l, we get an undirected network $U \in \mathcal{U}(n+1,k)$. This unique network, the *underlying undirected network* of N, is denoted $U(N)$. Similarly, for a network $\ddot{N} \in \ddot{\mathcal{N}}(n,k,m)$, the underlying undirected network $U(\ddot{N})$ is an undirected network in $\ddot{\mathcal{U}}(n+1,k,m)$.

An undirected network U is called *orientable* if there exists a directed network N such that $U(N) = U$. Note that this definition differs slightly from the definition in [HvIJ+19], where the root has degree two and, to get such a node in an undirected network, an edge is subdivided.

Not all undirected networks are orientable. For example, consider the network shown in Figure 2.14. Indeed, in any orientation of this graph, one of the arcs adjacent to a subdivided copy of K_4 must be directed towards that copy of K_4. As there is no degree-1 vertex in that part of the graph, any orientation will have a cycle there.

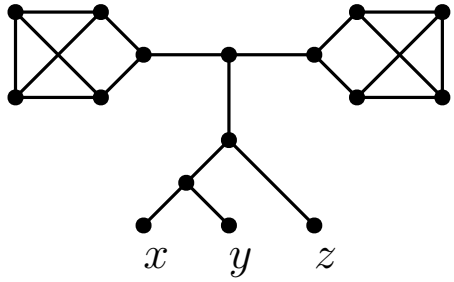


Figure 2.14: An undirected network that is not orientable.

Orientable networks can easily be recognized by looking at the blobs of the network. Two blobs are necessarily separated by an edge, whose removal disconnects the network. Any edge of an undirected network whose removal disconnects the network is called a *cut-edge*. If only one of the components resulting from the removal of an edge contains leaves, this edge is called a *redundant cut-edge*. The next lemma characterizes orientable networks via redundant cut-edges.

Lemma 2.68 ([JJE+18])**.** *An undirected network is orientable if and only if it has no redundant cut-edges.*

Proof. First let U be an undirected network with no redundant cut-edges. To show that U is orientable, we pick any leaf ρ of U and show how to construct a directed network with U as underlying graph and ρ as root. First, orient all cut-edges "away" from ρ. Then, it only remains to find a valid orientation of every blob. To this end, let B be a blob. After orienting all cut-edges, B has only one incoming edge (s, t) and, as (s, t) is not redundant, B has at least one outgoing edge (x, y). Since B is biconnected, there is a bipolar (i.e. acyclic) orientation of B with t as source and x as sink [LEC67]. Doing the same for all biconnected components, we get an acyclic orientation of U rooted at ρ.

Conversely, suppose that U has a redundant cut-edge e. Deleting e creates a component C without leaves. If we direct e towards C, then C has one source and no possible sinks (no leaves). If we direct e away from C then C has one sink but no possible sources (since the root is also a leaf). This implies there is no valid orientation of the edges in C and therefore in U. □

A *redundant terminal blob* of an undirected network U is a blob that is incident to exactly one cut-edge (which must be a redundant cut-edge). The next lemma, which follows directly from Lemma 2.68, characterizes orientable networks via redundant terminal blobs.

Lemma 2.69 ([JJE+18])**.** *An undirected network is orientable with any arbitrary leaf as the root if and only if it has no redundant terminal blobs.*

Given an orientable network, one can attempt to find an orientation (as directed network) of this network with a given leaf as the root. This problem has been studied extensively for the first time by Huber et al. ([HvIJ+19]). However, related results can already be found in Atallah's paper in 1983, where orientations with the fewest number of sinks are considered [Ata83].

An algorithm for this problem based on the proof of Lemma 2.68 would work as follows. Choose an arbitrary leaf of the network as the root and orient all cut-edges away from this root. To orient the internal edges of the blobs, let

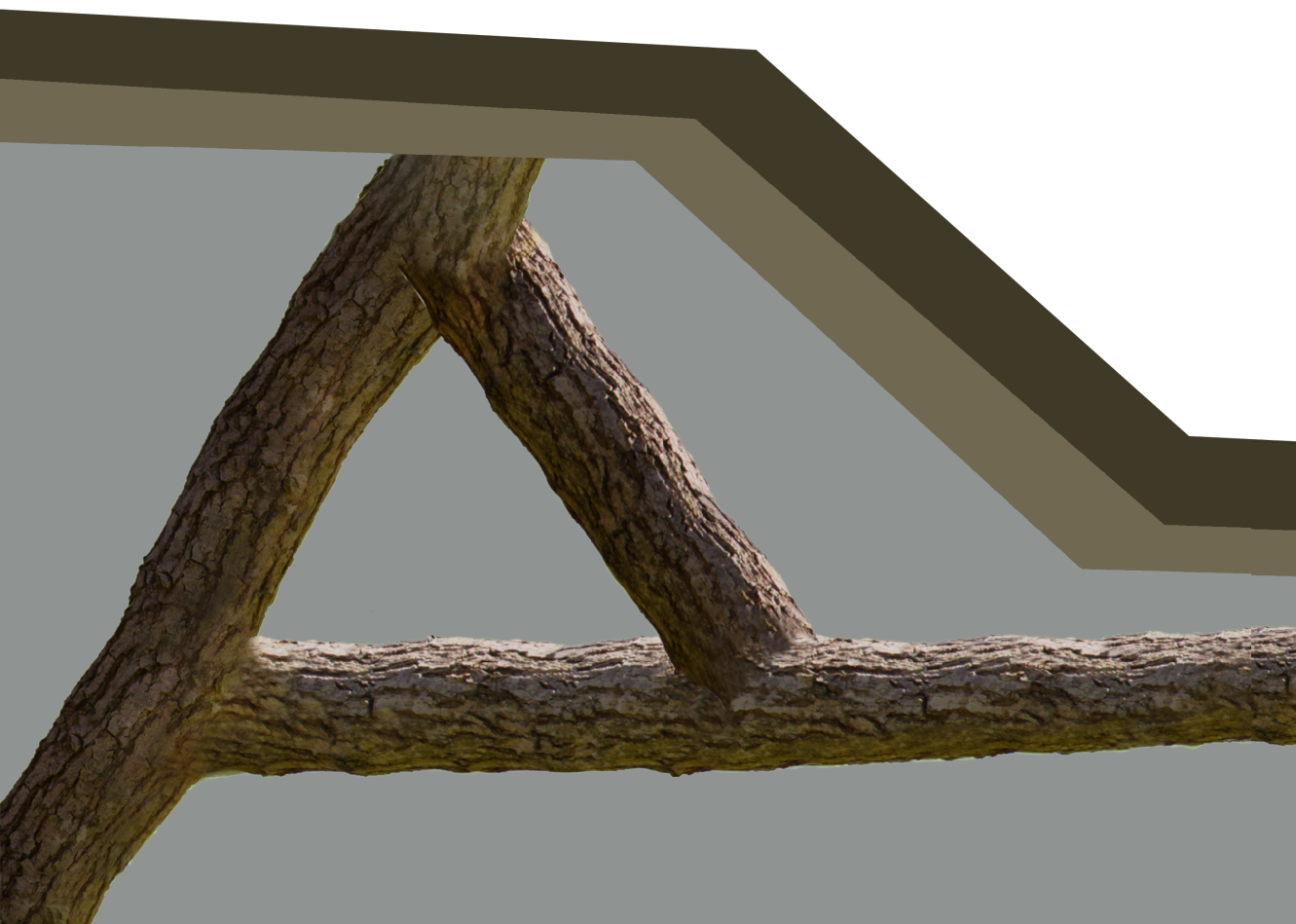
s be the head of the incoming edge of the blob and t the tail of an arbitrary other leaf-edge of the blob. Removing all cut-edges of the blob, we obtain a biconnected graph, which can be st-oriented [LEC67]. Doing this for all blobs, we obtain an orientation of the network.

Finding the st-orientation can be done in polynomial with a simple algorithm [LEC67]. If the running time is deemed more important, it can also be done in linear time [ET76]. Algorithms for this problem are still an active field of research (e.g., [SS19]).

Lemma 2.70 ([JJE+18])**.** *Each Echidna network is orientable.*

Proof. Let U be an arbitrary Echidna network. Then U consists of a cycle, leaf-edges, and edges between nodes of the cycle. As all edges are adjacent to the cycle, the network consists of one blob. This blob is not a redundant terminal blob, because all leaves are adjacent to this blob. Hence, the U has no redundant terminal blobs, and it is orientable by Lemma 2.69. □

3



Tail Moves

In this chapter, we study spaces of networks under tail moves. Tail moves are one of the natural extentions of rSPR moves from trees to networks, and are thus a logical choice in local search heuristics. Hence, it is important to know whether these moves are actually suited for this purpose. The most fundamental question in this assessment is whether the spaces are connected by tail moves. Establishing this connectedness result will be the main focus of this chapter.

To study the connectedness of the tiers of tail move spaces, we first focus on networks where only the leaves are labeled. Using a bottom-up approach, where we inductively build an isomorphism between two networks changing only the upper part, we show that there exists a sequence of tail moves between any pair of networks with the same number of leaves and reticulations (excepting one bad case; Section 3.2.1). This implies that the spaces $\mathcal{N}_{\text{tail}}(n,k)$ are connected for all $n \geq 1$ and $k \geq 0$, except when $n = 2$ and $k = 1$ (Theorem 3.8). This result still holds when we consider local moves, i.e., $\mathcal{N}_{\text{tail}_1}(n,k)$ is connected for all $(n,k) \neq (2,1)$ as well (Theorem 3.10). This is proven by showing that each tail move can be replaced by a sequence of local tail moves (Lemma 3.9).

Next, we extend these results to spaces with internally labeled nodes, and then also to networks with degree-2 nodes. For the first extension, we simply show that the tree nodes and the reticulations can be permuted (Propositions 3.19 and 3.22). Adding the degree-2 nodes to this result is quite simple: we show degree-2 nodes can, in most cases, be collected on the root path where they can be permutated (Section 3.3.2). This cannot be done in all cases, however, as the degree-2 nodes can be 'stuck' on some lowest arcs. We completely characterize the connected components of the spaces $\mathcal{N}_{\text{tail}}(n,k,m)$ that include degree-2 nodes.

All these connectedness proofs are constructive. Hence, by analysing these constructions carefully, we also obtain upper bounds on the diameters of the tiers of tail and tail_1 move spaces. In Chapter 7, the basic construction for leaf-labeled networks will be made explicit in the form of an algorithm, which finds a sequence of moves between any pair of networks in the same tier. We do not prove lower bounds in general, except for leaf-labeled networks in Subsection 3.2.3. There, we prove a lower bound based on the lower bound for trees.

In this chapter, each move is a tail move and movability always refers to tail-movability unless stated otherwise. We start with a short section discussing some basic results about tail-movability.

3.1 Tail movability

In this section, we relate movability of an arc to the existence of triangles, and we show that most arcs can be made movable by destroying a triangle. Recall that an arc (u, v) is movable iff u is a tree node and there is no triangle $\langle x, u, y\rangle$ with $y \neq v$.

Lemma 3.1 ([JJE+18] Observation 2.9)**.** *Suppose N has a triangle $\langle x, u, y\rangle_t \in N$ and an arc (u, v), then all arcs in $\langle x, u, y\rangle_t$ are movable, but (u, v) is not. After moving (x, y) up, the arc (u, v) is movable.*

Proof. First we check movability of the arcs of $\langle x, u, y\rangle$. The arc (x, u) is movable because there is no triangle $\langle t, x, b\rangle$ with $b \neq u$ for any $t, b \in V(N)$. Such a triangle would necessarily have $b = y$, and t would then have to be the common parent of x and y, which doesn't exist. Similarly, the arc (x, y) is movable because there is no triangle $\langle t, x, b\rangle$ with $b \neq y$ for any $t, b \in V(N)$. Lastly, (u, y) is movable, because there is no triangle $\langle x, u, b\rangle$ with $b \neq y$ for any $b \in V(N)$. □

Recall that moving (x, y) up is allowed iff x is not the child of the root and (x, y) is tail-movable. Using this, we can make most arcs movable using at most one move to destroy a triangle.

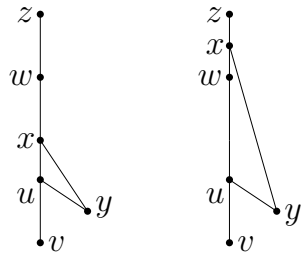


Figure 3.1: The immovable arc (u, v) whose tail u is a tree node can be made movable by moving the long arc of the triangle $\langle x, u, y\rangle$ up.

Lemma 3.2 ([JJE+18] Observation 2.9)**.** *Let u be a tree node, and (u, v) an immovable arc because of the presence of $\langle x, u, y\rangle$ in N. If x is not the child of the root, then (u, v) can be made movable by moving (x, y) up to an arc (z, w) above x. Furthermore, we may choose w to be the parent of x.*

Proof. As (u, v) is immovable and u is a tree node, the triangle $\langle x, u, y\rangle$ must be in N (Figure 3.1). As (x, y) is an arc in a triangle, it is movable by Lemma 3.1. If x is not the child of the root, there is an arc (z, w) with $w \neq x$ above x—in

particular, we can choose w to be the parent of x. By Corollary 2.47, we can move (x, y) to (z, w). After this move, the parents of y are u and x, but the parent of u is not equal to x. Hence, the arc (u, v) is now movable. □

The following lemma will play an important role in the arguments presented in the next chapter, by making it easy to show there is a movable arc where we need one.

Lemma 3.3 ([JJE+18] Observation 2.10)**.** *Let u be a tree node, then at least one of its child arcs is movable.*

Proof. Let x be the parent of u, and let y and v be the children of u. Note that the network includes at most one of the triangles $\langle x, u, y\rangle$ and $\langle x, u, v\rangle$. If it does not include $\langle x, u, y\rangle$, then (u, v) is movable; if it does not include $\langle x, u, v\rangle$, then (u, y) is movable. □

The following lemma will be used in this chapter to find a tail that can be moved down "sufficiently far". This will be used mainly to make an incoming arc of a low node movable.

Lemma 3.4 ([JJE+18] Lemma 2.12)**.** *Let x, y be nodes of a phylogenetic network N such that $x, y \notin \mathrm{LCA}(x, y)$. Then for any LCA u of x and y, one of the child arcs of u is a movable arc that is not both above x and above y.*

Proof. Consider an arbitrary $u \in \mathrm{LCA}(x, y)$. This LCA is a tree node, and both child arcs are above either x or y, but not both. Because at least one of the child arcs of a tree node is movable by Lemma 3.3, at least one of the child arcs of the LCA has the desired properties. □

3.2 Connectedness and diameter bounds

In this section, we study bounds for the diameters of tail move spaces, i.e., the maximum value of $d_{\mathrm{Tail}}(N_1, N_2)$ and $d_{\mathrm{Tail}_1}(N_1, N_2)$ over all possible networks N_1 and N_2 with the same reticulation number.

3.2.1 Bottom-up isomorphism

To obtain our diameter bound, we incrementally build an isomorphism between any given pair of networks. We keep an isomorphism between two down-closed sets, one in each of the networks.[1] To start, we let the down-closed sets consist

[1] The down-closed sets were initially thought of as parts of the networks below a certain line, which we drew in green. Hence, in our discussions, we called this technique the "green line approach". This name still pops up in discussions and source files related to this proof.

of only the leaves. Then, in each step, we add one node to each of the down-closed sets, while keeping them labeled isomorphic. To achieve this, we may need to change the networks. Of course, these changes are made using tail moves, and we aim to use as few tail moves for each step as possible.

Lemma 3.5 ([JJE+18] Lemma 4.6 Case 1)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ such that $N_1 \not\simeq N_A$, and let $Y_1 \supseteq L(N_1)$ and $Y_2 \supseteq L(N_2)$ be down-closed sets of nodes of N_1 and N_2 such that $N_1[Y_1] \overset{\phi}{\simeq}_X N_2[Y_2]$. Suppose there is a lowest node u_2 in $N_2 \setminus Y_2$ such that u_2 is a reticulation. Then there is a network N_1' with a down-closed set Y_1' such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\text{tail}}(N_1, N_1') \leq 3$.*

Proof. We first observe that ϕ maps reticulations (tree nodes) of N_1 to reticulations (tree nodes) of N_2. Indeed, every node in $N_1[Y_1]$ is mapped to a node in $N_2[Y_2]$ of the same outdegree, and the tree nodes are exactly those with outdegree 2. It follows that Y_1 and Y_2 contain the same number of reticulations and the same number of tree nodes. As N_1 and N_2 have the same number of nodes and reticulations by Observation 2.7, it also follows that $V(N_1) \setminus Y_1$ and $V(N_2) \setminus Y_2$ contain the same number of reticulations and the same number of tree nodes.

Let x_2 be the single child of u_2. Then x_2 is in Y_2, and therefore there exists a node $x_1 = \phi^{-1}(x_2) \in Y$. Furthermore, x_1 has the same number of parents in N_1 as x_2 does in N_2 (because the networks are binary and x_1 has the same number of children in N_1 as x_2 does in N_2 by the isomorphism between $N_1[Y_1]$ and $N_2[Y_2]$), and the same number of parents in Y_1 as x_2 has in Y_2. Thus x_1 has at least one parent z_1 such that z_1 is not in Y_1.

We now split into two cases:

1. **z_1 is a reticulation:** In this case, set $N_1' = N_1$ and let Y_1' be the down-closed set $Y_1 \cup \{z_1\}$ in N_1'. We may then extend ϕ to an isomorphism between $N_1'[Y_1']$ and $N_2[Y_2 \cup \{u_2\}]$ by setting $\phi(z_1) = u_2$ (see Figure 3.2). As $N_1 = N_1'$, we have $d_{\text{tail}}(N_1, N_1') = 0 \leq 3$.

2. **z_1 is not a reticulation:** then z_1 cannot be the root of N_1. Indeed, this would imply $|N_1 \setminus Y_1| = |N_2 \setminus Y_2| = 1$, and u_2 would be the root of N_2. This contradicts the fact that u_2 is a reticulation, so z_1 must be a tree node. It follows that the arc (z_1, x_1) is movable, unless there is a triangle $\langle c_1, z_1, d_1 \rangle \in N$ with $d_1 \neq x_1$.

    a) **(z_1, x_1) is movable:** In this case, let u_1 be any reticulation in $N_1 \setminus Y_1$ —such a node must exist, as u_2 exists and $N_1 \setminus Y_1$ has the same number of reticulations as $N_2 \setminus Y_2$. Let v_1 be the child of u_1 and observe that the arc (u_1, v_1) is not below x_1 (as $x_1 \in Y_1$ and

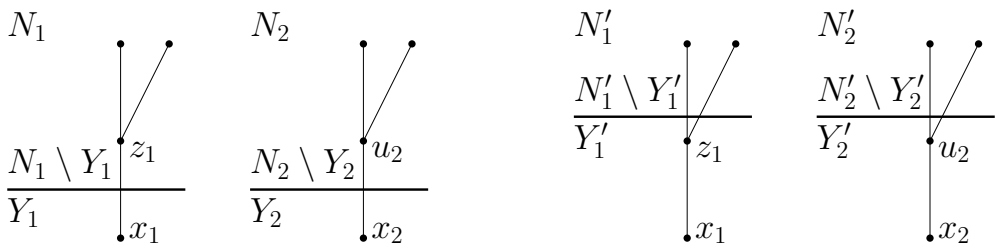


Figure 3.2: Proof of Lemma 3.5, Case 1: If u_2 is a lowest reticulation in $N_2 \setminus Y_2$ with child x_2, and the node $x_1 = \phi^{-1}(x_2) \in Y_1$ has a reticulation parent z_1 in $N_1 \setminus Y_1$, then we may add z_1 to Y_1 and u_2 to Y_2 to obtain Y_1' and Y_2'. The networks weren't changed, to $N_1' = N_1$ and $N_2' = N_2$.

$u_1 \notin Y_1$). If $v_1 = x_1$, then u_1 is a reticulation parent of x_1 that is not in Y_1, and by substituting u_1 for z_1, we have case 1. Hence, we may assume $v_1 \neq x_1$. Then it follows from Lemma 2.44 that the tail move $z_1 \xrightarrow{(z_1,x_1)} (u_1, v_1)$ resulting in M_1 is valid (see Figure 3.3). Thus, M_1 contains the arcs $(u_1, z_1), (z_1, v_1), (z_1, x_1)$. Note that if z_1 is immediately below u_1 in N_1 (i.e., $z_1 = v_1$) then in fact $M_1 = N_1$. In this case, we may skip the move from N_1 to M_1, and in what follows replace v_1 by the other child of z_1.

Note that (z_1, v_1) is movable in M_1, since the parent u_1 of z_1 is a reticulation node, and therefore deleting (z_1, v_1) and suppressing z_1 cannot create parallel arcs. Let w_1 be one of the parents of u_1 in M_1. Then the tail move $z_1 \xrightarrow{(z_1,v_1)} (w_1, u_1)$ resulting in M_1' is valid (see Figure 3.3). Indeed, $u_1 \neq v_1$ and (w_1, u_1) is not below v_1, as this would imply a cycle in M_1.

In M_1', the reticulation u_1 is the parent of x_1 (as z_1 was moved), so Case 1 applies to M_1' and N_2.

Set $N_1' = M_1'$ and let Y_1' be the down-closed set $Y_1 \cup \{u_1\}$ in N_1'. We may then extend ϕ to an isomorphism between $N_1'[Y_1']$ and $N_2[Y_2 \cup \{u_2\}]$ by setting $\phi(u_1) = u_2$. By construction, we have $d_{\text{tail}}(N_1, N_1') \leq 2 \leq 3$.

b) **There is a triangle $\langle c_1, z_1, d_1 \rangle \in N_1$ with $d_1 \neq x_1$:** As c_1 has outdegree 2 it is not the root of N_1, so let b_1 denote the parent of c_1.

   i. **b_1 is not the root of N_1:** In this case, let a_1 be a parent of b_1 in N_1. By Lemma 3.2, the tail move $c_1 \xrightarrow{(c_1,d_1)} (a_1, b_1)$ is valid, and (z_1, x_1) is movable in the resulting network M_1.

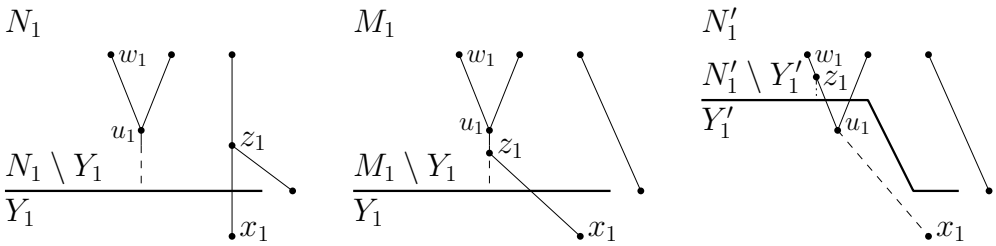


Figure 3.3: Proof of Lemma 3.5, Case 2a: If (z_1, x_1) is movable, the tail move of $z_1 \xrightarrow{(z_1,x_1)} (u_1, v_1)$ resulting in M_1 is valid. In M_1, the parent z_1 of x_1 is the child of u_1; so after the tail move $z_1 \xrightarrow{(z_1,v_1)} (w_1, u_1)$, the reticulation u_1 becomes a parent of x_1.

Now Case 2a (which uses at most 2 moves) applies to M_1 and N_2. It follows that there is a network N_1' with down-closed set Y_1' such that $N_1'[Y_1'] \simeq N_2[Y_2 \cup \{u_2\}]$ and $d(N_1, N_1') \leq 1 + d(M_1, N_1') \leq 1 + 2 = 3$.

ii. **b_1 is the root of N_1:** In this case, we observe that $d_1 \in Y$ implies that every reticulation in N_1 is in Y_1. This contradicts the fact that $N_1 \setminus Y_1$ and $N_2 \setminus Y_2$ contain the same number of reticulations, so we may assume that $d_1 \notin Y_1$. Then, we may proceed as follows. Let N_1' be the network derived from Y by the *head*-move $d_1 \xrightarrow{(c_1,d_1)} (z_1, x_1)$. We show that it is possible to replace this head move with a sequence of three tail moves.
Let e_1 be the child of d_1 in N_1. We note that if e_1 is a reticulation, then one of the parents of e_1 is a descendant of x_1 (otherwise z_1, b_1, or c_1 would have to be a parent of e_1, which is not the case). Thus, if e_1 is a reticulation, then it is a descendant of x_1. By a similar argument, if x_1 is a reticulation, then it is a descendant of e_1. Hence, either $x_1 = e_1$ and we can immediately extend the isomorphism by setting $\phi(d_1) = u_2$, or we may assume that one of e_1 and x_1 is not a reticulation, and that, furthermore, at least one of e_1 and x_1 is a tree node. Indeed, if both are leaves, then $N_1 \simeq N_A$; and, if one of e_1 and x_1 is a reticulation, then the other must be its ancestor, and must therefore be a tree node.
Our approach in this case will be to "swap" the positions of e_1 and x_1 via a sequence of tail moves. First, assume that x_1 is a tree node. If e_1 is a child of x_1 and q_1 is the other child,

we apply the tail move $x_1 \xrightarrow{(x_1,q_1)} (d_1, e_1)$. Otherwise, let the children of x_1 be s_1 and t_1. As $e_1 \notin \{s_1, t_1\}$ we may apply the sequence of tail moves $x_1 \xrightarrow{(x_1,s_1)} (d_1, e_1)$, $x_1 \xrightarrow{(x_1,e_1)} (z_1, t_1)$, $x_1 \xrightarrow{(x_1,t_1)} (d_1, s_1)$ (Figure 3.4). In both cases, the resulting network is isomorphic to the network N_1' resulting from the head move $d_1 \xrightarrow{(c_1,d_1)} (z_1, x_1)$. The case that e_1 is a tree node can be handled in a similar manner.
Observe that, in N_1', x_1 has a non-reticulate parent in $N_1' \setminus Y_1$, and that $N_1'[Y_1] \overset{\phi}{\simeq}_X N_1[Y_1]$. Hence, Case 1 applies to N_1' and N_2, so there exists a down-closed set Y_1' in N_1' such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$. Finally, by construction, we have $d_{\text{tail}}(N_1, N_1') \leq 3$. □

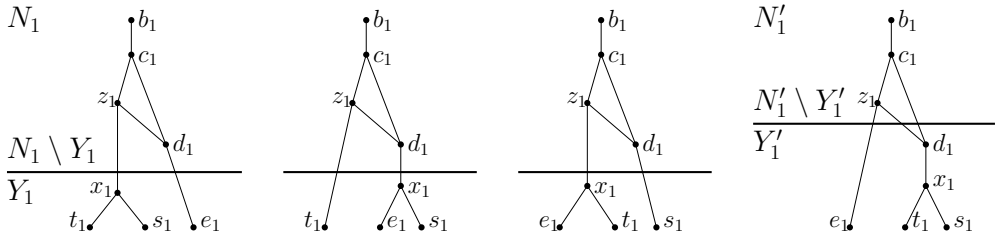


Figure 3.4: Proof of Lemma 3.5, Case 2(b)ii: Note that the sequence of tail moves depicted results in the same network as the head move $d_1 \xrightarrow{(c_1,d_1)} (z_1, x_1)$. After this sequence of moves, x_1 has a reticulation parent.

Lemma 3.6 ([JJE+18] Lemma 4.6 Case 3)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ such that $N_1 \not\simeq N_A$, and let $Y_1 \supseteq L(N_1)$ and $Y_2 \supseteq L(N_2)$ be down-closed sets of nodes of N_1 and N_2 such that $N_1[Y_1] \overset{\phi}{\simeq}_X N_2[Y_2]$. Suppose that each lowest node in $N_2 \setminus Y_2$, and each lowest node in $N_1 \setminus Y_1$ is a tree node, and let u_2 be such a lowest node in $N_2 \setminus Y_2$. Then there is a network N_1' with a down-closed set Y_1' such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\text{tail}}(N_1', N_1) \leq 2$.*

Proof. Let x_2 and y_2 be the children of u_2. As $x_2, y_2 \in Y_2$, there exist nodes $x_1 = \phi^{-1}(x_2)$ and $y_1 = \phi^{-1}(y_2)$ in Y_1. Furthermore, x_1 has the same number of parents in N_1 as x_2 does in N_2, and the same number of parents in Y_1 as x_2 has in Y_2 (again, because the networks are binary and x_1 has the same number of children in N_1 as x_2 does in N_2 by the isomorphism between $N_1[Y_1]$ and $N_2[Y_2]$). Thus, x_1 has at least one parent not in Y_1. Similarly, y_1 has at least one parent not in Y_1. We now split into two cases.

1. **x_1 and y_1 have a common parent u_1 in $N_1 \setminus Y_1$:** In this case, set $N_1' = N_1$ and let Y_1' be the down-closed set $Y_1 \cup \{u_1\}$ in N_1'. We may then extend ϕ to an isomorphism between $N_1'[Y_1']$ and $N_2[Y_2 \cup \{u_2\}]$ by setting $\phi(u_1) = u_2$ (see Figure 3.5). As $N_1 = N_1'$, we have $d_{\text{tail}}(N_1, N_1') = 0 \leq 2$.

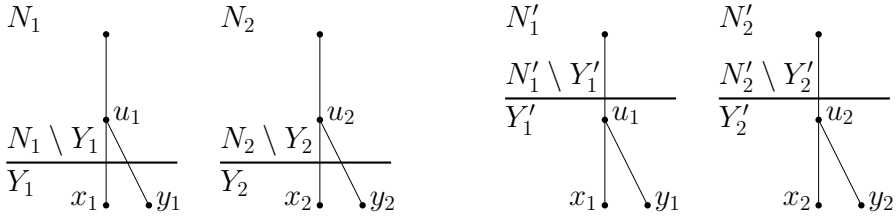


Figure 3.5: Proof of Lemma 3.6, Case 1: If u_2 is a lowest reticulation in $N_2 \setminus Y_2$ with children x_2, y_2, and the nodes $x_1, y_1 \in Y_1$ corresponding to x_2, y_2 share a reticulation parent u_1 in $N_1 \setminus Y_1$, then we may add u_1 to Y_1 and u_2 to Y_2.

2. **x_1 and y_1 do not have a common parent in $N_1 \setminus Y_1$:** In this case, let z_1^x be a parent of x_1 not in Y, and let z_1^y be a parent of y_1 not in Y_1. Recall that z_1^x and z_1^y are both tree nodes. Moreover, either one of $(z_1^x, x_1), (z_1^y, y_1)$ is movable, or neither (z_1^x, x_1) nor (z_1^y, y_1) is movable.

   a) **(z_1^x, x_1) is movable:** In this case, observe that the arc (z_1^y, y_1) is not below x_1 (as $x_1 \in Y_1$ and $z_1^y \notin Y$), and that $x_1 \neq y_1$. Then, by Lemma 2.44, the tail move $z_1^x \xrightarrow{(z_1^x, x_1)} (z_1^y, y_1)$ is valid.
   Let N_1' be the network derived from N_1 by applying this tail move (see Figure 3.6) and set $Y_1' = Y_1 \cup \{z_1^x\}$. Then, as z_1^x is a common parent of x_1 and y_1 in N_1' and $N_1'[Y_1] \simeq N_1[Y_1]$, we have $N_1'[Y_1'] \overset{\phi}{\simeq}_X N_2[Y_2]$. Additionally, we have $d_{\text{tail}}(N_1', N_1) = 1 \leq 2$.

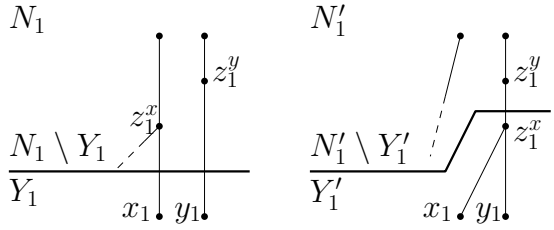


Figure 3.6: Proof of Lemma 3.6, Case 2a: If (z_1^x, x_1) is movable, we may move the tail of (z_1^x, x_1) to (z_1^y, y_1) so that x_1 and y_1 share the parent z_1^x in the resulting network.

b) **(z_1^y, y_1) is movable:** This case can be handled by exchanging the roles of x_1 and y_1 and applying Case 2a.

c) **Neither (z_1^x, x_1) nor (z_1^y, y_1) is movable:** In this case, there must exist nodes d_1^x, c_1^x, d_1^y, and c_1^y such that $\langle c_1^x, z_1^x, d_1^x\rangle \in N_1$, and $\langle c_1^y, z_1^y, d_1^y\rangle \in N_1$. Moreover, as z_1^x and z_1^y are different nodes with one parent each, we have $c_1^x \neq c_1^y$ and it follows that one of c_1^x, c_1^y is not the child of the root of N_1. Suppose without loss of generality that c_1^x is not the child of the root. Then, (z_1^x, x_1) can be made movable by the upwards tail move $c_1^x \xrightarrow{(c_1^x, d_1^x)} (a_1^x, b_1^x)$, where b_1^x is the parent of c_1^x and a_1^x is a parent of b_1^x (Lemma 3.2). In the resulting network M_1, (z_1^x, x_1) is movable, so Case 2a applies to M_1 and N_2.

It follows that there is a network N_1' with down-closed set Y_1' such that $N_1'[Y_1'] \simeq N_2[Y_2 \cup \{u_2\}]$ and $d_{\text{tail}}(N_1', N_1) \leq d_{\text{tail}}(N_1, M_1) + d_{\text{tail}}(M_1, N_1') \leq 2$. □

Lemma 3.7 ([JJE+18] Lemma 4.6)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ such that $N_1 \not\simeq N_A$, and let $Y_1 \supseteq L(N_1)$ and $Y_2 \supseteq L(N_2)$ be down-closed sets of nodes of N_1 and N_2 such that $N_1[Y_1] \overset{\phi}{\simeq}_X N_2[Y_2]$. Suppose $N_1 \setminus Y_1$ contains t tree nodes and r reticulations of N_1, then $d_{\text{Tail}}(N_1, N_2) \leq 3r + 2t$.*

Proof. We prove the claim by induction on t and r. First we consider the base case where $t = r = 0$, i.e., $|N_1 \setminus Y_1| \leq 1$. If $|N_1 \setminus Y_1| = 0$, then $N_1 = N[Y_1]$, which is isomorphic to $N_2[Y_2] = N_2$, and so $d_{\text{tail}}(N_1, N_2) = 0 = 3 \cdot 0 + 2 \cdot 0$. If $|N_1 \setminus Y_1| = 1$, then, as Y_1 is down-closed, $N_1 \setminus Y_1$ consists of ρ_1, the root of N_1, and by a similar argument $N_2 \setminus Y_2$ consists of ρ_2. Let x_1 be the only child of ρ_1 and x_2 the only child of ρ_2. Because x_1 (resp. x_2) is the only node of indegree 0, outdegree 2 in $N_1[Y_1]$ (resp. $N_2[Y_2]$), it follows that $\phi(x_1) = x_2$. Thus, we can extend ϕ to an isomorphism between N_1 and N_2 by setting $\phi(\rho_1) := \rho_2$. Again, this implies that $d_{\text{tail}}(N_1, N_2) = 0 \leq 3 \cdot 0 + 2 \cdot 0$.

Now, we may assume that $|N_1 \setminus Y_1| > 1$, so the sets of lowest nodes of $N_1 \setminus Y_1$ and $N_2 \setminus Y_2$ are non-empty. As $L(N_1) \subset Y_1$, each node in these sets is either a tree node or a reticulation. We consider three cases depending on the composition of these sets.

1. **There exists a lowest node u_2 of $N_2 \setminus Y_2$ such that u_2 is a reticulation.** By Lemma 3.5, there exists a network network N_1' with a down-closed set Y_1' such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\text{tail}}(N_1', N_1) \leq 3$. We now decompose the distance $d_{\text{tail}}(N_1, N_2) \leq d_{\text{tail}}(N_1, N_1') + d_{\text{tail}}(N_1', N_2)$. As $Y_2 \cup \{u_2\}$ contains one more reticulation than Y_2, so does Y_1' contain one more reticulation than Y_1. Hence, by induction $d_{\text{tail}}(N_1', N_2) \leq 3(r-1)+2t$, and it follows that $d_{\text{tail}}(N_1, N_2) \leq 3+3(r-1)+2t = 3r+2t$.

2. **There exists a lowest node u_1 of $N_1 \setminus Y_1$ such that u_1 is a reticulation.** By symmetric arguments to Case 1—exchanging the roles of N_1 and N_2—we have that there is a sequence of at most $3r + 2t$ tail moves turning N_2 into N_1. As all tail moves are reversible, there is also a sequence of at most $3r + 2t$ tail moves turning N_1 into N_2.

3. **No lowest node of $N_1 \setminus Y_1$ nor any lowest node of $N_2 \setminus Y_2$ is a reticulation.** By Lemma 3.6, there exists a network network N_1' with a down-closed set Y_1' such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\text{tail}}(N_1', N_1) \leq 2$. We now decompose the distance $d_{\text{tail}}(N_1, N_2) \leq d_{\text{tail}}(N_1, N_1') + d_{\text{tail}}(N_1', N_2)$. As $Y_2 \cup \{u_2\}$ contains one more tree node than Y_2, so does Y_1' contain one more tree node than Y_1. Hence, by induction $d_{\text{tail}}(N_1', N_2) \leq 3r + 2(t-1)$, and it follows that $d_{\text{tail}}(N_1, N_2) \leq 2 + 3r + 2(t-1) = 3r + 2t$. □

For all $(n, k) \neq (2, 1)$, we get the following bound on the distance between any two directed networks in the same tier by setting $Y_1 = L(N_1)$ and $Y_2 = L(N_2)$ and noting that a network in $\mathcal{N}(n, k)$ has $n + k - 1$ tree nodes and k reticulations (Observation 2.7)

This leaves the space $\mathcal{N}_{\text{tail}}(2, 1)$, which contains exactly two networks. Both consist of a triangle $\langle a, b, c \rangle_t$ where c is the child of the root, and the child x of c and the child $y \neq c$ of b are leaves. The two networks are found by swapping the labels of the leaves. By checking all possible moves, it is easy to see that there is no move that transforms the one into the other. Hence, combining all of the above, we obtain the following theorem.

Theorem 3.8 ([JJE+18] Theorem 4.7)**.** *Let $n \in \mathbb{Z}_{\geq 1}$ and $k \in \mathbb{N}$ such that $(n, k) \neq (2, 1)$, then $\text{diam}_{\text{tail}}(n, k) \leq 2n + 5k - 2$. The space $\mathcal{N}_{\text{tail}}(2, 1)$ is disconnected.*

As rSPR moves consist of head moves and tail moves, Theorem 3.8 also provides an upper bound for the number of rSPR moves needed to turn N_1 into N_2. In Chapter 5 (Theorem 5.19) we modify these arguments to improve this bound for the rSPR diameter.

3.2.2 The diameter of tail_1 spaces

Here, we show that the tail_1 move space $\mathcal{N}_{\text{tail}_1}(n, k)$ is connected whenever the tail move space $\mathcal{N}_{\text{tail}}(n, k)$ is connected, and we give diameter bounds for these spaces of local tail moves. The main difference between tail moves and tail_1 moves is that in a tail_1 move the tail may only be moved a small distance, whereas in tail moves the tail of the moving arc may be moved to any place in the network. The following lemma shows that moving a tail over a long

distance can always be done in steps, moving the tail to an adjacent arc each step. This way, we show that each tail move can be replaced by a bounded number of tail_1 moves. Hence, we find an upper bound for the diameter of tail_1 spaces.

Lemma 3.9 ([JJE+18] Lemma 4.11)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ be two networks such that N_2 is the result of $u \xrightarrow{(u,v)} (s,t)$ in N_1. Then there is a sequence of at most $d \leq n + 3k - 1$ tail_1 moves from N_1 to N_2 which moves the arc (u,v) in each step, where d is the length of an up-down path between u and s.*

Proof. Note that there exist directed paths P_u and P_s from any $t \in \text{LCA}(u,s)$ to u and s (which are not necessarily unique). We prove that $u \xrightarrow{(u,v)} (a)$ is valid for all arcs $a \in P_u \cup P_s$. This gives a sequence of tail_1 moves: Indeed, if $b, c \in P_u \cup P_s$ share a node and both moves $u \xrightarrow{(u,v)} b$ and $u \xrightarrow{(u,v)} c$ are valid, then there is a tail_1 move $u \xrightarrow{(u,v)} c$ between the resulting networks.

To see that $u \xrightarrow{(u,v)} (x,y)$ is valid in N_1 for all $(x,y) \in P_u \cup P_s$, we check the conditions of Lemma 2.44. Firstly, (u,v) is movable, because the move $u \xrightarrow{(u,v)} (s,t)$ is valid. Secondly, $v \neq y$, as that would imply v lies above u or above s, which contradicts the acyclicity of N_1 or the validity of the move $u \xrightarrow{(u,v)} (s,t)$. Lastly, v is not above y. Indeed, if $(x,y) \in P_u$ this implies there is a cycle in N_1 consisting of the path from v to y, together with the path from y to u via P_u. Similarly, if $(x,y) \in P_s$ and v were above y, there would be a cycle in N_2 consisting of the path from v to y together with the path from y to s in P_s and the arcs (s,u) and (u,v).

Noting that a path between two nodes uses at most $|E| - n$ arcs, we see that at most $|E| - n = n + 3k - 1$ tail_1 moves (Observation 2.7) suffice to simulate any tail move. □

Lemma 3.9 immediately gives us upper bounds on tail_1 distances and diameters in terms of the tail move distances and diameters.

Theorem 3.10 ([JJE+18])**.** *For all $n, k \in \mathbb{N}$ and all $N, N' \in \mathcal{N}(n,k)$, the tail move and local tail move distances satisfy $d_{\text{tail}_1}(N, N') \leq (n+3k-1) d_{\text{tail}}(N, N')$.*

Proof. Let S be a sequence of tail moves between N and N' of length $d_{\text{tail}}(N, N')$. By Lemma 3.9, each tail move in S can be replaced by a sequence of at most $n + 3k - 1$ tail_1 moves. This results in a sequence of tail_1 moves between N and N' of length at most $(n + 3k - 1) d_{\text{tail}}(N, N')$. □

Corollary 3.11 ([JJE+18])**.** *For all $n, k \in \mathbb{N}$, the tail move and local tail move diameters satisfy $\text{diam}_{\text{tail}_1}(n,k) \leq (n + 3k - 1) \,\text{diam}_{\text{tail}}(n,k)$.*

Corollary 3.12 ([JJE+18]). *For all $n, k \in \mathbb{N}$, the local tail move diameters satisfy $\operatorname{diam}_{\text{tail}_1}(n,k) \leq (n+3k-1)(2n+5k-2)$, except when $(n,k) = (2,1)$.*

3.2.3 Lower bounds

We will discuss lower bounds for $\mathcal{N}_{\text{tail}_1}(n,k)$ in Chapter 5. There, using the fact that each tail_1 move is an rNNI move, we give a bound of order $O_{n,k}((n+k)\log(n+k))$. For tail moves, we give a lower bound based on the lower bound for trees.

Lemma 3.13. *Let N, N' be networks such that $d_{\text{tail}}(N,N') \leq 1$. Then for each $T \in \mathcal{T}(N)$ there is a tree $T' \in \mathcal{T}(N')$ such that $d_{\text{tail}}(T,T') \leq 1$.*

Proof. Let the tail move from N to N' be $u \xrightarrow{(u,v)} (x,y)$, and let $T \in \mathcal{T}(N)$ be arbitrary with embedding i.

If $(u,v) \notin i(T)$, then T can be embedded in N without the arc (u,v). This means T can also be embedded in N'. Hence, we assume $(u,v) \in i(s,t)$ for some arc (s,t) of T. Let $P \subseteq i(s,t)$ be the path through $i(T)$ from $i(s)$, to $i(v)$.

If $(x,y) \in i(p,q)$ for some arc $(p,q) \in A(T)$, consider the subdivided tree $T'_s = i(T) \setminus (P \cup \{(x,y)\}) \cup \{(x,u),(u,y)\}$. As T'_s only uses arcs from N', it is a subgraph of N' and $T' = S(T'_s) \in \mathcal{T}(N')$. Furthermore, $d_{\text{tail}}(T,T') \leq 1$ as T' can be obtained from T by the move $s \xrightarrow{(s,t)} (p,q)$.

Otherwise, let Q be a path from a node w in the embedding of T to x that does not use any arcs in the embedding of T. This path exists as $(x,y) \notin i(T)$, so we can find it by going up from x using arcs not in $i(T)$ until we reach a node $w \in i(p,q)$ for some arc (p,q) of T (it is possible that $x = w$ and $Q = \emptyset$). Set $T'_s = i(T) \setminus (P \cup \{(x,y)\}) \cup Q \cup \{(x,u)\}$, and note that T'_s is a subgraph of N' again, as T'_s only uses arcs from N'. Hence, $T' = S(T'_s) \in \mathcal{T}(N')$ and $d_{\text{tail}}(T,T') \leq 1$ as T' can be obtained from T by the move $s \xrightarrow{(s,t)} (p,q)$. □

Proposition 3.14. *Let $T, T' \in \mathcal{N}(n,0)$ be any pair of trees, then for any $k > 1$, the two ladder trees $N, N' \in \mathcal{N}(n,k)$ with underlying trees T and T' have distance $d_{\text{tail}}(T,T') \leq d_{\text{tail}}(N,N') \leq d_{\text{tail}}(T,T') + 2$.*

Proof. Each tail move on the network does at most one tail move on each of the embedded trees (Lemma 3.13). Hence, for N, N', we have $d_{\text{tail}}(N,N') \geq d_{\text{tail}}(T,T')$. On the other hand, any sequence of tail moves on the trees T and T' can directly be applied to the networks as well. Hence, to change N into N' we fix the underlying tree with at most $d_{\text{tail}}(T,T')$ moves, and then swap the two pendant subtrees below the ladder with at most two moves to obtain the bound $d_{\text{tail}}(N,N) \leq d_{\text{tail}}(T,T') + 2$. □

Using the lower bound for diameters of tree space (Theorem 2.60), we can now get a lower bound on the diameter of tail network spaces.

Theorem 3.15. *The diameter of tail move spaces is of order* $\operatorname{diam}_{\text{tail}}(n,k) \geq n - \Omega(\sqrt{n})$.

Note that this lower bound only depends on the number of leaves and not on the number of reticulations. In Section 5.2.3, we will give a different lower bound that incorporates k as well.

3.3 Internal labels

Now we turn to spaces of networks with internally labeled nodes, too. This leads to interesting problems, as moves that result in a leaf-isomorphic network may give a network that is not labeled isomorphic with respect to all labels. For example, moving the bottom arc of a triangle to the incoming arc of the triangle swaps the labels of the two tree nodes of the triangle (Figure 3.7, bottom).

To characterize when these networks can be reached from one another, we prove that we can swap the labels of two tree nodes or reticulations. To swap the labels of reticulations, we first transform the networks into ladder caterpillars.

3.3.1 Labeled isomorphisms without degree-2 nodes

We will first show how to swap the labels of two tree nodes using a constant number of tail moves in any network. After establishing this fact, we turn to the labels of the reticulation nodes. To permute these, we transform the network into a ladder caterpillar. In those networks, we can swap the labels of *adjacent* reticulations using a constant number of tail_1 moves.

Tree node labels

To permute the tree node labels, we us the following strategy. We show that we can make any two tree nodes adjacent, and that, when two tree nodes are adjacent, we can swap them.

Lemma 3.16. *Let* $\dot{N} \in \dot{\mathcal{N}}(n,k)$ *be a network with two tree nodes* x *and* y *labelled* $l(x)$ *and* $l(y)$*. Then there is a network* $\dot{N}'$ *in which the nodes labelled* $l(x)$ *and* $l(y)$ *are adjacent such that* $d_{\text{tail}}(\dot{N},\dot{N}') \leq 1$ *and* $d_{\text{tail}_1}(\dot{N},\dot{N}') \leq n + 3k - 1$.

Proof. If x is above y and y is above x, there is a cycle. Hence, we may assume without loss of generality that y is not above x. Furthermore, we assume x

and y are not yet adjacent in N. Hence, one of the outgoing arcs of x is tail movable (Lemma 3.3), and it can be moved to any arc adjacent to y using one tail move. This move creates a node labelled $l(x)$ adjacent to y, which is labelled with $l(y)$. By Lemma 3.9, this move can be replaced by a sequence of at most $n + 3k - 1$ tail_1 moves via an up-down path, the label of the moving tail is the same for each move, hence, the resulting labelling is also the same for the single tail move, as for the sequence of tail_1 moves. □

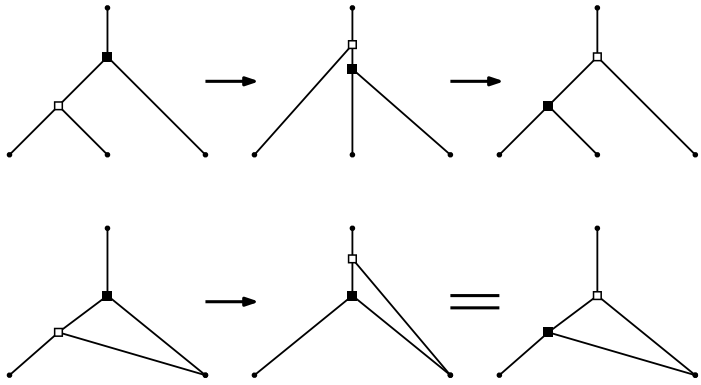


Figure 3.7: Swapping tree nodes as in Lemma 3.17. The bottom move is used when the intermediate network in the top sequence would contain parallel arcs. The labels of the tree nodes are represented by node shapes.

Lemma 3.17. *Let $\dot{N} \in \dot{\mathcal{N}}(n,k)$ be a network, and let x and y be a pair of adjacent tree nodes. Then, there is a sequence of at most 2 tail_1 moves that swaps the labels of x and y.*

Proof. The used sequences of tail_1 moves are shown in Figure 3.7. The top sequence in the figure works if the second and third bottom node in the figure are not the same: in that case, the intermediate network does not have parallel arcs. If these nodes are the same, then the situation shows a triangle as in the bottom sequence in the figure. In this case, we can swap the tree nodes with one tail_1 move. □

Lemma 3.18. *Let $\dot{N}$ be a network, and x and y two tree nodes. Then, using a sequence of at most 4 tail moves or $2n + 6k$ tail_1 moves, the labels of x and y can be swapped.*

Proof. By Lemma 3.16, we can use one tail move (or $n + 3k - 1$ tail_1 moves) in $\dot{N}$ to make the nodes labeled $l(x)$ and $l(y)$ adjacent. Then, using Lemma 3.17, the labels of these two nodes can be swapped with at most two tail_1 moves.

Lastly, the first move is reversed. This results in a network isomorphic to $\dot{N}$, where only the labels of x and y are swapped. □

Proposition 3.19. *Let $\dot{N}$ be a network with s tree nodes, and let π be a permutation of the tree node labels. Then there is a sequence of at most $4s$ tail moves and a sequence of at most $(2n+6k)s$ tail_1 moves that permutes the tree node labels with π and leaves all other labels the same.*

Proof. Each permutation of n elements decomposes in at most n swaps. Each of the swaps of two tree node labels can be achieved using at most 4 tail moves or $2n+6k$ tail_1 moves by Lemma 3.18. Hence, to permute the tree node labels with π, we need at most $4s$ tail moves, or $(2n+6k)s$ tail_1 moves. □

Reticulation labels

For the remaining part of this section, we will work with ladder caterpillars. These networks have all reticulations in one path, which makes it easy to permute the labels of reticulations.

In the following two lemmas, we show when and how we can permute the labels of the reticulations. The first lemma shows that this cannot always be done, and the second shows how it can be done when it is possible.

Lemma 3.20. *Let $\dot{N}$ be a network with one leaf, whose parent p is labelled $l(p)$. Then, after any tail move in $\dot{N}$, the parent of the leaf will still be labelled $l(p)$.*

Proof. The parent of the leaf of $\dot{N}$ is a reticulation, so it cannot be moved using a tail move. Hence, the only way to change the label of the parent of the leaf of $\dot{N}$, is to move another node to the leaf-arc. However, this arc is below all other arcs of $\dot{N}$. Therefore, no node can be moved there using a tail move. We conclude that no tail move can change the label of the parent of the leaf. □

Lemma 3.21. *Let $\dot{N}$ be a ladder caterpillar with two adjacent reticulations x and y, and at least one tree node. Then, using a sequence of at most 5 tail_1 moves, the labels of x and y can be swapped, except if $\dot{N}$ has exactly one leaf l and x or y is the parent of l.*

Proof. Without loss of generality, we assume that y is above x. The sequences shown in Figure 3.8 can then be used to swap the labels of x and y with at most 5 tail_1 moves, except if x is the lowest reticulation and there is only one leaf in the network. □

Now we know how to swap labels of reticulations in a ladder caterpillar, we can permute all reticulation labels of such a network.

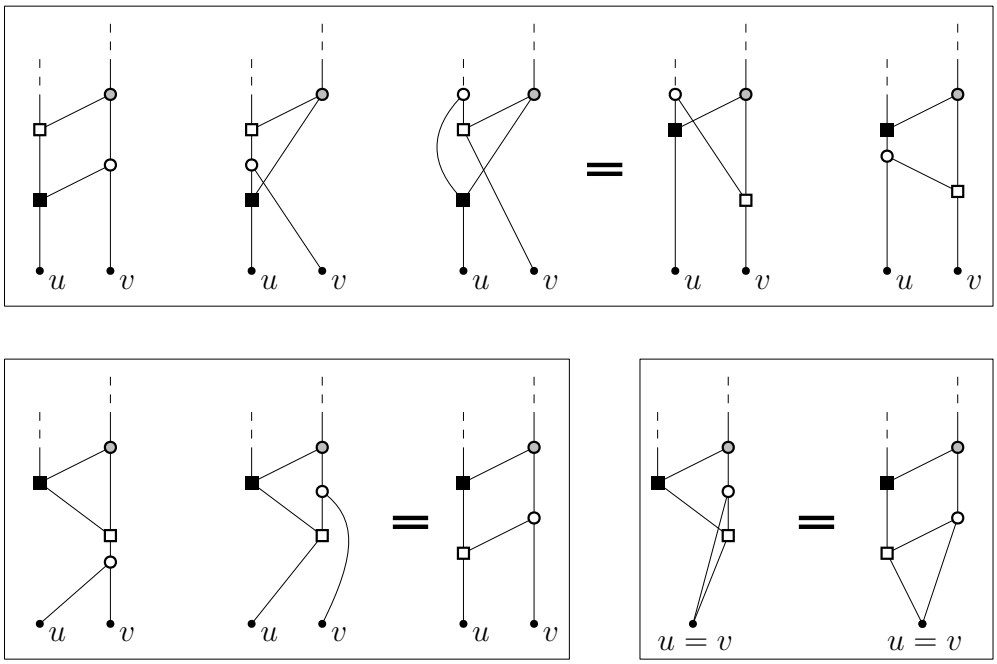


Figure 3.8: The sequence of moves used in Lemma 3.21 to swap two adjacent reticulations in the ladder (x: black square; y: white square), except for the lowest reticulation when there is only one leaf in the network. First apply the moves in the first row, then, depending on whether $u \neq v$ or $u = v$, apply the moves to and within the second or third box. The sequences in this figure take at most 5 tail_1 moves.

Proposition 3.22. *Let $\dot{N}$ be a ladder caterpillar with k reticulations, and let π be a permutation of the reticulation labels. Then there is a sequence of $5k^2$ $tail_1$ moves, that permutes the reticulation labels with π and leaves all other labels the same, except if $\dot{N}$ has one leaf with parent p labelled $l(p)$ and $l(p) \neq \pi(l(p))$.*

Proof. Note that all reticulations are in one path, and that we can swap adjacent labels of reticulations using at most 5 tail_1 moves (Lemma 3.21). Any permutation of n elements of a sequence can be decomposed into at most n^2 neighbour swaps. Hence, to permute the reticulation labels in a network, we need at most $5k^2$ tail_1 moves. □

Using the results above, we can permute all labels of a network with tail moves. Hence, we can find a sequence between any two internally labeled networks (with only a few exceptions).

Theorem 3.23. *Let $\dot{N}, \dot{N}' \in \dot{\mathcal{N}}(n,k)$ internally labelled networks. Then, there exists a tail move sequence of length at most $2\operatorname{diam}_{\text{tail}}(n,k)+5k^2+4(n+k-1)$ and a tail_1 sequence of length at most $2\operatorname{diam}_{\text{tail}_1}(n,k)+5k^2+(2n+6k)(n+k-1)$ from $\dot{N}$ to $\dot{N}'$, except in the following cases:*

- *$(n,k) = (2,1)$ and $\dot{N} \not\simeq_{X^l} \dot{N}'$;*
- *$n = 1$ and $l(p) \neq l'(p')$ for p and p' the parents of the single leaf in each network.*

Proof. We first look at the cases in which there is no sequence of moves possible. For the first case, note that $\dot{N}$ and $\dot{N}'$ cannot be made leaf-isomorphic by Theorem 3.8. For the second case, we only need to observe that the label of the parent of the leaf cannot be changed (Lemma 3.20).

In all remaining cases, Theorem 3.8 shows that we can turn $\dot{N}$ and $\dot{N}'$ into two ladder caterpillars (by Lemma 2.21) with the same labelling of the roots and leaves using a number of moves at most twice the diameter. Then, with $s = n+k-1$ the number of tree nodes (Observation 2.7), we can permute the labels of the tree nodes using at most $4(n+k-1)$ tail moves or $(2n+6k)(n+k-1)$ (Proposition 3.19). Lastly, using at most $5k^2$ tail_1 moves, we can permute the labels of the reticulations (Proposition 3.22). Combining these sequences, we get a sequence from $\dot{N}$ to $\dot{N}'$. □

Note that the bound for tail moves contains a quadratic term. We will see in Chapter 5 that this quadratic term disappears for rSPR moves, when we use a combination of tail and head moves. It is still an open question whether this quadratic term is necessary for tail moves.

3.3.2 Degree-2 nodes

To show connectedness in the presence of degree-2 nodes, we will reduce to the case where we are simply working with supressed networks. This will work in all cases, except for $\dot{\mathcal{N}}(1,k,m)$ and $\dot{\mathcal{N}}(2,1,m)$. In the general case, we first move all degree-2 nodes to a leaf edge in a some fixed order, and then fix all the labels of leaves and degree-3 nodes. This is not possible for the mentioned exceptions.

Lemma 3.24. *Let $\ddot{N} \in \dot{\mathcal{N}}(n,k,m)$ be a subdivided internally labeled network. Then there is a sequence of at most m tail_1 moves that turns $\ddot{N}$ into a network $\ddot{N}'$ without parallel paths, except when $(n,k) = (1,1)$.*

Proof. Let x and y be two nodes in $\ddot{N}$ such that there are parallel paths between x and y. If x is not the child of the root, then one of the outgoing arcs of x

is movable, and moving it up reduces the number of parallel paths by at least one. If x is the child of the root, then there is a highest tree node z directly below y, because $(n,k) \neq (1,1)$. One of the outgoing arcs of z can be moved up to an incoming arc of y (Lemma 3.3). Again, this reduces the number of parallel paths by at least one.

Indeed, for each of these moves, the parallel paths between x and y are destroyed, and there is only one way to create new parallel paths. That is if, in the second case, there is a subdivided triangle $\langle \cdot u, v, w \cdot \rangle \in \ddot{N}$ and the move we apply is $(u', v, w') \xrightarrow{(v,\cdot)} (x', y')$, where u' is on the subdivided path from u to v, w' is on the subdivided path from v to w, and (x', y') is an arc on the subdivided path between x and y. However, in that case, u is a higher tree node below y, a contradiction. □

Lemma 3.25. *Let $\ddot{N}$ be a subdivided network without parallel paths, and suppose there is a tail move changing $S(\ddot{N})$ into N', then there is a tail move changing $\ddot{N}$ into a network $\ddot{N}'$ such that $S(\ddot{N}') = N'$.*

Furthermore, if the tail move in $S(\ddot{N})$ is a tail_1 *move, then there is a sequence of at most $m+1$* tail_1 *moves changing $\ddot{N}$ into $\ddot{N}'$.*

Proof. Suppose the move changing $S(\ddot{N})$ into $\dot{N}'$ is $u \xrightarrow{(u,v)} (x,y)$. Let w be the first node on the path between u and v, and z be the first node on the path between x and y in $\ddot{N}$. Then the move $u \xrightarrow{(u,w)} (x,z)$ is valid in $\ddot{N}$, and results in a network $\ddot{N}'$ with $S(\ddot{N}') = N'$.

For the second part, consider the length one path in $S(\ddot{N})$ that determines the distance of the move. Then the corresponding path in $\ddot{N}$ has length at most $m+1$, and we can move the tail along this path. □

Lemma 3.26. *Let $\ddot{N}$ be a subdivided network with a tree node t such that $S(\ddot{N})$ has arcs (p,t), (t,c_1), and (t,c_2), which are subdivided by the set of degree-2 nodes Y. Let $y_1, \ldots, y_s$ be any ordering of Y, then there is a sequence of at most $2s+2$ tail moves or s^2+4s+2* tail_1 *moves from $\ddot{N}$ to the network obtained from $\ddot{N}$ by removing the subdivided arcs incident to t, and replacing them with (p,t), (t,c_1) and $(t, y_s, \ldots, y_1, c_2)$.*

Proof. First move $s-1$ nodes to the incoming arc of t using at most two tail moves—moving all s degree-2 nodes to this arc may lead to a pair of parallel arcs if $c_1 = c_2$. If the degree-2 nodes are not all at (p,t) already, then we may suppose (without loss of generality) that (t,c_1) is subdivided with at least one node. Let q_1 be the degree-2 node whose child is c_1, let (t,v_2) be the outgoing arc of t on the path to c_2, and (r_1,q_1) the incoming arc of q_1. Then, all $y \in Y \setminus \{q_1\}$ can be moved to (p,t) using the two moves $t \xrightarrow{(t,v_2)} (r_1,q_1)$ and $t \xrightarrow{(t,q_1)} (\cdot, c_2)$.

Secondly, we make y_1 the single node on the empty outgoing arc (t, c_2) of t using at most 2 tail moves. If y_1 subdivides (t, c_1), then we exchange the roles of c_1 and c_2. Otherwise, this can be achieved by applying the moves $t \xrightarrow{(t,c_2)} (y_1, x)$ and $t \xrightarrow{(t,x)} (\cdot, y_1)$ if $x \neq t$, or, if $x = t$, the single move $t \xrightarrow{(t,z)} (\cdot, y_1)$ where $z \neq c_2$. Note that, now, the arc (t, c_2) is subdivided with the single node y_1.

To sort the remaining degree-2 nodes, we do the following: Assume (t, c_2) is subdivided by $y_i, \ldots, y_1$. Apply the moves $t \xrightarrow{(t,y_i)} (y_{i+1}, w)$ and $t \xrightarrow{(t,w)} (\cdot, y_{i+1})$ (Figure 3.9). This takes at most two moves for each $i \in \{2, \ldots, s\}$. Together with the maximum of 4 moves used in the previous steps, we need at most $2s + 2$ tail moves in total

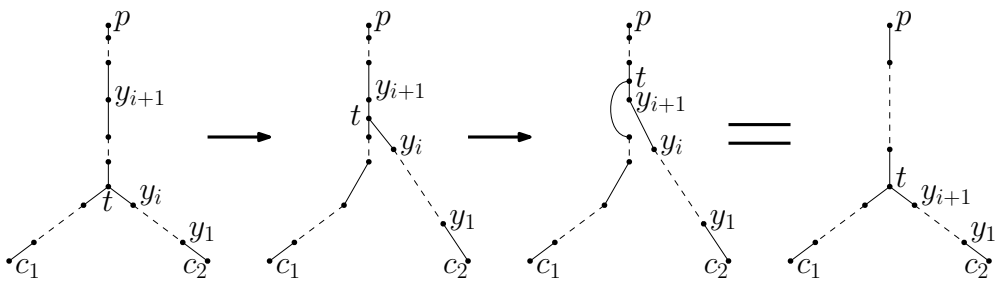


Figure 3.9: Collecting an additional arc on $(t, \ldots, c_2)$ in the proof of Lemma 3.26.

If $c_1 = c_2$, the sequence of moves still works, but one has to consistently identify one of the parallel paths as the "path above c_1", and the other as the "path above c_2".

For the tail_1 sequence, replace each tail move by a sequence of tail_1 moves, noting that each second move in the sorting sequence is a tail_1 move already, and all other moves are along an up-down path of length at most $s + 1$. This gives a tail_1 move sequence of length at most $(s+1)(s+2)+s = s^2+4s+2$. □

The case $\dot{\mathcal{N}}(1, k, m)$

We first consider the cases $\dot{\mathcal{N}}(1, k, m)$, where each network has one leaf. We will show that these spaces are connected, provided the degree-2 nodes on the leaf arc have the same labels. In fact, all these spaces (except for $k = 1$) have a connected component for each ordered sequence (of each subset) of the degree-2 node labels. To prove this, we first need to show that the degree-2 nodes on the leaf arc cannot be changed using tail moves.

By the same arguments as for Lemma 3.20, nothing on the subdivided leaf arc can be changed using tail moves.

Lemma 3.27. *Let $\ddot{N} \in \dot{\mathcal{N}}(1,k,m)$ and let $\ddot{N}'$ be the result of a tail move in $\ddot{N}$. Then the leaf path of $\ddot{N}$ is labeled isomorphic to the leaf path of $\ddot{N}'$.*

Proposition 3.28. *For $k = 0$, and for each $m \geq 0$, the space $\dot{\mathcal{N}}_{\text{tail}}(1,0,m)$ is the edgeless graph on $m!$ nodes.*

Proof. There is exactly one network in $\dot{\mathcal{N}}(1,0,m)$ for each permutation of the m degree-2 nodes. As there are no tree nodes, there are no tail moves possible in any of these networks. □

Observation 3.29. Each network in $\dot{\mathcal{N}}(1,1,m)$ consists of a root path, followed by a pair of parallel paths, and then a leaf path.

Lemma 3.30. *Each network in $\dot{\mathcal{N}}(1,1,m)$ has a leaf arc that is subdivided by at most $m-1$ degree-2 nodes.*

Proof. If the leaf arc of a network $\ddot{N} \in \dot{\mathcal{N}}(1,1,m)$ is subdivided by m degree-2 nodes, then all the degree-2 nodes are on the leaf path. This means the parallel paths of the network are actually parallel arcs (Observation 3.29), which contradicts the fact that $\ddot{N}$ is a network. □

To move the remaining degree-2 nodes around, we use Lemma 3.26.

Proposition 3.31. *For each $m \geq 1$ and ordered sequence of degree-2 nodes $S = (x_1, \ldots, x_s)$ with $0 \leq s < m$, there is a connected component of $\dot{\mathcal{N}}_{\text{tail}}(1,1,m)$, whose networks have a leaf arc subdivided by the degree-2 nodes S.*

Furthermore, these are all the components of $\dot{\mathcal{N}}_{\text{tail}}(1,1,m)$, and we have the bounds $d_{\text{tail}}(\ddot{N},\ddot{N}') \leq 2m+2$ and $d_{\text{tail}_1}(\ddot{N},\ddot{N}') \leq m^2+4m+2$ for each pair of networks in the same component.

Proof. Let m and S be arbitrary conforming to the restrictions of the lemma. Each network $\ddot{N}$ in $\in \dot{\mathcal{N}}(1,1,m)$ with the leaf arc subdivided by S, has all its other degree-2 nodes on the remaining three arcs and at least one of the degree-2 nodes subdivides the pair of parallel arcs.

By Lemma 3.27, each network in the connected component containing $\ddot{N}$ has its leaf arc subdivided by S. Hence, we must prove that each network with a leaf arc subdivided by S can be reached from $\ddot{N}$.

Each such network has the remaining $m-s$ degree-2 nodes on the other three arcs, and at least one on one of the parallel arcs. By Lemma 3.26, these nodes can be arranged on one of the parallel arcs in any predefined order using at most $2(m-s)+2$ tail moves, or $(m-s)^2+4(m-s)+2$ tail_1 moves. Therefore, each network with m degree-2 nodes whose leaf arc is subdivided by S can be reached from $\ddot{N}$.

Now note that each network is in a component of this type. Indeed, let $\ddot{N} \in \dot{\mathcal{N}}(1,1,m)$ be an arbitrary network whose leaf arc is subdivided by a sequence of degree-2 nodes S, then $|S| < m$ by Lemma 3.30. Furthermore, the restrictions of the lemma apply to m and S. Therefore, $\ddot{N}$ is in the component of $\in \dot{\mathcal{N}}(1,1,m)$ whose networks have a leaf arc subdivided by the degree-2 nodes S. □

Lemma 3.32. *For each $k > 1$, $m \geq 0$, and $0 \leq s \leq m$, there is a network $\ddot{N} \in \dot{\mathcal{N}}(1,k,m)$ whose leaf arc is subdivided by s degree-2 nodes.*

Proof. For each $k > 1$, let $\dot{N}_k \in \dot{\mathcal{N}}(1,k,0)$ be the ladder caterpillar with one leaf and k reticulations. Such a network exists for all $k > 1$. To obtain a network in $\dot{\mathcal{N}}(1,k,m)$ with s degree-2 nodes on the leaf arc, subdivide the root arc with $m - s$ degree-2 nodes, and the leaf arc with s degree-2 nodes. □

To characterize the connected components of $\dot{\mathcal{N}}(1,k,m)$ for all m and k, we first treat the case for $k = 2$ separately in the next proposition.

Proposition 3.33. *For each combination of a choice of $m \geq 1$, a reticulation r, and an ordered sequence of degree-2 nodes $S = (x_1, \ldots, x_s)$ with $0 \leq s < m$, there are $1 + (m-s)!$ connected components of $\dot{\mathcal{N}}_{\text{tail}}(1,2,m)$:*

- *One component for each ordering S' of the $m - s$ remaining degree-2 nodes, which consists of the two ladder caterpillars with one leaf and two reticulations, each with their leaf arc (r,l) subdivided by S and arc between the reticulations subdivided by S';*
- *The other component consists of all other networks in $\dot{\mathcal{N}}(1,2,m)$ whose leaf arcs (r,l) are subdivided by S.*

Furthermore, together with the single component where $s = m$, these are all the components of $\in \dot{\mathcal{N}}(1,2,m)$, and we have the bounds $d_{\text{tail}}(\ddot{N}, \ddot{N}') \leq 2(9+2(m-s))$ and $d_{\text{tail}_1}(\ddot{N}, \ddot{N}') \leq 2((m-s)^2 + 6(m-s) + 5)$ for each pair of networks in the same component.

Proof. First note that for each network $\ddot{N} \in \dot{\mathcal{N}}(1,2,m)$, the underlying DAG of $S(\ddot{N})$ is isomorphic to one of the following two. It is either the unique underlying DAG of $N \in \dot{\mathcal{N}}(1,2)$, which consists of two triangles sharing an arc; or it is the directed multi-graph consisting of a root arc, a pair of parallel arcs, a single arc, another pair of parallel arcs, and finally a leaf arc. In the last case, each pair of parallel arcs must be subdivided by at least one degree-2 node. Hence, if $s = m$, then there are only two networks, which are connected by a tail move. Therefore, we henceforth assume that $s < m$.

First, consider any network of the second type (i.e., with parallel paths) whose leaf path is $P_l = (r, x_1, \ldots, x_s, l)$. By moving an outgoing arc of the lowest tree node to an outgoing arc of the highest tree node, we obtain a network of the first type whose leaf path is P_l, in which one of the $m-s$ degree-2 nodes not on P_l subdivides an arc of the highest triangle. We show that any such network can be transformed into a network $\ddot{N}' \in \dot{\mathcal{N}}(1,2,m)$ without parallel paths, whose root arc is subdivided by $S' = (y_1, \ldots, y_{m-s})$, and whose leaf arc is P_l. Note that there are exactly two such networks (we may only swap the labels of the two tree nodes), and that these are exactly one tail move apart.

Let $\ddot{N} \in \dot{\mathcal{N}}(1,2,m)$ be an arbitrary network without parallel arcs whose leaf path P_l is subdivided by S, and at least one of whose remaining $m-s$ degree-2 nodes is not on the arc between the two reticulations. Then $\ddot{N}$ contains two subdivided triangles, a top triangle $\langle \cdot s, t, q \cdot \rangle_t$ and a bottom triangle $\langle \cdot t, q, r \cdot \rangle_r$.

Suppose $\ddot{N}$ has $s' > 0$ degree-2 nodes on the path between q and r. To reach $\ddot{N}'$ from $\ddot{N}$, we first move the $m-s-s'$ degree-2 nodes not on P_l away from the path between q and r. This can be done as follows. First move all degree-2 nodes on the outgoing paths of t up to the incoming path of t, using at most two tail moves or $m-s-s'$ tail_1 moves. Now, $\langle \cdot s, t, q \cdot \rangle$ is subdivided by at least one degree-2 node. Hence, using the tail_1 move $t \xrightarrow{(t,r)} (q, \cdot)$, we obtain a network with parallel paths. As all degree-2 nodes on the path from t to r were moved up, the resulting network contains the arcs (t,r) and (q,t). Then, go back to a network without parallel paths using the tail_1 move $t \xrightarrow{(t,z)} (\cdot, q)$, where $z \neq r$. This ensures the resulting network contains the arc (q,r) (Figure 3.10).

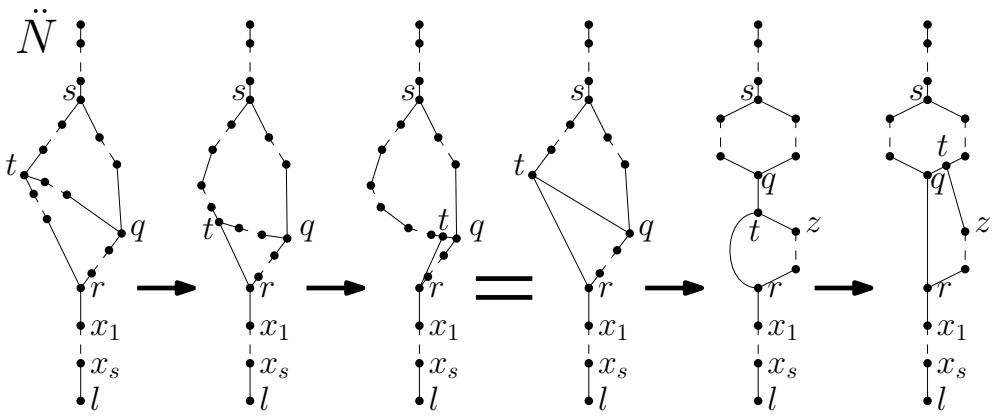


Figure 3.10: Removing all degree-2 nodes from the arc between the reticulations q and r in the proof of Proposition 3.33 using at most 4 tail moves.

Now, using at most one tail moves or s' tail_1 moves, we can move all $m-s$ degree-2 nodes not on P_l to an arc adjacent to the top split (only s' of them are not yet there). Using Lemma 3.26, these nodes can now be sorted as in S' with at most $2(m-s)+2$ tail moves or $(m-s)^2+4(m-s)+2$ tail_1 moves. With at most two tail moves or $m-s+1$ tail_1 moves, these nodes can be moved to the root arc, and the tree nodes can be swapped if needed. Hence, the distance between any pair of networks in such a component is at most $2(9+2(m-s))$ tail moves or $2((m-s)^2+6(m-s)+5)$ tail_1 moves.

For the other components, let $\ddot{N} \in \dot{\mathcal{N}}(1,2,m)$ be a network without parallel paths, with leaf path P_l, whose arc between the two reticulations is subdivided by S'. The component of $\dot{\mathcal{N}}_{\text{tail}}(1,2,m)$ containing $\ddot{N}$ consists of two networks, the other, $\ddot{N}'$, being $\ddot{N}$ with its tree nodes swapped. Indeed, in $\ddot{N}$, no tail move can be moved to the path $(r', y_1, \ldots, y_{m-s}, r, x_1, \ldots, x_s, l)$, and no arc on this path can be moved. Hence, after each tail move in $\ddot{N}$, this path will remain intact, and the only network that can be reached is $\ddot{N}'$. For each S of length s, there are $(m-s)!$ orderings of the remaining $m-s$ degree-2 nodes. This accounts for the $(m-s)!$ components of size two in $\dot{\mathcal{N}}_{\text{tail}}(1,2,m)$ for a fixed S. □

Proposition 3.34. *For each $k>2$ and $m \geq 1$ there is a connected component of $\in \dot{\mathcal{N}}_{\text{tail}}(1,k,m)$ for each choice of lowest reticulation r and ordered sequence of degree-2 nodes $S=(x_1,\ldots,x_s)$ with $0 \leq s \leq m$. The networks in this component are exactly those networks with the leaf path $(r, x_1, \ldots, x_s, l)$.*

Furthermore, these are all the components of $\in \dot{\mathcal{N}}_{\text{tail}}(1,k,m)$, and we have the bounds $d_{\text{tail}}(\ddot{N},\ddot{N}') \leq 6m + 2\,\text{diam}_{\text{tail}}(n,k,0) + 10$ and $d_{\text{tail}_1}(\ddot{N},\ddot{N}') \leq 2m(2+\text{diam}_{\text{tail}_1}(n,k,0)+n+3k+m)+2n+6k+2$ for each pair of networks in the same component.

Proof. Let $\ddot{N}$ be an arbitrary network in $\dot{\mathcal{N}}(1,k,m)$ whose leaf arc is subdivided by S, and let $\ddot{N}_l \in \dot{\mathcal{N}}(1,k,m)$ be a ladder caterpillar whose leaf arc is also subdivided by S, and all remaining degree-2 nodes are on the path $(t, y_1, \ldots, y_{m-s}, r)$, where r is the lowest reticulation, and t is a tree node. We show that there is a sequence of moves from $\ddot{N}$ to $\ddot{N}_l$. This then implies that any pair of networks in $\dot{\mathcal{N}}(1,k,m)$ is connected by a sequence of tail moves provided they have the same sequence of degree-2 nodes subdividing their leaf arcs. As no pair of networks with different sequences of degree-2 nodes on their leaf arcs are connected, this characterizes all connected components of $\dot{\mathcal{N}}_{\text{tail}}(1,k,m)$.

First, we turn $\ddot{N}$ into a network $\ddot{N}'$ such that $S(\ddot{N}') \in \dot{\mathcal{N}}(1,k,0)$ using at most m tail_1 moves (Lemma 3.24). Then, by Theorem 3.23 and Lemma 3.25, we may turn $\ddot{N}'$ into a network $\ddot{N}'_l$ such that $S(\ddot{N}'_l) \simeq_X S(\ddot{N}_l)$ using at most $\text{diam}_{\text{tail}}(n,k,0)$ tail moves, or $(m+1)\,\text{diam}_{\text{tail}_1}(n,k,0)$ tail_1 moves.

Like in the proof of Proposition 3.33, we can remove all degree-2 nodes from incoming arcs of the lowest reticulations using at most 4 tail moves or $m+2$ tail_1 moves. Let (t, r) be the incoming arc of r that has a tree node t as its tail. As there are no degree-2 nodes on the incoming paths of r, this arc may be moved to the outgoing arc of any degree-2 node not in S.

We use a similar strategy to the one used in the proof of Lemma 3.26 to collect all degree-2 nodes on (t, r) in the predefined order $S' = (y_1, \ldots, y_{m-s})$: Suppose (t, r) is subdivided by $(y_i, \ldots, y_{m-s})$, then we may move (t, y_i) to the outgoing arc of y_{i-1} and move the other outgoing arc of t up to the incoming arc of y_{i-1}. Repeating these two tail moves (or at most $n+3k+m$ tail_1 moves) for all $m - s \geq i \geq 1$ and finally moving (t, y_{m-s}) back to its original position (1 tail move or at most $n + 3k - 1$ tail_1 moves), we end up with the network $\ddot{N}_l$.

The sequences of moves from $\ddot{N}$ to $\ddot{N}_l$ consist of at most $3m+\text{diam}_{\text{tail}}(n, k, 0)+5$ tail moves and at most $m(2 + \text{diam}_{\text{tail}_1}(n, k, 0) + n + 3k + m) + n + 3k + 1$ tail_1 moves. □

Theorem 3.35. *For all $k \geq 0$ except $k = 2$, two networks $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}_{\text{tail}}(1, k, m)$ are in the same component iff their leaf paths are labeled isomorphic. For $k = 2$, $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}_{\text{tail}}(1, 2, m)$ are in the same component iff their leaf paths are labeled isomorphic and either there is a subdivided arc besides the leaf arc and the arc between the two reticulations, or there are no such subdivided arcs and the subdivided arcs between the two reticulations in both networks are isomorphic.*

Proof. This is a direct consequence of Proposition 3.28 (for $k = 0$), Proposition 3.31 (for $k = 1$), Proposition 3.33 (for $k = 2$), and Proposition 3.34 (for $k \geq 2$). □

The case $\dot{\mathcal{N}}(2, 1, m)$

Each network in $\dot{\mathcal{N}}(2, 1, 0)$ is isomorphic to N_A. Hence, as $\mathcal{N}(2, 1)$ is not connected, we cannot use results about leaf-labeled networks to prove connectedness. Therefore, we study this case separately as well.

None of the spaces $\dot{\mathcal{N}}(2, 1, m)$ $(m \geq 0)$ is connected, as each network where all degree-2 nodes are on the subdivided arc from the reticulation to a leaf is on a different connected component than the ones where not all degree-2 nodes are on this arc.

Lemma 3.36. *Let $\ddot{N} \in \dot{\mathcal{N}}(2, 1, m)$ such that all degree-2 nodes are on the subdivided arc between the reticulation and a leaf. If $\ddot{N}'$ is the result of a tail move in $\ddot{N}$, then the leaf path of $\ddot{N}$ is labeled isomorphic to the leaf path of $\ddot{N}'$.*

Proof. Let $P_l = (r, x_1, \ldots, x_m, l)$ be the path between the reticulation and the leaf containing all degree-2 nodes. No tail in $\dot{N}$ can be moved to P_l, and no arc on P_l can be moved. Hence, this path will stay intact after each move in $\ddot{N}$. □

The proof for the following theorem works essentially the same as the proof for Proposition 3.33, which characterizes the components of $\dot{\mathcal{N}}_{\text{tail}}(1, 2, m)$.

Theorem 3.37. *For each $m > 0$, the space $\dot{\mathcal{N}}_{\text{tail}}(2, 1, m)$ consists of $2(m!) + 1$ connected components. Moreover, two networks $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}_{\text{tail}}(2, 1, m)$ are in the same component iff*

- *either both networks have a degree-2 node not on the arc between the reticulation and a leaf;*
- *or both networks have all the degree-2 nodes on an arc between the reticulation and a leaf, and these paths are labeled isomorphic.*

and the distance between these networks is at most $18 + 4m$ tail moves and $2m^2 + 16m + 6$ tail_1 moves.

Proof. First suppose both networks $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}(2, 1, m)$ have a degree-2 node not on an arc between a reticulation and a leaf. If $\ddot{N}$ has no parallel paths, we can make sure there is a degree-2 node subdividing the triangle using at most 1 tail_1 move. Then, using one tail move or at most m tail_1 moves, we can reach a network with parallel paths. In that network, we can use at most 2 tail moves or m tail_1 moves to make sure the leaf paths contain no degree-2 nodes. After this, we can choose either leaf l, and use one tail_1 move or m tail_1 moves to go back to a network without parallel paths, in which there is an arc between the reticulation and l without degree-2 nodes. In this network, we can use at most 2 tail moves or m tail_1 moves to make sure all degree-2 nodes are adjacent to the top tree node. Lastly, using Lemma 3.26, we can sort the degree-2 nodes and move them to the root arc with one additional tail move. Doing this for $\ddot{N}'$ as well, we find a sequence between $\ddot{N}$ and $\ddot{N}'$ of at most $18 + 4m$ tail moves, or $2m^2 + 16m + 6$ tail_1 moves.

Now suppose $\ddot{N} \in \dot{\mathcal{N}}(2, 1, m)$ has all its degree-2 nodes on the arc between the reticulation and a leaf l. Then, after any move in $\ddot{N}$, this path will remain intact (Lemma 3.36). Therefore, for each choice of leaf l in combination with an ordering $(x_1, \ldots, x_m)$ of the m degree-2 nodes, there are two networks with path $(r, x_1, \ldots, x_m, l)$ between a reticulation r and l, and each such pair is connected by a tail move (swapping the tree nodes).

Now we show that this characterizes all components of $\dot{\mathcal{N}}_{\text{tail}}(2,1,m)$. Let $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}(2,1,m)$ have all the degree-2 nodes on an arc between the reticulation and a leaf. If these paths are labeled isomorphic, then $\ddot{N}$ and $\ddot{N}'$ are isomorphic. If these paths are not isomorphic, then $\ddot{N}$ and $\ddot{N}'$ are not isomorphic (as the paths cannot be changed by a tail move). Lastly, as the path between the reticulation and the leaf cannot change, we can in particular reach no network with a degree-2 node on a different arc of the network. Hence, we have found all $2(m!)+1$ components of $\dot{\mathcal{N}}_{\text{tail}}(2,1,m)$. □

General case

Finally, we investigate the spaces $\dot{\mathcal{N}}_{\text{tail}}(n,k,m)$ in general, where $n \neq 1$ and $(n,k) \neq (2,1)$. We will show that, in this more general case, all spaces $\dot{\mathcal{N}}_{\text{tail}_1}(n,k,m)$ are connected. The proof is similar to the proofs in the previous subsections for the connectedness of components of $\dot{\mathcal{N}}_{\text{tail}}(1,k,m)$ and $\dot{\mathcal{N}}_{\text{tail}}(2,1,m)$: it starts by removing parallel arcs, then it creates a movable arc on which we collect the degree-2 nodes, and, finally, the structure of the suppressed network is fixed.

Lemma 3.38. *Let $\ddot{N} \in \dot{\mathcal{N}}_{\text{tail}}(n,k,m)$ ($n > 1$ and $(n,k) \neq (2,1)$) be a network without parallel paths. Then there is a sequence of at most 2 tail moves or $n+3k+2m-1$ tail_1 moves from $\ddot{N}$ to a network $\ddot{N}'$ without parallel paths in which there is a movable arc (t,l) between a tree node and a leaf and there does not exist a subdivided triangle $\langle \cdot x,t,y \cdot \rangle$ in $\ddot{N}'$.*

Proof. First suppose $S(\ddot{N})$ has an arc (t,l) from a tree node to a leaf. In that case, all degree-2 nodes on the corresponding path in $\ddot{N}$ can be removed from this path using one tail move or at most m tail_1 moves. Note that there is a subdivided triangle $\langle \cdot x,t,y \cdot \rangle \in \ddot{N}$ iff (t,l) is not movable in $S(\ddot{N})$. If x is not the child of the root in $S(\ddot{N})$, then (t,l) can be made movable by moving (x,y) up using one tail move. Otherwise, the child c of y in $S(\ddot{N})$ is a tree node (by the restrictions on n and k), and (t,l) can be made movable by moving one of the outgoing arcs of c up to an incoming arc of y with one tail_1 move. By Lemma 3.25, these tail_1 moves in $S(\ddot{N})$ correspond to one tail move or at most m tail_1 moves in $\ddot{N}$ each, resulting in a network $\ddot{N}'$ with the required properties.

Now suppose each leaf in $S(\ddot{N})$ is the child of a reticulation. Because $n > 1$, we can choose two arbitrary leaves l and l', and consider an LCA x of these. By Lemma 3.4, one of its outgoing arcs (x,y) is movable in $S(\ddot{N})$, and this arc is not above at least one of l and l'. Assume without loss of generality that y is not above l, then the move $x \xrightarrow{(x,y)} (\cdot, l)$ is valid in $\ddot{N}$, and it results in

a network $\ddot{N}'$ with the required properties. The move can be replaced by a sequence of at most $n + 3k - 1 + m$ tail_1 (Lemma 3.9)

In the first case, two tail moves or $2m$ tail_1 moves suffice; in the second case, we need at most 1 tail move or $n + 3k + m - 1$ tail_1 moves. The number of tail moves can therefore be bounded by 2, and the number of tail_1 moves by $n + 3k + 2m - 1$. □

Lemma 3.39. *Let $\ddot{N} \in \dot{\mathcal{N}}(n, k, m)$ be a network without parallel paths in which there is a movable arc (t, l) between a tree node and a leaf and $\langle \cdot x, t, y \cdot \rangle \notin \ddot{N}'$, and let $(v_1, \ldots, v_m)$ be any ordering of its degree-2 vertices. Then there are sequences of at most $2m + 1$ tail moves and at most $(m + 1)(n + 3k + m)$ tail_1 moves to the network $\ddot{N}'$ with $S(\ddot{N}) = S(\ddot{N}')$ in which there is a path $(t, v_1, \ldots, v_m, l)$.*

Proof. Suppose the network already contains the path $(t, v_{i+1}, \ldots, v_m, l)$. To add the next degree-2 node v_i to the path, apply the moves $t \xrightarrow{(t,v_{i+1})} (v_i, z)$ and $t \xrightarrow{(t,z)} (\cdot, v_i)$. Because $\langle \cdot x, t, y \cdot \rangle \notin \ddot{N}'$ and $\ddot{N}'$ has no parallel paths, the resulting network cannot contain such a triangle either. Hence, we can continue this process for each v_j, using at most 2 moves per degree-2 node. Lastly, we need one additional move to place the moving path back to its original position, so $d_{\text{tail}}(\ddot{N}, \ddot{N}') \leq 2m + 1$.

Noting that m of the tail moves are actually tail_1 moves, and the remaining moves can be replaced by a sequence of at most $(n + 3k + m - 1)$ tail_1 moves, we get $d_{\text{tail}_1}(\ddot{N}, \ddot{N}') \leq (m+1)(n+3k+m-1)+m < (m+1)(n+3k+m)$. □

Theorem 3.40. *Let $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}(n, k, m)$ with $n > 1$ and $(n, k) \neq (2, 1)$, then $d_{\text{tail}}(\ddot{N}, \ddot{N}') \leq 2\,\text{diam}_{\text{tail}}(n, k, 0) + 5k^2 + 4n + 4k + 6m + 2$ and $d_{\text{tail}_1}(\ddot{N}, \ddot{N}') \leq 2\,\text{diam}_{\text{tail}_1}(n, k, 0) + 2n + 6k + 8m - 2 + m(2n + 6k + 2m) + 2n^2 + 8nk + 11k^2$.*

Proof. First, in both networks, we destroy parallel paths using at most m tail_1 moves (Lemma 3.24). Then, using at most 2 tail moves or $n+3k+2m-1$ tail_1 moves, we create a moving arc between a tree node and a leaf (Lemma 3.38). Then, we place all degree-2 leaves on this moving arc using at most $2m + 1$ tail moves or $(m + 1)(n + 3k + m)$ tail_1 moves using Lemma 3.39. Finally, Theorem 3.23 implies we have sequences of at most $2\,\text{diam}_{\text{tail}}(n, k, 0) + 5k^2 + 4(n + k - 1)$ tail moves or $2\,\text{diam}_{\text{tail}_1}(n, k, 0) + 5k^2 + (2n + 6k)(n + k - 1)$ tail_1 moves between the two resulting networks. Hence, there are sequences of at most $2\,\text{diam}_{\text{tail}}(n, k, 0) + 5k^2 + 4n + 4k + 6m + 2$ tail moves and $2(m+n+3k+2m-1+(m+1)(n+3k+m))+2\,\text{diam}_{\text{tail}_1}(n, k, 0)+5k^2+(2n+6k)(n+k-1) = 2\,\text{diam}_{\text{tail}_1}(n, k, 0) + 2n + 6k + 8m - 2 + m(2n + 6k + 2m) + 2n^2 + 8nk + 11k^2$ tail_1 moves from $\ddot{N}$ to $\ddot{N}'$. □

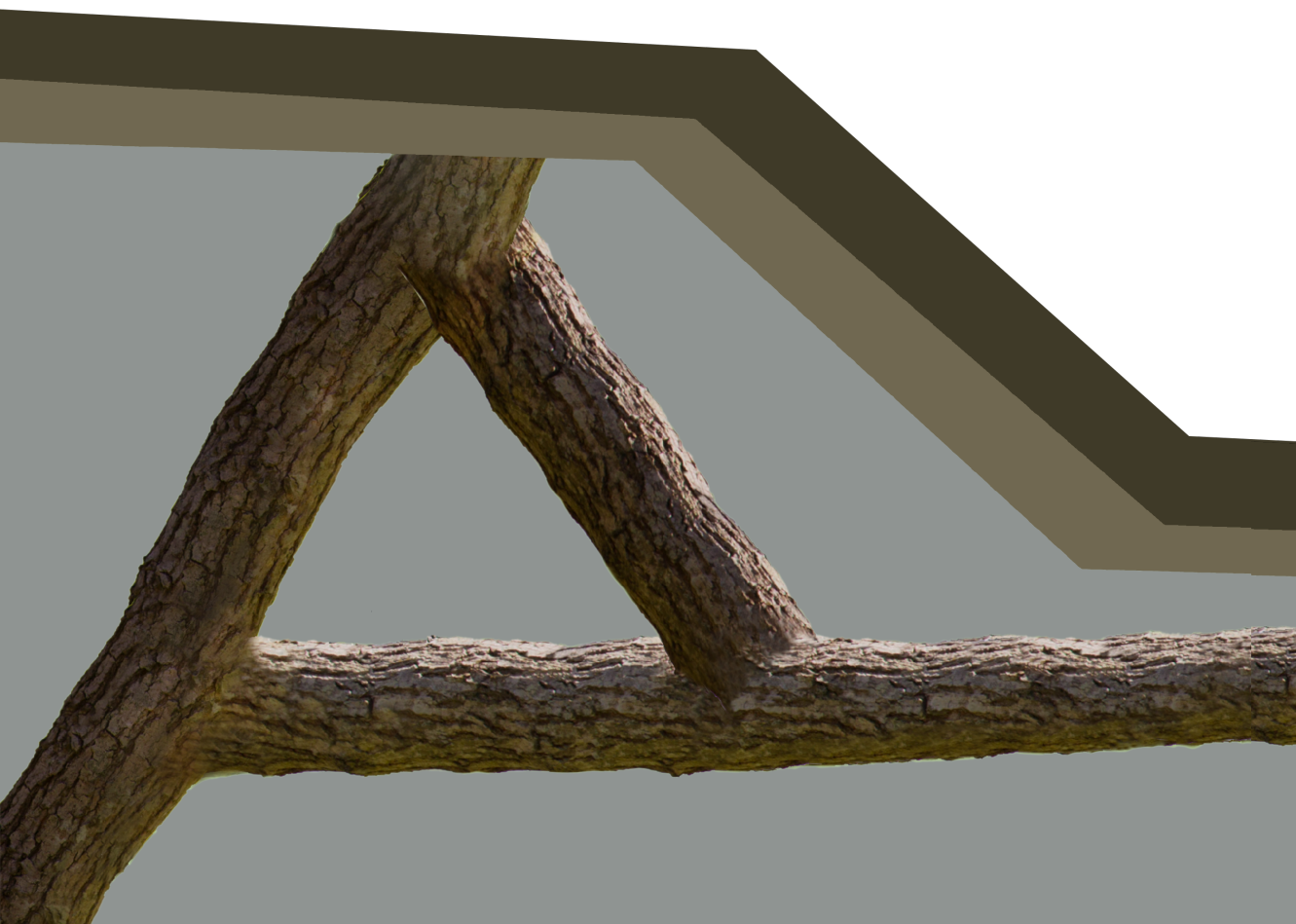
3.4 Conclusion

In this chapter, we have shown that all spaces $\mathcal{N}_{\text{tail}}(n,k)$ and $\mathcal{N}_{\text{tail}_1}(n,k)$ are connected, except when $(n,k) = (2,1)$. The diameters of these spaces are at most $2n+5k-2$ and $(n+3k-1)(2n+5k-2)$ respectively. For tail moves, we have also proven a lower bound of $n - \Omega(\sqrt{n})$, which is directly inherited from trees.

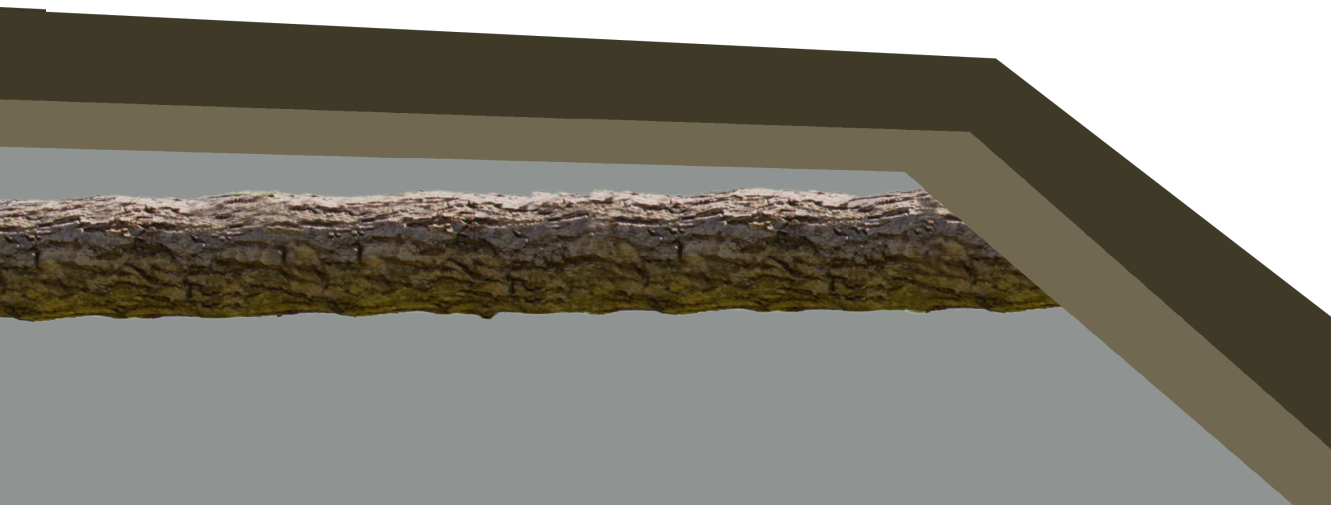
We have also extended the results to internally labeled networks, for which we show that most spaces are still connected by tail moves and tail_1 moves. The only exceptions are the spaces with one leaf, and the spaces where $(n,k) = (2,1)$. For these spaces, we have fully characterized all connected components. We have also bounded the diameters in these cases by $2\,\text{diam}_{\text{tail}}(n,k) + 5k^2 + 4n + 4k + 6m + 2$ tail for tail moves, and $2\,\text{diam}_{\text{tail}_1}(n,k) + 2n + 6k + 8m - 2 + m(2n + 6k + 2m) + 2n^2 + 8nk + 11k^2$ for tail_1 moves.

We do not know the exact diameter for any of these spaces (except for a few small ones where we can explicitly compute the whole space). Furthermore, in most cases we do not even know whether our bounds are asymptotically tight. Only for tail moves and networks without internal labels, we know that our upper bound is asymptotically tight, as we will see in Section 5.2.3. Determining asymptotically tight bounds for (local) tail move spaces remains an open problem.

4



Head Moves



Unlike tail moves and rSPR moves, head moves do not constitute a natural generalization of rearrangement moves from trees to networks. They are, however, part of rSPR moves, which are simply the combination of tail moves and head moves. Hence, to understand rSPR spaces, we have to understand head moves as well as tail moves. This, alone, is enough reason to study spaces of head moves. In this chapter, we show that spaces of head moves have several interesting properties on their own as well.

For example, we show that, even though the head move neighbourhood is typically much smaller than the tail move neighbourhood, the diameters of the respective spaces grow similarly fast in the number of leaves and reticulations. In Chapter 5, we will show that this even holds for the distance between each specific pair of networks in the following sense. Each tail move can be replaced by at most 16 head moves, and each head move by at most 13 tail moves. Hence, to find an optimal network using head moves, we have to consider fewer neighbours than for tail moves, but we will be able to find an optimal network in a similar number of steps.

Like the previous chapter, this chapter focuses on establishing connectedness and diameters of head move spaces. We start by proving that each tier of phylogenetic network space is connected by distance-2 head moves, but not by distance-1 head moves (Section 4.2). The connectedness of $\dot{\mathcal{N}}_{\text{head}_2}(n,k)$ for $n,k>0$ is proven by first constructing a sequence from each network to a ladder tree, and then describing how to find a sequence between any two ladder trees. For the first part, we simply move the heads of all reticulations up towards the root; for the second part, we observe that a ladder tree is (almost) defined by its embedded tree, so it suffices to find a sequence of head_2 moves that changes the embedded tree by one tree rearrangement move.

Then, in Section 4.3, we prove an upper bound of length $6n+6k-4$ for the diameter of $\dot{\mathcal{N}}_{\text{head}}(n,k)$ where $n,k>0$. Like for tail moves, the proof uses an isomorphism building strategy. In this case, however, we keep an up-closed isomorphism. As head moves are essentially tail moves in the digraph where he direction of all arcs are reversed, the proof is quite similar to that for tail moves. The main difference is that, here, we end up with an unlabeled isomorphism, as the leaves are the last to be added to the isomorphism. Hence, the proof has an additional last part, where we permute the leaf labels.

Lastly, in Section 4.4, we consider internally labeled networks again. The connectedness proofs here are quite similar to those in the last chapter. Indeed, here too, we first change each network into a ladder caterpillar and then we permute the leaf and reticulation labels. Similarly, to prove connectedness in the presence of degree-2 nodes, we collect them all at one arc again. Recall that

for tail moves, we could not change the leaf-path when the network only had one leaf. For head moves, the same is the case for the root path. We show that each space $\dot{\mathcal{N}}_{\text{head}}(n,k,m)$ has one connected component for each root-path.

In this chapter, unless stated otherwise, each move is a head move and movability always refers to head-movability. We start with a short section discussing some basic results about head-movability.

4.1 Preliminary observations

Before we turn to the connectedness of spaces, we start with a few facts that we will use repeatedly throughout this chapter. The first two are simple observations about possible moves, and the last relates the sets of displayed trees of networks that are one head move apart.

Observation 4.1 ([Jan21] Observation 1)**.** Let N be a network with a triangle $\langle x,y,z\rangle_t$. Then reversing the direction of the arc (y,z) gives a network N'. We say the *direction of the triangle is reversed*. This can be achieved using the head_1 move $z \xrightarrow{(x,z)} (y,c)$, where $c \neq z$ (Figure 4.1).

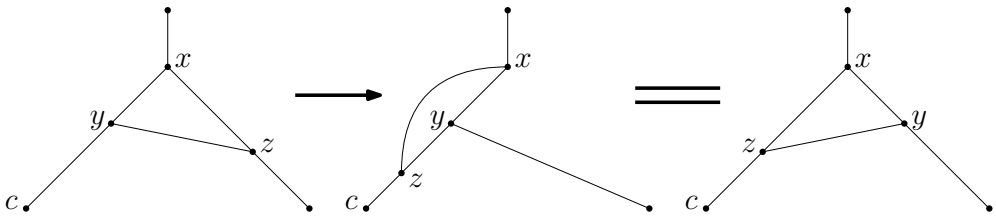


Figure 4.1: The head_1 move $z \xrightarrow{(x,z)} (y,c)$ used to change the direction of a triangle $\langle x,y,z\rangle_t$.

Lemma 4.2. *Let r be a reticulation, then at least one of its incoming arcs is movable.*

The proof of this lemma is analogous to the proof of Lemma 3.3 for the tree nodes and tail moves.

Lemma 4.3 ([Jan21] Lemma 3)**.** *Let N, N' be networks with $d_{\text{head}}(N,N') \leq 1$, then $\mathcal{T}(N) \cap \mathcal{T}(N') \neq \emptyset$ and for each $T' \in \mathcal{T}(N')$ there is a tree $T \in \mathcal{T}(N)$ such that $d_{\text{tail}}(T,T') \leq 1$.*

Proof. Let (u, v) be the arc that is moved in the head move from N to N'. Then v is a reticulation and it has another incoming arc (w, v). There is an embedded tree $T \in \mathcal{T}(N)$ that uses this arc, and therefore does not use (u, v). This means that changing the location of (u, v) does not change the fact that T is embedded in the network, and $T \in \mathcal{T}(N')$.

For the second part: first suppose the embedding of T' in N' does not use the new arc $(u, v) \in N'$. Then clearly T' can be embedded in N without the arc (u, v). This means it can also be embedded in N.

Now suppose the embedding of the arc (t, z) of T' in N' uses the arc $(u, v) \in N'$. Let P be the path through the embedding of T' in N' starting at the image of t, and ending at v. Note that this path passes through (u, v). Now consider the tree obtained by taking the embedding of T', removing P and adding a path leading from a node w in the embedding of T' to v via the other incoming arc of v. This tree only uses arcs that are also in N. Hence, it is embedded in N and it is at most one tail move away from T': the one that moves the subtree below v to w. □

4.2 Connectedness

In this section, we consider the connectedness of the tiers $\mathcal{N}(n, k)$ of network space under local head moves. One might hope that these spaces are connected under distance-1 head moves, because a similar result holds for distance-1 tail moves as well. We prove that this is, unfortunately, not the case. However, we will show that distance-2 head moves do suffice.

4.2.1 Distance-1 is not enough

We show by counterexample that distance-1 head moves are not enough to connect the tiers of phylogenetic network space (Figure 4.2). For tier-1 networks, this example can easily be checked, as there are no distance-1 head moves in the left network that result in a different network. For higher tiers, however, many valid distance-1 head moves may be possible. However, using such moves we are not free to change the structure of the network in any way we would desire. Indeed, using the following lemma, we will show that the reticulations remain roughly at the same place any network resulting from a distance-1 head move.

Lemma 4.4 ([Jan21] Lemma 4)**.** *Let N, N' be networks with $d_{\text{head}_1}(N, N') \leq 1$. If, in N, all reticulations and their parents are below some tree node s, then the same holds for N'.*

Proof. Suppose the head$_1$ move between N and N' is $(x, v, y) \xrightarrow{(u,v)} (z, w)$. Let q be a reticulation or a parent of a reticulation in N, with $q \neq v$, then there is a path from s to q in N. Furthermore, we may assume that this path does not pass through (u, v), as it could alternatively use a path using the other in-arc of v. Hence, after the head move, q is still below s. Now we show that the parents u and z of v in N' are below s. As u is the parent of the retculation v in N, it is still below s by the previous argument. For z, we note that the move is a distance-1 move, so either $z = y$ or $w = x$. In the first case, z is below x, which remains below s. In the second case, either x is a reticulation and its parent z is below s, or x is a tree node. If $x = s$, then the move is invalid, because u is below x; if $x \neq s$, then its parent z must be below s as well. We conclude that all reticulations and their parents are below s in N'. □

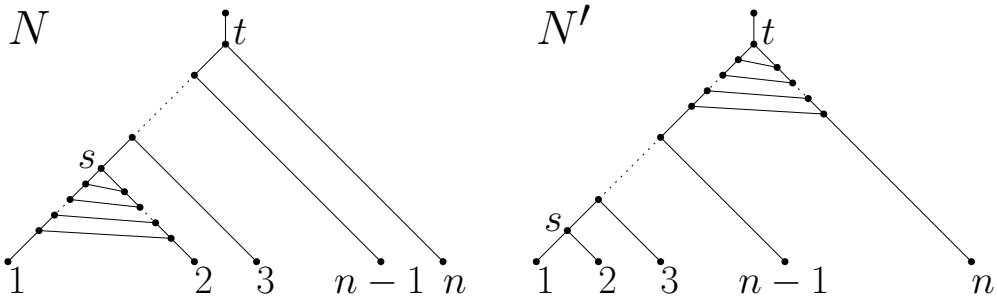


Figure 4.2: There is no sequence of distance-1 head moves between N and N' (Proposition 4.5).

Proposition 4.5 ([Jan21] Proposition 1)**.** *In all tiers of phylogenetic space with $n \geq 3$ leaves, there exist two networks not connected by a sequence of distance-1 head moves.*

Proof. Let T be a caterpillar on $n \geq 3$ leaves, and let s be the common parent of two of the leaves, and let t be the highest tree node in T. Now construct a network N by adding k reticulation arcs between the outgoing arcs of s, and the network N' by adding k reticulation arcs between the outgoing arcs of t (Figure 4.2). In N, all of the reticulations and their parents are below s. Lemma 4.4 implies that, using distance-1 head moves, only networks with all reticulations below s can be reached. Furthermore, because no part above s can ever be involved, the caterpillar structure above s will remain intact. Hence, any network reachable from N using distance-1 head moves consists of a chain of pendant leaves followed by the node s, which must still be above

all reticulations and parents of reticulations. Now note that N' is not such a network. We conclude that there is no distance-1 head move sequence between N and N'. □

Perhaps surprisingly, for networks with only one leaf, we can easily prove connectedness.

Proposition 4.6 ([Jan21] Proposition 2)**.** *The space $\mathcal{N}_{\text{head}_1}(1, k)$ is connected for all $k \in \mathbb{Z}_{\geq 1}$.*

Proof. This follows from the fact that all tiers of phylogenetic network space with one leaf are connected by distance-1 tail moves (Theorem 3.10). Indeed, if one reverses the direction of all arcs, a network with one leaf becomes another network with one leaf, and each distance-1 tail move in the reversed network is a distance-1 head move in the original network. □

4.2.2 Distance-2 suffices

To prove the connectedness of tiers of network space using distance-2 head moves, we present a procedure to generate a sequence between any two networks in the same tier. This sequence first turns both networks into ladder trees, where all reticulations are collected at the top. Next, the tree structure of these networks is adjusted, by simulating rSPR moves on the trees using distance-2 head moves.

Collecting the reticulations at the top

In this subsection, we show how all reticulations can be collected at the top of the network using distance-2 head moves (See Definition 2.17). This will be achieved by creating triangles, and moving these through the network.

The following lemma ensures that the top reticulations can be directed neatly using local head moves. The moves are similar to the one used to change the direction of a triangle (cf. Observation 4.1).

Definition 4.7. Let N be a network with k reticulations at the top. *Changing the direction* of an arc (a_i, b_i) (as in Definition 2.17) consists of changing N into a network N' that is isomorphic to N when (a_i, b_i) is replaced by (b_i, a_i). Note that this changes a_i from a tree node to a reticulation, and b_i from a reticulation to a tree node, which will not be possible when applying the rearrangement moves in this thesis. Changing the direction of a set of such arcs at the same time is defined analogously.

Lemma 4.8 ([Jan21] Lemma 5)**.** *Let N be a network with k reticulations at the top. Then the reticulation arcs (a_i, b_i) can be redirected so that they are neatly at the top with at most k distance-1 head moves. The network below a_k and b_k (notation as in Definition 2.17) is not altered in this process.*

Proof. We redirect the top reticulation arcs (a_i, b_i) starting with the lowest one ($i = k$). The move $b_i \xrightarrow{(u_{i-1}, b_i)} (a_i, v_{i+1})$, where u_{i-1} is the parent of b_i that is not a_i and v_{i+1} the child of a_i that is not b_i (Figure 4.3), changes the direction of the chosen arc (a_i, b_i) and all the reticulation arcs (a_j, b_j) above it (i.e., $j < i$); it leaves all other arcs fixed as they were. □

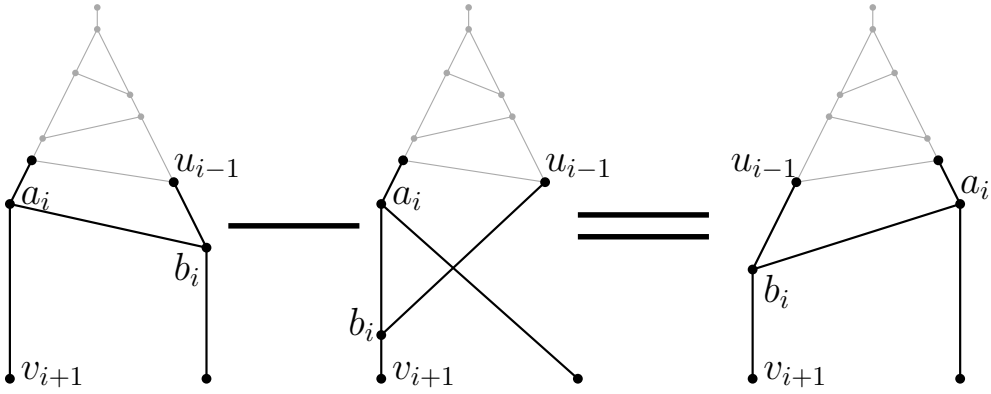


Figure 4.3: The move used in Lemma 4.8 to redirect the i highest reticulations at the top, use the move $b_i \xrightarrow{(u_{i-1}, b_i)} (a_i, v_{i+1})$. This move changes the direction of all reticulations at the top that are higher than the moved arc. The part of the network below a_i and b_i does not change.

Definition 4.9. Let N be a network with *a triangle at* u (i.e., a triangle $\langle u, v, w \rangle_t$) and let u and s be the children of a tree node s. *Moving a triangle from u to s* consists of removing the arc (v, w) and suppressing its endpoints, and subdividing the outgoing arcs of s with two new nodes, and adding an arc between them. The reverse of this process is called moving a triangle from s to u.

Definition 4.10. Let N be a network with k reticulations at the top (notation as in Definition 2.17) and a tree node x directly below a_k. *Removing a triangle from the top* consists of creating a triangle at x by a head move $b_k \xrightarrow{(a_k, b_k)} (x, c)$, where c is a child of x. *Adding a triangle to the top* is the reverse of this

operation. *Moving a triangle to the top* consists of first moving it to a tree node directly below the top, and then adding it to the top.

Lemma 4.11 ([Jan21] Lemma 6)**.** *Triangles can be moved between adjacent tree nodes using at most 4* head_2 *moves, and added to or removed from the top using at most* 7 head_2 *moves.*

Proof. Suppose a network N has a triangle $\langle u, v, w\rangle_t$, where u is the child of a tree node s. Let the other children of s, v, and w be a, b, and c respectively. To move the triangle up to s, we use the following sequence of distance-2 head moves: $w \xrightarrow{(v,w)} (s, a)$, $w \xrightarrow{(s,w)} (u, v)$, $w \xrightarrow{(u,w)} (v, b)$, and $w \xrightarrow{(v,w)} (s, u)$ (Figure 4.4). None of the intermediate networks in the sequence contain a directed cycle or parallel arcs, unless $a = b$. However, in that case, the move $w \xrightarrow{(u,w)} (s, a)$ is a head_2 move that moves the triangle up to s in N.

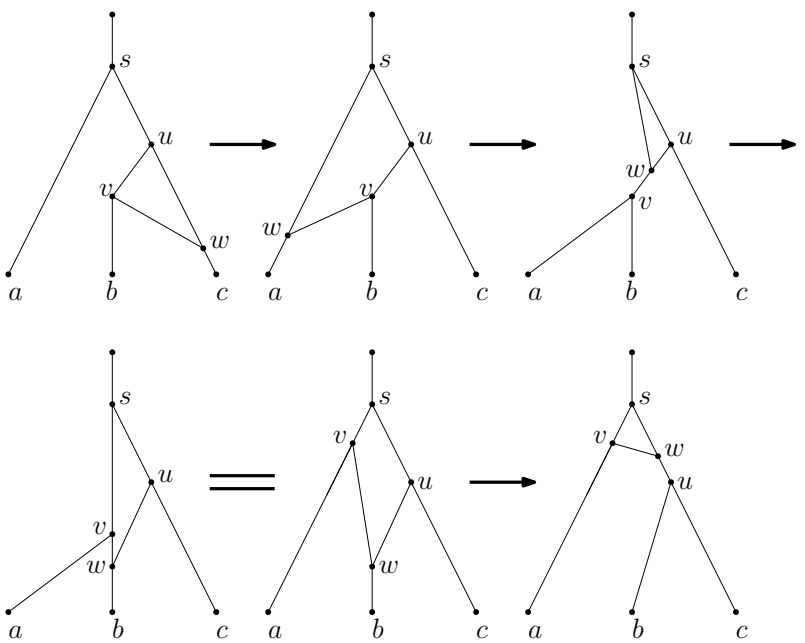


Figure 4.4: Sequence of moves used to move triangles with distance-2 head moves (Lemma 4.11).

Now suppose the network has k reticulations at the top, and there is a triangle $\langle u, v, w\rangle$ directly below $s = a_k$—if the triangle is below b_k, first redirect all reticulations at the top using one head_1 move. To move the triangle to the top, first move the triangle up to s using the previous sequence of at most 4 moves; then reverse the direction of the triangle using the distance-1 head move

$w \xrightarrow{(s,w)} (v, b_k)$, resulting in the triangle $\langle s, v, w\rangle$; and, lastly, apply $w \xrightarrow{(v,w)} (b_k, \cdot)$. Including the first move to redirect the reticulations at the top, this takes at most 7 head_2 moves. □

If the restriction to distance-2 moves is relaxed, the triangle can also be moved between adjacent tree nodes using one distance-3 head move $w \xrightarrow{(u,w)} (s, a)$. Adding or removing triangles from the top then takes at most two distance-3 moves, or one distance-4 move.

Lemma 4.12 ([Jan21] Lemma 7)**.** *Let N be a network and v a highest reticulation below the top reticulations. Suppose $v \xrightarrow{(u,v)} (x, y)$ is a valid head move resulting in a network N'. Then there is a sequence of at most $n + k + 5$ head_2 head moves from N to N'.*

Proof. Choose an up-down path from v to (x, y) not via (u, v). Note that if there is a part of this path above u, it is also above v and therefore we may assume it only contains tree nodes. Sequentially move the head of (u, v) to the pendant branches of this path as in Figure 4.5. This works for the entire path, except at the point where u is on the up-down tree path (the obvious move is a distance-3 move), and at the top (i.e., from 0 to 1 and from 3 to 5 in Figure 4.5).

Note that at the top, we need to move the head to the lowest reticulation arc at the top. This is only possible if this reticulation arc is directed away from u. If it is not, we redirect it using one distance-1 head move (Lemma 4.8), and redirect it back after we move the moving head down to the other branch of the up-down tree path. This accounts for two head_2 moves.

If u is on the up-down path, we use Lemma 4.11 to pass this point: Let $c \neq v$ be the other child of u and p the parent node of u; moving the head of (u, v) from an outgoing arc of c to the other outgoing arc (p, w) of p is equivalent to moving the triangle at u to a triangle at p using at most 4 head_2 moves and an addional move to reverse the direction of the triangle.

We have to be careful, because if the child c of u is not a tree node, this sequence of moves does not work. However, if c is a reticulation node, there exists a different up-down path from v to (x, y) not through u: such a path may use the other incoming arc of c.

At all other parts of the up-down path, the head may be simply moved to the arcs on the path. As we need one move for each arc on the path, and the path has length at most $n + k - 2$, the whole sequence takes at most $n + k - 2 + 2 + 5$ head_2 moves. □

Using these lemmas, it is easy to prove we can use distance-2 head moves to move reticulations to the top.

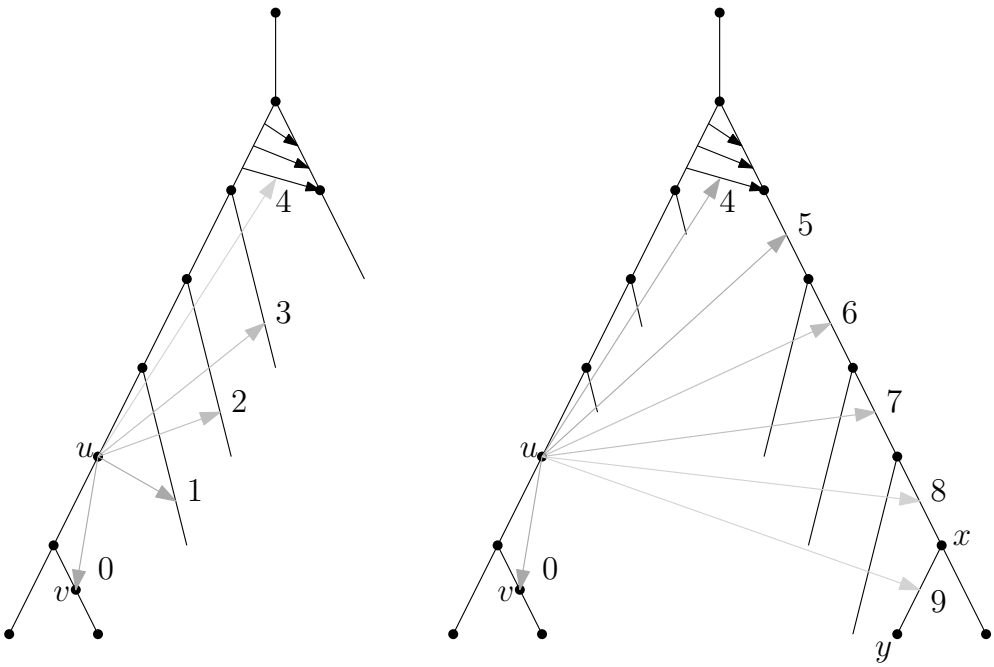


Figure 4.5: An example of a sequence as used in Lemma 4.12. Note that on the side of the tree containing the tail of the moving arc, we use the side branches to avoid cycles. The numbers represent the order of the distance-2 head moves. Note that the move from position 0 to position 1 is not a distance-2 head move, in this case we use the sequence of moves described in Lemma 4.11. Also note that position 4 is only allowed when the lowest reticulation at the top is directed away from the tail of the moving arc.

Lemma 4.13 ([Jan21] Lemma 8)**.** *Let $N \in \mathcal{N}(n,k)$, then there is a sequence of at most $5nk + 5k^2 + k$ distance-2 head moves turning N into a network with all reticulations at the top.*

Proof. Note that the network induces a partial order on the reticulation nodes. Suppose N has $l < k$ reticulations at the top. Let r be a highest reticulation node that is not yet at the top. One of the two corresponding reticulation arcs is head-movable. Let this be the arc (s, r).

If s is a child of a_l or b_l (as in Definition 2.17; i.e., s is directly below the top reticulations), then one head move suffices to get this reticulation to the top. By Lemma 4.12, this move can be replaced by a sequence of at most $n + k + 5$ distance-2 head moves.

Otherwise, there is at least one node between s and the top, let t be the lowest such node, that means that t is the parent of s. Because r is a highest reticulation that is not at the top, t is a tree node and there are arcs (t, s) and (t, q). The head move $r \xrightarrow{(s,r)} (t, q)$ is valid, and it creates a triangle. By Lemma 4.12, this head move can be replaced by a sequence of at most $n+k+5$ distance-2 head moves.

Now we move this triangle to the top using distance-2 head moves as in Lemma 4.11. This increases the number of reticulations at the top by one. To do this, we need at most 4 head_2 moves for each tree node on the path up to the top, and at most 7 moves to add the triangle to the top reticulations. As there are at most $n + k - 2$ tree nodes on this path, the total number of head_2 moves used amounts to at most $n + k + 5 + 4(n + k - 1) \leq 5n + 5k + 1$. □

Changing the tree

Networks with all reticulations at the top have exactly one embedded tree. Therefore, such networks are essentially determined by their embedded tree. This means we now only need to change this embedded tree. To achieve this, we use the lowest reticulation arc (a_k, b_k) to create a triangle that can move around the lower part of the network. Using the reticulation in this triangle, we simulate rSPR moves on the embedded tree.

Lemma 4.14 ([Jan21] Lemma 9). *Let $N, N' \in \mathcal{N}(n, k)$ $(k > 0)$ be networks with $k-1$ reticulations neatly at the top and the k-th reticulation at the bottom of a triangle. Suppose N and N' have the same embedded trees with the top reticulations oriented the same way along this tree, then $d_{\text{head}_2}(N, N') \leq 4n + 12$.*

Proof. Note that the network consists of $k-1$ reticulations at the top, and two pendant subtrees—isomorphic to the two pendant subtrees below the highest tree node of the embedded tree—one of which contains a triangle. The triangle can be moved through one of these subtrees using at most 4 head_2 moves for each tree-node it passes (Lemma 4.11). To move the triangle anywhere, we need to be able to move it between the pendant subtrees as well. This can be done by moving the triangle to the top, and then moving it down on the other side after redirecting all the top reticulations, using at most 14 head_2 moves (Lemma 4.11). None of these triangle moves changes the embedded tree: each of the intermediate networks has exactly one embedded tree, the triangle move can be done with one head move, and doing a head move keeps at least one embedded tree (Lemma 4.3). Hence, moving the triangle to the right place with at most $4(n-1) + 14$ head_2 moves and then redirecting the triangle and the

top reticulations as needed (using at most 2 moves to do this) gives a sequence of at most $4n + 12$ head_2 moves from N to N'. □

Lemma 4.15 ([Jan21] Lemma 10)**.** *Let $N, N' \in \mathcal{N}(n,k)$ be networks ($k > 0$) with all reticulations neatly at the top, and embedded trees T and T'. Then $d_{\text{head}_2}(N, N') \leq 2(6n + 24)d_{\text{rSPR}}(T, T') + 4n + 13$.*

Proof. Note that N and N' both have exactly one embedded tree, T and T' respectively, and we aim to change this embedded tree. It suffices to prove this for any T' that is one rSPR move removed from T, because the space of phylogenetic trees with the same leaf set is connected by rSPR moves (Theorem 2.60). Hence, let $u \xrightarrow{(u,v)} (x,y)$ be the rSPR move that transforms T into T'.

First suppose the rSPR move does not involve the root arc of the embedded trees. We can move triangles anywhere below the $k-1$ reticulations at the top by Lemma 4.14. Hence, there is a sequence of at most $4n + 12$ head_2 moves transforming N into a network M with the following properties: the tree T can be embedded in M; M has a reticulation arc (a,b) where a lies on the image of (x,y) in M, and the head b lies on the image of the other outgoing arc (x,z) of x if x is not the root and on the image of one of the outgoing arcs (y,z') of y otherwise.

This creates a situation where there are arcs (x,a), (a,b), (p,b), (a,y), and (b,ζ) with $p = x$ and $\zeta = z$ or $p = y$ and $\zeta = z'$. The case $p = x$ is depicted in Figure 4.6.

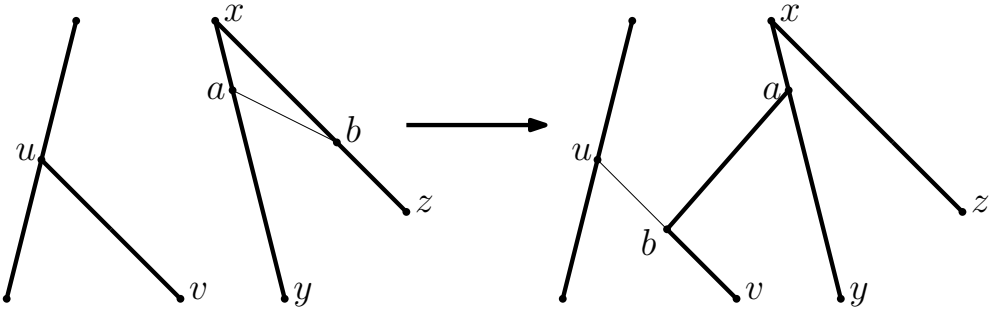


Figure 4.6: The head move used to simulate the rSPR move $u \xrightarrow{(u,v)} (x,y)$ on the embedded tree T to T'. The triangle is already in the position described in the proof of Lemma 4.15. The embedded trees T and T' are displayed with thick arcs in the left and right network respectively. The right network is obtained from the left one by the head move $b \xrightarrow{(a,b)} (u,v)$.

Now simulate the head move of (a, b) to the image of the T-arc (u, v) using at most $n+6$ head_2 moves (Lemma 4.12; noting that the up-down path contains at most n tree nodes). This is allowed because (a, b) is movable; b is not equal to the image of u as b is a reticulation node and the image of u a tree node; and the image of v is not above a, as otherwise the tail move $u \xrightarrow{(u,v)} (x, y)$ could not be valid. Let N' be the resulting network, and note that the embedded tree using the new reticulation arc is T'. Finally, create a new triangle at the parent p of u by moving (u, b) to the other outgoing arc of p using at most $n+6$ head_2 moves.

Next, suppose the rSPR move does involve the root arc of the embedded tree. Let $u \xrightarrow{(u,v)} (\rho, c)$ be such an rSPR move to the root arc of the tree, and let x and y be the nodes directly below the top: x on the side of the reticulations b_i, and y on the side of the tree nodes a_i. Apply $u \xrightarrow{(u,v)} (a_k, x)$, where (a_k, x) is an arc directly below the top. This produces the network with $k-1$ reticulations at the top, and (in Newick notation) embedded tree $((T \downarrow x, T \downarrow v), T \downarrow y)$, where $T \downarrow z$ denotes the part of tree T below z. Then do the rSPR move $u' \xrightarrow{(u',x)} (b_k, y)$, the other side of the top, producing a network with $k-1$ reticulations at the top and embedded tree $((T \downarrow x, T \downarrow y), T \downarrow v)$.

This creates the desired network with v below one side of the top, and x and y on the other side. Both these rSPR moves are performed as in the previous case, which did not involve the root arc. Hence, each rSPR move on the trees can be simulated with a sequence of at most $2(4n+12+2(n+6)) = 2(6n+24)$ head_2 moves.

To make the embedded trees isomrophic, we repeat this process for each rSPR move, which takes at most $(6n+24)d_{\text{rSPR}}(T, T')$ head_2 moves in total. Finally, we move the triangle back to the top without changing the embedded tree using at most $4n+12$ head_2 moves, and redirect the top reticulations using at most 1 move to produce N'. Tallying up all these moves, we get $d_{\text{head}_2}(N, N') \leq 2(6n+24)d_{\text{rSPR}}(T, T') + 4n + 13$. □

Theorem 4.16 ([Jan21] Theorem 1). *For all $n, k > 0$, the space $\mathcal{N}_{\text{head}_2}(n, k)$ is connected and* $\text{diam}_{\text{head}_2}(n, k) \leq 2(6n+24)\,\text{diam}_{\text{rSPR}}(n, 0) + 4n + 13 + 2k(5n + 5k + 1)$.

Proof. Let $N, N' \in \mathcal{N}(n, k)$ be two arbitrary networks. Use Lemma 4.13 and Lemma 4.8 to change N and N' into networks N_n and N'_n with all reticulations neatly at the top using at most $2k(5n + 5k + 1)$ distance-2 head moves. Now, Lemma 4.15 implies there is a sequence of at most $2(6n+24)\,\text{diam}_{\text{rSPR}}(n, 0) + 4n + 13$ distance-2 head moves from N_n to N'_n. Hence, tier-k of phylogenetic network space is connected by distance-2 head moves with diameter $\text{diam}_{\text{head}_2}(n, k) \leq 2(6n + 24)\,\text{diam}_{\text{rSPR}}(n, 0) + 4n + 13 + 2k(5n + 5k + 1)$ □

The next corollary follows immediately from Theorem 2.60, which contains an upper bound for $\mathrm{diam}_{\mathrm{rSPR}}(n, 0)$.

Corollary 4.17. *For all $n, k > 0$, we have* $\mathrm{diam}_{\mathrm{head}_2}(n, k) \leq 12n^2 + 10kn + 10k^2 + 52n + 2k + 13$

4.3 Diameter bounds

The tail move diameter bounds from the previous chapter are obtained using a technique where an isomorphism is built incrementally. For head moves, we can employ a similar technique. For any pair of networks, we build an isomorphism between growing subgraphs where, in each step, we only have to use a small number of moves to grow the isomorphism. For tail moves and rSPR moves, it is convenient to build this isomorphism bottom-up. Head moves are essentially upside-down tail moves. Hence, for head moves, we build an isomorphism starting at the root.

Lemma 4.18 ([Jan21] Lemma 22 Case 1)**.** *Let $N_1, N_2 \in \mathcal{N}(n, k)$ with $k > 0$, and let Y_1 and Y_2 be non-empty up-closed sets of nodes of N_1 and N_2 such that $N_1[Y_1] \overset{\phi}{\simeq} N_2[Y_2]$. Suppose there is a highest node x_1 of $N_1 \setminus Y_1$ such that x_1 is a tree node. Then there is a network N_2' with up-closed set of nodes Y_2' such that $N_1[Y_1 \cup \{x_1\}] \simeq N_2'[Y_2']$ and $d_{\mathrm{head}}(N_2, N_2') \leq 4$.*

Proof. Because x_1 is a highest node not in Y_1, the parent p_1 of x_1 is in Y_1 and there is a corresponding node $p_2 = \phi(p_1)$ in Y_2. This node must have at least one child x_2 that is not in Y_2, as otherwise the degrees of p_1 and p_2 in $N_1[Y_1]$ and $N_2[Y_2]$ do not coincide.

1. **The node x_2 is a tree node.** In this case we can add x_1 and x_2 to Y_1 and Y_2 and set $\phi(x_1) = x_2$ to get an extended isomorphism. No head moves are necessary to make this extension.

2. **The node x_2 is a reticulation.** We make sure p_2 has a tree node y_2 as a child not in Y_2, using at most 3 head moves. We can then add x_1 to Y_1 and y_2 to Y_2 and extend the isomorphism with $\phi(x_1) = y_2$. To create this tree node, we use a tree node $c_2 \in N_2 \setminus Y_2$, which exists because there is a tree node in $N_1 \setminus Y_1$.

    a) **The arc (p_2, x_2) is movable.** Let t_2 be the parent of c_2. Apply $x_2 \xrightarrow{(p_2, x_2)} (t_2, c_2)$, which is valid because c_2 cannot be above p_2 (otherwise $c_2 \in Y_2$, a contradiction) and $t_2 \neq p_2$—if $t_2 = p_2$ then we can add c_2 to the isomorphism directly, as it is a tree node child of

p_2 not in Y_2. Now the head move $x_2 \xrightarrow{(z_2,x_2)} (c_2, \cdot)$ where $z_2 \neq p_2$ is valid because it moves a head down. After this move, the tree node c_2 is the child of p_2, so we can extend the isomorphism with a tree node $\phi(x_1) = c_2$ using at most 2 head moves (Figure 4.7).

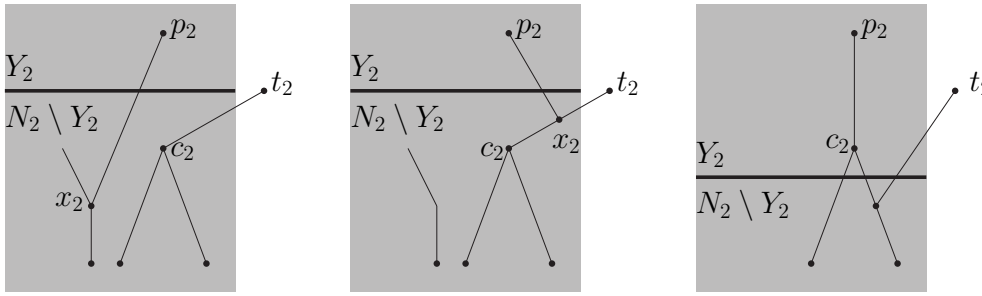


Figure 4.7: The moves and incremented isomorphism for Lemma 4.18 Case 2a. For nodes outside of the shaded region, it is not known whether they are in Y_2.

b) **The arc (p_2, x_2) is not movable.** This means that x_2 is on the side of a triangle $\langle z_2, x_2, d_2 \rangle_r$, with $z_2 \neq p_2$. The arc (z_2, d_2) is movable, and $d_2 \xrightarrow{(z_2,d_2)} (u_2, v_2)$ is valid for some arc (u_2, v_2) with v_2 not in Y_2 and (u_2, v_2) distinct from both (x_2, d_2) and from the outgoing arc of d_2. Such an arc (u_2, v_2) exists: choose a leaf l not equal to the child of d_2 (if that node is a leaf); as all leaves are added to the isomorphism last, the leaf is not in Y_2, furthermore, l is not above z_2, and the incoming arc of l is not equal to (x_2, d_2) nor to the outgoing arc of d_2. Applying the head move $d_2 \xrightarrow{(z_2,d_2)} (\cdot, l)$ creates the situation of the previous case (Case 2a), and we can use 2 more head moves to create a network with a tree node c_2 below p_2 which maintains the isomorphism of the upper part Y_2. Hence we can extend the isomorphism with a tree node $\phi(x_1) = c_2$ using at most 3 head moves.

3. **The node x_2 is a leaf.** Again, note that there is a tree node c_2 in $N_2 \setminus Y_2$, and let its parent be t_2. Note also that N_2 has a reticulation node r_2 with incoming arc (s_2, r_2), which is movable to (p_2, x_2) unless $r_2 = p_2$ (if $p_2 = s_2$, then the other incoming arc (q_2, r_2) is also movable, and can instead be moved to (p_2, x_2)).

   a) **The nodes p_2 and t_2 are the same.** The node $c_2 \in N_2 \setminus Y_2$ is a tree node and a child of p_2. Hence, we can immediately extend the isomorphism with a tree node by setting $\phi(x_1) = c_2$.

b) **The nodes p_2 and r_2 are the same.** First note that if $s_2 = t_2$ (resp. $q_2 = t_2$), then the move $p_2 \xrightarrow{(q_2,p_2)} (t_2, c_2)$ (resp. $p_2 \xrightarrow{(s_2,p_2)} (t_2, c_2)$) is valid, and does not affect the isomorphism between the up-closed sets because $x_2, c_2 \notin Y_2$. Furthermore, this move results in a network in which the tree node c_2 is a child of p_2. Hence, after this move, we can extend the isomorphism with a tree node by setting $\phi(x_1) = c_2$.

Now assume that $s_2, q_2 \neq t_2$. As $r_2 = p_2$ is in Y_2 and c_2 is not in Y_2, we know c_2 is not above p_2. Hence, as (s_2, r_2) is movable and $s_2 \neq t_2$, the move $p_2 \xrightarrow{(s_2,p_2)} (t_2, c_2)$ is valid. Then, as $q_2 \neq t_2$, the moves $p_2 \xrightarrow{(t_2,p_2)} (q_2, x_2)$ followed by $p_2 \xrightarrow{(q_2,p_2)} (s_2, c_2)$ are valid as well. In the resulting network, the tree node c_2 is a child of p_2, so we can extend the isomorphism by $\phi(x_1) = c_2$ with a tree node after at most three moves.

c) **The nodes t_2 and r_2 are the same.** First apply the move $r_2 \xrightarrow{(s_2,r_2)} (p_2, x_2)$. If $p_2 = q_2$, this results in a network where c_2 is a child of p_2, so that we can extend the isomorphism by $\phi(x_1) = c_2$ with a tree node. Otherwise, we can apply the additional two moves $r_2 \xrightarrow{(p_2,r_2)} (q_2, c_2)$ and $r_2 \xrightarrow{(q_2,r_2)} (s_2, x_2)$, which also results in a network where c_2 is a child of p_2. Hence, we can extend the isomorphism by $\phi(x_1) = c_2$ with a tree node after at most three moves in this case.

d) **The nodes s_2 and t_2 are different nodes and $r_2 \neq p_2, t_2$.** First move $r_2 \xrightarrow{(s_2,r_2)} (p_2, x_2)$. In the resulting network, the arc (p_2, r_2) is movable, and the move $r_2 \xrightarrow{(p_2,r_2)} (t_2, c_2)$ is valid because c_2 is not above p_2 and $p_2 \neq t_2$. This makes (t_2, r_2) movable, and we can apply the move $r_2 \xrightarrow{(t_2,r_2)} (s_2, x_2)$ because $s_2 \neq t_2$ and x_2 is a leaf, so it is not above t_2. Now lastly, we restore the reticulation by moving (s_2, r_2) back to its original position. Hence in this situation, 4 head moves suffice to make p_2 the parent of a tree node c_2, so that we can extend the isomorphism by $\phi(x_1) = c_2$ with a tree node (Figure 4.8).

e) **The nodes s_2 and t_2 are the same and $r_2 \neq p_2, t_2$.** Note that a child of t_2 is a tree node and a child of s_2 is a reticulation. This means that $s_2 = t_2$ is a tree node, as it has two distinct children. As (p_2, x_2) is a leaf arc, we may apply the move $r_2 \xrightarrow{(s_2,r_2)} (p_2, x_2)$. Now, the move $r_2 \xrightarrow{(p_2,r_2)} (s_2, c_2)$ is valid, because $p_2 \neq s_2$ and c_2 is not above p_2 (otherwise c_2 has to be in Y_2, contradicting our assumption). In the resulting network, we can move (s_2, r_2) back

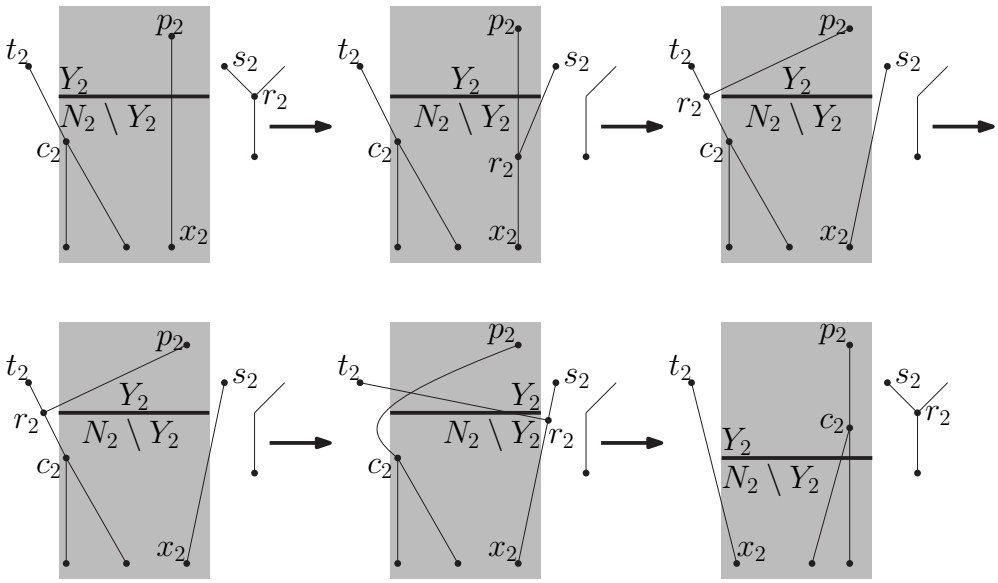


Figure 4.8: The moves and incremented isomorphism for Lemma 4.18 Case 3d. For nodes outside of the shaded region, it is not known whether they are in Y_2.

to its original position. This all takes three head moves, and makes sure that a child c_2 of p_2 is a tree node. This means we can extend the isomorphism by setting $\phi(x_1) = c_2$ (and if r_2 was in Y_2, changing $\phi(\phi^{-1}(r_2)) = r_2$ to $\phi(\phi^{-1}(r_2)) = r_2$) using at most 3 head moves to add a tree node. □

Lemma 4.19 ([Jan21] Lemma 22 Case 3)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ with $k > 0$, and let Y_1 and Y_2 be non-empty up-closed sets of nodes of N_1 and N_2 such that $N_1[Y_1] \overset{\phi}{\simeq} N_2[Y_2]$. Suppose each highest node of $N_1 \setminus Y_1$ and of $N_2 \setminus Y_2$ is a reticulation or a leaf, and suppose $N_1 \setminus Y_1$ has a highest node x_1 that is a reticulation. Then there is a network N_2' with up-closed set of nodes Y_2' such that $N_1[Y_1 \cup \{x_1\}] \simeq N_2'[Y_2']$ and $d_{\text{head}}(N_2, N_2') \leq 2$.*

Proof. This means the two parents p_1 and q_1 of x_1 are in Y_1, and consequently have corresponding nodes p_2 and q_2 in Y_2. Both these nodes also have at least one child not in Y_2, say c_2^p and c_2^q.

1. **The children of p_2 and q_2 are equal (i.e., $c_2^p = c_2^q$).** In this case, we can immediately extend the isomorphism with $\phi(x_1) = c_2^p$.

2. **Both nodes c_2^p and c_2^q are leaves.** Note that because x_1 is a reticulation node not in Y_1, there must also be a reticulation node $r_2 \in N_2$ not in

Y_2. Let its movable incoming arc be (s_2, r_2). As $p_2 \neq q_2$ we know that s_2 can be equal to at most one of p_2 and q_2, hence we can assume without loss of generality that $s_2 \neq p_2$. Then the head move $r_2 \xrightarrow{(s_2,r_2)} (p_2, c_2^p)$ is allowed, because the leaf c_2^p cannot be above s_2. Now (p_2, r_2) is movable because the child of r_2 is a leaf, and the move $r_2 \xrightarrow{(p_2,r_2)} (q_2, c_2^q)$ is valid because $p_2 \neq q_2$ and c_2^q is a leaf, and hence not above p_2. After this head move, p_2 and q_2 have a common child $x_2 := r_2$, and the isomorphism can be extended with one reticulation by setting $\phi(x_1) = x_2$ using at most 2 head moves.

3. **Both nodes c_2^p and c_2^q are reticulations.** Assume without loss of generality that c_2^p is not below c_2^q.

   a) **The arc (p_2, c_2^p) is movable.** Apply the move $c_2^p \xrightarrow{(p_2,c_2^p)} (q_2, c_2^q)$, which is allowed because c_2^q is not above p_2, and $p_2 \neq q_2$. Now p_2 and q_2 have a common child $x_2 := c_2^p$, so we can add one reticulation to Y_1 and Y_2 and extend the isomorphism by $\phi(x_1) = x_2$ using 1 head move.

   b) **The arc (p_2, c_2^p) is not movable.** Because (p_2, c_2^p) is not movable, c_2^p must be the side node of a triangle $\langle t, c_2^p, z\rangle$, and therefore its outgoing arc (c_2^p, z) is movable. If $t = q$, then we can extend the isomorphism by setting $\phi(x_1) = c_2^p$ without applying any move. Otherwise, c_2^q is not above c_2^p by assumption, so the move $z \xrightarrow{(c_2^p,z)} (q_2, c_2^q)$ is valid. After this move, the other incoming arc (t, c_2^p) of c_2^p becomes movable, and we apply $c_2^p \xrightarrow{(t,c_2^p)} (z, c_2^q)$ to move it down. Now p_2 and q_2 have a common child $x_2 := z$, and the isomorphism can be extended with one reticulation by setting $\phi(x_1) = x_2$ using at most 2 head moves (Figure 4.9).

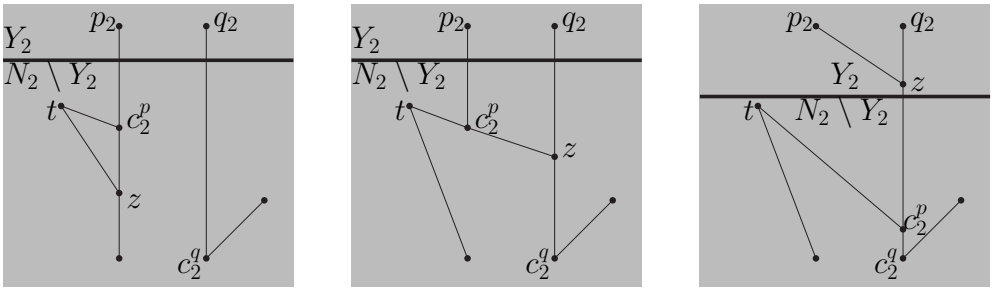


Figure 4.9: The moves and incremented isomorphism for Lemma 4.19 Case 3b.

4. **One of the nodes c_2^p and c_2^q is a reticulation, and the other is a leaf.** Assume without loss of generality that $\boldsymbol{c_2^p}$ is a reticulation and $\boldsymbol{c_2^q}$ is a leaf. The subcases here work exactly like the previous subcases in Case 3.

   a) **The arc (p_2, c_2^p) is movable.** Apply $c_2^p \xrightarrow{(p_2, c_2^p)} (q_2, c_2^q)$, which is allowed because c_2^q is not above p_2, and $p_2 \neq q_2$. Now p_2 and q_2 have a common child $x_2 := c_2^p$, so we can add one reticulation to Y_1 and Y_2 and extend the isomorphism by $\phi(x_1) = x_2$ using 1 head move.

   b) **The arc (p_2, c_2^p) is not movable.** Because (p_2, c_2^p) is not movable, c_2^p must be the side node of a triangle, and therefore its outgoing arc (c_2^p, z) is movable. Because c_2^q is a leaf, it is not above c_2^p, so the move $z \xrightarrow{(c_2^p, z)} (q_2, c_2^q)$ is valid. Now the other incoming arc (t, c_2^p) of c_2^p becomes movable, and we can move it down with $c_2^p \xrightarrow{(t, c_2^p)} (z, c_2^q)$. Now p_2 and q_2 have a common child $x_2 := z$, and the isomorphism can be extended with one reticulation by setting $\phi(x_1) = x_2$ using at most 2 head moves. □

Lemma 4.20 ([Jan21] Lemma 22)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ be networks with $n, k > 0$ networks and Y_1 and Y_2 be non-empty up-closed sets of N_1 and N_2 such that $N_1[N_1] \simeq \phi N_2[N_2]$. Suppose $N_1 \setminus Y_1$ contains t tree nodes and r reticulations, then there is a sequence of at most $4t + 2r$ moves that transforms N_1 into a network N_1' such that $N_1' \simeq N_2$.*

Proof. We prove the claim by induction on t and r. First we consider the base case where $t = r = 0$. In that case, $|N_1 \setminus Y_1|$ and $|N_2 \setminus Y_2|$ consist of only leaves, so N_1 and N_2 are unlabeled isomorphic. Therefore $d_{\text{head}}(N_1, N_2) = 0 \leq 4 \cdot 0 + 2 \cdot 0$.

Now, we may assume that $t \neq 0$ or $r \neq 0$, so there exists a highest reticulation or tree node in $N_1 \setminus Y_1$ and $N_2 \setminus Y_2$. We consider three cases depending on the existence of these nodes.

1. **There is a highest node x_1 of N_1 not in Y_1 such that x_1 is a tree node.** By Lemma 4.18, there exists a network network N_2' with an up-closed set Y_2' such that $N_1[Y_1 \cup \{x_1\}] \simeq N_2'[Y_2']$ and $d_{\text{head}}(N_2', N_2) \leq 4$. As $N_1 \setminus (Y_1 \cup \{x_1\})$ contains one tree node fewer than $N_1 \setminus Y_1$, by induction there is a sequence of length at most $4(t-1) + 2r$ from N_1 to a network N_1'' such that $N_1'' \simeq N_2'$. By applying the reverse of the 4 moves from N_2 to N_2' in N_1'', it follows that there is a sequence of length at most $4 + 4(t-1) + 2r = 4t + 2r$ from N_1 to a network N_1' (via N_1'') such that $N_1' \simeq N_2$.

2. **There is a highest node x_2 of N_2 not in Y_2 such that x_2 is a tree node.** By symmetric arguments to the previous case—exchanging the roles of N_1 and N_2—we have that there is a sequence of at most $4t + 2r$ head moves turning N_2 into a network N_2' such that $N_1 \simeq N_2'$. As all head moves are reversible, there is also a sequence of at most $4t+2r$ head moves turning N_1 into a network N_1' such that $N_1' \simeq N_2$.

3. **Each highest node x_1 of N_1 not in Y_1 and x_2 of N_2 not in Y_2 is a reticulation node or a leaf.**

   a) **There exists a highest node x_1 of N_1 not in Y_1 which is a reticulation node.** By Lemma 4.19, there exists a network network N_2' with an up-closed set Y_2' such that $N_1[Y_1 \cup \{x_1\}] \simeq N_2'[Y_2']$ and $d_{\text{head}}(N_2', N_2) \leq 2$. As $N_1 \setminus (Y_1 \cup \{x_1\})$ contains one reticulation fewer than $N_1 \setminus Y_1$, by induction there is a sequence of length at most $4(t-1)+2r$ from N_1 to a network N_1'' such that $N_1'' \simeq N_2'$. By applying the reverse of the 2 moves from N_2 to N_2' in N_1'', it follows that there is a sequence of length at most $2+4t+2(r-1) = 4t+2r$ from N_1 to a network N_1' (via N_1'') such that $N_1' \simeq N_2$.

   b) **There exists a highest node x_2 of N_2 not in Y_2 which is a reticulation node.** Do the same as in the previous case, switching the roles of N_1 and N_2.

   c) **All highest nodes of N_1 not in Y_1 and of N_2 not in Y_2 are leaves.** In this case $t = 0$ and $r = 0$, and we are in the base case of the induction. □

By setting $Y_1 = \{\rho_1\}$ and $Y_2 = \{\rho_2\}$, and noting that a network in $\mathcal{N}(n, k)$ has $n + k - 1$ tree nodes and k reticulations (Observation 2.7), we get the following bound on the distance between any two directed networks in the same tier.

Lemma 4.21 ([Jan21] Lemma 22). *Let $N_1, N_2 \in \mathcal{N}(n, k)$ be networks with $n, k > 0$ networks, then there is a head move sequence of at most $4n + 6k - 4$ moves from N_1 to a network N_1' such that and $N_1' \simeq N_2$.*

Lemma 4.22 ([Jan21] Lemma 23). *Let $N \simeq N' \in \mathcal{N}(n, k)$ with $n, k > 0$, then $d_{\text{head}}(N, N') \leq 2n$.*

Proof. Because $N \simeq N'$, the only difference between N and N' is a permutation of the leaves, say $\pi = (l_1^1, \ldots, l_{\Pi_1}^1)(l_1^2, \ldots, l_{\Pi_2}^2) \cdots (l_1^P, \ldots, l_{\Pi_q}^P)$ to get from N to N' (where all l_i^j are distinct). Note also that there is a reticulation in N with a head-movable arc (t, r), which is movable to the incoming arc of any leaf,

except the other child c_t of t, if it is a leaf. If $c_t = l^j_{\Pi_j}$, shift indices of the cycle by one. A sequence of moves from N to N' consists of the moves

- $r \xrightarrow{(t,r)} (p(l^j_{\Pi_j}), l^j_{\Pi_j})$;
- $r \xrightarrow{(p(l^j_{\Pi_j}),r)} (p(l^j_{\Pi_j-1}), l^j_{\Pi_j-1})$;
- $r \xrightarrow{(p(l^j_{\Pi_j-1}),r)} (p(l^j_{\Pi_j-2}), l^j_{\Pi_j-2})$;
- ...
- $r \xrightarrow{(p(l^j_2),r)} (p(l^j_1), l^j_1)$;
- $r \xrightarrow{(p(l^j_1),r)} (t, l^j_{\Pi_j})$;
- $r \xrightarrow{(t,r)} (s, c)$,

for each cycle π_j $(1 \leq j \leq q)$ of π, where $c = \pi_j(c')$, with c' the child of r in N (if c' is a leaf in the j-th permutation, the child of r becomes $\pi(c')$) and s is the other parent of r in N. This permutes the leaves in N by π so that the resulting network is N'. The sequence is allowed, provided that no two subsequent leaves in a cycle have a common parent (e.g., $p(l^j_i) = p(l^j_{i-1})$). There is always a permutation in which this does not happen. Indeed, suppose this were to happen with leaves x and y, so that we have w.l.o.g. $\pi(z) = x$, $\pi(x) = y$ and $\pi(y) = w$. Then, redefining and applying π with $\pi(z) := y$, $\pi(y) := w$, and $\pi(x) := x$ results in an isomorphic network, unless $z = y$ and $w = x$, in which case we redefine π as $\pi(x) := x$ and $\pi(y) := y$.

The worst case is attained when there are a maximal number of cycles in the permutation, which happens when π consists of only 2-cycles. In such a case there will be $n/2$ cycles of length 2. Each such a cycle takes four moves. An upper bound to the length of the sequence is therefore $4(n/2) = 2n$. □

A direct corollary of the previous two lemmas is the following theorem, giving an upper bound on the diameter of head move space. To see this, note that any head move is reversible, and hence we can concatenate sequences in different directions.

Theorem 4.23 ([Jan21] Theorem 4). *Let $N, N' \in \mathcal{N}(n,k)$ with $n, k > 0$, then $d_{\text{head}}(N, N') \leq 6n + 6k - 4$.*

4.4 Internal labels

Like in the previous chapter, we now extend the results to internally labeled networks. The techniques are quite similar, as we, again, use the structure of ladder caterpillars to permute the tree nodes and reticulations, and we collect all degree-2 nodes in one path.

4.4.1 Labeled isomorphisms without degree-2 nodes

Most of the proofs in this section follow essentially from the proofs in the previous chapter by reversing the direction of all the arcs. Of course, this only results in a network when there is only one leaf. Nevertheless, the results generalize with a little extra care.

We will first show how to swap the labels of two reticulations using a constant number of head moves in any network. After establishing this fact, we turn to the labels of the tree nodes nodes. To permute these, we transform the network into a ladder caterpillar. In those networks, we can swap the labels of *adjacent* tree nodes using a constant number of head moves or a number of head_1 moves linear in the number of leaves. Note that this gives a slightly worse result than for tail moves, as, there, we were able to swap adjacent reticulations using a constant number of tail_1 moves.

Reticulation labels

Lemma 4.24. *Let $\dot{N} \in \dot{\mathcal{N}}(n,k)$ be a network with two reticulations x and y labelled $l(x)$ and $l(y)$. Then there is a network $\dot{N}'$ in which the nodes labelled $l(x)$ and $l(y)$ are adjacent, such that $d_{\text{head}}(\dot{N},\dot{N}') \leq 1$. If $\dot{N}$ is a ladder caterpillar there is such a $\dot{N}'$ with $d_{\text{head}_1}(\dot{N},\dot{N}') \leq k$.*

Proof. If x is above y and y is above x, there is a cycle. Without loss of generality, assume that y is not above x. One of the incoming arcs of x is head movable (Lemma 4.2), and it can be moved to any arc adjacent to y using one head move. This move creates a node labelled $l(x)$ adjacent to y, which is labelled with $l(y)$.

Now suppose $\dot{N}$ is a ladder caterpillar. As all reticulations are on one path, and the incoming arc of each reticulation that is not on this path is movable, we may simply move the highest of the two reticulations down to the other reticulation, along this path. The path has length at most k, so we can do this using a sequence of at most k head_1 moves. □

Lemma 4.25. *Let $\dot{N}$ be an internally labeled network, and let x and y be a pair of adjacent reticulations. Then, there is a sequence of at most 2 head_1 moves that swaps the labels of x and y.*

Proof. The used sequences of head_1 moves are obtained by taking the moves shown in Figure 3.7 and reversing the direction of all arcs. □

Lemma 4.26. *Let $\dot{N}$ be a network, and x and y two reticulations. Then, using a sequence of at most 4 head moves the labels of x and y can be swapped. If $\dot{N}$ is a ladder caterpillar, x and y can be swapped using at most $2k + 2$ head_1 moves.*

Proof. By Lemma 4.24, we can use one head move in $\dot{N}$ to make the nodes labeled $l(x)$ and $l(y)$ adjacent. Then, using Lemma 4.25, the labels of these two nodes can be swapped with at most two head moves. Lastly, the first move is reversed. This results in a network isomorphic to $\dot{N}$, where only the labels of x and y are swapped.

Using the same strategy, two reticulations in a ladder caterpillar can be made adjacent using k head_1 moves. Then they can be swapped with 2 head_1 moves, and the first k moves can then be reversed again. This results in swapping x and y using at most $2k + 2$ head_1 moves. □

Proposition 4.27. *Let $\dot{N}$ be a network with k reticulations, and let π be a permutation of the reticulation labels. Then there is a sequence of at most $4k$ head moves that permutes the reticulation labels with π and leaves all other labels the same.*

If $\dot{N}$ is a ladder caterpillar, this can be achieved using a sequence of at most $2k^2 + 2k$ head_1 moves.

Proof. Each permutation of n elements decomposes in at most n swaps. Each of the swaps of two reticulation labels can be achieved using at most 4 head moves or $2k + 2$ head_1 moves in a ladder caterpillar (Lemma 4.26). Hence, to permute the reticulation labels with π, we need at most $4k$ head moves, or $2k^2 + 2k$ head_1 moves in a ladder caterpillar. □

Tree node labels

In this subsection, we will work with ladder caterpillars. These networks have all tree nodes in one path, which makes it easy to permute the labels of tree nodes. Note that, unlike for the reticulations and tail moves, here we have tree nodes that are outside of the ladder (i.e., the biconnected component) as well. Hence, these need some special attention beyond the observation that we can reverse the direction of all arcs.

In the following two lemmas, we show when and how we can permute the labels of the tree nodes. The first lemma shows that this cannot always be done, and the second shows how it can be done when it is possible. The proof of the first lemma is identical to the proof of Lemma 3.20 with the direction of all arcs reversed.

Lemma 4.28. *Let $\dot{N}$ be a network in which the child c of the root is labelled $l(c)$. Then, after any head move in $\dot{N}$, the child of the root will still be labelled $l(c)$.*

Lemma 4.29. *Let $\dot{N} \in \dot{\mathcal{N}}(n,k)$ $(k > 0)$ be a ladder caterpillar with two adjacent tree nodes x and y. Then, using a sequence of at most 8 head moves or $2n+6$ head_1 moves, the labels of x and y can be swapped, except if x or y is the child of the root.*

Proof. Without loss of generality, we assume that x is above y. First suppose x and y are both part of the ladder, then the sequences shown in Figure 3.8 can be used to swap the labels of x and y with at most 5 head_1 moves, except if x is the child of the root. Now suppose that y is in the caterpillar, then the the sequence in Figure 4.10 can be used to swap the labels of x and y using at most 8 head moves, or $2n+6$ head_1 moves. □

Now we know how to swap labels of tree nodes in a ladder caterpillar, we can permute all tree node labels of such a network.

Proposition 4.30. *Let $\dot{N} \in \dot{\mathcal{N}}(n,k)$ be a ladder caterpillar, and let π be a permutation of the tree node labels. Then there is a sequence of at most $8(n+k-1)^2$ head moves, or $(2n+6)(n+k-1)^2$ head_1 moves, that permutes the tree node labels with π and leaves all other labels the same, except if the child of the root c is labelled $l(c)$ and $l(c) \neq \pi(l(c))$.*

Proof. Note that all $n+k-1$ tree nodes are in one path, and that we can swap adjacent labels of tree nodes using at most 8 head moves or $2n+6$ head_1 moves (Lemma 4.29). Any permutation of m elements of a sequence can be decomposed into at most m^2 neighbour swaps. Hence, to permute the tree node labels in a network, we need at most $8(n+k-1)^2$ head moves, or $(2n+6)(n+k-1)^2$ head_1 moves. □

Using the results above, we can permute all labels of a network using head moves. Hence, we can find a sequence between any two internally labeled networks (with only a few exceptions).

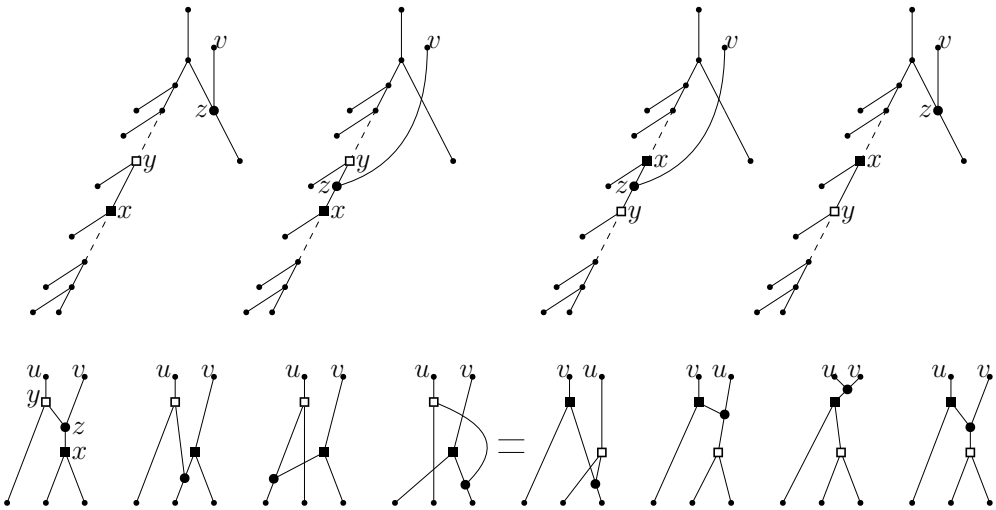


Figure 4.10: The sequence of 8 head moves, or $2n + 6$ head_1 moves used in Lemma 4.29. The first row shows the high level idea of the sequence: first move a reticulation between the tree nodes, then use the second row to swap the labels of the tree nodes, and, lastly, move the reticulation back to where it came from. Moving the reticulation takes at most n head_1 moves, or one head move, and the second row shows a sequence of 6 head_1 moves. If $u = v$, the last two moves in the sequence of the second row are not necessary.

Theorem 4.31. *Let $\dot{N}_1, \dot{N}_2 \in \dot{\mathcal{N}}(n,k)$ be networks. If $l_1(c_1) = l_2(c_2)$ for c_1 and c_2 the children of the roots of $\dot{N}_1$ and $\dot{N}_2$, there exists a head move sequence of length at most $2\,\text{diam}_{\text{head}}(n,k) + 4k + 8(n+k-1)^2$ and a head_2 sequence of length at most $2\,\text{diam}_{\text{head}_2}(n,k) + 2k^2 + 2k + (2n+6)(n+k-1)^2$ from $\dot{N}_1$ to $\dot{N}_2$. Otherwise, there is no sequence from $\dot{N}_1$ to $\dot{N}_2$.*

Proof. We first observe that the labels of c_1 and c_2 cannot be changed using head moves (Lemma 4.28). Hence, there is indeed no sequence between $\dot{N}_1$ and $\dot{N}_2$ if $l_1(c_1) \neq l_2(c_2)$.

In all remaining cases, Theorems 4.23 and 4.16 show that for any network $\dot{M} \in \dot{\mathcal{N}}(n,k)$, we can turn $\dot{N}_1$ and $\dot{N}_2$ into networks $\dot{N}_1'$ and $\dot{N}_2'$ such that $\dot{N}_1' \simeq_{X^l} \dot{N}_2' \simeq_{X^l} \dot{M}$. In particular, we may choose $\dot{M}$ to be a ladder caterpillar by Lemma 2.21. Then, we can permute the labels of the tree nodes using at most $8(n+k-1)^2$ head moves or $(2n+6)(n+k-1)^2$ head_2 moves (Proposition 4.30). Lastly, using at most $4k$ head moves or $2k^2 + 2k$ head_2 moves, we can permute the labels of the reticulations (Proposition 4.27). Combining these sequences, we get a sequence from $\dot{N}_1$ to $\dot{N}_2$ of the desired length. □

Like the bound for tail moves, the upper bound on the head move diameter for internally labeled networks contains a quadratic term. As mentioned before, this quadratic term disappears for rSPR moves, where we use a combination of tail and head moves (Chapter 5). As for tail moves, it is unclear whether this quadratic term is necessary.

4.4.2 Degree-2 nodes

In this section, we present a procedure to move all degree-2 nodes of a network to one arc. Thus, combining this with the previous results, we show how to find a sequence between any pair of networks in $\dot{\mathcal{N}}_{\text{head}}(n, k, m)$, with a some exceptions related to the root path.

For simplicity, we restrict our attention to non-local head moves. This is, most importantly, because distance-2 head moves suffer from a similar problem in the presence of degree-2 nodes, as distance-1 head moves did in their absence (Figure 4.11).

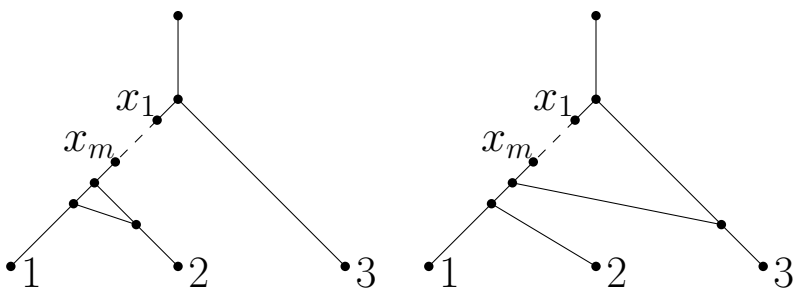


Figure 4.11: Two networks in $\dot{\mathcal{N}}(2, 1, m)$ not connected by a sequence of head_{m+1} moves. The m degree-2 nodes $x_1, \ldots, x_m$ form a barrier which prevents any distance-$(m-1)$ move that involves a part of the network above these nodes.

Networks with one leaf are treated separately, as their spaces are tightly related to tail move spaces. In fact, we have an isomorphism space $\dot{\mathcal{N}}_{\text{head}}(1, k, m) \overset{\phi}{\simeq} \dot{\mathcal{N}}_{\text{tail}}(1, k, m)$, where $\phi(\ddot{N})$ is the network obtained from $\ddot{N}$ by reversing the direction of each arc. Hence, by reversing the direction of all arcs, Theorem 3.35 fully characterizes $\dot{\mathcal{N}}_{\text{head}}(1, k, m)$ as follows.

Theorem 4.32. *For all $k \geq 0$ except $k = 2$, two networks $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}_{\text{head}}(1, k, m)$ are in the same component iff their root paths are labeled isomorphic. For $k = 2$, $\ddot{N}, \ddot{N}' \in \dot{\mathcal{N}}_{\text{head}}(1, 2, m)$ are in the same component iff their root paths are labeled isomorphic and either there is a subdivided arc besides the root arc and*

the arc between the two tree nodes, or there are no such subdivided arcs and the subdivided arcs between the two tree nodes in both networks are isomorphic.

We can now focus on spaces of networks with at least two leaves.

Lemma 4.33. *Let $n > 1$, $k > 0$, and $\ddot{N} \in \dot{\mathcal{N}}(n, k, m)$. Then there is a sequence of at most m* head *moves that turns $\ddot{N}$ into a network $\ddot{N}'$ without parallel paths.*

Proof. For each pair of parallel paths, move a bottom arc to a leaf that is not the child of the bottom node of the parallel paths. As each parallel path needs at least one degree-2 node, there are at most m pairs of parallel paths. □

Lemma 4.34. *Let $\ddot{N}$ be a subdivided network without parallel paths, and suppose there is a head move changing $S(\ddot{N})$ into N', then there is a head move changing $\ddot{N}$ into $\ddot{N}'$ such that $S(\ddot{N}') = N'$.*

Proof. Suppose the move changing $S(\ddot{N})$ into $\dot{N}'$ is $v \xrightarrow{(u,v)} (x, y)$. Let w be the last node on the path between u and v, and z be the first node on the path between x and y in $\ddot{N}$. Then the move $v \xrightarrow{(w,v)} (x, z)$ is valid, and results in a network $\ddot{N}'$ with $S(\ddot{N}') = N'$. □

The above two lemmas take care of the tree and reticulation labels. Indeed, we could already move those anywhere in networks in $\dot{\mathcal{N}}(n, k)$, and with the above two lemmas, we can use those moves on subdivided networks as well. To move and permute the degree-2 nodes as well, we do the following. First, we make a movable arc, which can be moved essentially anywhere in the network. Then, we use this arc to collect the degree-2 nodes in order, like we did in the proof of Lemma 3.39 for degree-2 nodes using tail moves. To collect the degree-2 nodes without changing the underlying suppressed network, we must first make sure this underlying network has no parallel arcs.

Lemma 4.35. *Let $n > 1$ and $\ddot{N} \in \dot{\mathcal{N}}(n, k, m)$ be a subdivided network without parallel paths whose root path ends in the node x. Then there is a sequence of at most 3 head moves to a network $\ddot{N}'$ without parallel paths in which there is a subdivided triangle $\langle \cdot x, y, z \cdot \rangle$ with (x, z) not subdivided.*

Proof. Create a subdivided triangle at the top by moving the head z of a highest reticulation arc up to the root at most twice. As this is possible in $S(\ddot{N})$ already, this can be done in $\ddot{N}$ without introducing parallel paths. After these moves, the last arc in the path $(x, \ldots, z)$ is movable, and so is the other incoming arc (w, z) of z. By moving (w, z) up to the highest arc in the $(x, \ldots, z)$ path, we create the desired network. □

Lemma 4.36. *Let $\ddot{N} \in \dot{\mathcal{N}}(n,k,m)$ be a network in which the root path consists of one arc (ρ, x), and there is a subdivided triangle $\langle \cdot x, y, z \cdot \rangle$ with (x,z) not subdivided. Let $v_1, \dots, v_m$ be any ordering of the degree-2 nodes of $\ddot{N}$, then there is a sequence of at most $2m+2$ moves to a network $\ddot{N}'$ with a path $(x, v_1, \dots, v_m, z)$ and $S(\ddot{N}') = S(\ddot{N})$.*

Proof. First, if (x,y) is subdivided, we use a sequence of three moves to make sure both (x,y) and (x,z) are not subdivided. Let c be the child of z, then this can be done using the sequence $z \xrightarrow{(x,z)} (w,y)$, $z \xrightarrow{(w,z)} (y,c)$, and $z \xrightarrow{(y,z)} (x,v)$, where (x,v) is on the path from x to z.

Now, we add each degree-2 node to (x,z) in order, exactly as for tail moves: Assume the network already contains the path $(x, v_1, \dots, v_i, z)$, then we first do $z \xrightarrow{(v_i,z)} (p, v_{i+1})$ and then $z \xrightarrow{(p,z)} (v_{i+1}, \cdot)$. After these two moves, the network contains the path $(x, v_1, \dots, v_{i+1}, z)$. Hence, by doing these two moves for each $i \in [m]$ and then moving (v_m, z) back to its original position by $z \xrightarrow{(v_m,z)} (y, \cdot)$, we obtain a network $\ddot{N}'$ with the desired properties. □

Theorem 4.37. *Let $n > 1$ and $\ddot{N}_1, \ddot{N}_2 \in \dot{\mathcal{N}}(n,k,m)$ be two networks with empty root paths ending in nodes with the same label, then $d_{\text{head}}(\ddot{N}_1, \ddot{N}_2) \leq \text{diam}_{\text{head}}(n,k,0) + 6m + 10$.*

Proof. First, remove all parallel paths from both networks using at most m head moves per network (Lemma 4.33). Let us denote the resulting networks as $\ddot{N}_1^p$ and $\ddot{N}_2^p$. Now, using at most $\text{diam}_{\text{head}}(n,k,0)$ moves, we can change $\ddot{N}_1^p$ into a network $\ddot{N}_1^{p\prime}$ such that $S(\ddot{N}_1^{p\prime}) \overset{\phi}{\simeq}_{X \setminus X^{(2)}} S(\ddot{N}_2^p)$ (Lemma 4.34 and Theorem 4.31).

Then, create a subdivided triangle at the top of each network, where the long path consists of one arc. This can be achieved using at most 3 head moves in each network and without re-introducing parallel paths (Lemma 4.35). By using the 'same' paths (i.e., mapped to each other by ϕ), we can additionally make sure that the resulting networks $\ddot{N}_1^t$ and $\ddot{N}_2^t$ satisfy $S(\ddot{N}_1^t) \simeq_{X \setminus X^{(2)}} S(\ddot{N}_2^t)$.

In the resulting networks, all degree-2 nodes can be collected in any order on the long path of the top triangle using at most $2m+2$ head moves in each network. By Lemma 4.36, this can be done without changing the underlying suppressed networks, so the resulting networks are labeled isomorphic. The total number of head moves used is $\text{diam}_{\text{head}}(n,k,0) + 6m + 10$. □

4.5 Conclusion

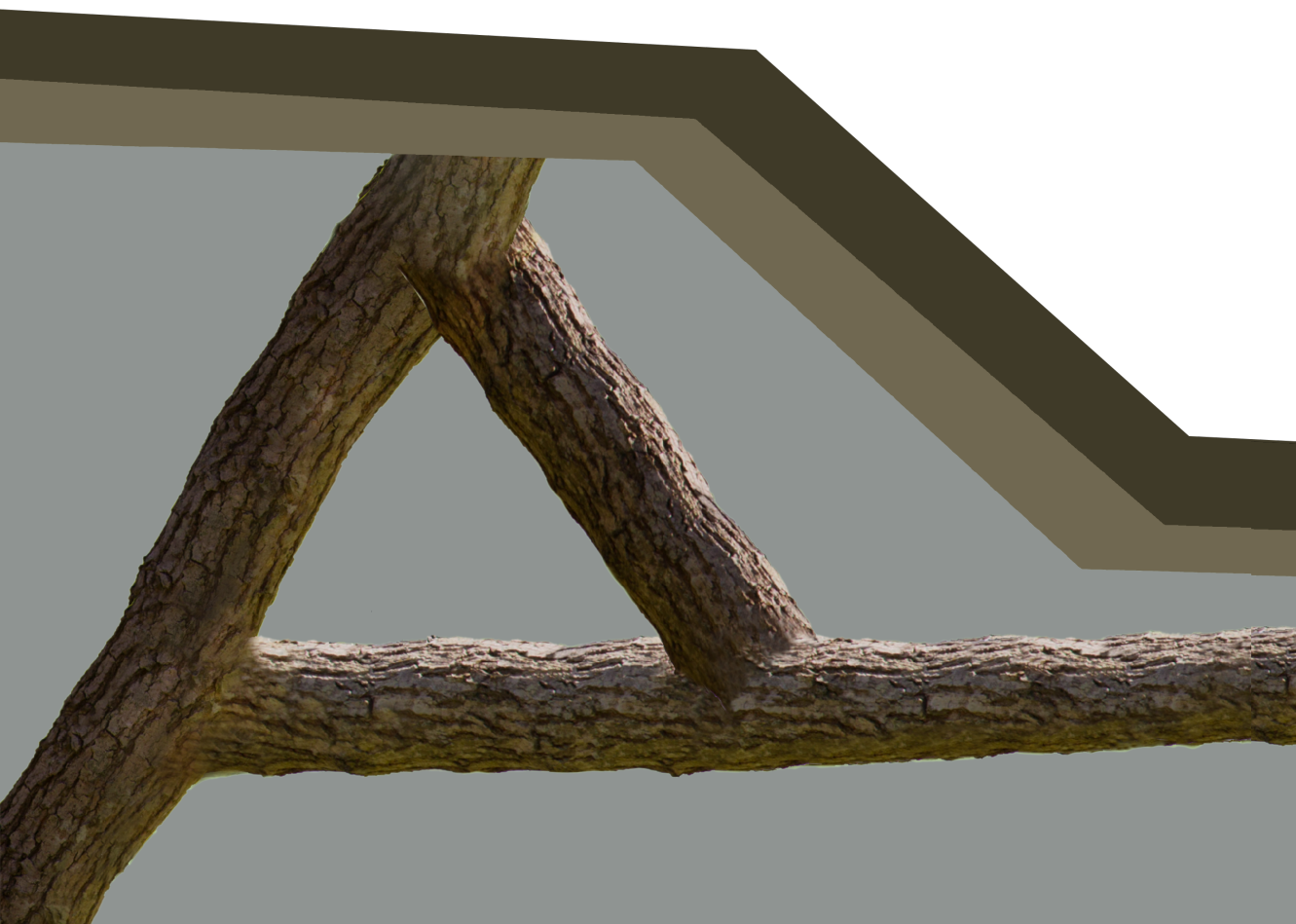
In this chapter, we have shown that all spaces $\mathcal{N}_{\text{head}}(n,k)$ and $\mathcal{N}_{\text{head}_2}(n,k)$ are connected. The constructive proofs that these spaces are connected also give upper bounds for the diameters of these spaces of $6n+6k-4$ for head moves and $12n^2+10kn+10k^2+52n+2k+13$ for head_2 moves. This bound for head moves is asymptotically tight, as we will see in Section 5.2.3. For all other (local) head move spaces, we have no lower bounds, so we do not know whether our upper bounds for these spaces are asymptotically tight.

Note that the local version of the head move we have considered is the distance-2 head move, and not the distance-1 head move. This is because no space of distance-1 head moves is connected. It may still be interesting to investigate the connected components of $\dot{\mathcal{N}}_{\text{head}_1}(n,k)$, in order to learn more about the often used rNNI move as well.

Like for tail moves, these results for head moves also extend to internally labeled networks, with a few exceptions. All spaces $\dot{\mathcal{N}}_{\text{head}}(n,k,m)$ and $\dot{\mathcal{N}}_{\text{head}_2}(n,k,m)$ are disconnected. However, the connected components of the spaces $\dot{\mathcal{N}}_{\text{head}}(n,k,m)$ and $\dot{\mathcal{N}}_{\text{head}_2}(n,k,0)$ can easily be characterized by the sequence of nodes in the root path: two networks are in the same component if their root paths are labeled isomorphic. In particular, when there are no degree-2 nodes there is a connected component for each label in X^t. We have also computed the diameters of the components of all these spaces.

Note that we have not characterized the connected components of the spaces $\dot{\mathcal{N}}_{\text{head}_2}(n,k,m)$ for $m>0$. This is because these spaces suffer from the same problem as $\mathcal{N}_{\text{head}_1}(n,k)$. It is an open question whether distance-$(m+2)$ moves provide sufficient connectedness (with one component for each root path structure).

5



rSPR and rNNI Moves

The previous two chapters concerned tail moves and head moves as separate types of moves. Originally, this distinction was only made as a convenient way to introduce rSPR moves in [GvIJ+17b]. In this chapter, we return to the original rearrangement move for directed networks, the rSPR move and its local variant, the rNNI move.

As rSPR moves consist of tail moves as well as head moves, the question of connectedness of rSPR and rNNI spaces has already been answered in the previous two chapters—in fact, it has been answered several times, using independent proofs.[1] For internally labeled networks, there were some exceptions for connectedness for tail and head moves separately. We will show that these exceptions disappear when we use the union of these moves.

Although the question of connectedness can be answered quickly using the results from the previous two chapters—in fact, we have $\mathcal{N}_{\text{rSPR}}(n,k) = \mathcal{N}_{\text{tail}}(n,k) \cup \mathcal{N}_{\text{head}}(n,k)$—we can still reconsider the diameter and distance bounds. It is far from clear that using only head moves or tail moves is close to being as efficient as using both at the same time. Hence, in this chapter, we will study the interplay between tail and head moves.

First, we will show that using both moves is not much more efficient than using either move separately. We show this by proving that the tail move distance and the head move distance between two networks (with $n > 1$) can differ by at most a multiplicative factor of 16. In these proofs, we give explicit methods to replace any tail move by a sequence of at most 13 head moves (Section 5.1.1), and any head move by a sequence of at most 16 tail moves (Section 5.1.2). Note that this section only concerns networks without internal labels.

Then, in Section 5.2, we will return to diameter bounds for rSPR and rNNI moves. Although the tail and head move diameters directly give upper bounds for the rSPR diameter, it is easy to improve on these bounds by adapting the proof for the tail move diameter to include head moves as well. This modified proof handles lowest reticulation nodes using one head move instead of a sequence of several tail moves (Section 5.2.1).

For upper bounds for the diameters of rNNI spaces, we employ a technique reminiscent of the technique used for distance-2 head moves (Section 5.2.2). In this technique, which was first used for undirected networks in [JK19], each network is first transformed into a highly tree-like network, whose underlying tree is then changed using a small number of moves. In the head move proof,

[1] It must be noted that [GvIJ+17b] were the first to provide a proof for this connectedness result. However, as we will see in Section 6.1, their proof was based on a false lemma, so the first correct proof was given in [JJE+18].

changing the tree required a quadratic number of moves, whereas, in the rNNI proof, changing the tree requires a linearithmic number (i.e., $O((n+k)\log(n+k))$) of moves. In the final part of this section (Section 5.2.3), we also prove several lower bounds for the diameters of network spaces.

Like in the last two chapters, the final part (Section 5.3) of this chapter concerns spaces of internally labeled networks. We show that these spaces are all connected for rNNI moves. This is mainly due to the fact that the root path can be rid of degree-2 nodes using tail moves, and, similarly, all degree-2 nodes on a leaf-paths starting at a reticulation can be moved up using head moves.

5.1 Rewriting head and tail moves

In this section, we show that each tail move can be replaced by a sequence of at most 13 head moves, and each head move can be replaced by a sequence of at most 16 tail moves.

5.1.1 Tail move replaced by head moves

Here, we show how to replace a tail move by a sequence of head moves (Theorem 5.7). The proof works by case distinction, where the main cases represent different types of tail moves. The first two lemmas prove that we can replace certain types of distance-1 tail moves: in Lemma 5.1, we replace a distance-1 tail move between the two outgoing arcs of a tree node, and in Lemma 5.2, we replace a distance-1 tail move between the two incoming arcs of a reticulation. Then, we turn to the remaining cases, where the tail move $(x_L, u, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ is such that $a_L \neq a_R$ and (w.l.o.g.) a_L is not above a_R (Lemma 5.5). This case is split up into two lemmas, depending on where the head-movable arcs are located in the network in relation to a_R (Lemmas 5.3 and 5.4).

In this section, unless stated otherwise, each move is a head move and movable means head-movable.

Lemma 5.1 ([Jan21] Lemma 11)**.** *Let $n, k > 0$ and $N, N' \in \mathcal{N}(n,k)$ such that the tail move $(x, u, a_L) \xrightarrow{(u,v)} (x, a_R)$ turns N into N', then $d_{\text{head}}(N, N') \leq 4$.*

Proof. To prove this, we have to find a reticulation somewhere in the network that we can use, as all of u, v, x, a_L, and a_R may be tree nodes.

Note that there exists a head-movable reticulation arc (t, r) in N with t not below both a_L and a_R. Indeed, if there is a highest reticulation node below a_L and a_R, then one of its incoming arcs is movable (Lemma 4.2) and this arc cannot be below both a_L and a_R. Otherwise, if there is no such reticulation,

then there is a reticulation r that is below at most one of a_L and a_R, and the same holds for its incoming arcs, one of which is movable (Lemma 4.2).

First, assume that $t = x$. Because u is a tree node and r is a reticulation, we have $r \neq u$. This means that $r = a_R$, and (x, a_R) is movable. We apply the move $(s, a_R, z) \xrightarrow{(x,a_R)} (u, a_L)$, which is allowed because $x \neq u$, a_L not above x and $(t, r) = (x, a_R)$ is movable. Now, we can obtain N' by applying $a_R \xrightarrow{(u,a_R)} (s, z)$.

Now, assume that $x \neq t$. Suppose w.l.o.g. that (t, r) is not below a_L, then we can use the following sequence of 4 moves except in the cases we mention in bold below the steps. For this lemma, we call this sequence the 'normal' sequence (Figure 5.1). The validity of each move is checked using Lemma 2.45.

- $(s, r, z) \xrightarrow{(t,r)} (u, a_L)$, keeping (x, a_R) except if $\boldsymbol{x = t}$. This can be done if (t, r) is movable, which it is by choice of (t, r); $t \neq u$, but we note that $\boldsymbol{t = u}$ may occur; and, a_L is not above t, which is true by choice of (t, r).

- $r \xrightarrow{(u,r)} (x, a_R)$, creating the arc (t, a_L), because $(x, a_R) \neq (t, r)$ and $(x, a_R) \neq (r, a_L)$. For this move, note that (u, r) is movable, except when $\boldsymbol{(t, a_L) \in N}$; $u \neq x$ as these nodes are distinct in the original network; and a_R is not above u, as it is not above x.

- $r \xrightarrow{(x,r)} (t, a_L)$. The arc (x, r) is movable because $v \neq a_R$ (otherwise the tail move would not be not valid); $t \neq x$, as we have assumed so for the first move; and a_L is not above x, as a_L is above x in N.

- $r \xrightarrow{(t,r)} (s, z)$, which moves r back to its original position. This move is allowed because it results in N'.

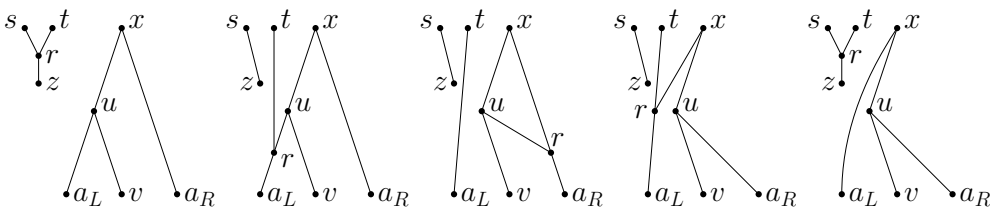


Figure 5.1: The "normal" sequence of head moves simulating a tail move in Lemma 5.1.

We now look at the situations where $t = x$, $t = u$, or $(t, a_L) \in N$ separately. We will split up in cases to keep the proof clear. Recall that (t, r) is a movable arc in N.

1. $\boldsymbol{t = u}$.

   a) $\boldsymbol{r = a_L}$. This case can simply be solved by the following sequence of 2 moves from N to N': $(s, r, z) \xrightarrow{(t,r)} (x, a_R)$ and $r \xrightarrow{(x,r)} (s, z)$.

   b) $\boldsymbol{r = v}$. To solve this case, first apply $(s, r, z) \xrightarrow{(t,r)} (x, a_R)$, creating a triangle at x. Now reverse the triangle using $r \xrightarrow{(x,r)} (t, a_L)$. Lastly, we obtain N' by moving r back to its original position using $r \xrightarrow{(t,r)} (s, z)$. This is a sequence of 3 head moves.

2. $\boldsymbol{t \neq u}$. Note that we can assume that (u, a_L) is not movable, as, otherwise, we are in the previous case. Because $t \neq u$, we may also assume that $(t, a_L) \in N$. Otherwise, this would not be a special case, and we can use the sequence of moves from the start of the proof. Hence, there is a triangle $\langle t, a_L, c(a_L)\rangle_r \in N$.

   a) **t is below a_R.**

      i. **t is below v.** Since t is below both a_R and v, there is a highest reticulation s strictly above t and below both a_R and v. As s is strictly above t, it is strictly above a_L. Therefore, we are either in the 'normal' case, or in Case 1b of this proof, where some incoming arc $(\cdot, s)$ of s is head-movable. This means this situation can be solved using at most 4 head moves.

      ii. **t is not below v.** As (t, a_L) is a reticulation arc in the triangle, it is movable, and, because t is not below v, the head move $r \xrightarrow{(t,a_L=r)} (u, v)$ is allowed. After this move, the *tail* move $u \xrightarrow{(u,r)} (x, a_R)$ is still allowed, because v is not above x. As $(u, c(a_L))$ is movable in this new network, we can simulate this tail move using the 2 head moves $(t, c(a_L), z) \xrightarrow{(u,c(a_L))} (x, a_R)$ and $c(a_L) \xrightarrow{(x,c(a_L))} (t, z)$ (Case 1a). Afterwards, we can put the triangle back in its place with one head move $r \xrightarrow{(t,r)} (x, c(a_L))$, which is allowed because it produces N'. The resulting sequence has length at most 4.

   b) **t is not below a_R.** Because $t \neq x$, we know that $(t, a_R) \notin N$. Therefore we can do the 'normal' sequence of moves from the start of this proof in reverse order, effectively switching the roles of a_L and a_R. Because we use the 'normal' sequence of moves, this case takes at most 4 head moves. □

To prove the case of a more general tail move, we need to treat another simple case first.

Lemma 5.2 ([Jan21] Lemma 12). *Let $n > 1$, $k > 0$, and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, u, r) \xrightarrow{(u,v)} (x_R, r)$ turns N into N', then $d_{\text{head}}(N, N') \leq 4$.*

Proof. Let z be the child of r, and note that not all nodes described must necessarily be unique. All possible identifications are $x_L = x_R$ and $v = z$, other identifications create cycles in either N or N'. First, note that if $x_L = x_R$, then $N \simeq_X N'$. Hence we can restrict our attention to the case that $x_L \neq x_R$. To prove the result, we distinguish two cases.

1. $\boldsymbol{z \neq v}$. This case can be solved with the following two head moves: $r \xrightarrow{(x_R,r)} (x_L, u)$ and $r \xrightarrow{(x_L,r)} (u, z)$. The first head move is allowed because $v \neq z$, so (x_R, r) is head-movable; $x_R \neq x_L$; and u is not above x_R because both its children aren't: z is below a_R, and if v is above x_R, the tail move $N \to N'$ is not allowed. The second head move is allowed because it produces the valid network N'. Hence the tail move can be simulated by at most 2 head moves (Figure 5.2).

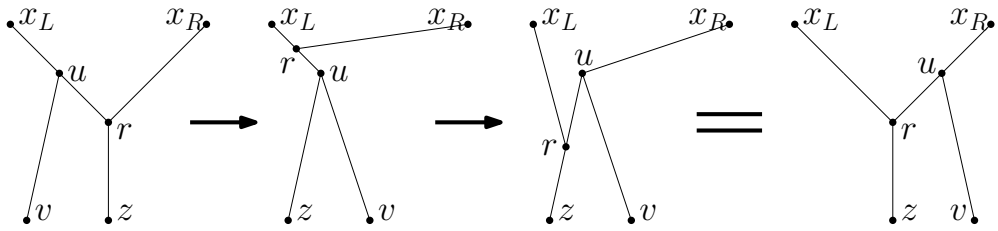


Figure 5.2: The two moves used to simulate a tail move in Case 1 of Lemma 5.2.

2. $\boldsymbol{z = v}$. The proposed moves of the previous case are not valid here, because they lead to parallel arcs in the intermediate network. To prevent this, we create the situation of the previous case by applying $v \xrightarrow{(u,v)} e$, where $e \neq (v, c)$ is an arc not above v (hence, neither above x_L nor above x_R). After this move, the tail move $u \xrightarrow{(u,v)} (x_R, r)$ is still allowed and can therefore be simulated by 2 head moves as in the previous case. Finally, moving (u, v) back with $v \xrightarrow{(u,v)} (r, c)$ results in N'. This sequence of moves uses 4 head moves (Figure 5.3).

   It remains to prove that an arc e with the required properties exist (i.e., not above v and excluding (v, c)). Because $n > 1$, there is a leaf $l \neq c$ whose incoming arc (p, l) is not above c and not equal to (v, c). Hence this arc $e = (p, l)$ suffices as a location for the first head move.

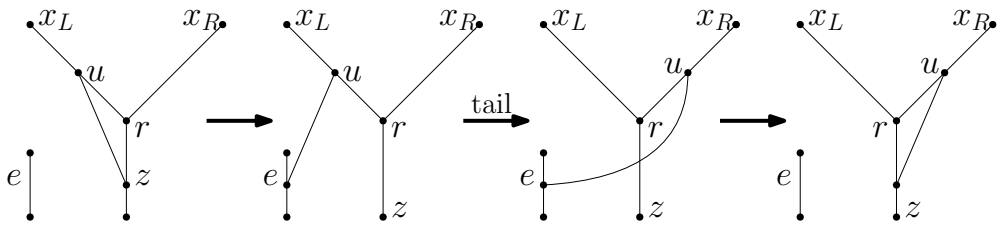


Figure 5.3: The four moves used in Case 2 of Lemma 5.2. The middle depicted move is the tail move of Case 1, which can be replaced by two head moves.

We conclude that any tail move of the form $(x_L, u, r) \xrightarrow{(u,v)} (x_R, r)$ can be simulated by at most 4 head moves. □

Lemma 5.3 ([Jan21] Lemma 13). *Let $n > 1$, $k > 0$, and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, u, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose $a_L \neq a_R$, a_L is not above a_R, and there exists a movable reticulation arc (t,r) not below a_R, then $d_{\text{head}}(N, N') \leq 7$.*

Proof. Note that v cannot be above either of x_L and x_R. The only possible identifications within the nodes a_L, a_R, x_L, x_R, u, and v are $a_L = a_R$, $x_L = x_R$ and $a_R = x_L$ (but not simultaneously), all other identifications lead to parallel arcs, cycles in either N or N', a contradiction with the condition "a_L is not above a_R", or a trivial situation where the tail move leads to an isomorphic network. The first of these two identifications have been treated in the previous two lemmas, so we may assume $\boldsymbol{a_L \neq a_R}$ and $\boldsymbol{x_L \neq x_R}$. We now distinguish several cases to prove the tail move can be simulated by a constant number of head moves in all cases.

1. $\boldsymbol{(t, a_R) \notin N}$.

   a) $\boldsymbol{r = x_R}$. As (t,r) is movable and not below a_L or v, we may apply the head move $(s, r, z) \xrightarrow{(t,r)} (x_L, u)$. As (x_L, r) is movable in the resulting network, we may move its head down with $r \xrightarrow{(x_L,r)} (u, a_L)$. Since $u \neq s$ (otherwise the original tail move was not allowed), the head move $r \xrightarrow{(u,r)} (s, a_R)$ is now valid. Lastly, $r \xrightarrow{(s,r)} (t, u)$ results in N'.

   b) $\boldsymbol{r \neq x_R}$. If $t = x_R$ then $(t, a_R) \in N$, which contradicts the assumptions of this case. Hence, $t \neq x_R$, and the move $(s, r, z) \xrightarrow{(t,r)} (x_R, a_R)$ is valid in N. Because neither a_L nor v can be above x_R and $x_L \neq x_R$, we can now apply the move $r \xrightarrow{(x_R,r)} (x_L, u)$. Now, if

$r \neq a_L$ and $u \neq t$ or $r \neq v$, we apply the moves $r \xrightarrow{(x_L,r)} (u, a_L)$ which is allowed because it moves a movable head down, and $r \xrightarrow{(u,r)} (t, a_R)$. If $u = t$ and $r = v$ or $r = a_L$, these moves are not valid, and we simply skip these moves. Lastly, we move $r \xrightarrow{(t,r)} (s, z)$ to arrive at N'. Hence the tail move of this situation can be simulated by at most 5 head moves.

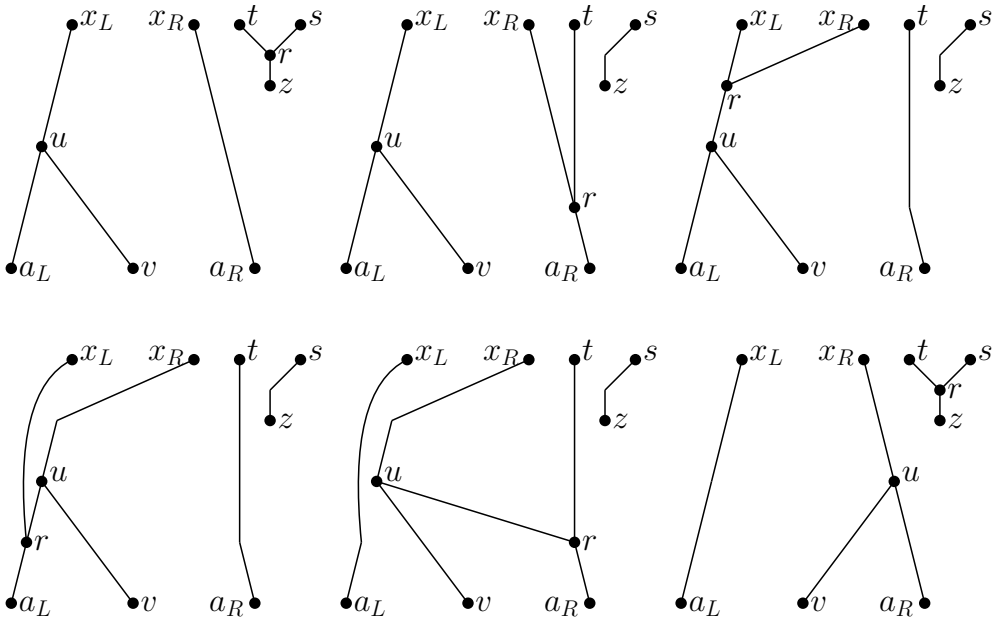


Figure 5.4: The five moves used to simulate a tail move in Case 1b of Lemma 5.3.

2. $\boldsymbol{(t, a_R) \in N}$. Again (t, r) is the head-movable arc. Let z be the child of r and $s \neq t$ the other parent of r.

   a) $\boldsymbol{z = a_R}$. First note that, in this case, we must have either $x_R = t$ or $x_R = r$. Otherwise, one of the arcs (t, a_R) and (r, z) is not in N.

      i. $\boldsymbol{x_R = t}$ Because r and $t = x_R$ are distinct, $a_R = z$ is a reticulation node with movable arc $(t = x_R, z = a_R)$. Hence, this case can be solved easily with the following 3 head moves: $(r, z, c) \xrightarrow{(t,z)} (x_L, u)$, $z \xrightarrow{(x_L,z)} (u, a_L)$, and $z \xrightarrow{(u,z)} (r, c)$.

      ii. $\boldsymbol{x_R = r}$ Note that the *tail* move $u \xrightarrow{(u,v)} (t, a_R)$ is also allowed because (u, v) is tail-movable, $v \neq a_R$ and v not above t (otherwise the tail move to (r, z) is not allowed either). This tail

move is of the type of the previous case, and can be replaced by at most 3 head moves. In the resulting network, the tail move $u \xrightarrow{(u,v)} (r,z)$ is of the type of Lemma 5.2, which takes at most 4 head moves to simulate. We conclude any tail move of this type can be simulated using at most 7 head moves.

b) $\boldsymbol{z \neq a_R}$.

i. $\boldsymbol{a_R \neq r}$.

A. $\boldsymbol{x_R = t}$ and $\boldsymbol{v \neq r}$. The sequence for this case is much like the sequence used in Case 2(a)i. The *tail* move $u \xrightarrow{(u,v)} (x_R, a_R)$ can be replaced with the 4 head moves $(s,r,z) \xrightarrow{(t,r)} (x_L, u)$, $r \xrightarrow{(x_L,r)} (u, a_L)$, $r \xrightarrow{(u,r)} (t, a_R)$, and $r \xrightarrow{(t,r)} (s,z)$.

B. $\boldsymbol{x_R = t}$ and $\boldsymbol{v = r}$. As a_L is not above x_R, the following sequence of 4 head moves suffices: $v \xrightarrow{(x_R,v)} (x_L, u)$, $v \xrightarrow{(x_L,v)} (u, a_L)$, $v \xrightarrow{(u,v)} (x_R, a_R)$, and $v \xrightarrow{(x_R,v)} (u,z)$.

C. $\boldsymbol{x_R \neq t}$ and $\boldsymbol{a_R \neq s}$. Because $a_R \neq s$, the arc (x_R, a_R) is movable. Also, because $x_R \neq x_L$ and u is not above x_R, the move $(t, a_R, c) \xrightarrow{(x_R,a_R)} (x_L, u)$ is valid. Now (x_L, a_R) is movable, so the downward head move $a_R \xrightarrow{(x_L,a_R)} (u, a_L)$ is allowed. Finally, the head move $a_R \xrightarrow{(u,a_R)} (t,c)$ results in N'. Hence, in this case we need at most 3 head moves.

D. $\boldsymbol{x_R \neq t}$ and $\boldsymbol{a_R = s}$. As a_R is a child of t and u has children v and a_L, both of which are distinct from a_R, we know $t \neq u$. Hence, the following sequence of 5 head moves suffices: $r \xrightarrow{(t,r)} (u, a_L)$, $s \xrightarrow{(x_R,s)} (x_L, u)$, $s \xrightarrow{(x_L,s)} (u, r)$, $s \xrightarrow{(u,s)} (t,z)$, and finally $r \xrightarrow{(t,r)} (s,z)$.

ii. $\boldsymbol{a_R = r}$. In this case either $x_R = t$ or $x_R = s$.

A. $\boldsymbol{x_R = t}$. This case is easily solved with 3 head moves: $a_R \xrightarrow{(x_R,a_R)} (x_L, u)$, $a_R \xrightarrow{(x_L,a_R)} (u, a_L)$, and $a_R \xrightarrow{(u,a_R)} (s,z)$.

B. $\boldsymbol{x_R = s}$. If (s,r) is movable (i.e. there is no arc (t,z)), then we can relabel $t \leftrightarrow s$ and use the sequence of the previous case. Otherwise, there is an arc (t,z) and we use the following sequence of moves: $a_R \xrightarrow{(t,a_R)} (u, a_L)$, $(t,z,c) \xrightarrow{(x_R,z)} (x_L, u)$, $z \xrightarrow{(x_L,z)} (u, a_R)$, $z \xrightarrow{(u,z)} (t,c)$, and $a_R \xrightarrow{(t,a_R)} (z,c)$. The tail move of this situation can therefore be replaced by at most 5 head moves. □

Lemma 5.4 ([Jan21] Lemma 14). *Let $n > 1$, $k > 0$, and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, u, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose $a_L \neq a_R$, a_L is not above a_R, and all movable reticulation arcs are below a_R, then $d_{\text{head}}(N, N') \leq 13$.*

Proof. Because the networks have at least one reticulation, we can pick a highest reticulation r in N. As each reticulation has at least one movable incoming arc, there is a movable arc (t, r) in N. As each movable reticulation arc is below a_R, so is (t, r). Let us denote the root of N with ρ, the child of r with z, and the other parent of r with $s \neq t$. We now distinguish two cases:

1. $\boldsymbol{x_R \neq \rho}$. Because x_R is above a_R, it must be a tree node, so it has another outgoing arc (x_R, b) with $b \neq a_R$ not above t: if b were above t, there would have to be a reticulation above r, contradicting our choice of r.

   a) $\boldsymbol{r \neq b}$. In this case, the move $r \xrightarrow{(t,r)} (x_R, b)$ is valid in both N and N'—using the natural map between nodes of N and N' induced by the tail move—resulting in networks M and M' such that (x_R, r) is movable in M. Now, by relabelling t as x_R, we can see that there is one tail move between M and M' of the same type as Case 2(b)i of Lemma 5.3, which can be simulated using at most 5 head moves. To see this, take r as the relevant reticulation with movable arc (t, r) and consider the tail move $u \xrightarrow{(u,v)} (x_R, a_R)$ producing M'. This case can therefore be solved with at most $5 + 2 = 7$ head moves.

   b) $\boldsymbol{r = b}$ **and** $\boldsymbol{(t, z) \notin N}$. In this case, (x_R, r) is movable, and not below a_R, contradicting our assumptions.

   c) $\boldsymbol{r = b}$ **and** $\boldsymbol{(t, z)) \in N}$. Because N has at least two leaves, there must either be at least 2 leaves below r, or there is a leaf not below r. Let l be an arbitrary leaf below r in the first case, or a leaf not below r in the second case. Note that the head move $(r, z, c) \xrightarrow{(t,z)} (\cdot, l)$ is valid, and it makes (x_R, r) movable. Now the tail move $u \xrightarrow{(u,v)} (x_R, a_R)$ is still allowed because $v \neq a_R$, v is not above x_R, and (u, v) is tail-movable. For this tail move we are in a case of Lemma 5.3 because (x_R, r) is not below a_R. Hence, this tail move takes at most 7 moves. After this move, we can do one head move $z \xrightarrow{(t,z)} (r, c)$ to put (t, z) back, so this case takes at most 9 moves.

2. $\boldsymbol{x_R = \rho}$. Let y, z be the children of a_R. Now first do the tail move $u \xrightarrow{(u,v)} (a_R, z)$. This is allowed because a_R is the highest tree node. The

sequence of head moves used to do this tail move is as in the previous case. Note that N' is now one tail move away: $u \xrightarrow{(u,z)} (a_R, y)$. This is a horizontal tail move along a tree node as in Lemma 5.1, which takes at most 4 head moves. As the previous case took at most 9 head moves, this case takes at most 13 head moves in total. □

Lemma 5.5 ([Jan21] Lemma 15)**.** *Let $n > 1$, $k > 0$, and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, u, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose $a_L \neq a_R$ and a_L is not above a_R, then $d_{\text{head}}(N, N') \leq 13$.*

Proof. This is a direct consequence of the previous two lemmas. □

Lemma 5.6 ([Jan21] Lemma 16)**.** *Let $n > 1$, $k > 0$, and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, u, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose $a_L \neq a_R$ and a_L is above a_R, then $d_{\text{head}}(N, N') \leq 13$.*

Proof. Note that in this case a_R is not above a_L in N'. Swapping the labels $x_L \leftrightarrow x_R$ and $a_L \leftrightarrow a_R$ we are in the situation of Lemma 5.5 for the reverse tail move N' to N. This implies the tail move can be replaced by a sequence of at most 13 head moves. □

Theorem 5.7 ([Jan21] Theorem 2)**.** *Let $n > 1$, $k > 0$, and $N, N' \in \mathcal{N}(n,k)$. If $d_{\text{tail}}(N, N') \leq 1$, then $d_{\text{head}}(N, N') \leq 13$.*

Proof. This follows from the previous lemmas. □

5.1.2 Head move replaced by tail moves

In this section, we show how to replace a head move $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ by a sequence of tail moves (Theorem 5.16). In the proof, we first show how to efficiently replace downward head moves by tail moves (i.e., when a_L is above x_R). This is then used repeatedly to simulate arbitrary head moves.

Unless stated otherwise, each move in this section is a tail move and movable means tail-movable.

Distance-1 head moves

We first recall a result from [JJE+18]: any distance-1 head move can be replaced by a constant number of tail moves, so the following result holds.

Lemma 5.8 ([JJE+18] Lemma 3.3)**.** *Let $n > 1$ and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ is a distance-1 head move that turns N into N'. Then $d_{\text{tail}}(N, N') \leq 4$, except if $N \not\simeq N'$ and $(n,k) = (2,1)$.*

Furthermore, this lemma has the following special cases, for which we repeat the proofs here, as we will use them repeatedly in the proofs of this section.

Lemma 5.9 ([JJE+18] Lemma 3.3 Case c)**.** *Let $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose that $a_L = x_R$ and x_R is a* tree node*, then $d_{\text{tail}}(N, N') \leq 1$.*

Proof. Let $c \neq a_R$ be the other child of x_R, then the tail move $x_R \xrightarrow{(x_R,c)} (x_L, v)$ suffices. □

Lemma 5.10 ([JJE+18] Lemma 3.3 Case a)**.** *Let $(n,k) \neq (2,1)$ and $N, N' \in \mathcal{N}(n,k)$ such that $(x, v, a_L) \xrightarrow{(u,v)} (x, a_R)$ turns N into N', then $d_{\text{tail}}(N, N') \leq 4$.*

Proof. Writing p for the parent of x, we distinguish two cases: $u \neq p$ and $u = p$. Except for this identification, all nodes in $\{p, u, v, x, a_L, a_R\}$ must be distinct—the only other identification is $a_L = a_R$, but in that case the head move results in an isomorphic network.

1. $u \neq p$**.** In this case, we can use the sequence of two tail moves $x \xrightarrow{(x,a_R)} (v, a_L)$ and $x \xrightarrow{(x,a_L)} (p, v)$ (Figure 5.5). The intermediate network contains no parallel arcs as $a_L \neq a_R$ and $u \neq p$, and it has no cycles for the following reason. If the intermediate network were cyclic, then the (directed) cycle must involve the arc (x, a_R), and there is a path from a_R to x. This implies there is a path from a_R to v, which then implies there is a path from a_R to either u or to p. This path must also exist in the networks before and after the head move. This leads to a contradiction, because, then, one of these networks already contains a cycle, making the head move invalid. Hence, this case can be solved using at most 2 tail moves.

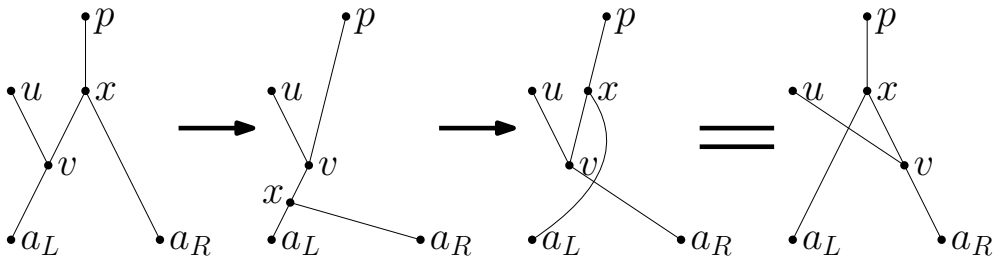


Figure 5.5: The sequence of tail moves needed to simulate the head_1 move in Lemma 5.10 Case 1.

2. $u = p$. Note that N contains the triangle $\langle p, x, v\rangle$. In this situation, we cannot directly use the same sequence as before, since this sequence would create parallel arcs. We can solve this problem in two ways.

   a) **There is a tail-movable arc (s,t) not above v.** Instead of moving arc (x, a_R) directly, we first subdivide it with s by moving $(q, s, c) \xrightarrow{(s,t)} (x, a_R)$. Then, we can do the sequence of moves depicted in Figure 5.6 and finally move s back with $s \xrightarrow{(s,t)} (q, c)$. Barring the addition of the extra tail, the sequence of moves is quite similar to the moves in case 1.

   b) **The root is not a parent of p.** The long arc of the triangle (p, v) is movable, and it can be moved to the root with $(r, p, x) \xrightarrow{(p,v)} (\rho, \cdot)$. Now, we are in the situation of case 1. Do the sequence of moves for that case, and move the (p, v) back with $p \xrightarrow{(p,v)} (r, x)$ to obtain N'.

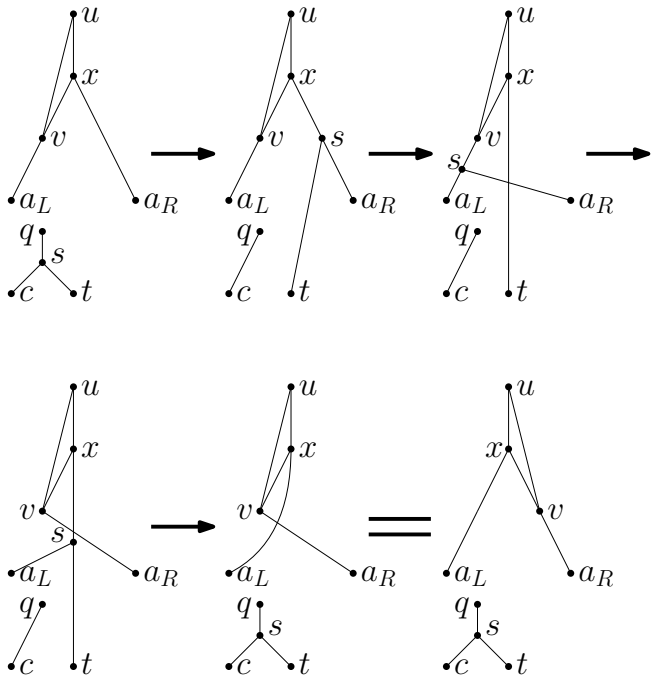


Figure 5.6: The sequence of tail moves used to simulate the head_1 move in Lemma 5.10 Case 2. The extra tail s of arc (s, t) is used in the sequence of moves: $(q, s, c) \xrightarrow{(s,t)} (x, a_R)$, $s \xrightarrow{(s,a_R)} (v, a_L)$, $s \xrightarrow{(s,a_L)} (x, t)$, and $s \xrightarrow{(s,t)} (q, c)$.

Option 2b is possible if there is at least one vertex above the triangle in addition to the root, and option 2a can be used if there is a tree node not above v.

There are two networks to which neither of these conditions apply. The first is the network on one leaf with two reticulations. In this network no head move leads to a different (non-isomorphic) network. The only non-trivial case is the excluded network where $(n, k) = (2, 1)$. The head move cannot be substituted by a sequence of tail moves, because there is no valid tail move. □

Downward head moves

In this section, we prove that the head move can be replaced by a sequence of constant length if a_L is above x_R. We start by considering the case that x_R is a tree node. In the proof we use a constant number of moves to create a situation where we simply need to do a distance-1 downward head move.

Lemma 5.11 ([Jan21] Lemma 19). *Let $N, N' \in \mathcal{N}(n, k)$ such that the move $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose that a_L is above x_R, $a_L \neq x_R$, and x_R is a tree node, then $d_{\text{head}}(N, N') \leq 4$.*

Proof. We split this proof in two cases: (x_R, a_R) is movable, or it is not. We prove in both cases there exists a constant length sequence of tail moves between N and N'.

1. **(x_R, a_R) is tail-movable.** As (x_R, a_R) is movable, we may move it up with $(s, x_R, z) \xrightarrow{(x_R,a_R)} (v, a_L)$ (Corollary 2.47). Now (u, v) is still head-movable, so we can move it down to (x_R, a_R). As this is exactly the situation of Lemma 5.9, we can replace this head move by the tail move $x_R \xrightarrow{(x_R,a_L)} (x_L, v)$. Now, tail-moving the subdivided (x_R, a_R) back down with $x_R \xrightarrow{(x_R,v)} (s, z)$ results in N', so this move is allowed, too. Hence, there is a sequence of 3 tail moves between N and N'.

2. **(x_R, a_R) is not tail-movable.** Because x_R is a tree node and (x_R, a_R) is not movable, there has to be a triangle $\langle p, x_R, c\rangle_t \in N$ with $c \neq a_R$. Note that (p, x_R) is tail-movable, and that it can be moved up by $(s, p, c) \xrightarrow{(p,x_R)} (v, a_L)$. After this move, the parent v of p is a reticulation, so (p, a_L) is movable, and it can be moved up with $p \xrightarrow{(p,a_L)} (x_L, v)$—this move is equivalent to the head move $v \xrightarrow{(u,v)} (p, x_R)$. The next step is to tail move the subdivided (p, x_R) back down with $p \xrightarrow{(p,v)} (s, c)$. The resulting network is allowed because it is one valid distance-1 head move away from

N' (as c is not above u). Lastly, we simulate this distance-1 head move, with the tail move $x_R \xrightarrow{(x_R,c)} (p,v)$ (Lemma 5.9). Note that this sequence is also valid if $p = a_L$. Hence, there is a sequence of at most 4 tail moves between N and N'. □

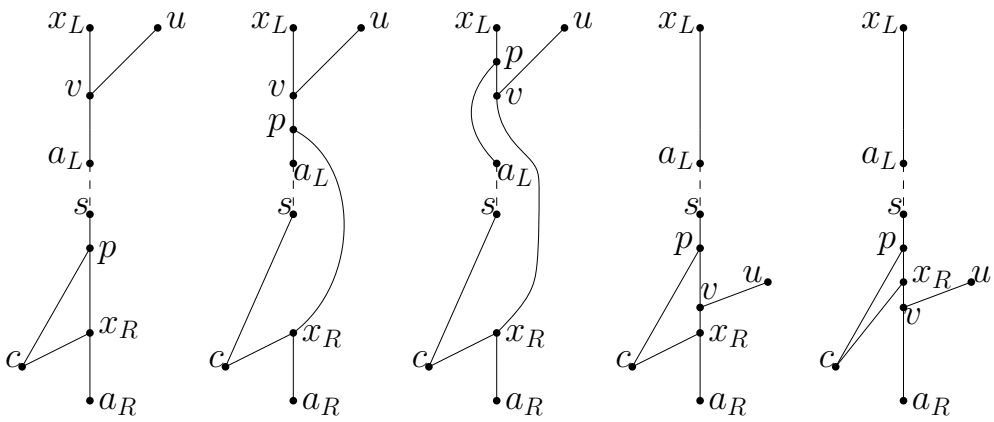


Figure 5.7: The four moves used in Case 2 of Lemma 5.11. The dased arc represents the ancestry relation a_L is above x_R, which must be via a path passing through s.

Lemma 5.12 ([Jan21] Lemma 20)**.** *Let $n > 1$, and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose that a_L is (strictly) above x_R and x_R is a reticulation, then there are networks M and M' such that the following hold:*

1. *$d_{\text{tail}}(N, M) \leq 1$;*
2. *$d_{\text{tail}}(N', M') \leq 1$;*
3. *there is a head move $(x'_L, v', a'_L) \xrightarrow{(u',v')} (x'_R, a'_R)$ turning M into M' such that a'_L is above x'_R and x'_R is a reticulation;*
4. *there is a tail-movable arc (s,t) in M with t not above x_R.*

Proof. Note that we have to find a sequence from N to N' consisting of a tail move followed by a head move and finally a tail move again, such that the head move is of the desired type and the network after the first tail move has a movable arc not above the top node x_R of the receiving arc of the head move.

If there is a tail-movable arc (s,t) in N with t not above x_R, we are done by the previous lemmas: simply take $M := N$ and $M' := N'$, and the head move of the statement of the lemma. Hence, we may assume that there is no such arc in N.

Recall that $n > 1$ and suppose that all leaves are below a_R, then there must also be a tree node below a_R. Furthermore, as one of its child arcs is movable, there is a tail-movable arc below a_R (and hence not above x_R). Hence, if all leaves are below a_R, we can again choose $M := N$ and $M' := N$.

Because our networks have at least 2 leaves, the remaining part is to show the lemma assuming that there is a leaf l_1 that is not below a_R. Note that there must also be a leaf l_2 below a_R. Now consider an LCA j of l_1 and l_2. As $j \notin \{l_1, l_2\}$, j is a tree node of which at least one outgoing arc **(j,m) is not above x_R**. If (j,m) is tail-movable, then $M := N$ and $M' := N'$ suffices, so assume **(j,m) is not tail-movable**. Let i be the parent of j, and k be the other child of j; because j is a tree node and (j,m) is not movable, there is a triangle $\langle i,j,k\rangle \in N$ (Figure 5.8).

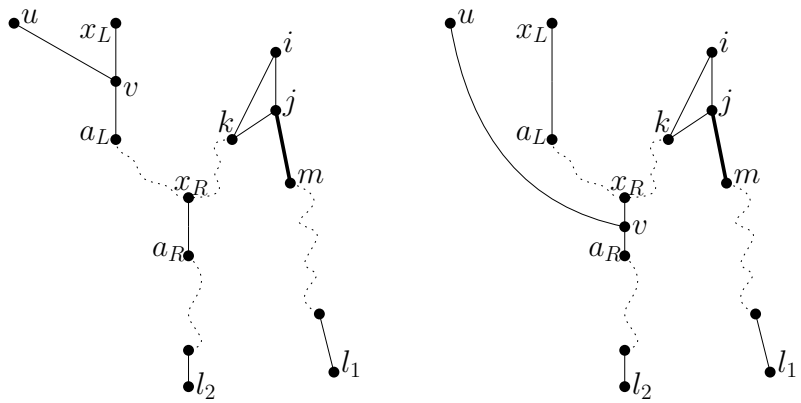


Figure 5.8: The situation of Lemma 5.12 in which we want to make the thick arc (j,m) movable in both the network before (left) and after (right) the head move $v \xrightarrow{(u,v)} (x_R, a_R)$. Dotted lines indicate ancestral relations, but are not necessarily arcs in the network.

The idea is to destroy this triangle with one tail move in N and N' simultaneously, essentially by doing 'the same move' in N and N'. If we can destroy the triangle in both networks keeping (u,v) movable, creating new networks M and M', then choosing $(s,t) := (j,m)$ in M will work. To show how to achieve this, we split into two cases:

- **i is the child of the root.** In this case, we destroy the triangle by

moving a tail to the triangle. As v is a reticulation and there is no path from any node below m to v (if so, there is a path from m to x_R), there must be a tree node p below k and (not necessarily strictly) above both parents of v. At least one of the outgoing arcs (p, q) is movable in N. If v is a child of p and (p, v) is movable, then we choose $q = v$, otherwise any choice of (p, q) will suffice.

Because (p, q) is movable and k is above p, the tail move $p \xrightarrow{(p,q)} (j, k)$ is valid. Now the head move $v \xrightarrow{(u,v)} (x_R, a_R)$ is valid, because x_R is below v, (u, v) is movable (as (u, v) was movable in N), and the only ways to create a triangle $\langle \cdot, v, \cdot \rangle_r$ with one tail move are:

- suppressing one node of a four-cycle that includes v to create a triangle by moving the outgoing arc of that node that is not included in the four-cycle. As this node is p, and p is above both parents of v, the suppressed node must be on the incoming arc of v in the four-cycle (Figure 5.9 top). However, in that case v is a child of p and (v, p) is tail-movable, so we choose to move (v, p) up for the first move, which keeps (u, v) head-movable.
- applying the move $x \xrightarrow{(x,v)} (y, c)$, where c is the child of v, $x \neq u$ and $y \neq v$. But as the tail move moves (p, q) to (j, k), we see that $k = c$ which contradicts the fact that v is strictly below k in N. Hence, this cannot result in a triangle with v on the side (Figure 5.9 bottom left).
- applying the move $y \xrightarrow{(y,c)} (x, v)$, where c is the child of v, $x \neq u$ and $y \neq v$. As we move $p \xrightarrow{(p,q)} (j, k)$, we see that $v = k$ and $u = i$. But, then, $c = q$ must be below the other child m of j, and, as x_R is below q, this contradicts the fact that (j, m) is not above x_R. Hence, this cannot result in a triangle with v on the side (Figure 5.9 bottom right).

The preceding shows that (u, v) is still head-movable after the first tail move. Because p is above x_R through two paths, a_L is still above x_R after the tail move $p \xrightarrow{(p,q)} (i, j)$. Now, the head move $v \xrightarrow{(u,v)} (x_R, a_R)$ is still valid and of the right type. Furthermore (j, m) is a tail-movable arc with m not above x_R. Finally, after the head head move $v \xrightarrow{(u,v)} (x_R, a_R)$, we can move (p, q) back to its original position to obtain N'.

Hence, we M is obtained from N by the tail move $p \xrightarrow{(p,q)} (i, j)$, and M' from N' by moving the corresponding arc $p' \xrightarrow{(p',q')} (i, j)$. We can do this because (i, j) is still an arc in N': indeed it is not subdivided by the head move.

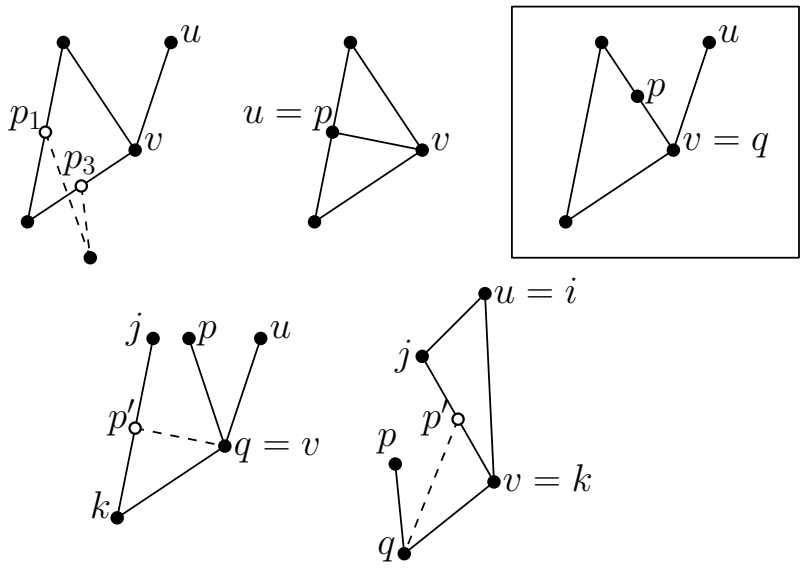


Figure 5.9: The ways of making (u, v) not head-movable in Lemma 5.12. Top: creating a triangle by suppressing a node in a four cycle. The first two of these are invalid because p is not above both parents of v. The right one does not give any contradictions, but forces us to choose to move (p, v), so that no triangle is produced. Bottom: creating a triangle by moving an arc to become part of the triangle. Both these options contradict our assumptions.

- ***i* is not the child of the root.** In this case, we can move $i \xrightarrow{(i,k)} (\rho, \cdot)$, where ρ is the root of N, and, possibly, $(i, j) = (u, v)$). Now note that j is a tree node, so the tail move cannot create any new triangles with a reticulation on the side. In particular, (u, v) is still movable after the tail move. Furthermore, after the tail move x_R is still a reticulation node below a_L, and (j, m) is movable and not above x_R. Hence, the head move $v \xrightarrow{(u,v)} (x_R, a_R)$ is allowed and of the appropriate type. Now moving the tail of (i, k) back with $i \xrightarrow{(i,k)} (\cdot, j)$ results in N'.

  Hence, this case works with M being the result of $i \xrightarrow{(i,k)} (\rho, \cdot)$ in N, and M' the result of $i \xrightarrow{(i,k)} (\rho', \cdot)$ in N'.

□

Lemma 5.13 ([Jan21] Lemma 21). *Let $n > 1$ and $N, N' \in \mathcal{N}(n, k)$ such that $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose that a_L is above x_R and x_R is a reticulation, then $d_{\text{tail}}(N, N') \leq 8$.*

Proof. By Lemma 5.12, with the cost of 2 tail moves, we can assume there is a tail-movable arc (s,t) that can be moved to (x_R, a_R). After applying $(p,s,c) \xrightarrow{(s,t)} (x_R, a_R)$, the head move $v \xrightarrow{(u,v)} (s, a_R)$ is allowed because it moves a head down and (u,v) is head-movable. By Lemma 5.11, there is a sequence of at most 4 tail moves simulating this head move. Now we need one more tail move to arrive at N': the move $s \xrightarrow{(s,t)} (p,c)$ which puts (s,t) back to its original position. This all takes at most 8 moves. □

Combining all the previous lemmas in this section gives us the following result about simulating downward head moves with tail moves.

Proposition 5.14 ([Jan21] Proposition 3)**.** *Let $n > 1$ and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N'. Suppose that a_L is above x_R or a_R is above x_L, then $d_{\text{tail}}(N, N') \leq 8$.*

Non-downward head moves

Finally, we consider head moves where the original position of the head and the location it moves to are incomparable.

Proposition 5.15 ([Jan21] Proposition 4)**.** *Let $n > 1$ and $N, N' \in \mathcal{N}(n,k)$ such that $(x_L, v, a_L) \xrightarrow{(u,v)} (x_R, a_R)$ turns N into N' and $(n,k) \neq (2,1)$ Suppose that a_L is not above x_R and a_R is not above x_L, then $d_{\text{tail}}(N, N') \leq 16$.*

Proof. Find an LCA s of x_L and x_R. We split into different cases for the rest of the proof:

1. $\boldsymbol{s \notin \{x_L, x_R\}}$**.** One of the outgoing arcs (s,t) of s is tail-movable and it is not above one of x_L and x_R (Lemma 3.4). Suppose t is not above x_L, then we can do the following (Figure 5.10):

   - $(p,s,c) \xrightarrow{(s,t)} (x_L, v)$;
     allowed because $t \neq v$, (s,t) movable, and t not above x_L.
   - $s \xrightarrow{(s,v)} (x_R, a_R)$;
     allowed because $(x_L, t) \notin N$: otherwise x_L was the only LCA of x_L and x_R; a_L and hence v is not above x_R; $(x_R, a_R) \neq (u,v)$.
   - $v \xrightarrow{(u,v)} (s, a_R)$, a distance-1 head move;
     No parallel arcs after pruning: if so, they are between s and $a_L = a_R$, but then the move actually resolves this; no parallel arcs by re-attaching: $u \neq s$; no cycles: a_R not above u, otherwise, there would be a cycle in N'.

- $s \xrightarrow{(s,a_L)} (x_L, t)$;
  Moving a tail up is allowed if the tail is movable.
- $s \xrightarrow{(s,t)} (p, c)$, which moves (s, t) back up to its original position.
  Moving a tail up is allowed if the tail is movable.

As the head move used in this sequence is a distance-1 sideways move through a tree node, it can be simulated with at most 4 tail moves as long as $(n, k) \neq (2, 1)$ (Lemma 5.10).

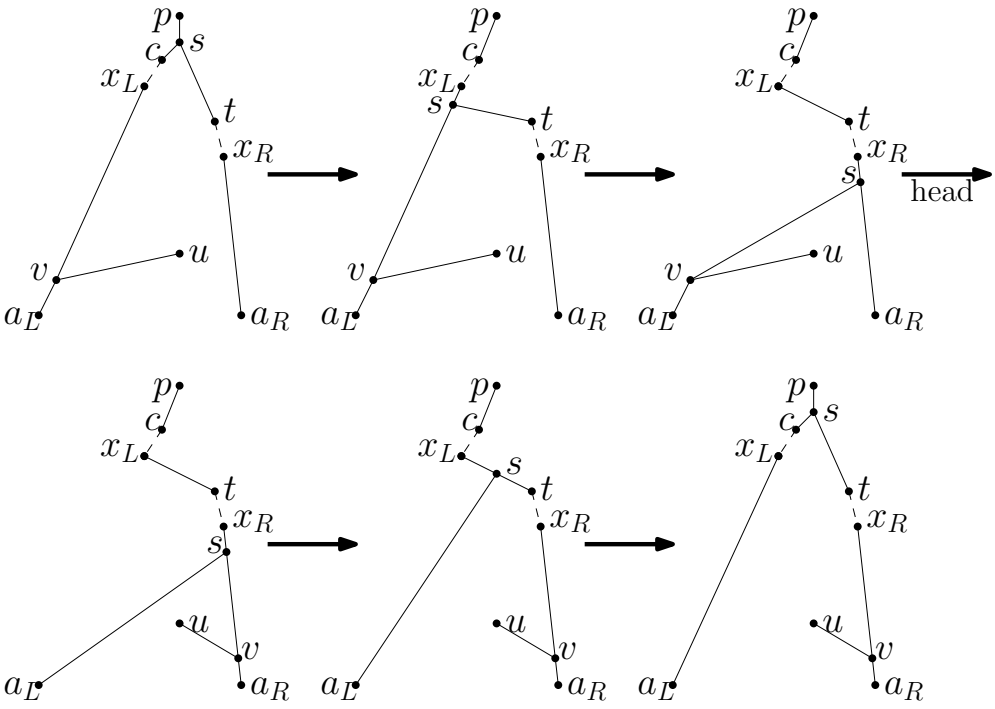


Figure 5.10: The sequence of moves used in Case 1 of Proposition 5.15.

2. $\boldsymbol{s = x_L}$.

   a) **$\boldsymbol{u}$ is not below $\boldsymbol{x_L}$.** As $s = x_L$ is above x_R and a_L is not above x_R, s must be a tree node. Hence, we may first apply the head move $v \xrightarrow{(u,v)} (x_L, t)$ where $t \neq a_L$, this takes at most 4 tail moves by Lemma 5.10. Now, we simply move v down with $v \xrightarrow{(u,v)} (x_L, a_L)$ to create N', this takes at most 8 tail moves by Proposition 5.14. Hence, for this case we need at most 12 tail moves.

b) $\boldsymbol{u}$ **is below** $\boldsymbol{x_L}$**.** In this case, the previous approach is not directly applicable, as moving the head of (u, v) to the other outgoing arc of x_L creates a cycle. Hence we need to take a different approach, where we distinguish the following cases:

   i. $\boldsymbol{(x_L, v)}$ **is tail-movable.** Apply the tail move $(s, x_L, z) \xrightarrow{(x_L,v)} (x_R, a_R)$, this is allowed because a_L is not above x_R. Then simulate the distance-1 head move $v \xrightarrow{(u,v)} (x_L, a_R)$ using at most 4 tail moves (Lemma 5.10). Then, move $x_L \xrightarrow{(x_L,a_L)} (s, z)$ back up to create N'. This takes at most 6 moves

   ii. $\boldsymbol{(u, v)}$ **is tail-movable.** Apply $(p, u, c) \xrightarrow{(u,v)} (t, s)$, which moves u up to an incoming arc of s. In the resulting network, the head move $v \xrightarrow{(u,v)} (x_R, a_R)$ is still allowed, except if there are triangles $\langle s, u, v\rangle$ and $\langle u, v, a_L\rangle$ in N. However, in that case x_L could not be the LCA of x_L and x_R. Hence, we can simulate the head move with at most 12 tail moves by Case 2a of this analysis. Because, afterwards, we can move the tail of (u, v) back to its original position with $u \xrightarrow{(u,v)} (p, c)$, this case takes at most 14 moves.

   iii. **Neither** $\boldsymbol{(x_L, v)}$ **nor** $\boldsymbol{(u, v)}$ **is tail-movable.** We create the situation of Case 1 by reversing the direction of the triangle at x_L, this takes at most 4 tail moves (Lemma 5.10). Notice that, if the bottom node of the (original) triangle is x_R, then we do not reach the situation of Case 1. However, in that case, the head move $v \xrightarrow{(u,v)} (x_R, a_R)$ can be decomposed as two distance-1 head moves $v \xrightarrow{(u,v)} (x_L, x_R)$ and $v \xrightarrow{(u,v)} (x_R, a_R)$. These moves can be simulated with at most 12 tail moves, as one is a sideways move (Lemma 5.10), and the other is a downward move (Lemma 5.14). Otherwise, we are in the situation of Case 1, and we can simulate the head move with at most 8 moves. This is allowed because it produces N' with the direction of a triangle reversed, which is a valid network. Then we can reverse the direction of the triangle again using at most 4 tail moves. This way, we obtain N' with at most 16 tail moves (Figure 5.11).

3. $\boldsymbol{s = x_R}$**.** This can be achieved with the reverse sequence for the previous case. □

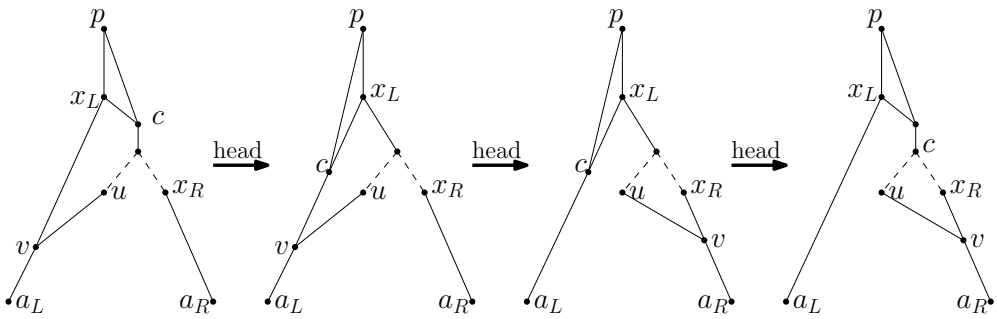


Figure 5.11: The sequence of head moves we simulate in Case 2(b)iii of Proposition 5.15. The first and the last move reverse the direction of a triangle, and the middle move is a move of the same type as dealt with in Case 1 of the same Proposition, unless $c = x_R$. If $c = x_R$, the head move $v \xrightarrow{(u,v)} (x_R, a_R)$ can be decomposed into two distance-1 head moves, which can be simulated using at most 12 tail moves.

Proposition 5.15 and Proposition 5.14 directly imply the following theorem

Theorem 5.16 ([Jan21] Theorem 3). *Let $n > 1$ and $N, N' \in \mathcal{N}(n,k)$ such that $d_{\text{head}}(N, N') \leq 1$ and $(n,k) \neq (2,1)$, then $d_{\text{tail}}(N, N') \leq 16$.*

5.2 Diameter bounds

In this section, we study the diameter of rSPR and rNNI spaces. First, we prove upper bounds for these spaces. The upper bound for rSPR spaces (Section 5.2.1) follows from an argument very similar to that for tail moves. The rNNI upper bound (Section 5.2.2) is based on the proof of the upper bound for NNI moves in [JK19], which heavily uses the NNI bounds for trees. Lastly, in Section 5.2.3 we also prove several lower bounds for the rSPR diameter.

5.2.1 rSPR upper bound: bottom-up isomorphism

Here, we modify the proof of Lemma 3.7 to get a better bound for the rSPR diameter. Recall that the proof of Lemma 3.7 works by gradually expanding two down-closed subsets $Y \subseteq V(N), Y' \subseteq V(N')$ for which $N[Y]$ is isomorphic to $N'[Y']$, using at most 3 tail moves each time the size of Y and Y' is increased. We show that in Cases 1 and 2 of the proof of Lemma 3.7, we may instead use one head move. As these cases both use Lemma 3.5, we improve this lemma for the case that we can use both tail and head moves.

Lemma 5.17 ([JJE+18] Theorem 4.8)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ such that $N_1 \not\simeq N_A$, and let $Y_1 \supseteq L(N_1)$ and $Y_2 \supseteq L(N_2)$ be down-closed sets of nodes of N_1 and N_2 such that $N_1[Y_1] \overset{\phi}{\simeq}_X N_2[Y_2]$. Suppose there is a lowest node u_2 in $N_2 \setminus Y_2$ such that u_2 is a reticulation. Then there is a network N_1' with a down-closed set Y_1' such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\mathrm{rSPR}}(N_1, N_1') \leq 1$.*

Proof. Let $x_2 \in Y_2$ be the child of u_2, let $x_1 = \phi^{-1}(x_2) \in Y_1$ be its corresponding node in N_1, and z_1 a parent of x_1 not in Y_1. If z_1 is a reticulation (Lemma 3.5 Case 1) then we set $N_1' = N_1$ and $Y_1' = Y_1 \cup \{z_1\}$. It then follows that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\mathrm{rSPR}}(N_1, N_1') = 0 \leq 1$.

If z_1 is not a reticulation, then there exists a reticulation node in $v_1 \in N_1 \setminus Y_1$ (again, such a node must exist, as u_2 exists and $N_1 \setminus Y_1$ and $N_2 \setminus Y_2$ have the same number of reticulations).

One of the incoming arcs of (u_1, v_1) is movable. Furthermore, x_1 is not above u_1, because $x_1 \in Y_1$, $u_1 \notin Y_1$ and Y_1 is down-closed. Hence, unless $u_1 = z_1$, the move $v_1 \xrightarrow{(u_1,v_1)} (z_1, x_1)$ is valid and it results in a network N_1' with the down-closed set $Y_1' = Y_1 \cup \{v_1\}$ such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\mathrm{rSPR}}(N_1, N_1') \leq 1$.

Now, if $u_1 = z_1$, then either $(v_1, x_1) \in A(N_1)$, or the move $v_1 \xrightarrow{(p_1,v_1)} (z_1, x_1)$ is valid (with $p_1 \neq u_1$) and it results in a network N_1' with the down-closed set $Y_1' = Y_1 \cup \{v_1\}$ such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\mathrm{rSPR}}(N_1, N_1') \leq 1$.

Finally, if both moves aren't valid, then $(v_1, x_1) \in A(N_1)$, where v_1 is a lowest reticulation above Y_1. Hence, in that case, we can immediately add v_1 to the isomorphism, setting $N_1' = N_1$ with the down-closed set $Y_1' = Y_1 \cup \{v_1\}$ such that $N_1'[Y_1'] \simeq_X N_2[Y_2 \cup \{u_2\}]$ and $d_{\mathrm{rSPR}}(N_1, N_1') = 0 \leq 1$.

□

The next lemma follows by an argument identical to the proof of Lemma 3.7, where we use Lemma 5.17 instead of Lemma 3.5 to add reticulations to the down-closed sets.

Lemma 5.18 ([JJE+18] Theorem 4.8)**.** *Let $N_1, N_2 \in \mathcal{N}(n,k)$ and let $Y_1 \supseteq L(N_1)$ and $Y_2 \supseteq L(N_2)$ be down-closed sets of nodes of N_1 and N_2 such that $N_1[Y_1] \overset{\phi}{\simeq}_X N_2[Y_2]$. Suppose that $N_1 \setminus Y_1$ contains r reticulations and t tree nodes of N_1, then $d_{\mathrm{rSPR}}(N_1, N_2) \leq 2t + r$.*

By setting $Y_1 = L(N_1)$ and $Y_2 = L(N_2)$, and noting that a network in $\mathcal{N}(n,k)$ has $n + k - 1$ tree nodes and k reticulations (Observation 2.7), we get the following bound on the distance between any two directed networks.

Theorem 5.19 ([JJE+18] Theorem 4.8)**.** *Let $n \in \mathbb{Z}_{\geq 1}$ and $k \in \mathbb{N}$, then* $\mathrm{diam}_{\mathrm{rSPR}}(n,k) \leq 2n + 3k - 2$.

5.2.2 rNNI upper bound: using tree diameters

In this section, we prove a linearithmic upper bound for the diameters of rNNI spaces. It is not clear whether each head move can be replaced by a short sequence of rNNI moves. Hence, we do not try to use the bounds for rSPR moves, but we use an independent proof.

This proof is a modified version of Theorem 13 in [JK19] for SPR moves. In that paper, all networks under consideration were orientable (or, proper, as it was called in that paper). Hence, with a little care, we can use the same proof as in to give a linearithmic bounds on the diameter of rNNI spaces. Our process for proving that $\mathcal{N}_{\text{rNNI}}(n,k)$ is connected with a linearithmic diameter is as follows (Figure 5.12).

Step 1. Transform any network N into a network N_T that is tree-based on $T \in \mathcal{T}(N)$.

Step 2. Transform N_T into handcuffed tree N_H by adding handcuffs between the root arc and the incoming arc of leaf 1.

Step 3. Transform N_H into a sorted handcuffed caterpillar N^*.

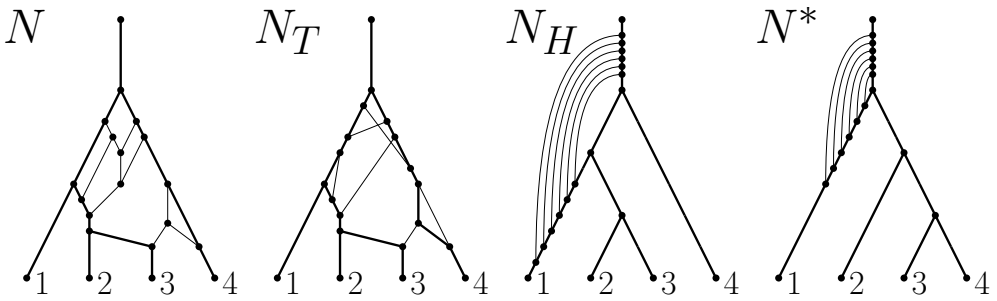


Figure 5.12: The process used in the proof of 5.23. We transform a network N into a tree-based network N_T, then into a handcuffed tree N_H, and, finally, into a sorted handcuffed caterpillar N^*.

Lemma 5.20 (Step 1). *Let $n > 1$ and $N \in \mathcal{N}(n,k)$, then there is a tree-based network N_T such that $d_{\text{rNNI}}(N, N') \leq 2k-2$.*

Proof. Let $T \in \mathcal{T}(N)$ with embedding ψ. Note that N is tree based if $|V(N)| - |V(\psi(T))| = 0$, and that this difference is at most $2k-2$.

Suppose $|V(N)| - |V(\psi(T))| = d > 0$, we prove that there is a network N' with T embedded as $\psi'(T)$ such that $|V(N)| - |V(\psi'(T))| < d$ and $d_{\text{rNNI}}(N, N') \leq 1$. As d is at most $2k-2$, this proves the lemma.

As not all nodes of N are in $\psi(T)$, there is an arc (u, v) with $u \in \psi(T)$ and $v \notin \psi(T)$. Choose (u, v) so that there is no other such arc (x, y) with x below u.

If v is a tree node, then at least one outgoing arc (v, w) of v is movable, and the rNNI move $v \xrightarrow{(v,w)} (p, u)$ is valid. Now, let $\psi'(T)$ be the embedding of T in the resulting network N', obtained by removing (p, u) from $\psi(T)$, and adding (p, v) and (v, u). Now $\psi'(T)$ has one more node than $\psi(T)$ (namely, the node v), so $|V(N')| - |V(\psi'(T))| < d$ (Figure 5.13 Left).

Now assume v is a reticulation and let $y \neq u$ be the other parent of v, w the child of v, and $z \neq v$ the other child of u. If (y, v) is not head-movable, then $z = w$ and we can simply find a new embedding ψ' of T into N, by removing (u, z) from $\psi(T)$ and adding (u, v) and (v, z) (Figure 5.13 Middle). Hence, for this new embedding, $|V(N)| - |V(\psi'(T))| < d$. Otherwise, (y, v) is head-moveable, and we can move the head of $v \xrightarrow{(y,v)} (u, z)$, because $y \neq u$ and y is not below u (Figure 5.13 Right). Again, T has an embedding $\psi'(T)$ in the resulting network N' consisting of $\psi(T)$ with (u, z) removed and (u, v) and (v, z) added, so $|V(N')| - |V(\psi'(T))| < d$. □

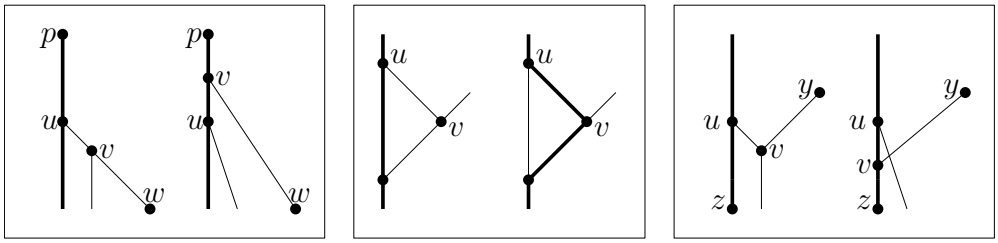


Figure 5.13: Transformation and rNNI moves used in Lemma 5.20 to obtain a tree-based network N_T.

Lemma 5.21 (Step 2). *Let $n > 1$ and $N_T \in \mathcal{N}(n, k)$ be a tree-based network, then there is a handcuffed tree N_H on leaf 1 such that $d_{\mathrm{rNNI}}(N_T, N_H) \leq 5k + 5n + (k + 2)\log(k + 2)$.*

Proof. Let $\psi(T)$ be the embedding of the base tree T into N_T, and let $M = A(N_T \setminus \psi(T)) = \{(u_1, v_1), \ldots, (u_k, v_k)\}$ be the arcs that are not in the embedding T into N_T. Without loss of generality, assume that for $i \in \{1, \ldots, r\}$ the distance between u_i and the root in $\psi(T)$ is at most the distance of u_{i+1} to root in $\psi(T)$. The idea is to carry all u_i to the root arc by sweeping from the leaves to the root, and then do something similar to move the v_i towards leaf 1.

For an arc a of T, let P_a be the path of $\psi(T)$ corresponding to a. Let a_ρ be the arc of T incident to the root. Let $\prec$ be a linear extension of the 'lower than' order on the arcs of T (so that $a' \prec a_\rho$ for all $a \in A(T)$)

Let $a = (x, y)$ be the minimum of $\prec$ restricted to arcs a' of T such that $P_{a'}$ contains a u_i. Let $P_a = (x, \ldots, y)$ be the corresponding path in $\psi(T)$. Proceed as follows along P_a from y to x.

(i) If there is an arc (v_l, u_i) in P_a, then tail_1-move $u_i \xrightarrow{(u_i,c)} (p, v_l)$ with $(u_i, c) \notin P_a$ $p \neq u_l$ (i.e., so that in $(p, v_L) \in P_a$).

(ii) If there is an arc (u_l, u_i) in P_a then tail_1-move $u_i \xrightarrow{(u_i,c)} (u_l, v_l)$, where $(u_i, c) \notin P_a$.

(iii) Otherwise, if there is an arc (x, u_i) in P_a, then tail_1-move $u_i \xrightarrow{(u_i,c)} (\cdot, x)$, where $(u_i, c) \notin P_a$.

This is illustrated in 5.14. After this process, all u_i on P_a are moved to the image $\psi(a')$ of the arc a' directly above a in T. Informally speaking, we stack u_i onto u_l so they can move together towards a_ρ. Repeat this process for each arc in the order given by $\prec$. For the last arc a_ρ, ignore case (iii). Next "unpack" the stacked u_i's on a_ρ.

We now count the number of tail_1 moves needed. Firstly, each v_l is swapped at most once with a u_i (k moves). Secondly, each u_i is moving to and from another arc (u_l, v_l) at most once ($2k$ moves). Furthermore, each vertex of $\psi(T)$ corresponding to a vertex of T is swapped at most twice ($2n$ moves). Hence, the total number of NNI moves required is at most $3k + 2n$.

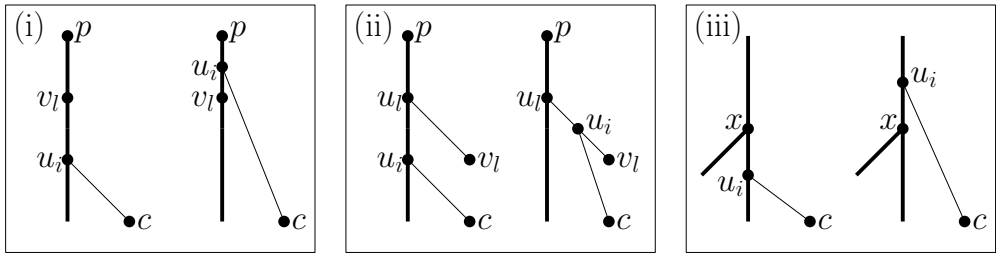


Figure 5.14: The rNNI move used in Lemma 5.21 to obtain a handcuffed tree N_H.

Repeat this process to move the v_i towards leaf 1 using head_1 moves, where $\prec$ is the linear order on the arcs of T induced by the the partial order induced by T rooted at leaf 1. Since the v_i do not have to be swapped with u_j, the total number of rNNI moves required for this is at most $2k + 2n$.

Note that the resulting network may not yet be a handcuffed tree as the order of the u_i and the v_j on P_ρ and P_1 may be different. To solve this, we sort the arcs of the caterpillar $P_\rho \cup M$ on $k+1$ leaves with the mergesort-like algorithm by Li et al. [LTZ96, Lemma 2]. In fact, we simply use the fact that the distance between two rooted caterpillars with $k+1$ leaves is at most $(k+2)\log(k+2)$ (Proposition 2.65). □

Lemma 5.22 (Step 3). *Let $N_H \in \mathcal{N}(n,k)$ be a handcuffed tree on leaf 1, and let N^* be the ladder caterpillar in $\mathcal{N}(n,k)$ with base tree $C((n,\dots,1))$ then $d_{\mathrm{rNNI}}(U_N, N^*) \leq (n+1)(1+\log(n+1))$*

Proof. Let $M = \{(u_1,v_1),\dots,(u_k,v_k)\}$ be the handcuffs of N_H. We simply transform the tree $S(N_H \setminus M)$ into the ladder caterpillar on n leaves using at most $(n+1)(1+\log(n+1))$ tail_1 moves (Proposition 2.65). In this process, the handcuffs remain between the root and the first leaf in the same order, so the resulting network is N^*. □

Theorem 5.23. *For all $n > 0$, the spaces $\mathcal{N}_{\mathrm{rNNI}}(n,k)$ are connected, and if $n > 1$, then $\mathrm{diam}_{\mathrm{rNNI}}(n,k) \leq 11n+14k-4+2(k+2)\log(k+2)+(n+1)(1+\log(n+1))$.*

Proof. Let $N, N' \in \mathcal{N}(n,k)$ be two arbitrary networks. If $n = 1$, then the spaces are connected by Theorem 3.10.

Otherwise, N can be transformed into a k-handcuffed tree N_H via a tree-based network using at most $5n + 7k - 2 + (k+2)\log(k+2)$ rNNI moves (Lemmas 5.20 and 5.21). Using n additional tail_1 moves, this network can be transformed into a k-handcuffed caterpillar N^*, by just changing the tree formed by the network without the handcuffs.

Then, to see there is a sequence of the desired length between N and N', note that N' can be transformed into N^* via a k-handcuffed tree using at most $5n + 7k - 2 + (k+2)\log(k+2) + (n+1)(1+\log(n+1))$ rNNI moves (Lemmas 5.20, 5.21, and 5.22). Hence, the distance between N and N' is at most $11n + 14k - 4 + 2(k+2)\log(k+2) + (n+1)(1+\log(n+1))$. □

Note that Steps 2 and 3 concern tree-based networks only, but the intermediate networks in these proofs are not all tree-based. This is mainly because we attempt to use an efficient sequence. If one desires to keep the networks tree based, it is still easy to find a sequence, as we can simply collect the heads and tails of the reticulations one at a time, without combining them; and we can sort the caterpillars while keeping the caterpillar structures as well. This results in a quadratic upper bound for the space of tree-based networks under rNNI. An independent proof of this is given in [EFM20]. There, they prove the stronger fact that this space is connected by tail_1 moves. Moreover, they also prove it for networks with outdegree-2 roots.

In the next chapter, we apply our proof of connectedness of directed networks under rNNI to undirected networks (as it originally was in [JK19]). Like for directed networks, our proof cannot be applied immediately to undirected tree-based networks, as the collection of the reticulation arcs and rearrangement of the caterpillars does not allow this. The proof can be modified, like for directed networks, to give a quadratic upper bound on the diameter of undirected tree-based network space. Nevertheless, Francis and Fischer claim that our proof technique is not applicable to undirected tree-based networks at all [FF20]. However, by performing the transformations naively as described for the directed case, we essentially obtain their proof. This becomes even clearer when inspecting their intermediate networks, so-called *shoat* networks, which are the same as our intermediate stage of handcuffed trees.

5.2.3 Asymptotic bounds

Francis et al. ([FHMW17]) proved lower bounds for the diameters of spaces of *undirected* networks under NNI and SPR moves. To apply these bounds to the case of rNNI and rSPR moves, we need more information about the relation between these moves and their spaces.

Lemma 5.24 ([JJE+18] Lemma 4.19)**.** *Let $N, N' \in \mathcal{N}(n,k)$ be networks such that $d_{\mathrm{rSPR}}(N,N') = 1$, then $d_{\mathrm{SPR}}(U(N),U(N')) \leq 1$.*

Furthermore, if $d_{\mathrm{rNNI}}(N,N') = 1$, then $d_{\mathrm{NNI}}(U(N),U(N')) \leq 1$.

Proof. Suppose the rSPR move in N resulting in N' moves endpoint u of arc e from f to another arc g. Then, moving endpoint u of edge e to edge g in $U(N)$ results in $U(N')$. If the rSPR move was actually a rNNI move, then the SPR move in $U(N)$ is an NNI move. This follows from the fact that f and g are adjacent in N iff they are adjacent in the undirected graph $U(N)$. □

Corollary 5.25 ([JJE+18] Corollary 4.20)**.** *For any pair of networks $N, N' \in \mathcal{N}(n,k)$, we have $d_{\mathrm{rSPR}}(N,N') \geq d_{\mathrm{SPR}}(U(N),U(N'))$ and $d_{\mathrm{rNNI}}(N,N') \geq d_{\mathrm{NNI}}(U(N),U(N'))$.*

For the application of this corollary, it is important to note that not every undirected network has an orientation as a directed network. In particular, we cannot deduce simple relations between diameter bounds for $\mathcal{U}_{\mathrm{SPR}}$ and $\mathcal{N}_{\mathrm{rSPR}}$. Indeed, the diameter of the space of undirected networks could be determined by parts of the space consisting of networks that are not orientable.

Nevertheless, we can use this corollary to study diameters. For lower bounds on $\mathcal{N}(n,k)$, we simply consider orientable networks in $\mathcal{U}(n,k)$; and for upper bounds on $\mathcal{U}(n,k)$, we bound the number of moves needed to make an undirected network orientable (Section 6.1).

For the lower bound on $\mathcal{N}_{\mathrm{rNNI}}(n,k)$, we use the following lower bound from Francis et al. for NNI diameters ([FHMW17]).

Theorem 5.26 ([FHMW17] Theorem 2 and Proposition 4)**.** *Let $n \in \mathbb{Z}_{\geq 2}$ and $k \in \mathbb{N}$, then there exist two Echidna networks $U, U' \in \mathcal{U}(n,k)$ such that $d_{\mathrm{NNI}}(U,U')$ is at least*

$$\frac{1}{20}\left((v_k^n-3)\log_6\left(\frac{v_k^n}{2}-2\right)-(2k-1)\log_6(k-1)-(v_k^n-2k)\log_6 e-2v_k^n\right),$$

and there exist two Echidna networks $U, U' \in \mathcal{U}(n,k)$ such that $d_{\mathrm{SPR}}(U,U')$ is at least

$$\mathrm{diam}_{\mathrm{SPR}}(n,k) \geq \frac{(v_k^n-3)\ln(\frac{v_k^n}{2}-2)-(2k-1)\ln(k-1)-(2n-2)}{4\ln(2(v_k^n+k))},$$

where v_k^n is the number of nodes in an undirected network with n leaves and reticulation number k. [2]

As Echidna networks are orientable (Lemma 2.70) the argument of Francis et al. extends to directed networks.

Lemma 5.27 ([JJE+18])**.** *Let $n \in \mathbb{Z}_{\geq 1}$ and $k \in \mathbb{N}$, then there exist two networks $N, N' \in \mathcal{N}(n,k)$ such that $d_{\mathrm{NNI}}(N,N')$ is at least*

$$\frac{1}{20}\left((v_k^n-3)\log_6\left(\frac{v_k^n}{2}-2\right)-(2k-1)\log_6(k-1)-(v_k^n-2k)\log_6 e-2v_k^n\right),$$

where v_k^n is the number of nodes in an undirected network with $n+1$ leaves and k reticulation nodes.

Proof. Fix n and k, and let U and U' be two Echidna networks in $\mathcal{U}(n,k)$ such that $d_{\mathrm{NNI}}(U,U')$ is at least

$$\frac{1}{20}\left((v_k^n-3)\log_6\left(\frac{v_k^n}{2}-2\right)-(2k-1)\log_6(k-1)-(v_k^n-2k)\log_6 e-2v_k^n\right),$$

which exist by Theorem 5.26. By Lemma 2.70, there are two networks $N, N' \in \mathcal{N}(n,k)$ such that $N(U) = U$ and $N(U') = U'$. By Corollary 5.25, we have $d_{\mathrm{rNNI}}(N,N') \geq d_{\mathrm{NNI}}(U(N),U(N'))$, which proves the statement. □

[2] The notation in their paper is somewhat unclear, as they claim $\Delta_i^{\mathrm{SPR}}(n)$ is the diameter of the SPR space $\mathcal{U}_{\mathrm{SPR}}(n,k)$, but they also use n for the number of nodes v_k^n in such a network.

Theorem 5.28 ([JJE+18] Theorem 4.12)**.** *The diameters of tiers of network space for metrics induced by rNNI and* tail_1 *moves satisfy* $\text{diam}_{\text{rNNI}}(n,k) = \Theta((n+k)\log(n+k))$ *and* $\text{diam}_{\text{tail}_1}(n,k) = \Omega((n+k)\log(n+k))$.

Using the same argument, we get a lower bound for the diameter of spaces of directed networks under non-local moves as well.

Theorem 5.29. *The diameters of tiers of network space for metrics induced by rSPR,* head*, and* tail *moves satisfy* $\text{diam}_{\text{rSPR}}(n,k) = \Theta(n+k)$*,* $\text{diam}_{\text{head}}(n,k) = \Theta(n+k)$*, and* $\text{diam}_{\text{tail}}(n,k) = \Theta(n+k)$.

5.3 Internal labels

As before, we start with spaces of networks without degree-2 nodes. For rSPR moves, we simply make the networks leaf-isomorphic, and then permute the tree nodes and the reticulations using a small number of tail and head moves respectively. For rNNI moves, we once again use the structure of ladder caterpillars.

Theorem 5.30. *For all* $n \geq 1$ *and* $k \geq 0$*, the spaces* $\dot{\mathcal{N}}_{\text{rSPR}}(n,k)$ *and* $\dot{\mathcal{N}}_{\text{rNNI}}(n,k)$ *are connected. Furthermore, their diameters are bounded by* $\text{diam}_{\text{rSPR}}(n,k,0) \leq \text{diam}_{\text{rSPR}}(n,k) + 4n + 8k - 4$ *and* $\text{diam}_{\text{rNNI}}(n,k,0) \leq 2\,\text{diam}_{\text{rNNI}}(n,k) + 2n^2 + 8nk + 8k^2 - 2n - 4k$.

Proof. Let $\dot{N}_1, \dot{N}_2 \in \dot{\mathcal{N}}(n,k,0)$ be arbitrary, we prove that there are rSPR and rNNI sequences from $\dot{N}_1$ to $\dot{N}_2$ of the desired length.

Because the spaces of rSPR moves are connected, there is a sequence of at most $\text{diam}_{\text{rSPR}}(n,k)$ moves from $\dot{N}_1$ to a network $\dot{N}_1'$ such that $\dot{N}_1' \simeq_{X_l} \dot{N}_2$ (Theorem 5.19). Then using at most $4k$ head moves and $4(n+k-1)$ tail moves, we can permute the reticulations and the tree nodes to make the networks labeled isomorphic with respect to the full label set X (Propositions 3.19 and 4.27). Counting the number of moves for each of these steps, we get $\text{diam}_{\text{rSPR}}(n,k,0) \leq \text{diam}_{\text{rSPR}}(n,k) + 4n + 8k - 4$.

For the rNNI moves, we take a similar approach, but changing both networks into leaf-isomorphic ladder caterpillars first using at most $\text{diam}_{\text{rNNI}}(n,k)$ rNNI moves in each network. Then we can permute the reticulations using at most $2k^2 + 2k$ head_1 moves, and the tree nodes using at most $(2n+6k)(n+k-1)$ tail_1 moves (Propositions 3.19 and 4.27). Counting the number of moves for each of these steps, we get $\text{diam}_{\text{rNNI}}(n,k,0) \leq 2\,\text{diam}_{\text{rNNI}}(n,k) + 2n^2 + 8nk + 8k^2 - 2n - 4k$. □

Note that $\mathrm{diam}_{\mathrm{rSPR}}(n,k,0) \geq \mathrm{diam}_{\mathrm{rSPR}}(n,k) = \Omega(n+k)$, which implies that we have an asymptotically tight bound $\mathrm{diam}_{\mathrm{rSPR}}(n,k,0)\Theta(n+k)$ for internally labeled networks as well.

5.3.1 Degree-2 nodes

To prove connectedness and diameter bounds in the presence of degree-2 nodes, we essentially reduce to the case of no degree-2 nodes by first collecting all degree-2 nodes at the root. This works for all (n,k) except when $(n,k) = (1,0)$ (then the space is not connected) and when $(n,k) = (1,1)$ (then there is no corresponding network without degree-2 nodes).

Lemma 5.31. *Let $\ddot{N} \in \dot{\mathcal{N}}(n,k,m)$ be a suppressable network, then the degree-2 nodes can be collected at the top in a predefined order using a sequence of at most $2n+3k+2m+2$ rSPR moves, or at most $m(2n+3k)+m^2+4m+3$ rNNI moves, except when $n=1$, $k=0$, and $m>0$.*

Proof. We may move all degree-2 nodes up to the root arc by starting at the bottom, and moving all degree-2 nodes up one arc in $S(\ddot{N})$ at a time. Moving all degree-2 nodes up one arc takes at most one rSPR move. Indeed, one can move the vertex incident to both arcs down below all the degree-2 nodes. The same can be achieved using one rNNI move per degree-2 node on the lower arc. As there are at most m degree-2 nodes on this arc, we need at most m rNNI moves. The network $S(\ddot{N})$ has $2n+3k-1$ arcs, so it takes at most $2n+3k-1$ SPR moves or $m(2n+3k-1)$ rNNI moves to collect all degree-2 nodes at the root arc. To sort the degree-2 nodes, we need an additional $2m+3$ rSPR or m^2+5m+3 rNNI moves (Lemmas 3.26). □

Theorem 5.32. *The spaces $\dot{\mathcal{N}}(n,k,m)$ are connected for all $n,k,m>0$, except when $n=1$, $k=0$, and $m>0$. Furthermore, the diameter of the space of subdivided networks under rSPR is bounded by $\mathrm{diam}_{\mathrm{rSPR}}(n,k,m) \leq \mathrm{diam}_{\mathrm{rSPR}}(n,k,0)+4n+6k+4m+2$ and $\mathrm{diam}_{\mathrm{rNNI}}(n,k,m) \leq \mathrm{diam}_{\mathrm{rNNI}}(n,k,0)+4mn+6mk+m^2+2m+2$.*

Proof. We first note that $\dot{\mathcal{N}}(1,0,m)$ is not connected. To see this, note that for each permutation $x_1,\ldots,x_m$ of the m degree-2 nodes, there is a unique network in $\dot{\mathcal{N}}(1,0,m)$ consisting of the path $(\rho,x_1,\ldots,x_m,l)$, where ρ is the root of the network and l is the leaf. As all networks in $\dot{\mathcal{N}}(1,0,m)$ necessarily consist of such a path and none of the arcs of such a network can be moved the space $\dot{\mathcal{N}}(1,0,m)$ consists of $m!$ nodes and no arcs.

Then, to prove connectedness in all other cases, we first turn to the case $n=k=1$ and $m>0$. Here, we can first move all degree-2 nodes from

the leaf path to one of the parallel paths using one head move or at most m head_1 moves. Then, by Lemma 3.26, all degree-2 nodes can be sorted on either of the parallel arcs using at most $2m + 2$ tail moves or $m^2 + 4m + 2$ tail_1 moves. Hence, $\dot{\mathcal{N}}(1, 1, m)$ is connected for all $m > 0$. Furthermore, as we can collect all degree-2 nodes on one arc in arbitrary order in one of the networks using at most 3 tail moves or $2m$ tail_1 moves, $\text{diam}_{\text{rSPR}}(1, 1, m) \leq 2m + 6$ and $\text{diam}_{\text{rSPR}}(1, 1, m) \leq m^2 + 7m + 2$.

For all other cases (i.e., $n > 1$ or $k > 1$) any network in $\dot{\mathcal{N}}(n, k, m)$ can be transformed into a suppressable network using at most m tail_1 moves (Lemma 3.24). Then, we may move all degree-2 nodes up to the root arc using at most 1 head move or m head_1 moves per reticulation, and at most 2 tail moves or $2m$ tail_1 moves per tree node. Hence, this can be achieved using at most $2n + 3k - 2$ rSPR moves or $m(2n + 3k - 2)$ rNNI moves in total. To do this, we start at the bottom, and move all degree-2 nodes up one arc at a time.

Again, these degree-2 nodes can be sorted at the root arc using at most $2m + 2$ tail moves or $m^2 + 4m + 2$ tail_1 moves (Lemma 3.26). Starting with two networks and applying this procedure to both (noting that we only have to sort the degree-2 nodes in one of the networks) we can easily see that $\text{diam}_{\text{rSPR}}(n, k, m) \leq \text{diam}_{\text{rSPR}}(n, k, 0) + 2(m + 2n + 3k - 2) + 2m + 2 = \text{diam}_{\text{rSPR}}(n, k, 0) + 4n + 6k + 4m + 2$ and $\text{diam}_{\text{rNNI}}(n, k, m) \leq \text{diam}_{\text{rNNI}}(n, k, 0) + 2(m + m(2n + 3k - 2)) + m^2 + 4m + 2 = \text{diam}_{\text{rNNI}}(n, k, 0) + 4mn + 6mk + m^2 + 2m + 2$. □

Corollary 5.33. *The spaces $\dot{\mathcal{N}}(n, k, m)$ under rSPR and and rNNI moves have diameters bounded by* $\text{diam}_{\text{rSPR}}(n, k, m) \leq 10n + 17k + 4m - 4$ *and*

$$\begin{aligned}\text{diam}_{\text{rNNI}}(n, k, m) \leq & 2n^2 + 8nk + 8k^2 + 4mn + 6mk + m^2 \\ & + 2(k + 2)\log(k + 2) + 2(n + 1)(1 + \log(n + 1)) \\ & + 20n + 24k + 2m - 6.\end{aligned}$$

5.4 Conclusion

In this chapter, we have studied the spaces of networks under rSPR and rNNI moves. We have started by showing that head moves and tail moves are tightly related in networks with at least two leaves: a head move can be replaced by a sequence of at most 13 tail moves (if $n > 1$), and each tail move by a sequence of at most 16 head moves.

This provides a new proof for connectedness of $\mathcal{N}_{\text{tail}}(n, k)$ using the connectedness of $\mathcal{N}_{\text{head}}(n, k)$ and vice versa. Furthermore, connectedness of $\mathcal{N}_{\text{tail}}(n, k)$ also implies the connectedness of $\mathcal{N}_{\text{rSPR}}$. Using both tail moves and head moves,

we give a better upper bound for the rSPR diameters than the one following directly from tail moves.

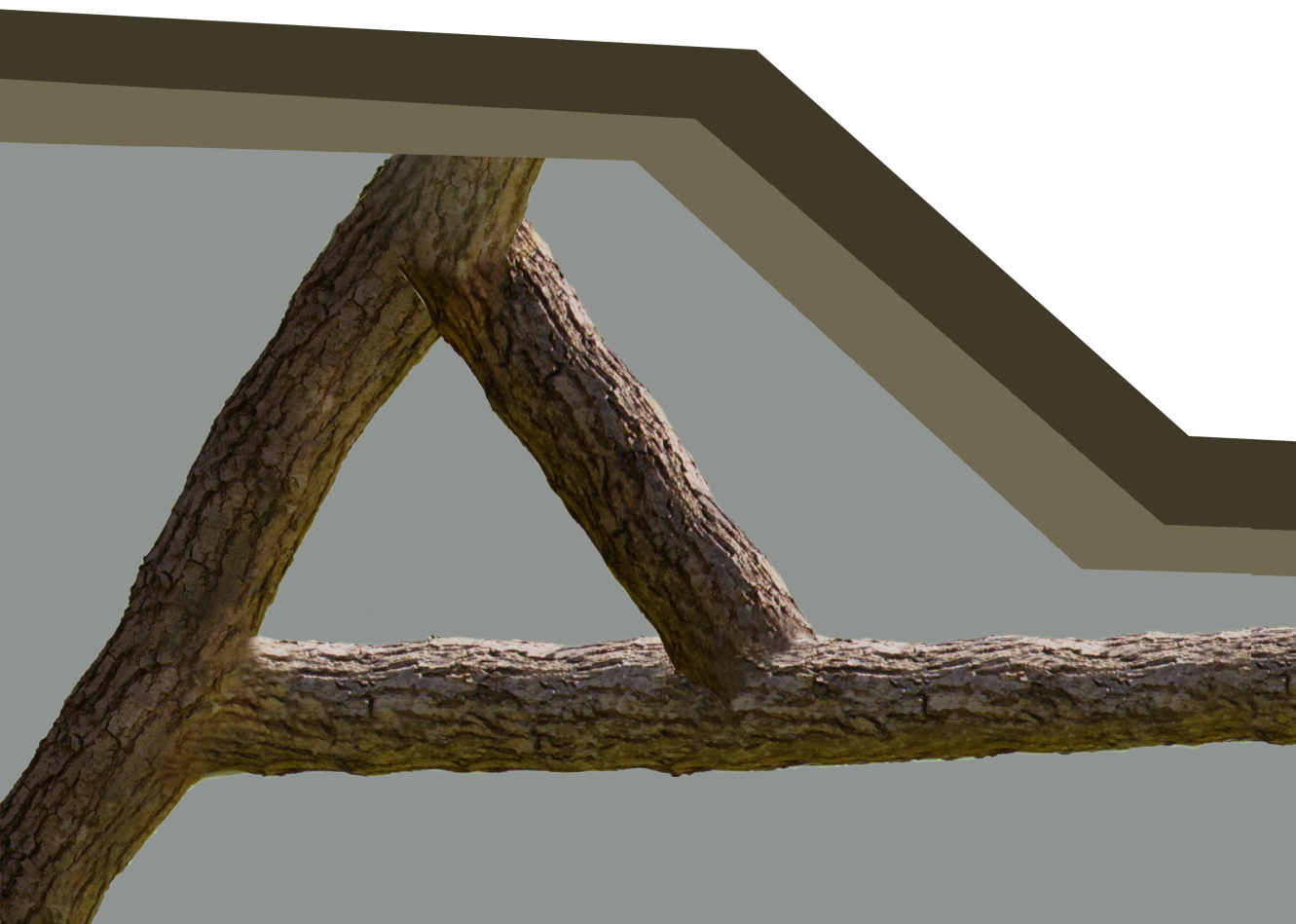
For the connectedness and diameter of rNNI moves we use a different technique. This technique is an adaptation of the connectedness proof for NNI moves in [JK19]. The construction gives a diameter bound for $\mathcal{N}_{\mathrm{rNNI}}(n,k)$ of order $O((n+k)\log(n+k))$ for networks with at least two leaves. This bound is asymptotically tight by Theorem 5.28, which uses the linearithmic lower bound for NNI spaces from [FHMW17].

For networks with one leaf, our diameter bound for $\mathcal{N}_{\mathrm{NNI}}(1,k)$ is order $O(k^2)$, which follows by using the bound for tail moves. Note that this is not as good as the linearithmic bounds for networks with at least two leaves. We conjecture that a linearithmic upper bound can be obtained for $n=1$ as well, by using the following strategy. Make a triangle $\langle a,b,c\rangle_r$ at the bottom of the network using a small number of moves, and then treat a and b as the leaves of a network in $\mathcal{N}(2,k-2)$. The result then follows from our linearithmic upper bound for networks with at least two leaves. It would also be interesting to check whether this proof technique can be used to prove linearithmic upper bounds for the diameters of $\mathcal{N}_{\mathrm{tail}_1}(n,k)$ and $\mathcal{N}_{\mathrm{head}_2}(n,k)$.

The results easily extend to internally labeled networks, too. All spaces $\dot{\mathcal{N}}_{\mathrm{rSPR}}(n,k,m)$ and $\dot{\mathcal{N}}_{\mathrm{rNNI}}(n,k,m)$ are connected, except when $(n,k)=(1,0)$ and $m>0$. The upper bound for the rSPR diameter is linear, even when considering degree-2 nodes as well. This is because we can swap tree nodes using a constant number of tail moves, and reticulations using a constant number of head moves.

Recall that our upper bounds for the diameters of the spaces $\dot{\mathcal{N}}_{\mathrm{tail}}(n,k,m)$ and $\dot{\mathcal{N}}_{\mathrm{head}}(n,k,m)$ contain quadratic terms. The fact that each tail move can be replaced by a constant number of head moves and vice versa makes it believable that these quadratic terms are not necessary. Indeed, if these replacement lemmas still hold for internally labeled networks, two reticulations can be swapped with a constant number of tail moves. Hence, it would be interesting to resolve this open question, by carefully checking the sequences of moves in these lemmas to see what happens to the labels of all nodes involved.

6



SPR and NNI Moves

In this chapter, we study spaces of undirected networks. The most common rearrangement moves for undirected networks are SPR moves and their local versions, NNI moves. Although there is a fundamental difference between directed and undirected networks, there is a strong connection between (parts of) the spaces of directed and undirected networks. Especially when we consider spaces of orientable networks. Indeed, each rSPR move on a directed network N gives an SPR moves in the orientable network $U(N)$.

To prove connectedness and diameter bounds for the spaces of undirected networks, we heavily use this connection between rSPR and SPR moves on directed and orientable networks. Hence, we start this chapter with a section about this relation (Section 6.1). Then, in Section 6.2, we prove connectedness and diameter bounds for SPR and NNI spaces. In these proofs, we turn each undirected network into an orientable network, and then leverage the relation between directed networks and orientable networks. In Section 6.3, we again generalize to spaces of internally labeled networks. The proofs in that section use techniques similar to those used for rSPR moves.

Note that connectedness of NNI spaces—and thus also SPR spaces—was first proven in [HMW16]. The proof was based on the connectedness of cubic graphs under NNI [Tsu96]. Although their proof was constructive in nature, like all proofs in this thesis, [HMW16] did not give any diameter bounds. Such bounds were first given in [FHMW17], but have since been improved [JJE+18]. The currently best bounds can be found in this chapter.

6.1 Relation with directed moves

As we have seen in Lemma 5.24, moves on directed networks can easily be translated to moves on their underlying undirected networks. The converse is not true in general.

However, the converse is true in a limited sense for orientable networks: if we have a network N and an SPR (or NNI) move $U(N)$ to an *orientable* network U', then we can find an rSPR (or rNNI) sequence between N and some network N' for which $U(N') = U'$. This follows simply from the fact that spaces of directed networks are connected under rSPR and rNNI.

This gives a relation between spaces of directed networks and spaces of *orientable* networks. The following faulty version of this result is used in [GvIJ+17b] to prove connectedness of spaces of rNNI moves, using connectedness of spaces of NNI moves.[1]

[1] The authors of the paper have been informed about this issue on 14 June 2019, but have

False Lemma 6.1 (Lemma 3, [GvIJ+17b]). *Let N be a directed network and U' an undirected network one NNI move away from $U(N)$, then there is a sequence of rNNI moves from N to a network N' with $U(N') = U'$.*

There is a serious issue with the claim in this lemma, which becomes clear when we investigate the following example. Consider the non-orientable undirected network U' obtained from a copy of K_4 and the network on two leaves by subdividing an edge in both and adding an edge between the two resulting degree-2 nodes (See Figure 6.1). As U' is not orientable, there is no N' such that $U(N') = U'$. However, using one NNI move in U', we can obtain an orientable network U, with orientation N. The lemma fails for N and U'.

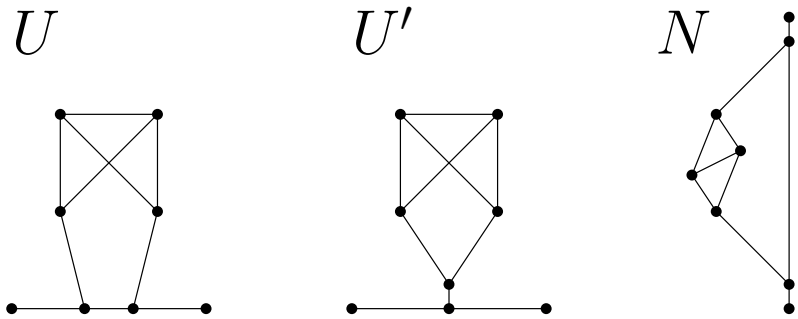


Figure 6.1: A counterexample showing that an NNI move on an orientable network U, with orientation N, can result in an unorientable network U'.

As mentioned above, the lemma can be fixed by restricting U' to be an orientable network, because it then simply follows from the fact that the tiers of rNNI space are connected—which leads to a cyclic argument in [GvIJ+17b], because this is what the lemma is ultimately used for. The first proof that spaces of orientable networks are connected under NNI can be found in [JK19], although it also easily follows from the results in [GvIJ+17b].

Lemma 6.2. *Let N be a directed network and U' an* orientable *network one NNI move away from $U(N)$, then there is a sequence of rNNI moves from N to a network N' with $U(N') = U'$.*

A more interesting question is whether an NNI (SPR) move from $U(N)$ to U' can be replaced with a constant number of rNNI (rSPR) moves from N to a network N' with $U(N') = U'$. The proof given in [GvIJ+17b] does not help in answering this question as, firstly, it does not give a constant number of

not corrected or retracted their paper yet.

moves—and secondly, there is a mistake in the proof. Hence, we have to find another proof, or a counterexample.

Considering the following example, it seems unlikely that such sequences of constant moves always exist. This is hard to prove, because we have no good algorithms for computing distances yet (Chapter 7).

Example 6.3. Let $b > 0$, and N be the network obtained from a series of $3b$ in-degree-1, outdegree-1, level-2 blobs by adding $c > 0$ arcs to the b-th blob, and then adding an arc between the root arc and the incoming arc of the single leaf of the network. Using one NNI move in $U(N)$, we can get a network U' for which the chain of blobs is reversed, and in which the $(2b+1)$-th blob has c additional arcs. To simulate this in the oriented networks, we essentially have to move all these c arcs, which, intuitively, has to take a number of moves increasing with c; or we have to move the level-$(2+c)$ blob, over a distance increasing with b.

In light of this example, the best expectable result would be that we can replace an NNI move with a (non-constant) sequence of rNNI moves, which is trivial by Theorem 5.23. This example does not tell us anything about the relation between SPR moves and rSPR moves—the level-$(2+c)$ blob can be moved over a large distance using a small number of moves (note that it also needs to be turned upside-down, which may still take a large number of moves).

However, as we have seen in Chapter 2, a similar claim about the relation between NNI and SPR fails already on trees. This follows from a result from Atkins and McDiarmid ([AM19], Example 2.3; Lemma 2.67). Recall, also, that the claim holds for NNI moves on trees by Lemma 2.64.

6.1.1 Moving towards orientable networks

In this section, we bound the number of SPR and NNI moves needed to reach an orientable network (Propositions 6.7 and 6.10). For SPR moves, this entails bounding the number of redundant terminal blobs (Lemma 6.4), and then showing that we can reduce the number of redundant terminal blobs with one SPR move (Lemma 6.6). As it may take many moves to reduce the number of redundant terminal blobs using NNI moves, the proof for NNI moves uses a different quantity, the number of redundant cut-edges. We bound the number of redundant cut-edges (Lemma 6.8), and then show that this number can be reduced using one NNI move (Lemma 6.9). As a network with redundant terminal blobs or without redundant cut-edges is orientable (Lemma 2.69), this shows that we can reach an orientable network in a bounded number of moves.

SPR moves

Each redundant terminal blob accounts for at least three reticulations in the absence of degree-2 nodes. If degree-2 nodes are present, the redundant terminal blobs may account for fewer reticulations. The following lemma makes this precise, and provides a bound on the number of redundant terminal blobs in the presence of degree-2 nodes.

Lemma 6.4 ([JJE+18] Lemma 4.16)**.** *Let $\ddot{U} \in \dot{\mathcal{U}}(n,k,m)$, then the number of redundant terminal blobs in $\ddot{U}$ is at most the optimum value of the ILP* $\max(c_0 + c_1 + c_2)$ *s.t.* $3c_0 + 2c_1 + c_2 \leq k$ *and* $c_1 + 2c_2 \leq m$ *with decision variables* $c_i \in \mathbb{N}$.

Proof. Let V' be the set of nodes of a redundant terminal blob, and E' the set of edges incident to V'. Suppose the component has m' degree-2 nodes and i degree-3 nodes.

As each edge of E' except for one is adjacent to two nodes of V', we have $2|E'| - 1 = 3i + 2m'$, so i must be odd. Furthermore, $|V'|$ must be be at least 3, because $i \geq 1$ and a degree-3 node in V' has at least two neighbours in V'. Hence, $|E'| - |V'| \geq (3i + 2m' + 1)/2 - (i + m') = 1/2i + 1/2 \geq 1$ for each redundant terminal blob.

If $m' \leq 1$, then V' has at least two degree-3 nodes, and one of these has all its neighbours in V'. Hence, $|V'| \geq 4$ and $i \geq 3$. Hence, using the same calculation as in the previous paragraph using $i \geq 3$, we get $|E'| - |V'| \geq 2$ if $m' = 1$. If $m' = 0$, we have $|V'| = i \geq 4$. This, together with the fact that $|V'|$ must be odd implies that $i \geq 5$. Using the same calculation again, we get $|E'| - |V'| \geq 3$ if $m' = 0$.

Now observe that every node and edge of $\ddot{U}$ appears in at most one such V' or E'. Furthermore, if all nodes and edges in all such sets V' and E' are deleted, the resulting graph (V'', E'') is still connected because only redundant terminal blobs are removed. Hence, the remaining part satisfies $|E''| - |V''| \geq -1$. Note that $|E| - |V|$ is equal to $|E''| - |V''|$ plus the sum of all $|E'| - |V'|$ for every redundant terminal blob with nodes V' and incident edges E'.

Now suppose $\ddot{U}$ has c_0 redundant terminal blobs with no degree-2 nodes, c_1 redundant terminal blobs with one degree-2 nodes, and c_2 redundant terminal blobs with at least two degree-2 nodes.

Then

$$|E| - |V| = |E''| - |V''| + \sum_{j=0}^{2} \sum_{i=1}^{c_j} \left(|E'_{i,j}| - |V'_{i,j}| \right) \geq -1 + 3c_0 + 2c_1 + c_2,$$

where $V_{i,j}$ and $E_{i,j}$ denote the nodes and the edges of the ith redundant terminal blob with $j = 0, 1$ or $j \geq 2$ degree-2 nodes of $\ddot{U}$.

Furthermore, as the total number of degree-2 nodes is m, we have $c_1 + 2c_2 \leq m$. Using the fact that $k - 1 = |E| - |V|$, we see that the numbers of redundant terminal blobs c_0, c_1, and c_2 must be such that $3c_0 + 2c_1 + c_2 \leq k$ and $c_1 + 2c_2 \leq m$. □

The next lemma follows immediately.

Lemma 6.5 ([JJE+18] Lemma 4.16). *Let $\ddot{U} \in \dot{\mathcal{U}}(n, k, m)$, then the number of redundant terminal blobs in $\ddot{U}$ is at most k. If $m = 0$ or $\ddot{U}$ is suppressable, then there are at most $k/3$ redundant terminal blobs.*

Lemma 6.6 ([JJE+18] Lemma 4.17). *Let $\ddot{U} \in \dot{\mathcal{U}}(n, k, m)$ be a network with c redundant terminal blobs. Then there exists a network $\ddot{U}'$ with at most $c - 1$ redundant terminal blobs such that $d_{\mathrm{SPR}}(\ddot{U}, \ddot{U}') = 1$.*

Proof. Pick any redundant terminal blob B and let $\{u, v\}$ be the unique edge for which $u \notin B$, $v \in B$. Let x and y be the other neighbours of v. Now SPR move $v \xrightarrow{\{v,x\}} \{l, \cdot\}$ to an arbitrary leaf edge of $\ddot{U}$. Suppressing v cannot give parallel edges, because $\{u, v\}$ is a cut-edge (Figure 6.2). In the resulting undirected network, B is extended to a biconnected component with a pendant leaf, and because no new cut-edges have been created, the network has at most $c - 1$ redundant terminal blobs and is one SPR move away from the original undirected network. □

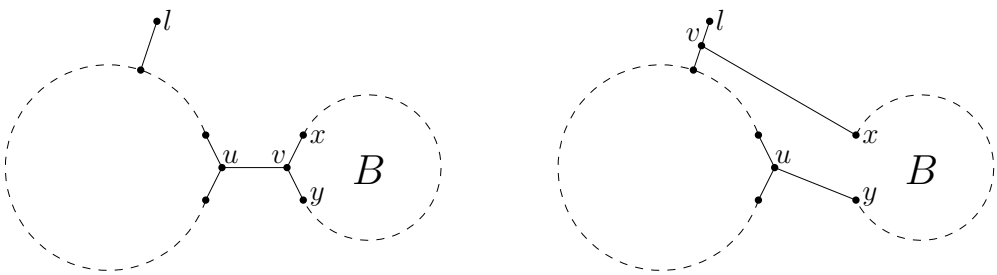


Figure 6.2: The SPR move used to remove redundant terminal blob B. The big circle represents all of U except for B. Note that in the undirected network on the right, B is not a redundant terminal blob anymore, and no extra redundant terminal blobs have been created.

Lemmas 2.69, 6.5 and 6.6 imply the following.

Proposition 6.7 ([JJE+18] Corollary 4.18). *Let $\ddot{U} \in \dot{\mathcal{U}}(n, k, m)$, then there exists an orientable network $\ddot{U}'$ such that $d_{\mathrm{SPR}}(\ddot{U}, \ddot{U}') \leq k$. If $m = 0$, then there exists such a network with $d_{\mathrm{SPR}}(\ddot{U}, \ddot{U}') \leq k/3$*

NNI moves

The number of redundant cut-edges can simply be bound by the maximal number of edges that are not in some embedded tree.

Lemma 6.8. *Let $\ddot{U} \in \dot{\mathcal{U}}(n,k,m)$, then the number of redundant cut-edges in $\ddot{U}$ is at most $3k+m$.*

Proof. Let G be the subgraph of U consisting of all paths between the leaves. Note that U has at most $2n+3k+m-3$ edges (Observation 2.7), and, as G has an embedded tree on n leaves, G has at least $2n-3$ edges. None of these edges can be redundant cut-edges. Indeed, if such an edge were a cut-edge, it would separate two of the leaves of the embedded tree, and it would not be redundant. Therefore, U has at most $3k+m$ redundant cut-edges. □

Lemma 6.9. *Let $\ddot{U} \in \dot{\mathcal{U}}(n,k,m)$ be a network with c redundant cut-edges. Then there exists an undirected network $\ddot{U}'$ with at most $c-1$ redundant cut-edges such that $d_{\mathrm{NNI}}(\ddot{U},\ddot{U}')=1$.*

Proof. Let $G \subseteq \ddot{U}$ be the graph consisting of the union of all paths between the leaves of $\ddot{U}$.

Let e be a redundant cut-edge closest to G. Note that e must be incident to G, that is, $e=\{u,v\}$ with $u \in G$ and $v \notin G$. Let $w_1, w_2 \neq v$ be adjacent to u. As e is a redundant cut-edge, there is no edge $\{u,w_2\}$, so the u-end of $\{u,w_1\}$ is movable. Let $z \neq u$ be adjacent to v, then the NNI move $u \xrightarrow{\{u,w_1\}} \{v,z\}$ is valid, because $w_1 \notin \{v,z\}$. Furthermore, the resulting network $\ddot{U}'$ has one redundant cut-edge fewer than $\ddot{U}$. □

Proposition 6.10. *Let $\ddot{U} \in \dot{\mathcal{U}}(n,k,m)$, then there is a sequence of at most $3k+m$ NNI moves that makes $\ddot{U}$ orientable.*

Proof. By Lemma 6.8, $\ddot{U}$ has at most $3k+m$ cut-edges. Using Lemma 6.9, we can reduce the number of redundant cut-edges to zero using at most $3k+m$ NNI moves. Such a network necessarily has no redundant terminal blobs, so it is orientable (Lemmas 2.69). □

6.2 Connectedness and diameters

In this section, we prove diameter bounds for SPR and NNI spaces. For the proofs, we use that each network can be turned into an orientable network using a small number of SPR or NNI moves, as we proved in the previous section. The bound for SPR moves then follows from a modified version of the rSPR bound, and the NNI bound follows directly from the rNNI bound.

6.2.1 SPR moves

In this section, we will prove an upper bound on the diameter of SPR moves on undirected networks, using results for rSPR moves on directed networks. This linear bound improves on the previously best quadratic bound: $(v_k^n)^2 + 4v_k^n$, where v_k^n denotes the number of nodes in a tier-k network with n leaves [FHMW17].

Recall from Corollary 5.25 that any rSPR sequence transforming a directed network N into another directed network N' provides a corresponding SPR sequence transforming $U(N)$ into $U(N')$.

Moreover, since any orientable network is the underlying graph $U(N)$ of some directed network N, the diameter for rSPR moves on directed networks gives an upper bound for the diameter of SPR moves on the space of undirected networks restricted to orientable networks.

Using this fact, and the techniques from the previous section to get to orientable networks we obtain the following proposition. This proposition gives an upper bound for $\mathrm{diam}_{\mathrm{SPR}}(n,k)$ in terms of $\mathrm{diam}_{\mathrm{rSPR}}(n,k)$.

Proposition 6.11 ([JJE+18] Proposition 4.21)**.** *The diameter of the k-tier of undirected network space has upper bound*

$$\mathrm{diam}_{\mathrm{SPR}}(n,k) \leq \mathrm{diam}_{\mathrm{rSPR}}(n,k) + 2M,$$

where M denotes the maximal SPR distance from any undirected network to an orientable network.

Proof. Let $U, U' \in \mathcal{U}(n,k)$ be two undirected networks. Then we can apply at most M moves to U and at most M moves to U' to get to orientable networks U_o and U'_o (M exists by Lemma 6.6). Choose an orientation for these orientable networks, taking the same leaf as the root in both. Then, by Corollary 5.25, there is a sequence of at most $\mathrm{diam}_{\mathrm{rSPR}}(n,k)$ moves to go from U_o to U'_o. Because all moves are reversible, there is a sequence of moves:

$$U \overset{M}{\longleftrightarrow} U_o \overset{\mathrm{diam}_{\mathrm{rSPR}}(n,k)}{\longleftrightarrow} U'_o \overset{M}{\longleftrightarrow} U',$$

of length at most $\mathrm{diam}_{\mathrm{rSPR}}(n,k) + 2M$ from U to U'. □

This proposition together with Proposition 6.7 and Theorem 5.19 gives a reasonable upper bound on the diameter of $\mathcal{U}(n,k)$, namely $2n + 3k + \frac{2}{3}k$. However, using the (lack of) structure in an undirected network, we can do better. The next theorem again uses the bottom-up isomorphism building technique. This time, we apply it to orientable networks, where we dynamically re-orientate the networks between steps.

Theorem 6.12 ([JJE+18] Theorem 4.22)**.** *The SPR diameter of the k-tier of undirected network space has upper bound*

$$\mathrm{diam}_{\mathrm{SPR}}(n,k) \leq n + \frac{8}{3}k.$$

Proof. Let $U_1, U_2 \in \mathcal{U}(n,k)$ be arbitrary undirected networks. By Proposition 6.7, we can apply a total of $\frac{2}{3}k$ moves to produce orientable networks U_1^o and U_2^o. Now we do induction as in the proof of the rSPR diameter (Theorem 5.19). Choose network orientations $N_1, N_2 \in \mathcal{N}(n-1,k)$ for both orientable networks, i.e., networks such that $U(N_1) = U_1^o$ and $U(N_2) = U_2^o$. As before, we construct the down-closed subsets Y_1 and Y_2 of N_1 and N_2 such that $N_1|_{Y_1}$ and $N_2|_{Y_2}$ are isomorphic. We show that we need at most $n + 2k$ SPR moves in total to reach U_2^o from U_1^o, by inductively increasing the size of Y_1 and Y_2.

Recall that, in the proof of Theorem 5.19, we use Lemma 3.5 to add reticulations and Lemma 3.6 to add tree nodes to the down-closed sets. For reticulations, this takes at most one rSPR move per reticulation, and at most two rSPR moves per tree node. The only case that requires two moves is Case 2c of Lemma 3.6, where all lowest nodes are tree nodes and the nodes in the other network corresponding to their child nodes have tail-immovable incoming edges. We now show how one SPR move suffices to deal with this case when we consider undirected networks.

Suppose we are in the situation of Case 2c. We shortly recall the situation: *Every lowest node of $N_1 \setminus Y_1$ and every lowest node of $N_2 \setminus Y_2$ is a tree node, so there exists a node $u_2 \in N_2 \setminus Y_2$ with children $x_2, y_2 \in Y_2$. Let x_1, y_1 be the nodes in Y corresponding to x_2 and y_2. The nodes x_1 and y_1 do not have a common parent not in Y_1. Let z_1^x and z_1^y be parents of x_1 and y_1 not contained in Y_1. Neither (z_1^x, x_1) nor (z_1^y, y_1) is movable, so N_1 contains triangles $\langle c_1^x, z_1^x, d_1^x\rangle$ and $\langle c_1^y, z_1^y, d_1^y\rangle$ such that $d_1^x \neq x_1$ and $d_1^y \neq y_1$.* We distinguish three subcases:

1. $\boldsymbol{d_1^y \notin Y_1}$**.** Reversing the direction of (z_1^y, d_1^y) gives a valid orientation with the same underlying undirected network, and conserves the down-closedness of Y_1 and the isomorphism between $N_1[Y_1]$ and $N_2[Y_2]$ (Figure 6.3). The resulting directed network has a reticulation node directly above Y_1, so we can add a node to Y_1 and Y_2 with at most one move.

2. $\boldsymbol{d_1^x \notin Y_1}$**.** This case is handled symmetrically to Case 1 by reversing the direction of (z_1^x, d_1^x).

3. $\boldsymbol{d_1^x, d_1^y \in Y_1}$**.** Note that one of (u_2, x_2) and (u_2, y_2) is tail-movable (Lemma 3.3).

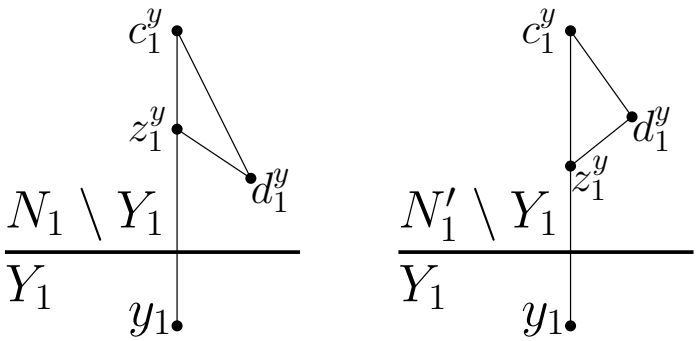


Figure 6.3: The re-orientation of the bottom edge of a triangle in Theorem 6.12 Case 1. All of the nodes of the triangle are above Y_1, so re-orientation does not affect the isomorphism between $N_1[Y_1]$ and $N_2[Y_2]$.

a) **(u_2, x_2) is movable.** Tail moving (u_2, x_2) to an incoming edge of d_2^x, the node in Y_2 corresponding to d_1^x, we create a lowest tree node u_2 with children x_2 and d_2^x in Y_2 (Figure 6.4). We can add z_1^x (with children x_1 and d_1^x) and u_2 (with children x_2 and d_2^x) to Y_1 and Y_2 at the cost of one move.

b) **(u_2, y_2) is movable.** This case is handled symmetrically to the previous case by interchanging the roles of x_1 and y_1.

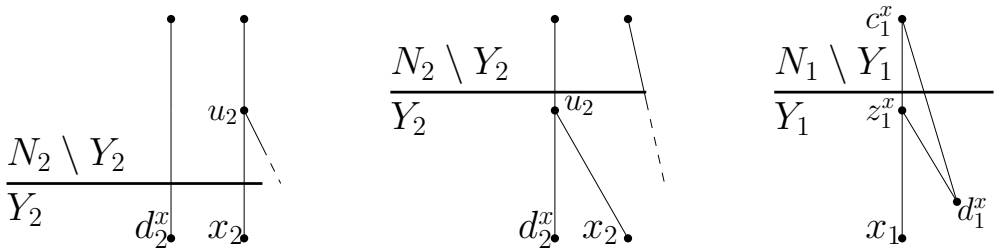


Figure 6.4: The tail move used in the proof of Case 3a in the proof of Theorem 6.12.

This proves that we can always add a node to Y_1 and Y_2 using at most one SPR move. Hence the number of SPR moves needed to transform U_1^o into U_2^o is at most $|V| \leq n + 2k$. Together with the moves needed to get to U_1^o and U_2^o from U_1 and U_2, we get an upper bound of $n + \frac{8}{3}k$ for the number of SPR moves needed to go from U_1 to U_2. □

6.2.2 NNI moves

Here, we prove upper bounds on the diameter of NNI spaces. First, we prove a simple bound using the results from the previous section about SPR moves. Then, we prove a better bound using the bound for rnni spaces from the previous chapter.

For the simple bound, we apply the following lemma, which is a variation on [JK19] Lemma 3.4 where it is shown that each SPR move can be replaced by a sequence of NNI moves. The proof here deviates from the proof in [JK19], mainly because that proof does not work for internally labeled networks, but also because this thesis uses different notation. With this lemma in hand, we can easily prove connectedness of NNI spaces, using the connectedness of SPR spaces.

Lemma 6.13. *Let $\ddot{U}, \ddot{U}' \in \dot{\mathcal{U}}(n,k,m)$ with $d_{\mathrm{SPR}}(\ddot{U}, \ddot{U}') = 1$, then $d_{\mathrm{NNI}}(\ddot{U}, \ddot{U}') \leq 2n + 3k - 2$ NNI moves.*

Proof. Assume that $\ddot{U}$ can be transformed into $\ddot{U}'$ by the move $u \xrightarrow{\{u,v\}} e$. Note that there is then a path $P = (p_0, u, p_1, p_2, \ldots, p_l)$ in $\ddot{U} \setminus \{e\}$ such that $e = \{p_{l-1}, p_l\}$ and $v \notin \{p_0, p_1\}$, since otherwise $\ddot{U}'$ would be disconnected. Like in Lemma 3.9, the idea is now to move u along P to e with NNI moves.

Let $\ddot{U}_i$ be the result of the SPR move $u \xrightarrow{\{u,v\}} \{p_i, p_i + 1\}$ in $\ddot{U}$ for all $i = 1, \ldots, l-1$. Note that such a move is valid—thus $\ddot{U}_i$ exists—iff $v \notin \{p_i, p_i+1\}$. Hence, if $v \notin P$, then each $\ddot{U}_i$ is a valid network. As it is trivial that $d_{\mathrm{NNI}}(\ddot{U}_i, \ddot{U}_{i+1}) \leq 1$ if these networks exist, the statement holds when $v \notin P$.

Now suppose that $v = p_j$, then only the networks $\ddot{U}_{j-1}$ and $\ddot{U}_j$ do not exist, and we need to find an NNI sequence between $\ddot{U}_{j-2}$ and $\ddot{U}_{j+1}$—these must exist because $\ddot{U}_0 = \ddot{U}$ and $\ddot{U}_{l-1} = \ddot{U}'$. The following sequence works for this purpose: $u \xrightarrow{\{p_{j-1},u\}} \{v, p_{j+1}\}$, $u \xrightarrow{\{v,u\}} \{p_{j+1}, p_{j+2}\}$, and $v \xrightarrow{\{u,v\}} \{p_{j-1}, p_{j+1}\}$. It can be easily verified that these moves are valid, except when p_{j-1} is adjacent to p_{j+1}. In that case, however, there exists a path P that does not contain v.

The number of moves in such a sequence is at most $l + 1$. As l is at most the number of edges in $\ddot{U}$, which is $2n + 3k - 3$, we have that $d_{\mathrm{NNI}}(\ddot{U}, \ddot{U}') \leq 2n + 3k - 2$. □

However, using this lemma, we get a diameter of order $O((n+k)^2)$, whereas we can get an asymptotically tight bound for $\mathcal{U}_{\mathrm{NNI}}(n,k)$ using the strategy used for rNNI moves in the previous chapter.

Applying the rNNI diameter proof to NNI moves, we recover the original proof for the NNI diameter from Theorem 13 in [JK19]. In that paper, all networks under consideration were orientable (or, proper, as it was called in that paper). Leveraging the fact that each undirected network can be transformed

into an orientable network with a small number of NNI moves (Proposition 6.7) we can use Theorem 13 from [JK19] or Theorem 5.23 to get the following result.

Theorem 6.14. *The space* $\mathcal{U}_{\mathrm{NNI}}(n,k)$ *is connected and* $\mathrm{diam}_{\mathrm{NNI}}(n,k) \leq 12n + 20k - 15 + 2(k+2)\log(k+2) + n\log n$.

Proof. Let $U, U' \in \mathcal{U}(n,k)$. First, using at most $3k$ NNI moves per network, we transform U and U' into orientable networks U_o and U'_o (Proposition 6.10). By Lemma 2.69, there exist orientations $N, N' \in \mathcal{N}(n-1,k)$ of U_o and U'_o with the same leaf chosen as the root. Applying Theorem 5.23 to these networks, we see that there is a sequence of at most $12n+14k-15+2(k+2)\log(k+2)+n\log n$ rNNI moves between N and N'. Using Lemma 5.24, this gives $d_{\mathrm{NNI}}(U,U') \leq 12n + 20k - 15 + 2(k+2)\log(k+2) + n\log n$. □

Recall the lower bounds from Francis et al. Theorem 2 (5.26), where they gave a lower bound of order $\Omega((n+k)\log(n+k))$ for $\mathrm{diam}_{\mathrm{NNI}}(n,k)$. Note that our upper bound is of order $O((n+k)\log(n+k))$, so we can conclude that $\mathrm{diam}_{\mathrm{NNI}}(n,k) = \Theta((n+k)\log(n+k))$.

6.3 Internal labels

Like in the previous chapters, we will now prove that the spaces of internally labeled networks are also connected by SPR and NNI moves. The proofs follow a similar strategy as for directed networks: we first prove that the internal degree-3 nodes can be permuted, and then we extend this to networks with degree-2 nodes.

6.3.1 Permuting internal nodes

The results for directed networks cannot directly be used for this purpose, because fixing an orientation of the undirected network also fixes a partition of the nodes into tree nodes and reticulations. Doing this for both networks, these partitions might not be compatible (a label attached to a tree node in the first network might be mapped to a reticulation in the other network). This problem cannot be circumvented by simply choosing some internal nodes to become reticulations, because for some choices the network may not be orientable [HvIJ+19].

Hence, we simply reprove that there is a short sequence of moves making any two internal nodes adjacent, and that there is a short sequence of moves swapping the labels of adjacent nodes.

Lemma 6.15. *Let $\dot{U} \in \dot{\mathcal{U}}(n,k)$, and let x and y be two internal nodes of $\dot{U}$. Then there is a network $\dot{U}'$ in which x and y are adjacent, and $d_{\mathrm{SPR}}(\dot{U},\dot{U}') \leq 1$.*

Proof. Note that at least two out of the three edges incident to x are movable with x as the moving endpoint, and that at most one of these edges is a cut-edge that separates x and y. Hence, there is an edge $\{u,x\}$ that is movable with x as the moving endpoint and that is not a cut-edge separating x and y.

As the x-endpoint of $\{u,x\}$ is movable and $\{u,x\}$ does not separate x and y, the SPR move $x \xrightarrow{\{x,u\}} \{y,z\}$ is valid as long as $u \notin \{y,z\}$. If $u = y$, x is adjacent to y in $\dot{U}$, and $\dot{U}' = \dot{U}$ suffices. If $u = z$, then we may simply choose another edge $\{y,z'\}$ incident to y, and apply $x \xrightarrow{\{x,u\}} \{y,z'\}$ to obtain a network $\dot{U}'$ in which x and y are adjacent. □

Lemma 6.16. *Let $\dot{U} \in \dot{\mathcal{U}}(n,k)$, and let x and y be two adjacent internal nodes of $\dot{U}$. Let $\dot{U}'$ be the network obtained by swapping the labels of x and y, then $d_{\mathrm{NNI}}(\dot{U},\dot{U}') \leq 2$.*

Proof. Let a and b be the other neighbours of x, and c and d the other neighbours of y. If $\{a,b\} \cap \{c,d\} \neq \emptyset$, then x and y are part of a triangle, and we can swap two nodes in a triangle with one NNI move. Otherwise, all of a, b, c, and d are distinct, and we may apply $x \xrightarrow{\{a,x\}} \{c,y\}$ and $y \xrightarrow{\{b,y\}} \{a,x\}$ to reach the desired result. □

Theorem 6.17. *For all $n > 1$ and $k \geq 0$, we have* $\mathrm{diam}_{\mathrm{SPR}}(n,k,0) \leq \mathrm{diam}_{\mathrm{SPR}}(n,k) + 4n + 8k - 8$ *and* $\mathrm{diam}_{\mathrm{NNI}}(n,k,0) \leq \mathrm{diam}_{\mathrm{NNI}}(n,k) + (n + 2k - 2)(4n + 6k - 2)$.

Proof. Let $\dot{U}_1, \dot{U}_2 \in \dot{\mathcal{U}}(n,k,0)$ be arbitrary internally labeled networks. Using at most $\mathrm{diam}_{\mathrm{SPR}}(n,k)$ moves, we can transform $\dot{U}_1$ into a network $\dot{U}_1'$ such that $\dot{U}_1' \simeq_{X^l} \dot{U}_2$.

For any permutation of the i internal nodes, we need at most i swaps, so in total, the permutation costs $4i$ SPR moves (Lemmas 6.15 and 6.16). Using that $i = n + 2k - 2$, there is a sequence of at most $\mathrm{diam}_{\mathrm{SPR}}(n,k) + 4n + 8k - 8$ SPR moves from $\dot{U}_1$ to $\dot{U}_2$.

For the NNI bound, use Lemma 6.13 to see that we need at most $2n + 3k - 2$ NNI moves to make two internal nodes adjacent; and note that we can make the networks leaf-isomorphic using at most $\mathrm{diam}_{\mathrm{NNI}}(n,k)$ moves. □

Like for rSPR moves, $\mathrm{diam}_{\mathrm{SPR}}(n,k,0) \geq \mathrm{diam}_{\mathrm{SPR}}(n,k) = \Omega(n+k)$, which implies that we have an asymptotically tight bound $\mathrm{diam}_{\mathrm{SPR}}(n,k,0)\Theta(n+k)$ for internally labeled networks as well.

6.3.2 Degree-2 nodes

Theorem 6.18. *All spaces $\dot{\mathcal{U}}_{\mathrm{NNI}}(n,k,m)$ are connected with diameter bounds $\mathrm{diam}_{\mathrm{SPR}}(n,k,m) \leq \mathrm{diam}_{\mathrm{SPR}}(n,k,0)+4n+8k+2m+1$ and $\mathrm{diam}_{\mathrm{NNI}}(n,k,m) \leq \mathrm{diam}_{\mathrm{NNI}}(n,k,0)+4mn+6mk+m^2+6k+5m+3$, except if $n=2$, $k=0$ and $m>0$, in which case $\dot{\mathcal{U}}_{SPR}(2,0,m)$ is the edgeless graph with $m!$ nodes.*

Proof. Let $\ddot{U},\ddot{U}' \in \dot{\mathcal{U}}(n,k,m)$ be two networks. To find a sequence between these networks, we first make the networks orientable using at most k SPR moves or $3k+m$ NNI moves per network (Proposition 6.7 and 6.10).

Then, using at most $2n+3k-1$ rSPR moves or at most $m(2n+3k-1)$ rNNI moves in the oriented versions of each network, we collect the degree-2 nodes at one leaf which we choose to be the root of the oriented versions (Lemma 5.31). To sort the degree-2 nodes in one of the networks, we need an additional $2m+3$ SPR moves or m^2+5m+3 NNI moves (Lemma 5.31).

Lastly, to make the networks labeled isomorphic w.r.t. the full label set, we use at most $\mathrm{diam}_{\mathrm{SPR}}(n,k,0)$ SPR moves or at most $\mathrm{diam}_{\mathrm{NNI}}(n,k,0)$ NNI moves (Theorem 6.17). The total sequence of moves thus consists of at most $\mathrm{diam}_{\mathrm{SPR}}(n,k,0)+4n+8k+2m+1$ SPR moves or $\mathrm{diam}_{\mathrm{NNI}}(n,k,0)+4mn+6mk+m^2+6k+5m+3$ NNI moves. □

6.4 Conclusion

In this chapter, we have investigated the spaces of undirected networks. First, we considered the relation between directed and undirected networks. This relation turns out to be slightly weaker than assumed in [GvIJ+17b]. We have investigated their line of thought in more detail, and showed that an SPR move cannot generally be replaced by a constant length sequence of rSPR moves in the oriented version of the network. This is unsurprising, as this already holds for trees [AM19]. It is still unclear whether this result holds for NNI moves in networks. For trees, however, we already know that each NNI can be replaced with a single rNNI move on each orientation of the tree (Lemma 2.64). We expect it to be false for networks in light of Example 6.3.

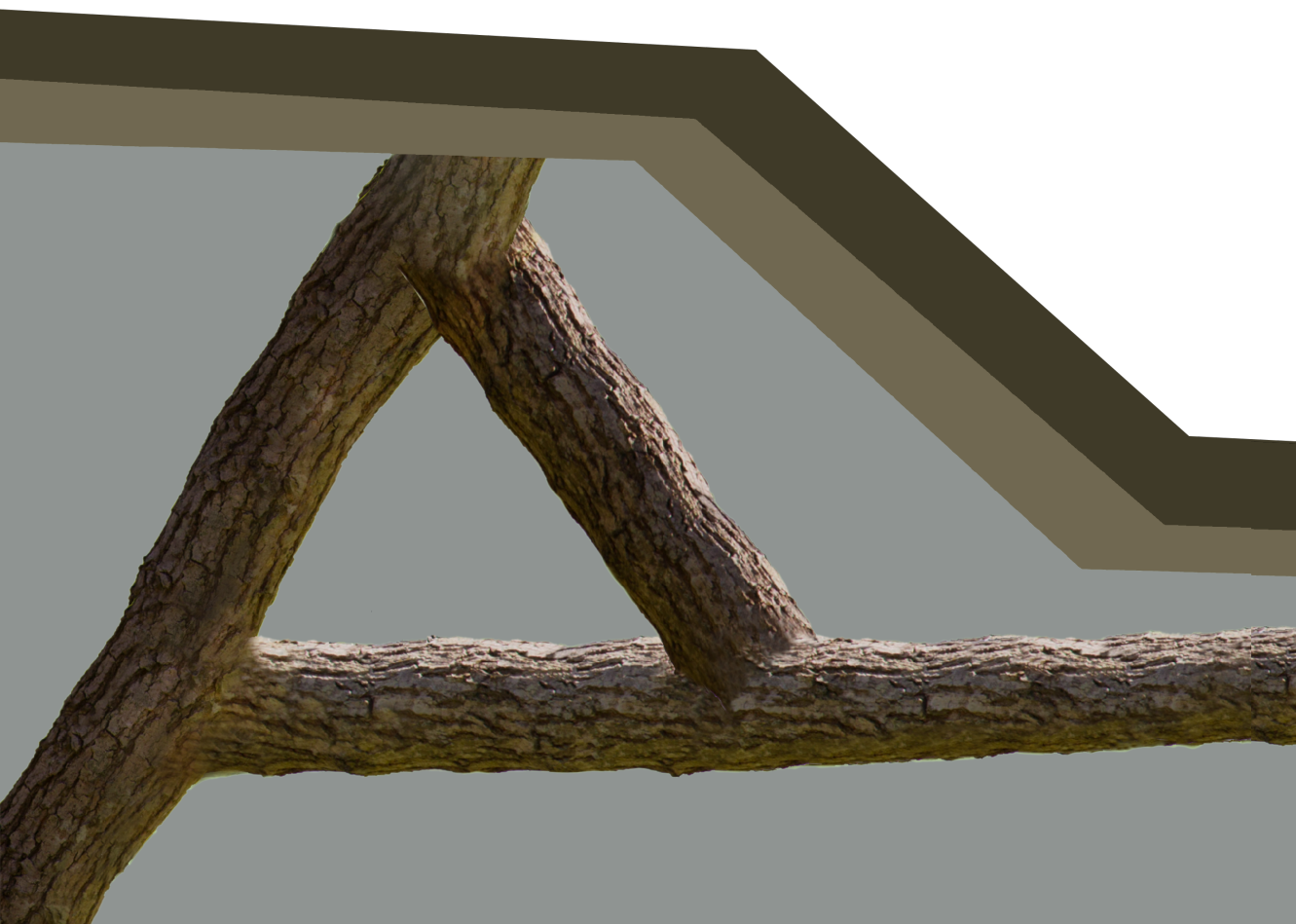
To get good upper bounds for the diameters of SPR and NNI spaces, we therefore cannot directly use the results for rSPR and rNNI moves. To use these results, we have considered orientable networks, and we have shown that each undirected network can be turned into an orientable network using a small number of SPR or NNI moves: k SPR moves if $m>0$, $k/3$ SPR moves if $m=0$, and $3k+m$ NNI moves. This last bound can possibly be improved because for each cut-edge, we need a blob containing at least 5 edges (if $m=0$). By orienting the obtained orientable networks and using the results from the

previous chapter, we obtain upper bounds on the diameter of SPR and NNI spaces. For the SPR diameter, we slightly improve this bound by making small alterations in the use of the rSPR result.

The space of directed networks is in some sense embedded in the space of orientable networks $\mathcal{O}(n,k)$: take the quotient $\mathcal{N}(n,k)/\sim$, where $N \sim N'$ if $U(N) = U(N')$, then $\mathcal{N}(n,k)/\sim$ is a subgraph of $\mathcal{O}(n,k)$. It would be interesting to investigate these graphs in more detail. We could, for example, ask whether these graphs are isomorphic? In other words, for each pair of orientable networks $U, U' \in \mathcal{O}(n,k)$ such that $d_{\mathrm{SPR}}(U,U') = 1$, are there orientations N, N' of these networks such that $d_{\mathrm{rSPR}}(N,N') = 1$?

The extension to internally labeled networks is again quite straightforward. The strategy we used is quite similar to the ones used in the previous chapters: to swap the labels of two nodes, make them adjacent, swap them, and return them to their original positions. For SPR moves, this gives linear upper bounds for the diameters of $\dot{\mathcal{U}}_{\mathrm{SPR}}(n,k,m)$. However, for NNI moves, our strategy reintroduces quadratic terms into the diameter bounds of $\dot{\mathcal{U}}_{\mathrm{NNI}}(n,k,m)$, which we have painstakingly removed for $\mathcal{U}_{\mathrm{NNI}}(n,k)$. It is still an open question whether these quadratic terms are necessary.

7



Computing Sequences

In the previous chapters, we have shown that most spaces of networks are connected, when restricted to spaces corresponding to the moves studied in this thesis. Furthermore, we have bounded the maximal distance between pairs of networks in these spaces. The proofs for these bounds are constructive, so they provide sequences of moves between networks. However, these sequences are unlikely to be as short as possible.

Knowing an exact distance between a pair of networks could be useful for comparing networks. Although there are many other distance measures between networks, rearrangement distances are most closely related to the local search heuristics. Hence, these distances are quite insightful for assessing the quality of local search heuristics.

In this chapter, we investigate the problem of finding short sequences of moves between a pair of networks. We only consider spaces of networks without internal labels. First, we prove that, in most cases, computing the shortest sequence is NP-hard (Section 7.1). Then, we investigate some practical aspects of finding a shortest sequence (Section 7.2.1). Finally, in Section 7.2, we leverage the results from the previous chapters, and investigate several algorithms that follow from the constructive proofs for sequences between networks. These algorithms are implemented and tested for the quality of their solutions.

7.1 Complexity of M Distance

The complexity of M Distance where M is a type of horizontal move is easily determined. In fact, it is NP-hard to determine the distance between two trees for SPR [HDRCB08], rSPR [BS05], NNI [JLTZ00], and rNNI moves.[1] This immediately implies that M Distance is NP-hard for SPR, rSPR, NNI, rNNI, tail, and tail_1 moves.

Since this does not immediately imply that these problems are also NP-hard for networks with reticulations, we modify the question to determining, for all $k \geq 0$, the complexity of M Distance Tier-k, for networks with k reticulations. In Section 7.1.1, we prove that this problem is NP-hard for rSPR and tail moves.

This only leaves the question of the complexity of Head Distance and Head_1 Distance. In Section 7.1.2, we prove that Head Distance is NP-hard when the reticulation number is allowed to vary, as in the original definition of

[1] The NP-hardness of rNNI Distance Trees follows very easily from [JLTZ00] and Lemma 2.64, but, as mentioned in Section 2.5.1, it is unclear whether there is a paper that explicitly states this result.

the problem. The complexity of Head_1 Distance and the fixed-tier versions of these problems are still open.

For all these problems, NP-completeness follows directly from NP-hardness because a certificate can consist of a sequence of moves of the required length plus an isomorphism between the resulting networks. Alternatively, for binary networks as considered in this thesis, one could take a certificate consisting only of the sequence of moves. Indeed, the sequence can be applied to a network in polynomial time, and the resulting networks can be compared in polynomial time as well, because Binary Network Isomorphism, the problem of checking network isomorphism, is polynomial time for binary networks [MR12]. This follows easily from the facts that Directed Graph Isomorphism is polynomial for graphs in which the degrees of the nodes are bounded by a constant [Luk82, KST93]—for the reduction, simply attach the reversed caterpillar with n leaves to the leaf with label n in each network.

Lemma 7.1 ([MR12])**.** *The problem* Binary Network Isomorphism *can be solved in polynomial time.*

7.1.1 M Distance Tier-k

In this subsection we prove that computation of the rSPR and tail distance between two networks in a fixed tier is NP-complete. To prove this, we use that the set of embedded trees of a network cannot change much after a move of any of these types. As such, this proof is different from the original proof in [JJE+18], where agreement forests are used.

Lemma 7.2. *Let N and N' be networks such that $d_{\text{rSPR}}(N, N') \leq 1$. Then for each $T \in \mathcal{T}(N)$ there is a tree $T' \in \mathcal{T}(N')$ such that $d_{\text{rSPR}}(T, T') \leq 1$.*

Proof. The statement holds for tail moves and head moves by Lemmas 3.13 and 4.3. As each rSPR move is either a tail move or a head move, the statement also holds for rSPR moves. □

Theorem 7.3 ([JJE+18] Theorem 4.1)**.** *When M is rSPR or tail, the problem* M Distance Tier-k *is NP-complete for all $k \geq 0$.*

Proof. We prove this using a reduction from rSPR Distance Tier-0. Let (T, T') be an instance of rSPR Distance Tier-0 with $T, T' \in \mathcal{N}(n, 0)$. Then, in polynomial time, we can construct the instance (N, N') of M Distance Tier-k consisting of the two ladder trees $N = H_k(T^+, c(\rho), n+1)$ and $N' = H_k(T'^+, c(\rho), n+1)$ in $\mathcal{N}(n, k)$.

Note that a sequence of moves between T and T' gives a sequence of moves between N and N', as T and T' are pendant trees of N and N'. This gives the inequality $d_M(N, N') \leq d(T, T')$.

For the other direction, we observe that each rSPR or tail move in a network changes each embedded tree of the network by at most one move (Lemma 7.2). Hence, for any $t \in \mathcal{T}(N)$ and $t' \in \mathcal{T}(N')$, a sequence of length l from N to N' gives a sequence of length at most l from t to t'. In particular, we have $d_M(N, N') \geq d(T^+, T'^+)$.

Because T and T' are pendant trees of T^+ and T'^+, we also have $d(T^+, T'^+) \geq d(T, T')$ and we can conclude that $d_M(N, N') \geq d(T, T')$ and thus $d_M(N, N') = d(T, T')$. Finally, because N and N' can be constructed from T and T' in polynomial time, any instance of rSPR Distance Tier-0 can be transformed into an equivalent instance of rSPR distance Tier-k in polynomial time. □

One would hope that an analogous proof works for SPR moves. However, we will show using a counterexample for the analogue of Lemma 7.2 that it does not.

Example 7.4. Let U and U' be the networks in Figure 7.1, and T the embedded tree of U shown in bold in the same figure. Notice that U' is one SPR move away from U, and that all trees in $\mathcal{T}(U')$ are at least two moves away from T. Indeed, each embedded tree of U' contains the cut-edge $\{u, v\}$, which separates the tree into to trees on $\{a_1, b_1, c_1, a_2, b_2, c_2\}$ and $\{a_3, b_3, c_3, a_4, b_4, c_4\}$. All options for these pendant trees are not isomorphic to the restrictions of T to these leaves. Hence, to reach T from an embedded tree of U', we need at least one move in each of these pendant trees.

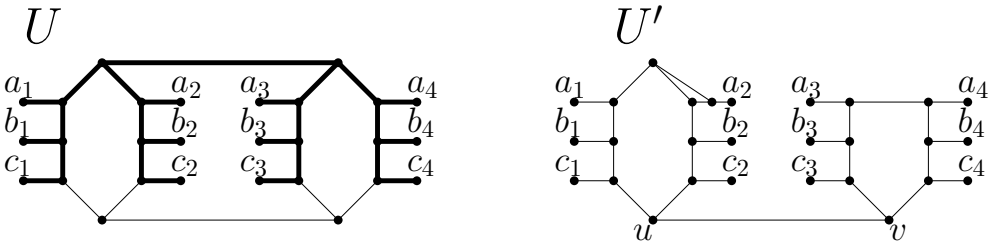


Figure 7.1: The networks of Example 7.4. The networks are one SPR move apart, but the thick tree in the left networks is not at distance at most one from any embedded tree of the right network.

For a similar reason, the same proof strategy also fails for local moves: A local move on a network can introduce embedded trees that are further than

one local move away from a previously embedded tree. See Proposition 6.3.1 of [Mar20] for an analysis of some cases where this cannot happen.

Example 7.5. Let N be the network obtained from a network with two leaves and one reticulation by adding three leaves to the bottom arc of the triangle. Let N' be the network obtained from N by the tail_1 (or head_1) move that creates a cherry $(1, 2)$ below the bottom node of the triangle. Let T be the embedded tree of N in which the leaves 1 and 2 are at large distance from each other (bold part of N in Figure 7.2). As 1 and 2 are far apart in T, they will still be far apart in any tree one rNNI move away from T. In particular, such a tree cannot contain the cherry on 1 and 2.

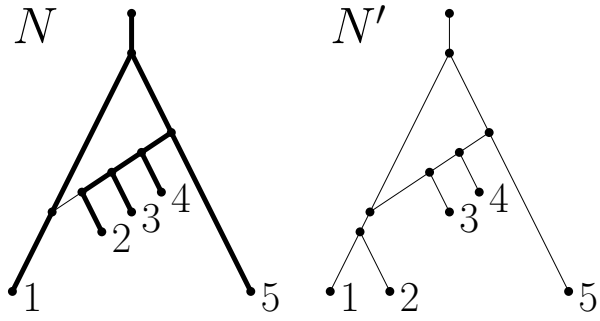


Figure 7.2: The networks of Example 7.5. The networks are one SPR move apart, but the thick tree T in the left networks is more than one rNNI move away from each embedded tree of the right network.

Although this proof strategy fails for local moves and SPR moves directly, it does not mean that the theorem is false. In fact, it seems feasible that the M-distances between the networks and the trees in Theorem 7.3 have the same relation when considering local moves or SPR moves. The main reason for believing this is that it is unlikely that the blob—which already has the right shape—can help in rearranging the pendant trees. For SPR moves, this can probably be proven by applying agreement forest techniques directly to networks in which only the pendant trees are different.

To enforce this distance relation for local moves, a more suitable technique would be to separate the blobs from the trees by a chain of length n^2, where n is the number of leaves in the original trees. It should then be pointless to rearrange the blob at all, as the only differences between the networks are in the trees and the trees can be rearranged using at most $O(n \log n)$ moves. We will not attempt to prove this in this thesis.

7.1.2 Head Distance

In this section, we prove that the problem Head Distance of computing the head move distance between two networks is NP-complete. The proof uses a reduction from rSPR Distance Trees, which is the problem of finding the rSPR distance between two rooted trees. The rough idea is to convert rSPR moves on trees into head moves on specifically constructed networks.

Because rSPR moves change the location of the tail and not the head of an arc, we use the following trick: we turn the tree upside down, which turns each tail into a head, and hence a tail move into a head move. Just reversing the direction of the arcs of the tree is not sufficient, as this gives a graph with multiple roots and one leaf. Hence, we connect all these roots and add a second leaf to create a phylogenetic network. To distinguish the leaves of the original tree in this network, we finally attach chains of leaves. This construction is formalized in the following definitions.

After these definitions, we will show that the minimal number of head moves between two upside down trees is equal to the number of rSPR moves between the two original trees. This proof uses the concepts of agreement forests.

Definition 7.6. Let T be a phylogenetic tree with labels $X = \{x_1, \ldots, x_n\}$, the *upside down version* of T is a network $\perp$ with $2n^2+2$ leaves ($e_{x,i}$ for $x \in X$ and $i \in [2n]$, y, and ρ) constructed by:

1. Creating the labelled digraph S, which is T with all the arcs reversed;
2. Creating the tree D by taking $C(X \cup \{y\})$ and adding $2n$ pendant arcs with leaves labelled $e_{x,1}, \ldots, e_{x,2n}$ to each pendant arc $e = (\cdot, x)$ of $C(X)$;
3. Taking the disjoint union of D and S;
4. Identifying the node labelled x_i in D with the node labelled x_i in S and subsequently suppressing this node for all i.

The *bottom part* of $\perp$ is the subgraph of $\perp$ below (and including) the parents of the $e_{x,1}$ (Figure 7.3).

The rSPR distance between two trees can be characterized alternatively as the size of an agreement forest [BS05]. Here, we use this alternative description as part of the reduction. To define agreement forests, we need the following definitions, which we have generalized slightly to work for networks.

Definition 7.7. Let G and G' be digraphs labeled by a set X. If $S(G) \simeq_X S(G')$ (i.e., G and G' are labelled isomorphic after suppression of all their

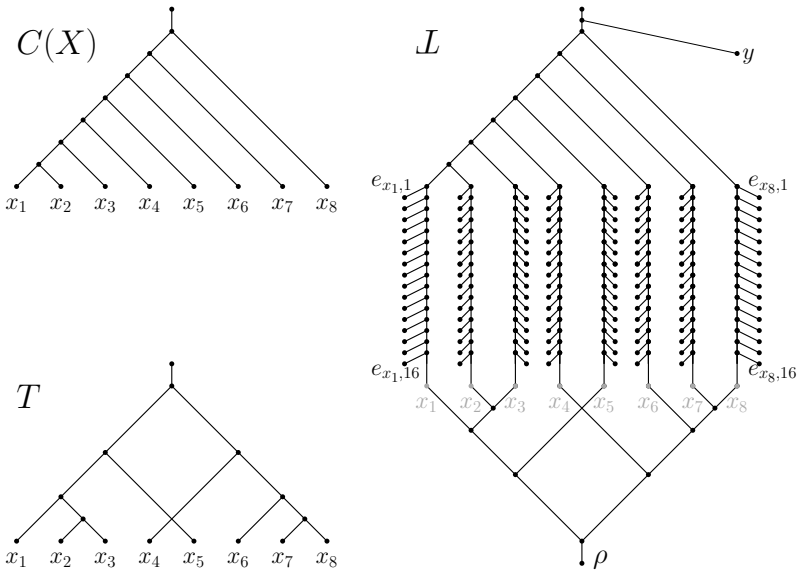


Figure 7.3: Left, the caterpillar $C(X)$ and the tree T, right, the upside down version of T. In the upside down version of T, the original leaves x_i (grey) are suppressed.

degree-2 nodes) then we write $G \equiv G'$, or say G is *s-isomorphic* to G' (for suppressed isomorphic).

An *s-embedding* of a graph H in G is an *s-isomorphism* $H \equiv S$ of H with a subgraph S of G. We say that H can be s-embedded in G if an s-embedding of H in G exists. Note that any subgraph H of G can be embedded in G as $H \equiv H$.

Now we look at an important property of s-embeddings relating to subgraphs, which implies that being s-embeddable is transitive.

Lemma 7.8 ([Jan21] Lemma 24)**.** *Let A, B and H be digraphs with all degree-1 nodes labelled. Suppose $A \equiv B$ and H is a subgraph of A, then H can be s-embedded in B.*

Proof. The s-isomorphism $A \equiv B$ is an isomorphism of graphs (topological minors) without degree-2 nodes. This isomorphism is a bijection between the non-degree-2 nodes of A to the non-degree-2 nodes of B. The map of the arcs is a map of paths of A to paths of B, where the internal nodes of these paths may only be degree-2 nodes. Now consider the subgraph H of A, and note that the non-degree-2 nodes of H are non-degree-2 nodes of A as well. Indeed, the only

way to create new non-degree-2 leaves by taking a subgraph, is to create a leaf from a degree-2 node, but $L(H) \subseteq L(A) = L(B)$, so each degree-1 node of H corresponds to a degree-1 node of A and of B. This means each non-degree-2 node of H corresponds to a non-degree-2 node of B, and each arc of H to a path between such nodes in B, and there is an s-isomorphism of H with the subgraph of B formed by these nodes and arcs. □

Now we turn to the definition of an agreement forest, which, as mentioned earlier, characterizes the rSPR distance for trees. Following the definition of the agreement forest, we define a tool similar to an agreement forest tailored to upside down versions of trees. This upside down agreement forest (udAF) can be turned into an agreement forest of the two original trees.

Definition 7.9. Let T_1 and T_2 be phylogenetic trees with labels X and root ρ. Then a partition $\mathcal{P} = \{P_i\}$ of $X \cup \{\rho\}$ is an *agreement forest (AF)* for T_1 and T_2 if the following hold:

- $T_1|_{P_i} \equiv T_2|_{P_i}$ for all i;
- $T_t|_{P_i}$ and $T_t|_{P_j}$ are node-disjoint for all pairs i, j with $i \neq j$ and fixed $t \in \{1, 2\}$.

Definition 7.10. Let T be a tree with label set X, and let $\perp\!\!\!\perp$ be the upside down version of T. Then an *upside down agreement forest (udAF)* for $\perp\!\!\!\perp$ is a directed graph F such that:

- The underlying undirected graph of F is an (undirected) forest;
- F is a leaf-labelled graph with label set $\{e_{x,i} : x \in X, i \in [2n]\} \cup \{\rho\}$, where each label appears at most once;
- $F \equiv S$ for some subgraph S of the bottom part of $\perp\!\!\!\perp$.

Note that the third requirement implies the first (Figure 7.4).

Lemma 7.11 ([Jan21] Lemma 25)**.** *Let T and T' be phylogenetic trees with label set X. If F is an udAF for $\perp\!\!\!\perp$ and for $\perp\!\!\!\perp'$, then there exists an AF of T and T' of size at most $|F|$, where the size $|F|$ denotes the number of components of F.*

Proof. Let $\mathcal{K}$ be the set of components of F. For each $K \in \mathcal{K}$ we define the following part of the agreement forest:

$$K_{AF} := \{x \in X | e_{x,i} \in K \ \forall i \in [1, 2n]\} \cup (K \cap \{\rho\}),$$

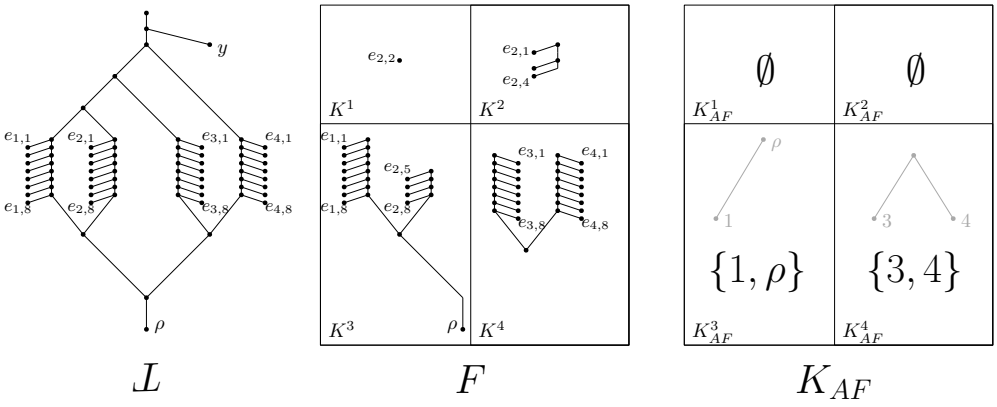


Figure 7.4: An example of a udAF F of the ud-version of the balanced tree on four leaves T. The udAF F consists of four components $K^1, \ldots, K^4$. If F is a udAF for the ud-versions of T and another tree T', then the non-empty K^i_{AF} are parts of a AF for T and T' by Lemma 7.11.

where $e_{x,i}$ indicates the i-th leaf of $\mathcal{L}$ corresponding to x. The agreement forest consists of these parts (ignoring the empty ones, resulting from components that have no complete sets of leaves), together with one part for each leaf that is in none of these parts, that is

$$\begin{aligned} AF := &\big\{Y \subseteq X \cup \{\rho\} | \exists K \in \mathcal{K} \text{ s.t. } Y = K_{AF}\big\} \\ &\cup \big\{\{x\} \subset X \cup \{\rho\} | \forall K \in \mathcal{K} : x \notin K_{AF}\big\} \setminus \{\emptyset\}. \end{aligned}$$

Note that each component Y of AF corresponds either uniquely to a component K of F which has all $e_{x,i}$ for some leaf x, or it corresponds to a leaf x for which not all $e_{x,i}$ are contained in one component of F. In the last case, for this x, there exists a component of F containing only leaves $e_{x,i}$, where x is fixed and $i \in [2n]$. Note that this correspondence $AF \to \mathcal{K}$ must therefore be injective, and AF has size at most $|F|$. What remains to prove is that AF is indeed an agreement forest for T and T'.

Let F' be the subgraph of F where each component K is restricted to the subgraph consisting of all paths between the leaves in K_{AF}. As (per definition of an udAF) F can be s-embedded in the bottom part of $\mathcal{L}$, F' can also be s-embedded in the bottom part of $\mathcal{L}$ as it is a subgraph of F (Lemma 7.8). This s-embedding must be unique, because it is of a labelled forest into a labelled tree.

Let E_x be the subgraph of $\mathcal{L}$ induced by the leaves $e_{x,i}$ and their parents for all i and a fixed x. Now replace each subgraph E_x with one leaf x in

both F' and in the bottom part of $\perp\!\!\!\perp$. Let the resulting graphs be F^s and B^s. Subsequently reverse the direction of each arc in both B^s and in F^s with resulting graphs B^r and F^r. Note that the resulting graphs are $B^r = T$ and the union $F^r = \cup_{K \in \mathcal{K}} T|_{K_{AF}}$, and all the restricted trees $T|_{K_{AF}}$ are node disjoint.

We repeat this argument for $\perp\!\!\!\perp'$, and note that the modifications from F to F^r are independent of $\perp\!\!\!\perp$, so we have the equality $F^r = \cup_{K \in \mathcal{K}} T'|_{K_{AF}}$, where the parts $T'|_{K_{AF}}$ are again node disjoint. This means $T|_{K_{AF}} \equiv T'|_{K_{AF}}$ for each $K \in AF$ corresponding to a non-trivial component of F, and $T|_{P_i}$ and $T|_{P_j}$ are node disjoint for all nontrivial parts P_i and P_j of AF (similarly for T'). Hence, so far, the elements of AF corresponding to non-trivial components of F, meet all the requirements of an AF.

The only other elements of AF contain only one label, each of which is not in any of the non-trivial components of AF. Hence, for any such label x, the restriction $T|_{\{x\}}$ consists of only the node labelled x, which is not contained in any other component by definition (and similarly for T'). Furthermore, the s-isomorphism $T|_{\{x\}} \equiv T'|_{\{x\}}$ is trivial. Hence, AF is indeed an agreement forest. □

The preceding lemma shows that an udAF for two upside down trees gives an AF for the original trees of the same size. We still lack a connection between the number of head moves and an udAF, however. The following lemma shows that appropriate head move sequences correspond to udAFs of size related to the number of head moves.

Lemma 7.12 ([Jan21] Lemma 26)**.** *Let T and T' be trees with label set X, and $|X| = n$. Suppose S is a sequence of head moves $\perp\!\!\!\perp = N_0, \ldots, N_{|S|} = \perp\!\!\!\perp'$ of length $|S| < 2n$. Then there is an udAF F of $\perp\!\!\!\perp$ and $\perp\!\!\!\perp'$ with at most $|S| + 1$ components.*

Proof. Let B be the bottom part of $\perp\!\!\!\perp$. We prove this result using induction on the number of moves to prove that there exist subgraphs F_i of N_i which can be s-embedded in the bottom part of $\perp\!\!\!\perp$ and have $|F_i| \leq i$ components. Finally we prove the subgraph $F_{|S|}$ of $N_{|S|} = \perp\!\!\!\perp'$ must actually be a subgraph of the bottom part of $\perp\!\!\!\perp'$.

As a base of the induction, set $F_0 = B$, which is connected and can clearly be s-embedded in itself and is a subgraph of $\perp\!\!\!\perp$.

Now suppose we have subgraphs F_i of N_i with s-embeddings of F_i in B and $|F_i| \leq i$ for all $i < j \leq |S|$. We prove that there also exists a subgraph F_j of N_j with at most j components that can be s-embedded in B.

Note that F_{j-1} is a subgraph of N_{j-1} and therefore the moving arc $e_j = (u, v)$ can be either an arc of F_{j-1}, or it is in the complement $N_{j-1} \setminus F_{j-1}$.

In the last case e_j can have only its endpoints in F_{j-1}. Now construct F_j as follows:

- remove arc $e_j = (u, v)$ from F_{j-1} if it was contained in it;
- clean up the resulting graph by removing all arcs not contained in any undirected path between two leaves, and suppressing v if it is a degree 2 vertex after removal of (u, v).
- add the new endpoint if necessary. That is, let the target arc of the move be t, if t is contained in the graph after cleaning up, subdivide t.

Note that F_j can be s-embedded in F_{j-1} because the only operations were: restriction to a subset of labels, subdivision, and suppression (Lemma 7.8).

Because F_{j-1} s-embeds in B, there is also an s-embedding of F_j into B. Furthermore, F_j is a subgraph of N_j by construction: the three steps correspond exactly to the pruning and reattaching steps of a head move in N_{j-1}. Lastly, F_j has at most one more component than F_{j-1}, because the only operation that can increase the number of components is the removal of the arc in the first step. Furthermore, because that is an arc removal in a graph, it creates at most one extra component.

We conclude that the desired subgraphs F_i of N_i exist for all $i \in [|S|]$.

Note that we have not yet proven that $F := F_{|S|}$ is an udAF for $\mathbb{L}'$, as F might not s-embed in the bottom part of $\mathbb{L}'$. We now prove that F is in fact a subgraph of the bottom part of $\mathbb{L}'$.

By construction, F is a directed subgraph of $\mathbb{L}'$. Suppose (for a contradiction) that F is not a subgraph of the bottom part of $\mathbb{L}'$, i.e., some part of F lies in the top part of $\mathbb{L}'$. This means that there is a node t of F that corresponds to a tree node (which we also call t) in the upper part of $\mathbb{L}'$. A tree node of F necessarily has two children c_1 and c_2, as F s-embeds in the bottom part of $\mathbb{L}$. One of these children (w.l.o.g. c_1) must have a unique leaf descendant $e_{x,i}$. The other child (c_2) either has a leaf descendant $e_{x,j}$—with the same x as the descendant of c_1—or the next non-degree-2 descendant is a reticulation node.

If c_2 has a leaf descendant $e_{x,j}$, we note the following: t is mapped to a tree node in the top part of $\mathbb{L}'$. Hence, the leaves below the one child of t and the leaves below the other child of t can never correspond to the same $x \in X$: indeed if $e_{y,i}$ is below c_1, then $e_{y,j}$ is also below c_1, and similarly for c_2; furthermore, as the only reticulations of $\mathbb{L}'$ are in the lower part of the network after the $e_{\cdot,\cdot}$ split off, the leaves below c_1 and c_2 are disjoint (except for the leaf corresponding to the root of T'). Hence, as c_1 has a descendant $e_{x,i}$, and c_2 has a descendant $e_{x,j}$, we have a contradiction.

Now, if c_2's first non-degree-2 descendant d is a reticulation node, then this node maps to a reticulation in the bottom part of L'. This means the arc (t, d) maps to a path from the top part of L' to a reticulation in the bottom part of L'. Such a path must necessarily contain the all parents of the leaves $e_{x,i}$ for some $x \in X$. As the s-embeddings of all components in F are node disjoint, and each leaf $e_{.,.}$ is a node of F, each leaf $e_{x,i}$ (fixed x, for all $i \in [2n]$) is its own component in F. Hence F has at least $2n+1$ components, implying that $|S| \geq 2n$, which gives us a contradiction with the assumptions of the lemma.

We conclude that there exists an s-embedding of F in the bottom part of L, and F is an udAF for L' with at most $|S|+1$ components, as F has at most $|S|+1$ components. □

Finally, we put everything together in the following lemma and theorem: each candidate head move sequence defines an udAF, which in turn gives an AF for the original trees, which bounds the rSPR distance between these trees.

Lemma 7.13 ([Jan21] Lemma 27)**.** *Let T_1 and T_2 be trees with a common label set, then*

$$d_{\mathrm{rSPR}}(T_1, T_2) = d_{\mathrm{head}}(\mathrm{L}_1, \mathrm{L}_2).$$

Proof. The inequality $d_{\mathrm{rSPR}}(T_1, T_2) \geq d_{\mathrm{head}}(\mathrm{L}_1, \mathrm{L}_2)$ is obvious, as the rSPR sequence for the trees directly translates into a head move sequence for the upside down trees.

We now prove the other inequality. As $2n > d_{\mathrm{rSPR}}(T_1, T_2) \geq d_{\mathrm{head}}(\mathrm{L}_1, \mathrm{L}_2)$ we only have to consider head move sequences of length less than $2n$. Suppose we have a sequence of head moves S between L_1 and L_2 of length less than $2n$, then there exists a udAF of size at most $|S|+1$ for L_1 and L_2 (Lemma 7.12). Now Lemma 7.11 tells us that there is an AF for T and T' of size at most $|S|+1$. Using the fact that the size of the MAF of T and T' minus one is equal to the rSPR distance between T and T' [BS05], we get the following inequalities:

$$|S| \geq |AF| - 1 \geq d_{\mathrm{rSPR}}(T_1, T_2).$$

We conclude that $d_{\mathrm{rSPR}}(T_1, T_2) = d_{\mathrm{head}}(\mathrm{L}_1, \mathrm{L}_2)$. □

Theorem 7.14 ([Jan21] Theorem 5)**.** *The problem* Head Distance *is NP-complete.*

Proof. This is a direct consequence of the previous lemma, as computing the rSPR distance between two trees is NP-complete [BS05]. □

7.2 Algorithms

In this section, we present pseudocode of algorithms for computing (bounds for) rearrangement distances. The exact algorithm is simply a breadth first search. The algorithms for the upper bounds are based on the isomorphism building proofs presented in the previous chapters. For rSPR and tail moves, we use the bottom-up isomorphism strategy (Theorems 3.8 and 5.19), and, for head moves, we use the top-down isomorphism construction (Theorem 4.23).

7.2.1 Exact algorithm

In this section, we give a simple breadth first search algorithm. This trivially shows that the distance is computable, albeit with a very bad running time. At the end of this chapter, we will discuss several options for improving this running time, either by making algorithmic changes, or by using some properties of the M-distance related to the structure of the networks.

If the breadth first search is implemented as it is most often defined—with a queue—the queue will naturally grow rather long. This is because the neighbourhood of each network is quite large. Furthermore, normally, a list of visited nodes (networks) is kept to prevent searching in cycles. This would also take quite a bit of memory. Moreover, checking whether a network is already visited might not be very straightforward, as the candidate network has to be compared to each network that has already been visited. Although this NETWORK ISOMORPHISM problem can be solved in polynomial time (for binary networks), this still takes a lot of time.

Algorithm 1: ITERATEDDFS(N, N')

```
   Data: Two binary networks N and N' in the same space S.
   Result: An M sequence from N to N'
 1 Set i = 0;
 2 Let D be an upper bound for the largest diameter among the components of S;
 3 while i ≤ D do
 4     Let s be a stack containing one element, the empty sequence of moves;
 5     while |s| > 0 do
 6         Take a sequence of moves S off the top of the stack;
 7         Let N_S be the result of performing S on N;
 8         if |S| = i and N_S ≃_X N' then
 9             return S;
10         if |S| < i then
11             for each valid M move t in N_S do
12                 Add S ∘ t to the top of the stack;
13     Set i = i + 1;
14 return FALSE;
```

Hence, the breadth first search is implemented as an iterated depth first search (DFS) with increasing maximal depth (Algorithm 1). In this implementation, we do not keep a list of visited networks. This makes to possible to visit a network multiple times, but, of course, such a sequence will never be optimal.

Theorem 7.15. *Let M be a move type, and N and N' two networks in the same space of (subdivided internally labeled) networks. If N and N' are in the same component, then* BfsDistanceM(N, N') *returns a minimal length M-sequence between N and N'; otherwise, it returns* False.

Proof. The algorithm halts, because each line in the algorithm runs in polynomial time, the largest diameter D among all components of $\mathcal{S}$ is finite (the number of networks in $\mathcal{S}$ is finite), and the number of sequences of length at most D is finite as well. An upper bound D can simply be found by looking up the appropriate upper bound in the corresponding theorem in this thesis if it concerns one of the moves studied here, otherwise, simply take an upper bound for $|\mathcal{S}|$ as D. Correctness of the algorithm follows from the fact that all possible sequences of moves of sufficient length are tried. □

The running time of this algorithm depends on the space and the type of move. In general we get a running time of $O(\text{poly}(n,k)|\text{Nbh}|^D)$, where |Nbh| is the maximal number of neighbours of a network in the space of networks. For rSPR moves without internal labels, for example, we may take $D = 2n+3k-2$ (Theorem 5.19) and the number of valid rSPR moves in any network in $\mathcal{S}$ can be bounded by $2(2n+3k-1)^2$ (moving each endpoint of each arc to each arc in the network). The running time can then be bounded by $O(\text{poly}(n,k)(\sqrt{2}(2n+3k-1))^{4n+6k-4})$. We will not study this running time in more detail for two reasons. First, we cannot calculate the running time very well because it depends on the size of the neighbourhood and on the diameters of the spaces. For both of these, we have some asymptotic bounds, but they aren't all asymptotically tight. The second reason is that the algorithm is so slow in practice (Section 7.3.4) that it is not very interesting to know the theoretical running time.

7.2.2 Upper bound: rSPR distance

We now give a heuristic for finding an rSPR sequence between any pair of networks. The structure of this heuristic is taken from the proof of Theorem 5.19. Each of the following subroutines corresponds to a lemma used to prove this theorem, starting with Algorithm 2 which is modeled after Lemma 5.17.

Algorithm 2: LowestRetic-rSPR(N, N', Y, Y', ϕ, u')

Data: Two binary tier-k networks N and N' with down-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq}_X N[Y']$ and a lowest node u' in $N \setminus Y$ that is a reticulation.
Result: Sequences of moves S, S' and a node u of N such that after applying S to N and S' to N' we get $N[Y \cup \{u\}] \overset{\phi'}{\simeq} N'[Y' \cup \{u'\}]$, where ϕ' is ϕ extended with $\phi'(u) = u'$.

Set S and S' to be empty sequences of moves.
Let x' be the child of u' and $x = \phi^{-1}(x)$;
Let z be a (random) parent of x not in Y;
if *z is a reticulation* **then**
    Set $u = z$;
else
    Let v be a (random) reticulation in $N \setminus Y$;
    if *N has an arc (u, v) such that $v \xrightarrow{(u,v)} (z, x)$ is valid* **then**
        Add $v \xrightarrow{(u,v)} (z, x)$ to S;
    Set $u = v$;
return S, S', u;

The algorithm follows the proof structure of Lemma 5.17, so correctness of the algorithm follows almost immediately from the proof of this lemma, save for the following small adjustment. The algorithm only checks one parent of x for being a reticulation. If the other parent of x is a reticulation in $N \setminus Y$, this node could be added to the isomorphism. However, the algorithm only catches this if there is a triangle $\langle z, v, x \rangle$. To keep the correspondence between the algorithm and the proof of Lemma 5.17, we have not implemented this possible improvement.

Lemma 7.16. *Let $N, N' \in \mathcal{N}(n, k)$ be networks with down-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq}_X N'[Y']$ and u' a lowest node of $N' \setminus Y'$ that is a reticulation. Moreover, let S, S', u be the output of* LowestRetic-rSPR(N, N', Y, Y', ϕ, u')*, and M and M' the networks obtained by applying S to N and S' to N'.*

Then $|S| + |S'| \leq 1$, $Y \cup \{u\}$ and $Y' \cup \{u'\}$ are down-closed sets in M resp. M', and $M[Y \cup \{u\}] \overset{\phi'}{\simeq}_X M'[Y' \cup \{u'\}]$, where ϕ' is ϕ extended with $\phi(u) = u'$. In other words, Algorithm 2 is correct.

The correctness of Algorithm 3 follows directly from Lemma 3.6 when all lowest nodes above the down-closed sets are tree nodes. However, we claim this algorithm is applicable in more general cases as well, but not in all cases, it does return False sometimes. There is always a lowest node for which this algorithm does not return False, so if the algorithm does return False, we may simply pick a new lowest node and try again.

Algorithm 3: LowestTree-Tail(N, N', Y, Y', ϕ, u')

Data: Two binary tier-k networks N and N' with down-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq}_X N[Y']$ and a lowest node u' in $N \setminus Y$ that is a tree node.
Result: Sequences of tail moves S, S' and a node u of N such that after applying S to N and S' to N' we get $N[Y \cup \{u\}] \overset{\phi'}{\simeq} N'[Y' \cup \{u'\}]$, where $\phi(u) = u'$, or False.

```
1  Set S and S' to be empty sequences of moves.
2  Let x', y' be the children of u';
3  Set x = φ^{-1}(x') and y = φ^{-1}(y');
4  if x and y have a common parent in N \ Y then
5  |  Set u to be a (random) common parent of x and y in N \ Y;
6  else
7  |  Let z_x be a (random) parent of x not in Y;
8  |  Let z_y be a (random) parent of y not in Y;
9  |  if (z_x, x) is tail-movable then
10 |  |  Add z_x --(z_x,x)--> (z_y, y) to S;
11 |  |  Set u = z_x;
12 |  else if (z_y, y) is tail-movable then
13 |  |  Add z_y --(z_y,y)--> (z_x, x) to S;
14 |  |  Set u = z_y;
15 |  else if Both z_x and z_y are tree nodes then
16 |  |  Let ⟨c_x, z_x, d_x⟩ be the triangle with d_x ≠ x;
17 |  |  Let ⟨c_y, z_y, d_y⟩ be the triangle with d_y ≠ y;
18 |  |  if c_x is the child of the root then
19 |  |  |  Swap the roles of x and y;
20 |  |  Let b_x be the parent of c_x;
21 |  |  Let a_x be a (random) parent of b_x;
22 |  |  Add c_x --(c_x,d_x)--> (a_x, b_x) and z_x --(z_x,x)--> (z_y, y) to S;
23 |  |  Set u = z_x;
24 |  else
25 |  |  return False;
26 return S, S', u;
```

Lemma 7.17. *Let $N, N' \in \mathcal{N}(n,k)$ be networks with down-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq}_X N'[Y']$ and u' a lowest node of $N' \setminus Y'$ that is a tree node. Moreover, let O be the output of* LowestTree-Tail(N, N', Y, Y', ϕ, u').

*If $O =$*False*, then N has a lowest node above Y that is a reticulation. Otherwise, the output O is a triple S, S', u. Let M and M' be the networks obtained by applying S to N and S' to N'.*

Then $|S| + |S'| \leq 2$, $Y \cup \{u\}$ and $Y' \cup \{u'\}$ are down-closed sets in M resp. M', and $M[Y \cup \{u\}] \overset{\phi'}{\simeq}_X M'[Y' \cup \{u'\}]$, where ϕ' is ϕ extended with $\phi(u) = u'$. In other words, Algorithm 3 is correct.

Proof. First note that for the algorithm to return False, it has to reach Line 25, which it only does if the condition in Line 15 is not satisfied. In that case, at least one of z_x and z_y is a reticulation. Note that this also implies one of these nodes is a lowest node of $N \setminus Y$. Hence, the algorithm can only return False if N has a lowest node above Y that is a reticulation.

The correctness in all other cases follows almost immediately from the proof of Lemma 3.6. We only need to be careful in Lines 7–14. In the proof of

Lemma 3.6, these cases assume that all lowest nodes above Y are tree nodes. Hence, we have to prove these lines give the desired outcome, even if this is not true.

First suppose that x and y have no common parent, and let z_x and z_y be parents of these nodes not in Y. If (z_x, x) is tail-movable, then $z_x \xrightarrow{(z_x,x)} (z_y, y)$ is valid, even if z_y is not a tree node. Indeed, $z_x \neq z_y$ because x and y have no common parent, and x is not above z_y because Y is down-closed, $x \in Y$, and $z_y \notin Y$. Furthermore, in the resulting network M, $u = z_x$ is a common parent of x and y, so $Y \cup \{u\}$ and $Y' \cup \{u'\}$ are down-closed sets in M resp. $M' = N'$, and $M[Y \cup \{u\}] \overset{\phi'}{\simeq}_X M'[Y' \cup \{u'\}]$, where ϕ' is ϕ extended with $\phi(u) = u'$.

The case in which (z_y, y) is movable is correct by symmetry in x and y. □

Like for Algorithm 2, some cases that could have been used to improve the algorithm are not actually implemented. For example, when (z_x, x) is not tail-movable because of a triangle and z_y is a reticulation, it could still be possible to make z_x movable using one tail move, and then use one tail move to make z_x a common parent of x and y. Again, this improvement is not necessary and would make the algorithm deviate from the proof of Lemma 3.6.

Algorithm 4: BOTTOM-UP RSPR(N, N')

Data: Two binary tier-k networks N_1 and N_2 on X
Result: An rSPR sequence from N_1 to N_2
Set $\phi(x_1) = x_2$ for all leaves x_1 and x_2 of N_1 and N_2 with the same label;
Let S_1 and S_2 be empty sequences of moves;
while $|\operatorname{im}(\phi)| < |V(N)| - 1$ **do**
  if N_2 *has a lowest reticulation* u_2 *above* $\operatorname{im}(\phi)$ **then**
    $S_1', S_2', u_1 =$LOWESTRETIC-RSPR$(N_1, N_2, \operatorname{dom}(\phi), \operatorname{im}(\phi), \phi, u_2)$;
  else if N_1 *has a lowest reticulation* u_1 *above* $\operatorname{dom}(\phi)$ **then**
    $S_2', S_1', u_2 =$LOWESTRETIC-RSPR$(N_2, N_1, \operatorname{im}(\phi), \operatorname{dom}(\phi), \phi^{-1}, u_1)$;
  else
    Let u_2 be a lowest tree node above $\operatorname{im}(\phi)$;
    $S_1', S_2', u_1 =$LOWESTTREE-TAIL$(N_1, N_2, \operatorname{dom}(\phi), \operatorname{im}(\phi), \phi, u_2)$;
  Apply S_1' to N_1 and add it to S_1;
  Apply S_2' to N_2 and add it to S_2;
  Set $\phi(u_1) = u_2$;
Set S_2^{-1} to the inverse of S_2;
return $S_1 \circ S_2^{-1}$;

With Algorithms 4 and 5, we can finally find sequences between any pair of networks. The first of these follows the proof of Lemma 5.18: a lowest reticulation node is chosen whenever possible, and no randomness is required in the subroutines Algorithms 2 and 3 (i.e., simply choosing nodes or arcs using an arbitrary method works as well). In Algorithm 5, we let go of this restriction of first choosing lowest reticulations, and we use the versions of the subroutines that actually choose random nodes whenever meaningful (i.e., whenever an instruction includes the word random within parentheses).

Algorithm 5: Bottom-Up rSPR Random(N, N')

```
   Data: Two binary tier-k networks N_1 and N_2 on X
   Result: An rSPR sequence from N_1 to N_2
 1 Set φ(x_1) = x_2 for all leaves x_1 and x_2 of N_1 and N_2 with the same label;
 2 Let S_1 and S_2 be empty sequences of moves;
 3 while |im(φ)| < |V(N)| − 1 do
 4     Let L be the set of all lowest nodes in N_1 \ dom(φ) and N_2 \ im(φ);
 5     for l in L do
 6         if l is a reticulation in N_1 then
 7             Let O be the output of LowestRetic-rSPR(N_2, N_1, im(φ), dom(φ), φ^{-1}, l);
 8         else if l is a reticulation in N_2 then
 9             Let O be the output of LowestRetic-rSPR(N_1, N_2, dom(φ), im(φ), φ, l);
10         else if l is a tree node in N_1 then
11             Let O be the output of LowestTree-Tail(N_2, N_1, im(φ), dom(φ)^{-1}, φ, l);
12         else
13             Let O be the output of LowestTree-Tail(N_1, N_2, dom(φ), im(φ), φ, l);
14         if O is not False then
15             if l is a node in N_1 then
16                 Set u_1 = l;
17                 Set S'_2, S'_1, u_2 = O;
18             else
19                 Set u_2 = l;
20                 Set S'_1, S'_2, u_1 = O;
21             Exit the for-loop;
22     Apply S'_1 to N_1 and add it to S_1;
23     Apply S'_2 to N_2 and add it to S_2;
24     Set φ(u_1) = u_2;
25 Set S_2^{-1} to the inverse of S_2;
26 return S_1 ∘ S_2^{-1};
```

We now prove that both these algorithms always correctly return a sequence of moves, and that their running time is polynomial.

Theorem 7.18. Bottom-Up rSPR *and* Bottom-Up rSPR Random *(Algorithms 4 and 5) are correct and run in polynomial time. Furthermore, for* $N_1, N_2 \in \mathcal{N}(n,k)$*, the sequence they return has length at most* $2n + 3k - 2$*.*

Proof. We will show that, in each of the algorithms, each iteration of the while-loop adds one node to an isomorphism between down-closed sets. This implies that the while-loop ends. Furthermore, after this loop, $|\mathrm{im}(\phi)| = |V(N)| - 1$, and by adding the root to the isomorphism ϕ, we obtain an isomorphicm between the networks N_1 and N_2 (at that point, so after performing all moves S_1 on the original N_1 and S_2 on the original N_2). By reversing the moves of S_2, which we can easily do using the explicit isomorphism ϕ, we obtain a sequence of rSPR moves from (the original) N_1 to N_2.

For Bottom-Up rSPR, the claim about the while-loop follows from the fact that LowestTree-Tail is only called when all lowest nodes are tree nodes (so it will never return False; Lemma 7.17) and LowestRetic-rSPR always returns a way to extend the isomorphism (Lemma 7.16). For Bottom-Up rSPR Random, this is because there is always either a lowest reticulation, or

all lowest nodes are tree nodes. Hence, there is always some lowest node for which the call to LowestRetic-rSPR or LowestTree-Tail returns a way to extend the isomorphism.

Furthermore, because LowestRetic-rSPR is used to add reticulations and adds at most 1 move, and LowestTree-Tail is used to add tree nodes and adds at most 2 tail moves, the sequence of moves has length at most $2n + 3k - 2$.

For the running time of these algorithms, we note that finding a reticulation or tree node in a given subset of the nodes of a network can be done in polynomial time, and, similarly, a movable arc with the endpoints in given sets can also be found in polynomial time. Hence, LowestRetic-rSPR and LowestTree-Tail clearly run in polynomial time. Moreover, the loops of Bottom-Up rSPR and Bottom-Up rSPR Random have at most $|V|$ iterations, and each operation within these loops takes polynomial time, so these loops take polynomial time in total as well.

Finally, to reverse a sequence of moves, we simply exchange the from-arc and the to-arc of each move, and replace the nodes by their pre-image in ϕ, which can also be done in polynomial time, as the sequence has polynomial length. □

Provable (in)effectiveness of the heuristic

The following lemma shows that the algorithm is in fact a heuristic, and no guarantees about the quality can be given that improve significantly upon the global upper bound of $2n + 3k - 2$ moves.

Lemma 7.19. *There exists a family of networks $N_i \in \mathcal{N}(1, 3{\cdot}2^{i-1}-1)$ for which* Bottom-Up rSPR Random(N_i, N_i) *returns a sequence of length $\Omega(|V(N_i)|)$ with non-zero probability.*

Proof. To construct N_i, take the caterpillar $C([2^i])$ and identify leaf j with the root of a copy $M_j \simeq N$ of the unique network $N \in \mathcal{N}(1, 2)$. Then take the balanced tree $B([2^i])$, reverse the direction of all arcs, and combine the result with the previously constructed network by identifying the original leaves of the balanced trees with the leaves of the M_j.

To see that Bottom-Up rSPR Random(N_i, N_i) may return a sequence of length $\Omega(|V(N_i)|)$, observe that the reversed balanced tree is highly symmetric, and there is an (non-labeled) isomorphism $\phi : B([2^i]) \to B([2^i])$ sending leaf l to leaf $2^i - l + 1$ for all $l \in [2^i]$ (Figure 7.5).

As the algorithm first adds lowest reticulations to the bottom-closed set, it will first construct an isomorphism between the reversed versions of the bal-

anced trees. This isomorphism may end up being the previously mentioned one, sending leaf l to leaf $2^i - l + 1$ for all $l \in [2^i]$.

The algorithm then proceeds to make the parts of the network above the reversed balanced tree isomorphic. These parts are two caterpillars, where the order of the leaves is reversed for one of these caterpillars. By Lemma 2.67, this distance is of order $\Omega(2^i)$. As the networks have $|V(N_i)| = \Theta(2^i)$ nodes, this implies the algorithm may give a sequence of length of order $\Omega(|V(N_i)|)$. □

Note that this means that Bottom-Up rSPR may also return sequences of length $\Omega(|V(N_i)|)$. Indeed, when parent nodes are chosen, this arbitrary choice may be the worst choice possible, depending on the implementation of the algorithm and on the encoding of the copies of N_i. If these networks are encoded in such a way that the worst parent is chosen each time, the algorithm can indeed perform very badly.

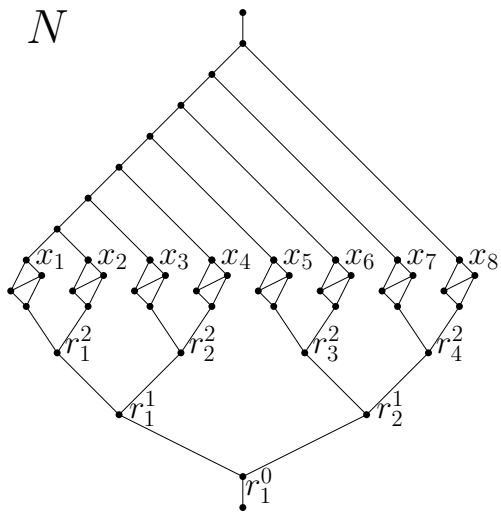


Figure 7.5: An example showing the ineffectiveness of the heuristic as constructed in Lemma 7.19. Bottom-Up rSPR(N, N) may first construct a down-closed isomorphism sending r_i^j to $r_{2^j-i+1}^j$. This essentially leaves two caterpillars T, T' at the top, whose leaf labels are reversed, so $d_{\text{rSPR}}(T, T') \geq (2^i - 2)/2$.

This example, unfortunately, shows that the heuristic is not an approximation algorithm. On the other hand, we may still expect good results from the random version, as the next proposition shows.

Proposition 7.20. *If $N \in \mathcal{N}(n,k)$, then* Bottom-Up rSPR Random(N,N) *(Algorithm 5) returns the empty sequence with probability at least 2^{-k}. In particular, if N is a tree, then this probability is 1.*

Proof. Let ϕ be a fixed labeled isomorphism between N and $N' = N$. We calculate the probability that the algorithm finds this isomorphism.

Suppose Y and Y' are down-closed sets of N and N' such that $N[Y] \overset{\phi}{\simeq} N'[Y']$, and let u' be a lowest node of $N' \setminus Y'$.

If u' is a reticulation and x' its child, then $x' \in Y'$, so $x = \phi^{-1}(x') \in Y$. If x has two parents in $N \setminus Y$, then the parent $z = \phi^{-1}(u')$ of x is chosen with probability $1/2$ in Line 3 of Algorithm 2. If x has one parent in $N \setminus Y$, then the parent $z = \phi^{-1}(u')$ of x is chosen with probability 1 in the same line.

Similarly, if u' is a tree node and x' and y' its children, then $x = \phi^{-1}(x')$ and $y = \phi^{-1}(y')$ have at least one common parent $\phi^{-1}(u')$ in $N \setminus Y$. If x and y have two such parents, then Algorithm 3 chooses $\phi^{-1}(u')$ with prability $1/2$ in Line 5, otherwise, it chooses it with probability 1.

Hence, we have an independent probability of at least $1/2$ for mapping the parents of a given reticulation correctly. Therefore, the algorithm will map all parents of reticulations correctly (and thus all nodes according to ϕ) with probability at least 2^{-k}. □

It is still an open question whether a similar statement is true when N and N' are not isomorphic, i.e., when $d_{\mathrm{rSPR}}(N,N') > 0$. More precisely, is the probability of finding an optimal sequence always positive? And, if so, can we find (a lower bound for) this probability?

7.2.3 Upper bound: tail move distance

For tail moves, a similar algorithm can be used. We cannot directly use the same algorithm, however, as the subroutine LowestRetic-rSPR of Algorithm 4 and Algorithm 5 may return head moves, so it cannot be used to find a sequence of tail moves. Hence, we replace this subroutine with the following algorithm based on Lemma 3.5.

Lemma 7.21. *Let $N, N' \in \mathcal{N}(n,k)$ be networks with down-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq}_X N'[Y']$ and u' a lowest node of $N' \setminus Y'$ that is a reticulation. Moreover, let O be the output of* LowestRetic-Tail(N, N', Y, Y', ϕ, u').

*If $O =$*False*, then $N \simeq N_A$ and $N \not\simeq_X N'$. Otherwise, the output O is a triple S, S', u. Let M and M' be the networks obtained by applying S to N and S' to N'.*

Then $|S|+|S'| \leq 3$, $Y \cup \{u\}$ *and* $Y' \cup \{u'\}$ *are down-closed sets in* M *resp.* M', *and* $M[Y \cup \{u\}] \overset{\phi'}{\simeq}_X M'[Y' \cup \{u'\}]$, *where* ϕ' *is* ϕ *extended with* $\phi(u) = u'$. *In other words, Algorithm 6 is correct.*

Algorithm 6: LowestRetic-Tail(N, N', Y, Y', ϕ, u')

Data: Two binary tier-k networks N and N' with down-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq}_X N[Y']$ and a lowest reticulation u' in $N \setminus Y$.
Result: Sequences of tail moves S, S' and a node u of N such that after applying S to N and S' to N' we get $N[Y \cup \{u\}] \overset{\phi'}{\simeq} N'[Y' \cup \{u'\}]$, where $\phi(u) = u'$; or False if $N \simeq N_A$ and $N \not\simeq_X N'$

1 Set S and S' to be empty sequences of moves.
2 Let x' be the child of u' and $x = \phi^{-1}(x)$;
3 Let z be a (random) parent of x not in Y;
4 **if** *z is a reticulation* **then**
5 | Set $u = z$;
6 **else if** (z, x) *is tail-movable* **then**
7 | Let u be a (random) reticulation in $N \setminus Y$;
8 | Let v be the child and w a (random) parent of r;
9 | **if** $v \neq x$ **then**
10 | | **if** $v = z$ **then**
11 | | | Set $v \neq x$ to be the other child of z;
12 | | | Add $z \xrightarrow{(z,v)} (w, u)$ to S;
13 | | **else**
14 | | | Add $z \xrightarrow{(z,x)} (u, v)$ and $z \xrightarrow{(z,v)} (w, u)$ to S;
15 **else**
16 | Let $\langle c, z, d\rangle$ be the triangle with $d \neq x$, and b the parent of c;
17 | **if** b *is not the root of* N **then**
18 | | Let a be a (random) parent of b;
19 | | Add $c \xrightarrow{(c,d)} (a, b)$ to S and apply it to N;
20 | | Execute Lines 7–14;
21 | **else**
22 | | Let e be the child of d;
23 | | **if** $e = x$ **then**
24 | | | Set $u = d$;
25 | | **else if** x *is a tree node* **then**
26 | | | Let s, t be the children of x;
27 | | | **if** $e \in \{s, t\}$ **then**
28 | | | | Let $q \in \{s, t\} \setminus \{e\}$;
29 | | | | Add $x \xrightarrow{(x,q)} (d, e)$ to S;
30 | | | **else**
31 | | | | Add $x \xrightarrow{(x,s)} (d, e)$, $x \xrightarrow{(x,e)} (z, t)$, and $x \xrightarrow{(x,t)} (d, s)$ to S;
32 | | | Set $u = d$;
33 | | **else if** e *is a tree node* **then**
34 | | | Let s, t be the children of e;
35 | | | **if** $x \in \{s, t\}$ **then**
36 | | | | Let $q \in \{s, t\} \setminus \{x\}$;
37 | | | | Add $e \xrightarrow{(e,q)} (z, x)$ to S;
38 | | | **else**
39 | | | | Add $e \xrightarrow{(e,s)} (z, x)$, $e \xrightarrow{(e,x)} (d, t)$, and $e \xrightarrow{(e,t)} (z, s)$ to S;
40 | | | Set $u = d$;
41 | | **else**
42 | | | **return** False;
43 **return** S, S', u;

Proof. The correctness follows almost immediately from Lemma 3.5. This is easy to see by comparing the cases of the proof of Lemma 3.5 with the following lines of the algorithm. Case 1 of the proof corresponds to Lines 4–5, Case 2(a) to Lines 6–14, Case 2(b)i to Lines 17–20, and Case 2(b)ii to Lines 21–40.

The algorithm returns False precisely when (z, x) is not movable which implies the existence of the triangle $\langle c, z, d\rangle$, the parent b of c is the root of N, and neither x nor e (the child of d) is a tree node. As noted in the proof of Lemma 3.5, e and x cannot be reticulations, so they have to be leaves. Hence, N consists of the nodes b, c, d, e, x, and z, and $N \simeq N_A$. Furthermore, as the parent u' of $x' = \phi(x)$ is a reticulation, we have $N \not\simeq_X N'$. □

The algorithms Bottom-Up Tail and Bottom-Up Tail Random are now defined exactly as Bottom-Up rSPR and Bottom-Up rSPR Random with the following two changes. Each occurrence of LowestRetic-rSPR is replaced with LowestRetic-Tail, and a few lines are added that catch the False output of LowestRetic-Tail when $N \simeq N_A$ and $N \not\simeq_X N'$, in this case, the algorithms should return False as well. The correctness and polynomial running time of these algorithms follow from the same argument as for their rSPR versions.

7.2.4 Upper bound: head move distance

For head moves, like for tail and rSPR moves, we present pseudo-code for a heuristic based on the top-down isomorphism approach we took to find an upper bound for the head move diameter in Theorem 4.23. Again, each algorithm corresponds to a lemma used to prove this bound.

Lemma 7.22. *Let $N, N' \in \mathcal{N}(n, k)$ be networks with up-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq} N'[Y']$ and x' a highest node of $N' \setminus Y'$ that is a tree node.*

Let S, S', c be the output of HighestTree(N, N', Y, Y', ϕ, x')*, and let M and M' be the networks obtained by applying S to N and S' to N'.*

Then $|S| + |S'| \leq 4$, $Y \cup \{c\}$ and $Y' \cup \{x'\}$ are up-closed sets in M resp. M', and $M[Y \cup \{c\}] \overset{\phi'}{\simeq} M'[Y' \cup \{x'\}]$, where ϕ' is ϕ extended with $\phi(c) = x'$. In other words, Algorithm 7 is correct.

Proof. The correctness follows immediately from Lemma 4.18. This is easy to see by comparing the cases of the proof of Lemma 4.18 with the following lines of the algorithm. Case 1 of the proof corresponds to Lines 3–4.

Case 2 consists of two parts, where Case 2a corresponds to Lines 8–13 and Case 2b to Lines 14–21.

Finally, Case 3 has five subcases, where Case 3b corresponds to Lines 24–33, Case 3c to Lines 34–37, Case 3d to Lines 38–39, and Case 3e to Lines 40–41.

Lastly, Case 3a is the subcase where $t = p$, which is caught by Line 23. In this case, the added node is simply the tree node c. □

Algorithm 7: HighestTree(N, N', Y, Y', ϕ, x')

Data: Two binary tier-k networks N and N' with up-closed sets Y and Y' such that $N[Y] \overset{\phi}{\cong} N[Y']$ and a highest tree node x' in $N' \setminus Y'$.
Result: Sequences of tail moves S, S' and a node c of N such that after applying S to N and S' to N' we get $N[Y \cup \{c\}] \overset{\phi}{\cong} N'[Y' \cup \{x'\}]$, where $\phi(c) = x'$.

```
1  Set S and S' to be empty sequences of moves.
2  Let p' be the parent of x', p = φ^{-1}(p'), and x ∉ Y a (random) child of p;
3  if x is a tree node then
4  |  Set c = x;
5  else if x is a reticulation then
6  |  Let c be a (random) tree node in N \ Y;
7  |  Let t be the parent, and b ≠ x a (random) child of c;
8  |  if p ≠ t and (p, x) is head-movable then
9  |  |  if x = t then
10 |  |  |  Let q ≠ p be the other parent of x;
11 |  |  |  Add x --(q,x)--> (c, b) to S;
12 |  |  else
13 |  |  |  Add x --(p,x)--> (t, c) and x --(t,x)--> (c, b) to S;
14 |  else if p ≠ t (and (p, x) is not head-movable) then
15 |  |  Let ⟨z, x, d⟩ be the triangle with z ≠ p;
16 |  |  Let (w, l) be the incoming arc of a (random) leaf l with w ≠ d;
17 |  |  Let b ∉ {l, x} be a (random) child of c;
18 |  |  if d = t then
19 |  |  |  Add d --(z,d)--> (w, l) and x --(z,x)--> (c, b) to S;
20 |  |  else
21 |  |  |  Add d --(z,d)--> (w, l), x --(p,x)--> (t, c), and x --(t,x)--> (c, b) to S;
22 else
23 |  Let c be a (random) tree node in N \ Y with parent t;
24 |  if t ≠ p then
25 |  |  Let (s, r) ∈ A(N) be a head-movable arc with s ≠ p;
26 |  |  Let q be the other parent of r, and w the child of r;
27 |  |  if r = p then
28 |  |  |  if t = s then
29 |  |  |  |  Add p --(q,p)--> (t, c) to S;
30 |  |  |  else
31 |  |  |  |  Add p --(s,p)--> (t, c) to S;
32 |  |  |  |  if t ≠ q then
33 |  |  |  |  |  Add p --(t,p)--> (q, x) and p --(q,p)--> (s, c) to S;
34 |  |  |  else if r = t then
35 |  |  |  |  Add r --(s,r)--> (p, x) to S;
36 |  |  |  |  if p ≠ q then
37 |  |  |  |  |  Add r --(p,r)--> (q, c) and r --(q,r)--> (s, x) to S;
38 |  |  |  else if s ≠ t then
39 |  |  |  |  Add r --(s,r)--> (p, x), r --(p,r)--> (t, c), r --(t,r)--> (s, x), and r --(s,r)--> (q, w) to S;
40 |  |  |  else
41 |  |  |  |  Add r --(s,r)--> (p, x), r --(p,r)--> (s, c), and r --(s,r)--> (q, w) to S;
42 return S, S', c;
```

Algorithm 8: HIGHESTRETIC(N, N', Y, Y', ϕ, x')

Data: Two binary tier-k networks N and N' with up-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq} N[Y']$ and a highest reticulation x' in $N' \setminus Y'$.
Result: Sequences of tail moves S, S' and a node r of N such that after applying S to N and S' to N' we get $N[Y \cup \{r\}] \overset{\phi}{\simeq} N'[Y' \cup \{x'\}]$, where $\phi(r) = x'$; or FALSE.

1 Set S and S' to be empty sequences of moves.
2 Let p' and q' be the parents of x', $p = \phi^{-1}(p')$, and $q = \phi^{-1}(q')$;
3 Let $c_p \notin Y$ and $c_q \notin Y$ be (random) children of p resp. q;
4 **if** *c_p or c_q is a tree node* **then**
5 **return** FALSE;
6 **if** $c_p = c_q$ **then**
7 Set $r = c_p$;
8 **else if** *both c_p and c_q are leaves* **then**
9 Let (s, r) be a (random) movable arc with $r \in N \setminus Y$;
10 **if** $s = p$ **then**
11 Swap the roles of p and q;
12 Add $r \xrightarrow{(s,r)} (p, c_p)$ and $r \xrightarrow{(p,r)} (q, c_q)$ to S;
13 **else**
14 **if** *c_p is above c_q or c_p is a leaf* **then**
15 Swap the roles of p and q;
16 **if** *(p, c_p) is movable* **then**
17 Add $c_p \xrightarrow{(p,c_p)} (q, c_q)$ to S;
18 Set $r = c_p$;
19 **else**
20 Let $\langle t, c_p, z \rangle$ be the triangle with $t \neq p$;
21 **if** $t = q$ **then**
22 Set $r = c_p$;
23 **else**
24 Add $z \xrightarrow{(c_p,z)} (q, c_q)$ and $c_p \xrightarrow{(t,c_p)} (z, c_q)$ to S;
25 Set $r = z$;
26 **return** S, S', r;

Lemma 7.23. *Let $N, N' \in \mathcal{N}(n, k)$ be networks with up-closed sets Y and Y' such that $N[Y] \overset{\phi}{\simeq} N'[Y']$ and x' a highest node of $N' \setminus Y'$ that is a reticulation. Moreover, let O be the output of* HIGHESTRETIC(N, N', Y, Y', ϕ, x').

*If $O =$*FALSE*, then N has a highest node below Y that is a tree node. Otherwise, the output O is a triple S, S', r. Let M and M' be the networks obtained by applying S to N and S' to N'.*

Then $|S| + |S'| \leq 2$, $Y \cup \{r\}$ and $Y' \cup \{x'\}$ are up-closed sets in M resp. M', and $M[Y \cup \{r\}] \overset{\phi'}{\simeq} M'[Y' \cup \{x'\}]$, where ϕ' is ϕ extended with $\phi(r) = x'$. In other words, Algorithm 8 is correct.

Proof. The correctness follows immediately from Lemma 4.19, by a straightforward mapping of the cases of its proof to parts of the algorithm. The structure of the algorithm diverges from the proof, as most cases use a very similar sequence of moves.

First note that the algorithm outputs FALSE only in Line 5. This line is only executed if c_p or c_q is a tree node. Cases 1 and 2 correspond to Lines 6–7 and Lines 8–12. The remaining cases all correspond to Lines 13–25. More

precisely, for Case 3, the assumption that c_2^p is not below c_2^q is enforced by Lines 14–15, and the remaining lines simply follow the structure of the proof. Similarly, for Case 4, the assumption that c_p is a reticulation and c_q a leaf is enforced by Lines 14–15, and the remaining lines simply follow the structure of the proof. □

Algorithm 9: Permute Leaves Head(N_1, N_2, ϕ)

```
   Data: Two binary tier-k networks N1 ≅^φ N2 ∈ N(n,k) labeled l1 and l2
   Result: A head move sequence making N1 labeled isomorphic to N2
 1 Find the permutation (as a set of disjoint cycles) of the leaves π : L(N1) → L(N2) that makes
     the networks labeled isomorphic;
 2 Let S be an empty sequence of moves;
 3 Let (t, r) be a head movable arc;
 4 Let c be the child, and s ≠ t the other parent of r;
 5 for each cycle (l1, ..., lm) in π with m ≥ 2 do
 6     if t is the parent of lm then
 7         Set (l1, ..., lm) := (lm, l1, ..., lm−1);
 8     Let pm be the parent of lm;
 9     if r ≠ pm then
10         Apply r --(t,r)--> (pm, lm) and add this move to S;
11     Let e_moved = (t, r);
12     for i = m, ..., 1 do
13         Let pi ≠ tail(e_moved) be a parent of r, and pi−1 the parent of li−1;
14         if pi ≠ pi−1 then
15             Apply r --(pi,r)--> (pi−1, li−1) and add this move to S;
16             Set e_moved = (pi, r);
17     if c = li then
18         Set c = li−1;
19 Apply r --(t,r)--> (s, c) and add this move to S;
20 return S;
```

Lemma 7.24. *Let $N_1 \overset{\phi}{\simeq} N_2$, then* Permute Leaves Head(N_1, N_2, ϕ) *returns a sequence of head moves from N_1 to N_2.*

Proof. First, in Line 1, the algorithm finds the permutation of the leaves that makes the networks labeled isomorphic. To use this permutation in the algorithm, it must be given in terms of a set of disjoint cycles. Such a representation of the permutation can easily be found as follows. Start with any leaf x_1 of N_1, find the node $x_2 = \phi(x_1) \in N_2$ this node is mapped to by the current isomorphism, and finally find the leaf $x_1' = l_1^{-1}(l_2(x_2))$ of N_1 with the same label as x_2. Now repeat this with $x_1 := x_1'$, and continue until x_1' is the original x_1. This gives one of the cycles of the permutation. All other cycles of the permutation can be found in this way, simply by starting with a leaf that is part of this cycle.

The standard sequence of $m+2$ moves for a cycle $(l_1, \ldots, l_m)$ are performed in Lines 8–18. The first of these moves is skipped if r is the parent of l_m, as the purpose of that move is to make r the parent of l_m. Moves 2 up to $m+1$ Lines 12–16. The if-statement in this for-loop ensures that no consequtive leaves

in the cycle are part of a cherry. Finally, the last move is skipped except for each cycle, and is only performed once at the end in Line 19. This is possible because the first and the last move for each cycle in the permutation move the same arc. To make sure this last move actually puts (t, r) back in its original position, the head of the from-arc (s, c) is changed as needed in Lines 17–18. □

Algorithm 10: TOP-DOWN HEAD(N_1, N_2)

```
   Data: Two binary tier-k networks N1, N2 ∈ N(n, k), with k ≥ 2
   Result: A head move sequence making N1 isomorphic to N2
 1 Set φ(ρ1) = ρ2 for the roots, and let S1 and S2 be empty sequences of moves;
 2 while |im(φ)| < |V(N1)| − n do
 3     if N1 has a highest node x1 below dom(φ) that is a tree node then
 4         Set S2', S1', x2 =HighestTree(N2, N1, im(φ), dom(φ), φ^-1, x1);
 5     else if N2 has a highest node x2 below im(φ) that is a tree node then
 6         Set S1', S2', x1 =HighestTree(N1, N2, dom(φ), im(φ), φ, x2);
 7     else
 8         Let x1 be a highest reticulation node in N1 \ dom(φ);
 9         Set S2', S1', x2 =HighestRetic(N2, N1, im(φ), dom(φ), φ^-1, x1);
10     Apply S1' to N1 and add it to S1;
11     Apply S2' to N2 and add it to S2;
12     Set φ(x1) = x2;
13 Extend the isomorphism φ to include the leaves;
14 Set S^π =Permute Leaves Head(N1, N2, φ);
15 Set S2^-1 to the inverse of S2;
16 return S1 ∘ S^π ∘ S2^-1;
```

Algorithm 11: TOP-DOWN HEAD RANDOM(N_1, N_2)

```
   Data: Two binary tier-k networks N1, N2 ∈ N(n, k), with k ≥ 2
   Result: A head move sequence making N1 isomorphic to N2
 1 Set φ(ρ1) = ρ2 for the roots, and let S1 and S2 be empty sequences of moves;
 2 while |im(φ)| < |V(N1)| − n do
 3     Let H be the set of all highest nodes in N1 \ dom(φ) and N2 \ im(φ);
 4     for h in H do
 5         if h is a tree node in N1 then
 6             Let O be the output of HighestTree(N2, N1, im(φ), dom(φ), φ^-1, h);
 7         else if h is a tree node in N2 then
 8             Let O be the output of HighestTree(N1, N2, dom(φ), im(φ), φ, h);
 9         else if h is a reticulation in N1 then
10             Let O be the output of HighestRetic(N2, N1, im(φ), dom(φ), φ^-1, h);
11         else
12             Let O be the output of HighestRetic(N1, N2, dom(φ), im(φ), φ, h);
13         if O is not False then
14             if h is a node in N1 then
15                 Set x1 = h;
16                 Set S2', S1', x2 = O;
17             else
18                 Set x2 = h;
19                 Set S1', S2', x1 = O;
20             Exit the for-loop;
21     Apply S1' to N1 and add it to S1;
22     Apply S2' to N2 and add it to S2;
23     Set φ(x1) = x2;
24 Extend the isomorphism φ to include the leaves;
25 Set S^π =Permute Leaves Head(N1, N2, φ);
26 Set S2^-1 to the inverse of S2;
27 return S1 ∘ S^π ∘ S2^-1;
```

Using the previous algorithms together with their correctness proofs, it is immediately clear that the Algorithms 10 and 11 that give upper bounds for the head move distance are correct.

7.3 Testing the heuristics

In this section, we present an implementation of the algorithms in this chapter. This python implementation is tested to determine its running time in practice. Furthermore, we investigate the quality of the solutions by running the heuristics on a large number of inputs. As we cannot easily find the actual distance between pairs of networks—this problem is NP-hard, and no efficient algorithms are known—we construct experiments in which we can judge the quality of the solutions by comparing them to each other, or to other known bounds. All data from these experiments can be found at the 4tu data repository, DOI:10.4121/13604387.

7.3.1 Implementation details

We have implemented the heuristics in python (v2.7) with a very simple interface for a (linux) terminal, which can be found at `https://github.com/RemieJanssen/RearrangementHeuristics`. The script requires an input file containing two networks (in extended Newick format, or as set of edges in python list format). Other terminal options are then used to choose the heuristic and the output options.

The implementation relies on the python package 'networkx' (v2.2), to represent the networks as graphs. Leaf labels are attached to nodes as a node attribute called 'label'. With this representation, it is easy to determine whether two networks are (labeled) isomorphic using the networkx function is_isomorphic. For our purposes, this makes isomorphism checking of networks fast enough. The only other ways this package is used are to check for ancestral relations between nodes, and to manipulate the graphs (applying moves and renaming nodes).

7.3.2 Running time in practice

In Section 7.2, we have proven that the running time of the heuristics is polynomial, but we have not determined the degree of the polynomial. Here, we investigate the practical running time of the heuristics.

To test the running time, we generated a large number of network pairs. Networks were obtained using the online 'Ntk generator' [Zha16].[2] We gener-

[2] `http://phylnet.univ-mlv.fr/tools/randomNtkGenerator.php`

ated 20 networks for each of the following parameters: the number of leaves is $n \in \{5, 10, 20, 50, 100\}$ and the number of reticulations $k \in \{1, 2, 5, 10, 20, 50, 100\}$. For all pairs among these with the same number of leaves and reticulations (including the pairs consisting of two copies of the same network), all six heuristics—non random or random for each of the three move types tail, head, and rSPR—were run once, and the time used to run the heuristic was recorded using the built-in 'time' package of python (the time to read the trees and parameters was not included).

All tests were run on a Linux system with an Intel Core i7-8650U CPU running at 1.90GHz and 8192MB of DDR4 RAM clocking in at 2400MT/s. The operating system was Ubuntu 18.04.4 with a 4.15.0-118-generic kernel. The software was written in Python version 2.7.17.

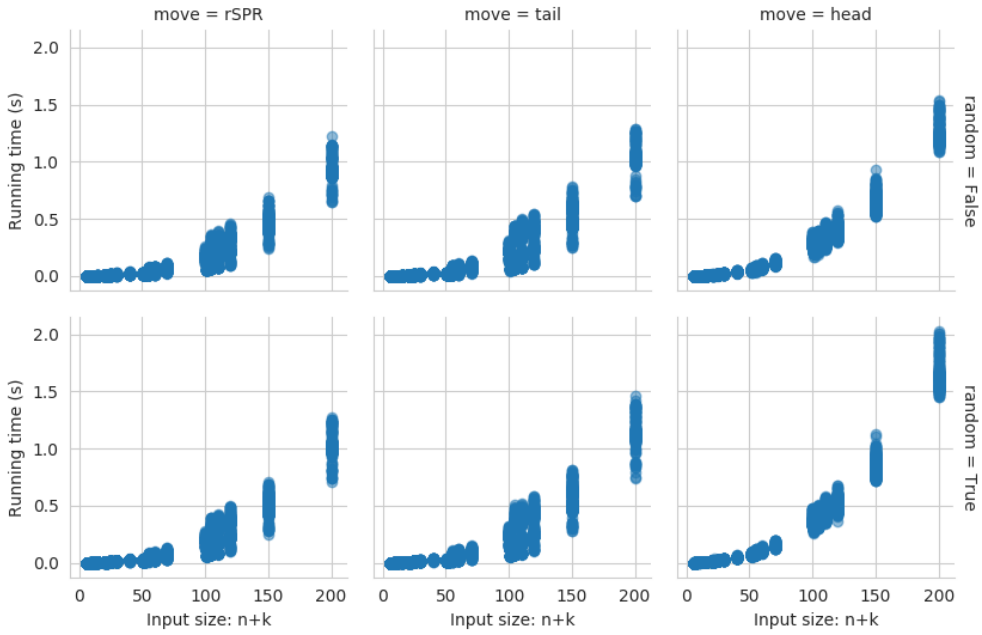


Figure 7.6: The dependence of the running time on the input size for the six different heuristics.

Results

The results of our experiments are shown in Figure 7.6. Note that the random versions seem to be slightly slower than non-random versions. Furthermore, the head move heuristics appear slower as well, which may be a result of the

permutation parts of these. This can most likely be explained by the failing attempts to add nodes to the isomorphism in the random algorithms, which cannot happen in the non-random algorithms. In general, however, the running times are all quite similar, and all are very sufficiently fast for our purposes.

To determine the degree of the polynomial running time, we performed a least squares linear fit in log-log space using the sklearn.LinearRegression function from the python package sci-kit learn. The independent variable in this analysis was $\log(n+k)$ and the dependent variable $\log(t)$, where t is the running time of the heuristic.

Table 7.1: Linear fits of the running times versus the input size $(n+k)$ for log-log transformed data.

Move	Random	Slope	Intercept	R^2
rSPR	False	2.146	-16.96	0.9644
	True	2.130	-16.73	0.9652
tail	False	2.162	-16.85	0.9556
	True	2.146	-16.68	0.9563
head	False	2.065	-15.72	0.9909
	True	2.079	-15.41	0.9947

Note that all these fits are decent (high R^2), and the slopes are all close to 2. As the slope in log-log space is equal to the degree of a polynomial function, the running time is slightly worse than quadratic, but certainly better than cubic. The apparently worse running time of the random versions is not visible in these fits, but the worse running time of the head move heuristics is clearly visible in the intercepts, which have a value roughly one higher than the tail move and rSPR heuristics.

In practice, this dependency may not be very relevant because, even for large networks (e.g., 100 leaves and 100 reticulations) the heuristics all run within two seconds. Even running a random heuristic 100 times only takes a few minutes on rather large pairs of networks. Therefore, when one wants to compare two networks, the heuristics are fast enough.

7.3.3 Performance

Although we cannot give a guarantee for the quality of the solutions the heuristics provide, it may still be possible that the heuristics output short sequences of moves in some cases. In this section, we investigate the quality of the solutions of the heuristic for three different types of input. First, we run the heuristics

on isomorphic networks, to see how frequently the heuristic recognizes that the input networks are isomorphic. Then, we supply it with more realistic inputs, where the two input networks are not isomorphic. We do this in two experiments, in the first of these, we use only small networks; in the second, we use larger networks, where we make sure the distance between the input networks is small.

Quality for isomorphic inputs

In this experiment, we use the heuristic to determine whether two networks are isomorphic. This can be done in polynomial time (Lemma 7.1), but not with our heuristics, as the heuristics are not made to check for network isomorphism. Indeed, the heuristics may return non-trivial sequences of moves between the already isomorphic networks (indicating that it thinks they are not isomorphic). As we have shown in Proposition 7.20, a sequence of length zero will be returned with probability at least 2^{-k} by the random rSPR/tail heuristic, but it is still interesting to see how long the sequences can get, and how large the probability of returning a sequence of length zero is in practice.

To test this, we create an input for each network of the previously mentioned data set by simply taking two copies of this network as the input. For each such pair, we run each of the three random heuristics 100 times.

To judge the quality of the heuristics, we estimate the probability of finding the optimal sequence, which has length zero as the pairs are all isomorphic. For each pair of networks, we estimate this probability by calculating the fraction of the 100 tests in which a sequence of length zero was found (Figure 7.7).

For rSPR and tail moves, two similar pictures emerge. The fraction in which an optimal sequence is found decreases sharply with k, but slightly slower than we may expect from Proposition 7.20, and the fraction increases very slightly with n. Furthermore, there seems to be an inverse correlation between the fraction optimal runs and the length of the worst solution. For head moves, however, the results are abysmal. A sequence of length zero is found only in a few cases, and only for very small networks (at most 10 leaves). Furthermore, the head move heuristic finds very long sequences sometimes.

This shows that, even though the top-down isomorphism approach works well to find a head move diameter upper bound, the corresponding heuristic cannot effectively determine a good upper bound for the head move distance between a specific pair of networks. For rSPR and tail moves, however, the heuristic seems to be quite effective for small k. Hence, it is interesting to see how these results hold up when the networks are not isomorphic to begin with. The next two experiments investigate this question.

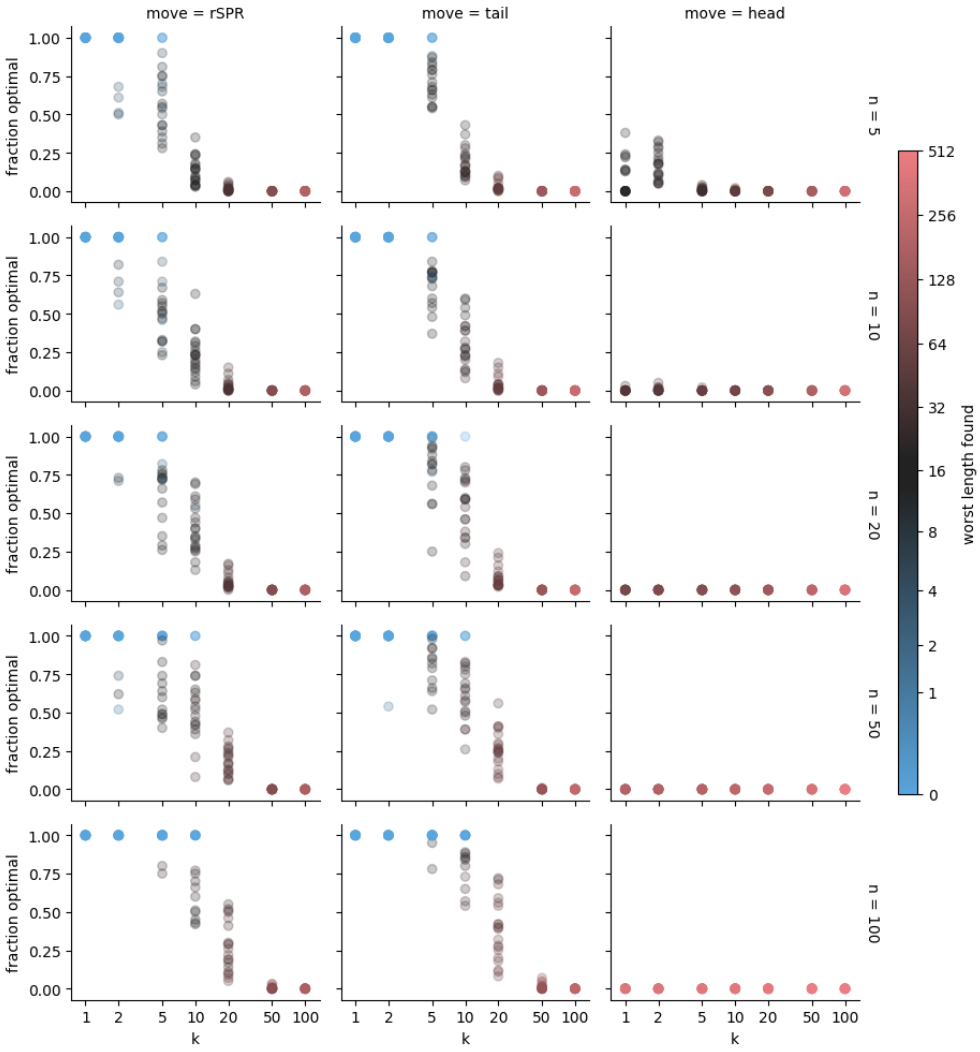


Figure 7.7: The quality of the solutions of the random heuristics when applied to a pair of isomorphic networks. Each dot represents a pair of networks, and the colour of the dot the worst found solution. The independent variable is k, and the dependent variable is the fraction of the 100 tests for that pair that results in a sequence of length zero.

7.3.4 Quality for small networks

For small networks, the number of valid moves in a network is small. Nevertheless, it is not easy to find the distance between two small networks quickly.

Indeed, as we will see shortly, even for networks with 3 leaves and 3 reticulations, or 4 leaves and 2 reticulations, a naive breadth first search may take more than ten minutes to find the distance. Therefore, we investigate the quality of the heuristic solutions for small networks.

For each of the combinations of $n \in \{3, 4\}$ leaves and $k \in \{1, 2, 3\}$ reticulations, we generated 10 networks using the aforementioned 'Ntk generator'. We then used a breadth first search (Algorithm 1) to determine exact distances between these pairs of networks (with the same number of leaves and reticulations). As the running time of this algorithm increases sharply with the actual distance, we limited the time for each pair to ten minutes. If the time limit was reached before a sequence was found, the breadth first search gave a lower bound on the distance. Indeed, as all sequences shorter than the currently considered sequence were already tried, the actual distance is at least the length of the sequence under consideration when the lime limit was reached. We compared these distances (or lower bounds) to the solutions of the heuristics, by running the random heuristic 100 times for each pair of networks.

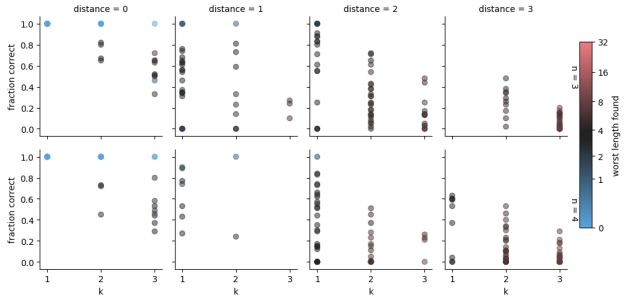
As we have seen in the previous experiment, the rSPR and tail move heuristics often give the correct sequence for small networks if this distance is 0. With this experminent, we see that the fraction of tests in which the heuristics give the right answer (for a given pair of networks) decreases quite sharply with the distance between the networks (Figure 7.8). Again, the fraction of correct sequences is markedly larger for rSPR moves and tail moves than for head moves.

We do see that, in many cases, the fraction of correct sequences is not zero, so taking the best sequence out of 100 runs could still often give the correct distance d for a given pair. To see how frequently we can obtain a correct sequence by running the heuristic 100 times, we determine whether the exact distance/lower bound found by the breadth first search is attained in at least one of the runs. Then, for each combination of d and k, we determine for how many pairs of networks this happens (Figure 7.9a).

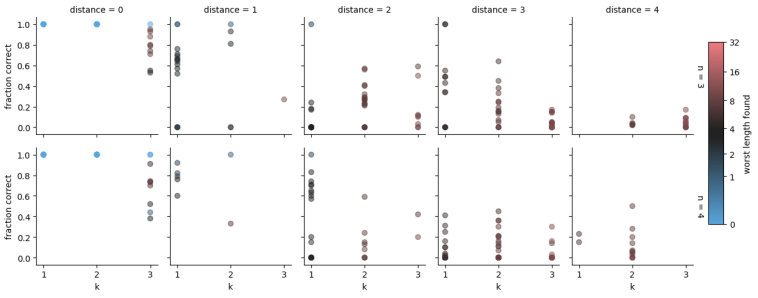
For some combinations of move, d, n, and k there is no data. This is either because no such pairs exist, or because they simply did not occur in our inputs. Observe that the correct length is found in many cases for rSPR and tail moves, and a sequence of length $d + 1$ is found in almost all cases (Figure 7.9b). For head moves, the results are observably worse again, but the heuristic still gives sequences that are only one move too long in many cases.

7.3.5 Quality for short distances

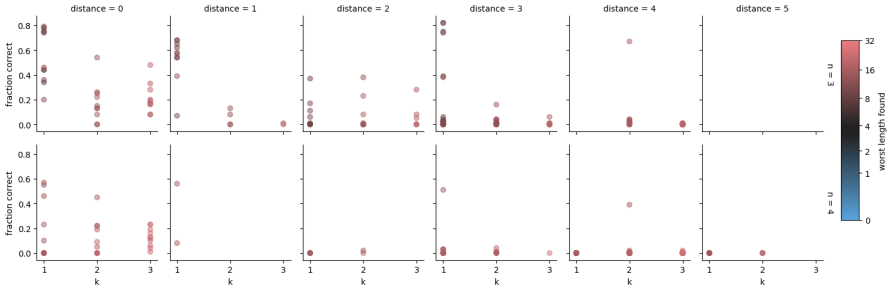
At first glance, it seems reasonably easy to calculate the distance between two networks that are only a small distance apart. As we have seen, for small



(a) rSPR



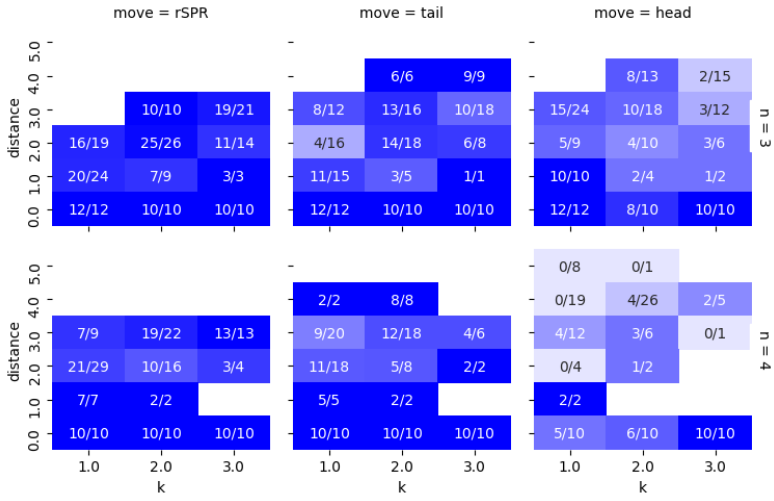
(b) tail



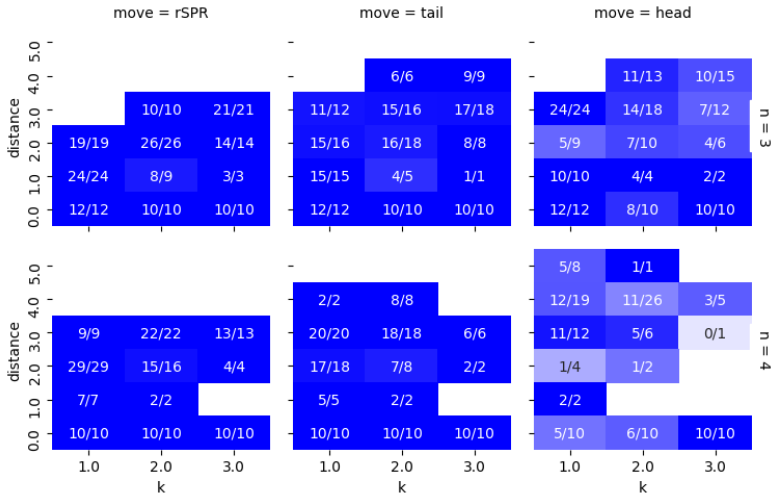
(c) head

Figure 7.8: The quality of the solutions of the random heuristics when applied to small networks. Each dot represents a pair of networks, and the colour of the dot the worst solution found for that pair. The independent variable is k and the dependent variable is the 'fraction optimal'. This is the fraction of the 100 runs for that pair that resulted in a sequence of length at most the actual distance d between the pair of networks.

networks this is the case. However, for large networks, this does not hold anymore, because the runtime dependency of the breadth first search on the distance and the size of the network is very bad. Because the heuristics still



(a) length at most d



(b) length at most $d+1$

Figure 7.9: The fraction of pairs of small networks for which the heuristic found a good answer of length at most d or $d+1$ in at least one of the 100 runs, where d is the actual distance between the networks.

perform quite well for larger isomorphic networks, it may be reasonable to expect them to perform well for short distances as well. We investigate this by running the random heuristics on pairs of networks that are a small distance apart.

As we cannot simply calculate the distance between pairs of networks, we create pairs of networks that are a small distance apart. To do this, we take a network, and apply a small number of moves chosen uniformly at random from all possible moves. The number of applied moves is an upper bound on the actual distance. Indeed, the actual distance may be shorter, as moves could be reversed or there may be shorter sequences between the networks.

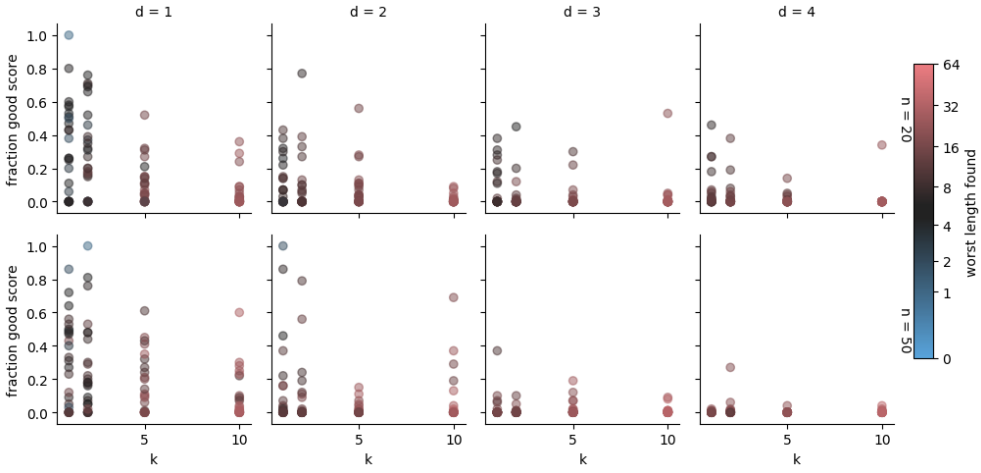
The data set for this experiment is created by starting with five networks in the previously mentioned data set for each $n \in \{20, 50\}$ and $k \in \{1, 2, 5, 10\}$. For each of these networks, we create 60 pairs: 20 per move type, where we make 5 pairs for each upper bound in $d = \{1, 2, 3, 4\}$—for the tail/head move upper bound, we, of course, only apply tail/head moves.

Like for the isomorphic networks, we judge the quality of the heuristics by estimating the probability of finding an optimal sequence. In this case, we do not know the exact distance between the networks, and it takes too long to compute this distance for all the pairs using the breadth first search algorithm. Hence, we take the number of moves d we used to create the pair of networks as being a good score for this pair. Then, for each pair, we calculate the fraction of the 100 tests in which we find a sequence of length at most d.

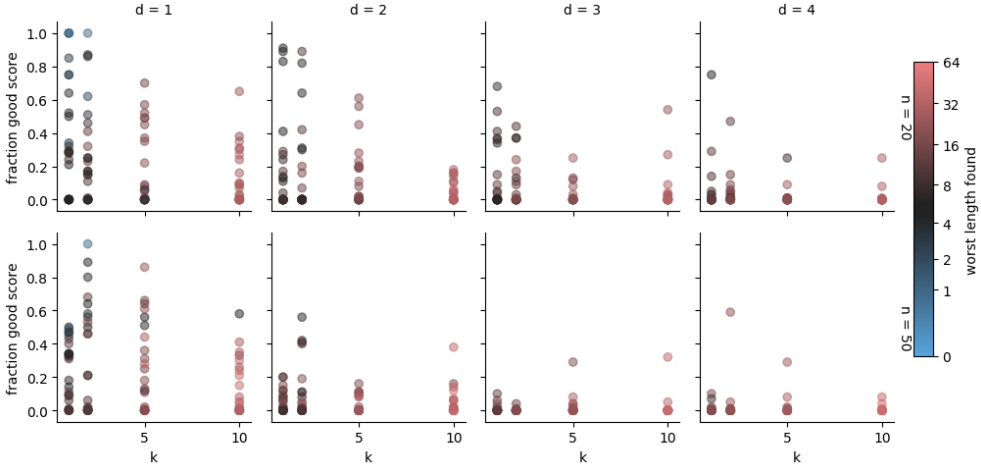
In Figure 7.10, we show the results of this analysis for rSPR and tail moves, where each dot represents a pair of networks. We see a similar dependence on d and k for rSPR and tail moves: an increase in d as well as k leads to a lower fraction of good scores. There also seems to be a less pronounced effect of n, where a larger n leads to lower fractions of good scores. Contrast this to isomorphic networks, where an increase in n seemed to give higher fractions of good scores.

For many pairs, the fraction of good scores is quite low, so we cannot discern pairs for which a good score was found from pairs for which no good scores were found in Figure 7.10. Hence, we also investigate the fraction of pairs for which at least one of the 100 tests finds a good score (i.e., a sequence of length at most d). These fractions are shown in Figure 7.11a. Observe that the fraction of pairs for which a good score is found decreases with both k and d. In Figure 7.11b, we show a similar plot for the fraction of pairs for which we found a sequence of at most length $2d$ at least once. For rSPR and tail moves, this is almost in all cases, and this even happens occasionally for head moves.

This shows that, for small distances, the heuristics work rather well and find a (near) optimal sequence in many cases. Moreover, a sequence at most twice the length of the original sequence is found in most cases. We conclude that the tail and rSPR heuristics provide decent upper bounds when run often enough.

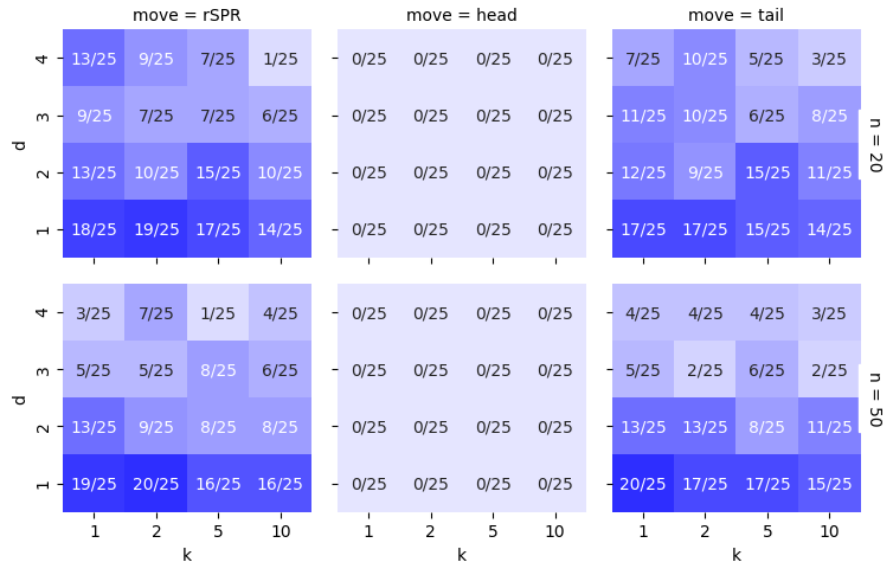


(a) rSPR

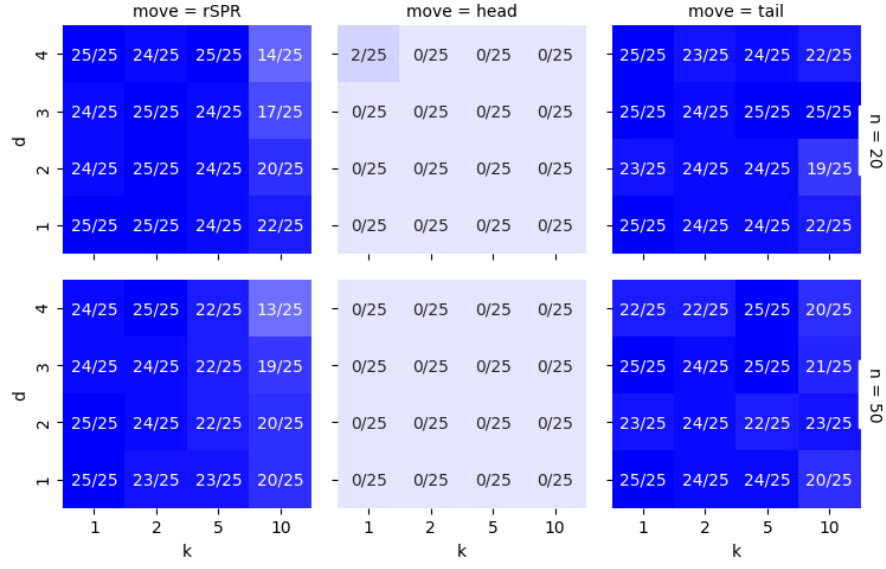


(b) tail

Figure 7.10: The quality of the solutions of the random heuristics when applied to networks a small distance apart. Each dot represents a pair of networks, and the colour of the dot the worst solution found for that pair. The independent variable is k and the dependent variable is the 'fraction good score'. This is the fraction of the 100 runs for that pair that resulted in a sequence of length at most the number d of moves used to create the pair of networks.



(a) length at most d



(b) length at most $2d$

Figure 7.11: The fraction of pairs which are a small distance d apart for which the heuristic found a good answer (length at most d or $2d$) in at least one of the 100 runs.

7.3.6 Discussion

With the experiments, we have first shown that it is practically feasible to run the (random) heuristics a large number of times for a large number of pairs of networks. Indeed, the running time is roughly quadratic in practice, and does not exceed a few seconds, even for large networks with 100 leaves and 100 reticulations.

Then we tested the quality of the heuristics: we have applied the heuristics a large number of times to a wide range of networks. First, we have confirmed that the tail and rSPR heuristics frequently return empty sequences for isomorphic inputs, as may be expected from Proposition 7.20. The frequency with which this happened increased with the number of leaves while keeping the number of reticulations constant. This may be the result of a kind of dilution effect, where correct (0-length) sequences become more likely if the reticulations are far apart in the networks. The results for the head moves heuristic were strikingly worse, and sequences of length zero were rarely found. Note that using these heuristics to determine network isomorphism is not very useful, as Binary Network Isomorphism is polynomial (Lemma 7.1).

Then, we investigated the quality of the solutions for non-isomorphic inputs. We started this investigation by considering small networks of at most 4 leaves and 3 reticulations. The fraction of runs in which an optimal sequence was found decreased with the actual distance between the networks. However, for most pairs, the tail and rSPR heuristics still found an optimal sequence at least once in one of the 100 runs, and, in almost all cases, the best sequence was at most one move too long. For head moves, the picture is again a lot worse, even though optimal sequences were still found in many cases.

We continued by considering larger networks that were only a small number of moves apart. For these networks, the head move heuristic was essentially worthless. The tail move and rSPR heuristics, however, still performed quite well. The quality of the solutions did decrease with the number of reticulations in the networks and with the distance between the networks. Nevertheless, in most cases, these heuristics found solutions that were within a factor 2 of the length of the original sequence.

From these experiments, we conclude that the rSPR and tail move heuristics are able to provide decent upper bounds, but the head move heuristic is rather worthless in this aspect. It seems it is often possible to find an optimal sequence using the tail move and rSPR heuristic, but not with the head move heuristic. For larger networks with a larger distance, we found fewer optimal sequences. This may be because the probability of finding an optimal sequence within 100 runs is small, or it may even be impossible to find an optimal sequence using the heuristics.

7.4 Conclusion

In this chapter, we have studied the problem of finding the rearrangement distance between a pair of directed networks without internal labels.

7.4.1 Computational complexity

First we have shown that these problems are NP-complete for non-local moves, and they remain NP-complete for rSPR and tail moves when restricted to a fixed tier of network space. It remains open whether the same holds for head moves. A first step for this might be to study head move distance computation in tier-1. This seems intriguingly simple, as head move distance in tier-1 seems to correspond to a weighted rSPR distance on the corresponding trees, where the weight of a sequence of moves is the number of moves plus the graph distance between the moving arc and the previous target arc for each move.

For local moves and moves on undirected networks, several of these questions also remain open. Perhaps, these questions can be answered using different techniques. For example, it seems reasonable to expect that the problems for local moves can be proven to be NP-complete by separating the reticulate parts from a tree part, and using the fact that it is hard to compute the rearrangement distance on trees. For SPR moves, the question can probably be answered using an agreement forest technique similar to the one used in [JJE+18]. The proofs can possibly be simplified by seeing these agreement forests as restrictions of agreement graphs of the networks to the trees. Agreement graphs have already been studied by Klawitter for directed networks [Kla18a] and undirected networks [Kla20a].

7.4.2 Heuristics

Next, we turned to algorithms for computing sequences between networks. We started with a simple breadth first search, which is guaranteed to find an optimal sequence, but is very slow in practice: it often exceeds 10 minutes running time on very small networks.

To compute sequences more efficiently, we employed heuristics based on the bottom-up and top-down isomorphism proofs for diameter bounds. The bottom-up heuristics for rSPR and tail moves worked quite well. They gave optimal sequences in many cases, and sequences at most twice as long in most cases. This shows that, by making the right choices in the heuristics, we can find short sequences between networks. In their theses, Husanovic and Versendaal have attempted to find rules for good choices. Husanovic did this by explicitly formulating rules, and testing whether they improved the solutions [Hus20]; and

Versendaal used machine learning to find such rules [Ver20]. Unfortunately, no guarantees for the quality of the solutions can be given, as there exists a family of networks for which the heuristics can return long distances (Lemma 7.19), even if the network is used as both parts of the input. This holds for the random and non-random versions of the heuristics in this thesis, but also for the versions used by Versendaal and Husanovic.

In contrast to the tail move and rSPR heuristics, the head move heuristic does not provide good bounds at all. This is probably because the heuristic does not take the leaf labels into account until the end. This could be checked by running all heuristics on networks with one leaf, but as these cases are biologically irrelevant, we have not done this. For a better heuristic, it might be better to build an isomorphism bottom-up for head moves as well. This is possible by taking the rSPR heuristic, and replacing each tail move by a sequence of at most 13 head moves (Theorem 5.7). Perhaps, this replacement can be done more efficiently in the specific cases of the heuristic, to get a decent head move bound.

Although we have exclusively studied heuristics for directed networks and non-local moves in this chapter, the diameter bounds for local and non-directed networks from the previous chapters can also be converted to heuristics. For example, for tail_1 and rNNI moves, we can take the same heuristic where we use the tail move heuristic and replace each long distance move with a sequence of tail_1 moves via an up-down path (Lemma 3.9). This is quite straightforward to implement. For SPR and NNI moves, the modifications are quite minimal as well.

For head_2 moves, the heuristic that follows from the diameter proof is slightly less simple to implement, as it involves computing an rSPR sequence between trees. To get a good solution, this rSPR sequence should not only be short, but the distance between the to-arcs of the moves should be small as well. Indeed, each rSPR move is simulated by moving a triangle through the tree to the from-arc first. This problem of finding such a sequence has not been studied yet, so it is unclear whether this can be done efficiently. Alternatively, we can take any sequence—for example, as returned by our rSPR heuristic applied to the trees—but this may result in low quality solutions in the head_2 heuristic.

Network generator

We must note that our input networks may not be representative of realistic networks, as the Ntk generator was not designed to produce such networks. In fact, this generator is supposed to draw networks uniformly at random from the set of networks [Zha16]. It is known that the *PDA* (Proportional to Distinguishable Arrangements) distribution for trees is not representative of realistic

phylogenetic trees [Ald96, BF06]. This makes it unlikely that drawing phylogenetic networks uniformly at random results in a set of networks representative of realistic networks.

Hence, to test the quality of the heuristic in more realistic cases, one should generate input networks that are representative of realistic networks. This could be done by extending tree generators to networks. Two examples of such generators are the birth-hybridization models of [ZODS18] and [PSC19]. The first of these adds reticulation arcs uniformly to the current network while generating the network, and the latter adds reticulations where it is more likely that the reticulation arcs are local. However, the underlying tree model—obtained by turning reticulations off—are *ERM* (Equal Rate Markov, also nown as Yule) models, which are also known to be unrealistic despite their wide use [Ald96, BF06]. To remedy this, one could take a more realistic tree model and add reticulation edges between taxa with increased probability for closely related taxa[3], or use in silico evolution models to generate realistic networks [e.g., vDMCH19, KMC+07].

7.4.3 Better exact algorithms

Of course, the goal is not to have an efficient heuristic, but an efficient algorithm for computing exact distances. For this, we may need a more abstract characterization of the exact distance between two networks. There are currently no exact characterizations of distances between networks given by rearrangement moves. A first attempt was recently made using agreement graphs, but this approach currently only yields constant factor approximations for the distance between two networks which cannot be calculated in polynomial time (unless P=NP) [Kla18a, KL18, Kla20a].

Reduction rules

Alternatively, we can try to improve the exhaustive search approach. The breadth first search presented in this chapter is quite inefficient. This is already true when applied to trees (results not shown). For trees, however, many

[3] I have implemented two such generators, but neither is published. The first is an extension of the Heath model for tree generation with autocorrelated speciation and extinction rates [HZKH08]. The generator uses this model and adds hybridization rates that depend on the current distance between each pair of taxa; the second network model uses a beta-splitting model [Ald96] for tree generation—which contains the ERM and PDA models as special cases—and adds reticulation edges to the tree where the probability of a reticulation arc between closely related taxa is larger than between distant taxa. Both can be found at `https://github.com/RemieJanssen/NetworkGenerators`.

improvements have been made, which make an exhaustive search much more effective.

First, we note that several common reduction rules for the problem on trees, such as subtree and cluster reduction [BS05, BSTW17], are not always safe when adapted for networks. For subtree reduction, this means that the tail move distance may increase after removing a common subtree. As an extreme example, consider the networks in Figure 7.12, which are three tail moves apart. After reducing the subtrees on $\{y, z\}$ to a single leaf there is no tail move sequence between the networks anymore.

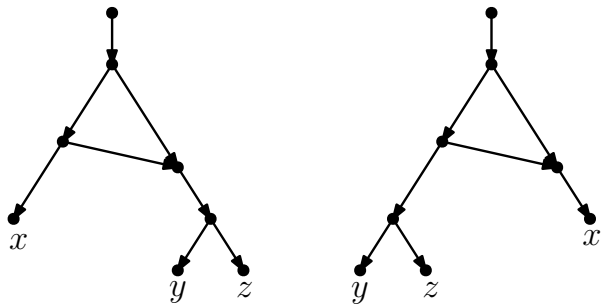


Figure 7.12: Two networks for which subtree reduction does not preserve tail move distance.

The same example shows that cluster reduction does not conserve tail distance: The sum of the tail distance within the common cluster $\{y, z\}$ and the tail distance for the networks resulting from the removal of this cluster does not give the tail distance between the original networks. Here, we use a non-standard definition of common cluster which seems necessary for rearrangement moves. A common cluster of two networks N and N' is a pair of pendant subnetworks M, M' of N and N' such that $X(M) = X(M')$. To make sure a sequence of moves between M and M' exists, we additionaly require that $r(M) = r(M')$.

Note that these examples only prove that these reductions fail for tail moves. They may still work for rSPR or head moves. Furthermore, Lemma 4.5 of [JJE+18] indicates that reducing "tree clusters" might still be possible for tail move distance, when the networks after removing the tree clusters do not become networks with 2 leaves and one reticulation. In fact, it could even still be safe to reduce subtrees to size 2.

Nevertheless, it is even easier to see that cluster reduction fails for head moves. Indeed, we can consider a pair of networks with two clusters: one cluster consists of two non-isomorphic trees, and the other cluster consists of

two isomorphic networks with at least one reticulation. There is a sequence between the two networks, but there is no head move sequence between the non-isomorphic trees that form one of the clusters.

A* search

As an alternative to breadth first search, we could attempt to design a progressive A* search, like the one for trees introduced by Whidden and Matsen [WM18]. To implement such a search, we need quickly computable upper and lower bounds. As we have shown in this chapter, our heuristics give reasonable upper bounds quickly. These can be made progressively better by repeating the random heuristic an increasing number of times.

Currently, some lower bounds are known, but none of these can be calculated efficiently. For example, we can obtain lower bounds from the sets of embedded trees. Indeed, to make two networks isomorphic, we in particular need to make sure the sets of embedded trees are equal. As each rSPR move changes each of the embedded trees by at most one rSPR move (Lemma 7.2), this results in the following lower bound.

Lemma 7.25. *Let N and N' be two networks in the same tier, and $\mathcal{T}$ and $\mathcal{T}'$ their sets of displayed trees. Then the following bound holds for all $T \in \mathcal{T}$*

$$d_{\mathrm{rSPR}}(N, N') \geq \min_{T' \in \mathcal{T}'} d(T, T'),$$

and therefore

$$d_{\mathrm{rSPR}}(N, N') \geq \max_{T \in \mathcal{T}} \min_{T' \in \mathcal{T}'} d_{\mathrm{rSPR}}(T, T').$$

Unfortunately, calculating such a lower bound entails calculating about 2^k (first bound) or 2^{2k} (second bound) distances on trees. As this problem is NP-hard as well, there is no fast way to obtain these bounds. However, Whidden et al. [WM18] have implemented quickly computable lower bounds for the rSPR distance between trees, such as a 3-approximation for the rSPR distance between two trees that runs in linear time [WZ09]. Using this linear time approximation algorithm, we can get a $O(2^k n)$ time lower bound for $d_{\mathrm{rSPR}}(N, N')$ by calculating the 3-approximation for $d_{\mathrm{rSPR}}(T, T')$ for a fixed $T \in \mathcal{T}$ and all $T' \in \mathcal{T}'$, each in $O(n)$ time. As long as k is small, this lower bound can be computed efficiently. For better lower bounds, one could also use the cubic time 2-approximation for tree rSPR distance by [CHW17].

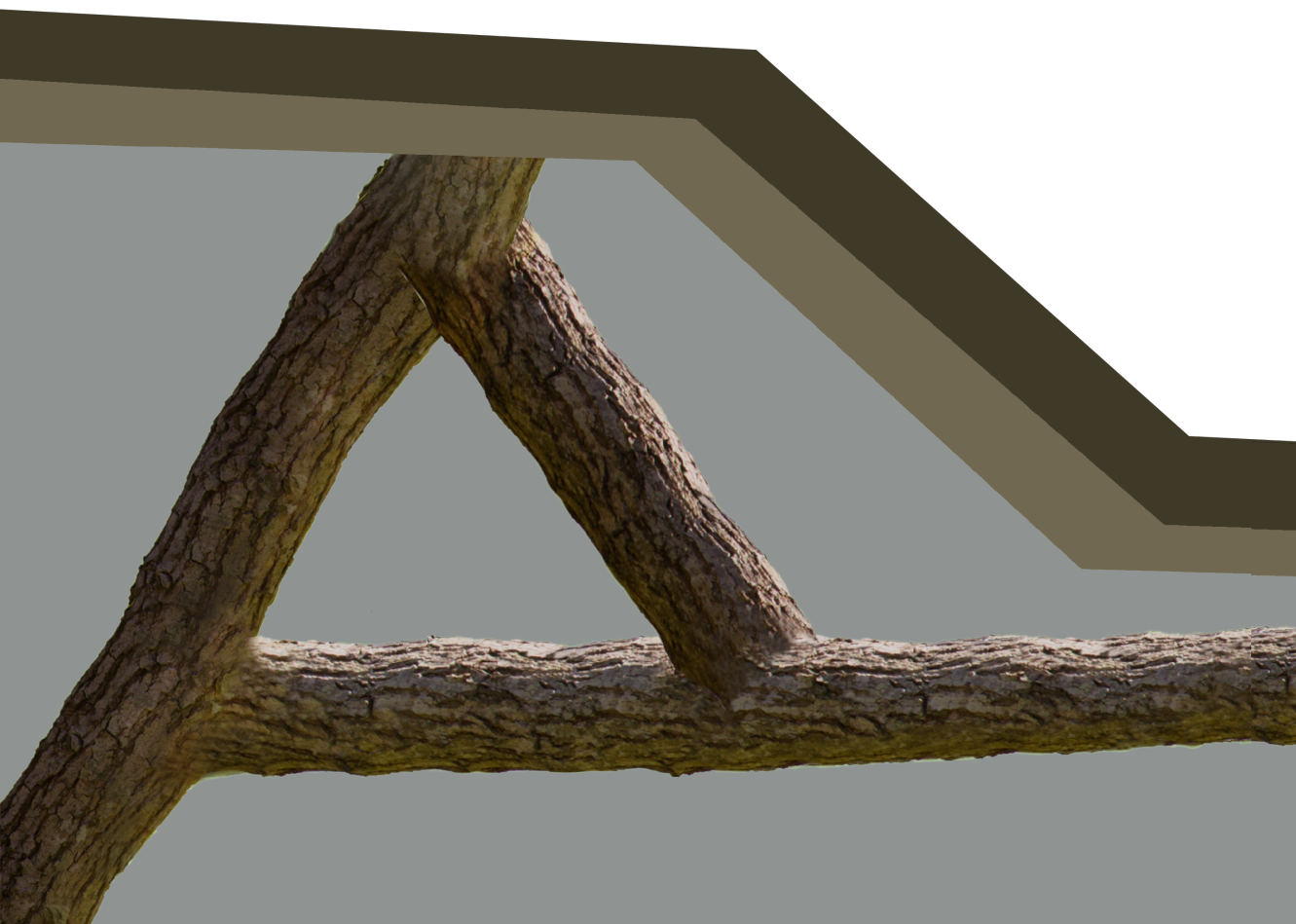
A second candidate for a lower bound is based on agreement graphs. The agreement distance between a pair of directed networks is a 3-approximation for their SNPR distance [Kla18a]. Hence, a lower bound on the agreement distance also gives an approximate lower bound for the SNPR distance. Unfortunately,

computing the agreement distance is NP-hard. Therefore, to make this work, we need an algorithm to compute a lower bound for the agreement distance.

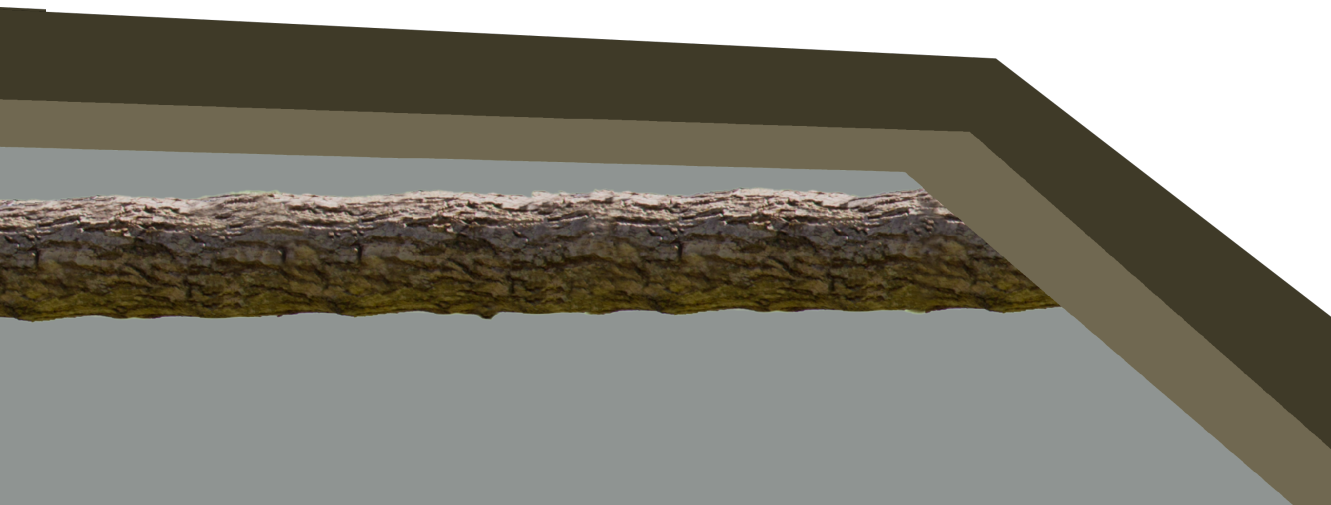
Of course, it may be possible to find lower bounds using different techniques. For example, it may be useful to consider some distance between the *blob trees* of the networks, where the blob tree of a network is the tree obtained by identifying all nodes of each blob of the network. Again, to make the networks isomorphic, we in particular need to make the blob trees isomorphic. To find such a lower bound, we need to study the what effects a move can have on the blob tree of a network.

We conclude that we do not have an efficient algorithm for rearrangement distance problems yet. However, the results in this thesis provide a strong basis on which such algorithms can be based.

8



Discussion



In this chapter, we will summarize all results from this thesis—including three tables with all our diameter upper bounds—and discuss the results from this thesis in light of applications to biology and to other mathematical research fields. In particular, we will discuss the interpretation of a phylogenetic network in a biological context, and the actual use of rearrangement moves in commonly used software for phylogenetic networks. In this discussion, we will encounter several open problems, which can also be found in the comprehensive list of open problems in Appendix B. Overall, it will become clear that our connectedness results suffice to justify the selections of rearrangement moves used in several software tools.

8.1 Overview of the results

Although this thesis is quite long, its results can easily be summarized. The central results all pertain to diameter upper bounds, which are summarized in Tables 8.1, 8.2, and 8.3. Since we have seen that not all spaces are connected, the tables show the diameters of the connected components. For networks without internal labels, all spaces are connected except for $\mathcal{N}(2,1)$ under tail and tail_1 moves. For internally labeled networks, there are several more exceptions, which all concern the highest tree node for head moves, and the lowest nodes for tail moves. For degree-2 nodes there are similar exceptions. Note that, for rSPR and rNNI moves, all spaces are connected, except $\mathcal{N}(1,0,m)$ for $m > 0$.

Although these diameter bounds were the main focus of the thesis, there are several other results that deserve to be mentioned. First, disregarding the internal labels, each tail move can be replaced by a constant length sequence of head moves, and (if $n > 1$) the converse holds as well. Secondly, embedded trees of a network cannot change by more than one move for each move on the network. This result was used to find lower bounds for the distance between networks, and to prove NP-hardness of M Distance Tier-k. Lastly, computing exact distances is generally NP-hard, and our heuristics (`https://github.com/RemieJanssen/RearrangementHeuristics`) work quite well for rSPR and tail moves for networks that are a small distance apart.

8.2 Revisiting networks

Our definition of a network contains several assumptions, some of which are common throughout phylogenetic network literature and others are less agreed upon. The latter category obviously deserves attention, but, as we will argue,

Table 8.1: Upper bounds for the diameters network spaces without internal labels. Note that among these, the only disconnected spaces are $\mathcal{N}_{\text{tail}}(2,1)$ and $\mathcal{N}_{\text{tail}_1}(2,1)$.

	$\text{diam}_M(n,k)$	
M	asymptotic	upper bound
tail	Theorem 3.8 $\Theta(n+k)$	$2n+5k-2$
head	Theorem 4.23 $\Theta(n+k)$	$6n+6k-4$
rSPR	Theorem 5.19 $\Theta(n+k)$	$2n+3k-2$
SPR	Theorem 6.12 $\Theta(n+k)$	$n+\frac{8}{3}k$
tail_1	Corollary 3.12 $O((n+k)^2)$	$(2n+5k-2)(n+3k-1)$
head_2	Corollary 4.17 $O((n+k)^2)$	$12n^2+10kn+10k^2+52n+2k+13$
rNNI	Theorem 5.23 $\Theta(n\log n+k\log k)$	$11n+14k-4+2(k+2)\log(k+2)+(n+1)(1+\log(n+1))$
NNI	Theorem 6.14 $\Theta(n\log n+k\log k)$	$12n+20k-15+2(k+2)\log(k+2)+n\log n$

Table 8.2: Upper bounds for the diameters of the (largest components of) network spaces with internal labels. Note that not all these spaces are connected. For each move, the upper bound is given twice: once in terms of the diameter without internal labels, and once written out completely.

	$\mathrm{diam}_M(n,k,0)$	
M	asymptotic	upper bound
tail	Theorem 3.23 $O(n+k^2)$	$2\,\mathrm{diam}_{\mathrm{tail}}(n,k)+5k^2+4(n+k-1)$ $5k^2+8n+14k-8$
head	Theorem 4.31 $O((n+k)^2)$	$2\,\mathrm{diam}_{\mathrm{head}}(n,k)+4k+8(n+k-1)^2$ $8n^2+16nk+8k^2-4n$
rSPR	Theorem 5.30 $\Theta(n+k)$	$\mathrm{diam}_{\mathrm{rSPR}}(n,k)+4n+8k-4$ $6n+11k-6$
SPR	Theorem 6.17 $\Theta(n+k)$	$\mathrm{diam}_{\mathrm{SPR}}(n,k)+4n+8k-8$ $5n+\frac{32}{3}k-8$
tail_1	Theorem 3.23 $O((n+k)^2)$	$2\,\mathrm{diam}_{\mathrm{tail}_1}(n,k)+5k^2+(2n+6k)(n+k-1)$ $6n^2+30nk+41k^2-10n-28k+4$
head_2	Theorem 4.31 $O((n+k)^3)$	$2\,\mathrm{diam}_{\mathrm{head}_2}(n,k)+2k^2+2k+(2n+6)(n+k-1)^2$ $2n^3+10n^2k+14nk^2+6k^3+20n^2+4kn+10k^2+106n+12k+26$
rNNI	Theorem 5.30 $O((n+k)^2)$	$2\,\mathrm{diam}_{\mathrm{rNNI}}(n,k)+2n^2+8nk+8k^2-2n-4k$ $2n^2+8nk+8k^2+(4k+8)\log(k+2)+(2n+2)(1+\log(n+1))+20n+24k-8$
NNI	Theorem 6.17 $O((n+k)^2)$	$\mathrm{diam}_{\mathrm{NNI}}(n,k)+(n+2k-2)(4n+6k-2)$ $4n^2+14nk+12k^2+2n+4k-11+2(k+2)\log(k+2)+n\log n$

Table 8.3: Upper bounds for the diameters of the (largest components of) network spaces with internal labels and degree-2 nodes. Note that not all these spaces are connected. For each move, the upper bound is given twice: once in terms of the diameter without internal labels, and once written out completely.

	$\mathrm{diam}_M(n,k,m)$	
M	asymptotic	upper bound
tail	Theorem 3.40 $O(n+k^2+m)$	$2\,\mathrm{diam}_{\mathrm{tail}}(n,k,0)+5k^2+4n+4k+6m+2$ $7k^2+20n+32k+6m-14$
head	Theorem 4.37 $O((n+k)^2+m)$	$\mathrm{diam}_{\mathrm{head}}(n,k,0)+6m+10$ $8n^2+16nk+8k^2-4n+6m+10$
rSPR	Corollary 5.33 $O(n+k+m)$	$\mathrm{diam}_{\mathrm{rSPR}}(n,k,0)+4n+6k+4m+2$ $10n+17k+4m-4$
SPR	Theorem 6.18 $O(n+k+m)$	$\mathrm{diam}_{\mathrm{SPR}}(n,k,0)+4n+8k+2m+1$ $9n+\frac{56}{3}k+2m-7$
tail_1	Theorem 3.40 $O((n+k+m)^2)$	$2\,\mathrm{diam}_{\mathrm{tail}_1}(n,k,0)+2n+6k+8m-2+m(2n+6k+2m)+2n^2+8nk+11k^2$ $14n^2+68nk+93k^2+2nm+6km+2m^2-18n-50k+8m+6$
head_2	? ?	? ?
rNNI	Corollary 5.33 $O((n+k+m)^2)$	$\mathrm{diam}_{\mathrm{rNNI}}(n,k,0)+4mn+6mk+m^2+2m+2$ $2n^2+8nk+8k^2+4mn+6mk+m^2$ $+2(k+2)\log(k+2)+(2n+2)\log(n+1)+22n+24k+2m-4$
NNI	Theorem 6.18 $O((n+k+m)^2)$	$\mathrm{diam}_{\mathrm{NNI}}(n,k,0)+4mn+6mk+m^2+6k+5m+3$ $4n^2+14nk+12k^2+4mn+6mk+m^2$ $+2(k+2)\log(k+2)+n\log n+2n+10k+5m-8$

so does the former. These assumptions are closely tied to the interpretation of networks as representations of evolutionary histories. Hence, in this section, we revisit the definition and interpretation of a phylogenetic network.

We only focus on biological evolutionary histories for this section. This may seem quite restrictive, as interpretations of networks in other fields such as linguistics and archaeology may require very different assumptions. However, we will see that the biological interpretation of networks already requires us to consider many different variations on the definition of a network. Furthermore, these variations are quite general as they correspond to natural extensions of the mathematical description of a network.

8.2.1 Biological interpretation

In the introduction, we introduced directed networks as a generalization of trees, where the arcs represent periods of descent with modification, tree nodes represent speciation, and reticulation nodes represent combination of hereditary material from two taxa. Unlike for trees, the interpretation of networks depends heavily on the assumptions on the reticulate processes involved.[1]

Perhaps the simplest example is when we assume the only reticulate process is HGT. In that case, there is a direct and unidirectional transfer of information from one group to another. Such a transfer may be singular, or it may consist of a burst of transfers, and it may be between closely related groups, or highly diverse groups [KP08]. These transfers are assumed to mostly occur within Bacteria or Archaea, but also from Bacteria to Fungi [JST15], or even within eukaryotes such as plants [LVK$^+$14, LRM$^+$15, DW04].

Suppose we draw a network for a scenario in which there is a single transfer, with time passing in the vertical direction. The transfer would be represented by a horizontal arc—no time passes—between the two involved taxa. This arc itself does not represent descent with modification, and the new tree node does not represent a speciation event. If we ignore the lengths of the arcs, as we have done in this thesis, we cannot tell which of the parent arcs is a transfer, and which is simply an arc representing descent with modification. Therefore, for these kinds of situations, it is common to use networks with additional structure, such as tree-based networks in which the support tree is

[1] For trees, the interpretation is not always as clear-cut either, as it may not be obvious to decide which lineages should be grouped into one arc of the tree [Kwo11]. For example, when sequence homology is mostly caused by recombination and not by vertical inheritance, the reconstruction of evolutionary histories becomes quite complicated, as arcs in the tree may represent groups that easily recombine, instead of groups related by descent [SFvN19]. This problem is tightly related to the Species Problem of defining a taxon or species [e.g., May97, Hey01, Ere10].

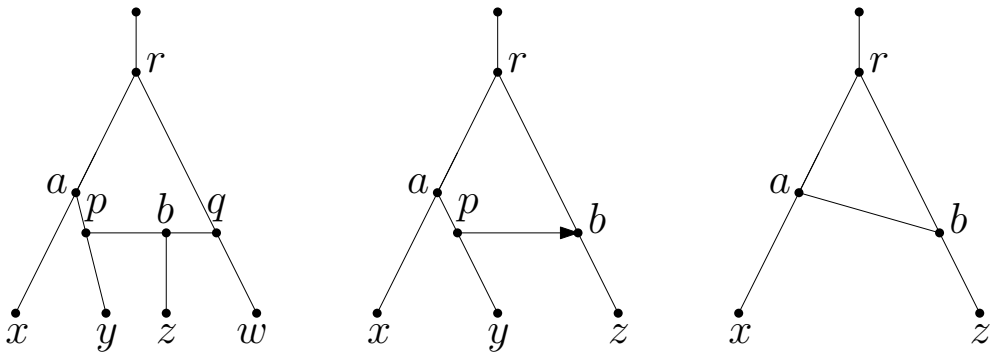


Figure 8.1: Evolutionary histories of incompletely sampled taxa. A network for x, y, z, and w on the left, where z is a hybrid between y and w. In the middle network, the arc (r, b) represents both descent with modification and hybrid speciation. Without knowing that the middle network was sampled incompletely, (r, b) could erroneously be interpreted as descent with modification. The restriction to x and z on the right contains arcs (a, b) and (r, b), which both represents descent with modification as well as hybrid speciation.

given [KGDO05, VWD$^+$17], which has also been studied in the form of *lateral gene transfer* (LGT; synonym of HGT) networks [CPR15].

When considering other processes as well, such as recombination and hybridization in the form of hybrid speciation or hybrid introgression, the interpretation becomes even more muddled. In hybrid speciation, for example, neither incoming arc of the reticulation node represents descent with modification. Hence, to represent an evolutionary history that includes multiple types of vertical processes, we may need to annotate phylogenetic networks with additional information. This information may consist of the types of processes represented by each arc of the network. To annotate a network with such information, additional biological information may be needed, provided it is possible to actually infer such an annotation from biological data.

To make it even more complicated, one could consider incomplete sampling of the involved species. Consider a tree-like evolutionary history (Figure 8.1) for three species x, y, and z, where x and y are most closely related, and a HGT from a recent ancestor of species y to a recent ancestor of species z occured. If we draw the network for only the taxa x and z that fully captures the flow of information, there should be an arc from the LCA of x and y to the very recent ancestor of z that received the genetic material. This arc is not horizontal, as it represents descent with modification as well as HGT.[2] Not only does this

[2]The contrived network shown in Figure 1.2 of the introduction suffers from many of

mean the interpretation of a network becomes more complicated, it also means that using tree-based or LGT networks to represent evolution with HGT may be too limited.

Another complication is that this kind of incomplete sampling can lead to parallel arcs in the true phylogeny, for example by restricting any of the networks in Figure 8.1 to only the taxon z. Note that incomplete sampling like this can occur simply when a too small number of species is studied, but also when some of the lineages have become extinct.

8.2.2 Network definitions

Parallel arcs

In our definition of a network, there cannot be parallel arcs. This can easily be defended, as parallel arcs do not give any information about the evolutionary relationships between the studied species—the same holds for all blobs with two incident cut-edges for that matter. Moreover, in some settings a network with parallel arcs is indistinguishable from the same network where the parallel arcs are removed [GL18, SLCA20]. However, as we have seen in the previous subsection, it may be necessary to include parallel arcs to fully show the phylogeny of the studied species, especially when arcs have lengths as well [PS15]. It is therefore easy to argue for as well as against the use of parallel arcs.

Spaces of directed networks with parallel arcs were first studied in [BLS17]. However, they included vertical moves for their investigation of the full space of networks (these vertical moves were excluded when studying certain classes of networks). Hence, the connectedness of the tiers of network spaces with parallel arcs is still an open problem.

Although we have worked with networks that do not include parallel arcs, it is easy to generalize our connectedness results to networks that do include parallel arcs. To remove parallel arcs, one can use the same techniques as used to remove parallel paths for tail_1 moves (Lemma 3.24) and head moves (Lemma 4.33). This shows that tail and head move spaces $\mathcal{N}^{()}(n,k)$ of networks with parallel arcs are connected if their counterparts $\mathcal{N}(n,k)$ without parallel arcs are connected. Moreover, $\mathcal{N}^{()}(n,k)$ may even be connected if $\mathcal{N}(n,k)$ is disconnected—which is the case for $\mathcal{N}_{\text{tail}}(2,1)$—and similar results also hold for $\dot{\mathcal{N}}^{()}(n,k,m)$.

It is not immediately clear how the diameters of spaces on $\mathcal{N}(n,k)$ and $\mathcal{N}^{()}(n,k)$ relate. On the one hand, using the technique of removing parallel arcs first, we obtain a larger diameter bound for $\mathcal{N}^{()}(n,k)$ than for $\mathcal{N}(n,k)$.

these problems. Indeed, it is unlikely that two groups merge completely as shown in that figure.

On the other hand, when parallel arcs are allowed, the distance between networks without parallel arcs may become smaller than when parallel arcs are not allowed. Asymptotically, the diameters of $\mathcal{N}(n,k)$ and $\mathcal{N}^{()}(n,k)$ will most likely be the same, as it takes only a linear number of moves in n and k to remove parallel arcs for most types of moves.

Internal labels and degree-2 nodes

This thesis contains the first results on rearrangement of internally labeled phylogenetic networks. We have chosen to generalize to such networks for three reasons: mathematical generality (why only label the leaves?), the existing research on trees that uses internal labels (e.g., [BBDVP19, BDEM$^+$20, JBZ20]), and the possible applications for such networks in research where ancestral data is available. Using these internally labeled networks, we can label ancestral taxa in a reticulate evolutionary history as well.

In the absence of degree-2 nodes, the following problem arises: it is unlikely that an ancestral taxon should always be located exactly at a speciation or reticulate event. Furthermore, using the moves as defined in this thesis, if we choose to assign an ancestral taxon to a speciation node, a local search cannot change this assignment to a reticulation node. This justifies the introduction of labeled degree-2 nodes.

It could then be argued that only leaves and degree-2 nodes should be labeled; one could go one step further by arguing that degree-2 nodes should not be used, and ancestral taxa should be represented by regular leaves. The former, only labeling leaves and degree-2 nodes, is quite reasonable for the reasons mentioned in the previous paragraph. Moreover, most spaces of networks with only labels for degree-2 nodes and leaves are connected. This follows by a restriction of our connectedness results for $\mathcal{N}(n,k,m)$: simply ignore the labels of reticulations and speciation nodes.[3]

The second, representing ancestral taxa by leaves, is reasonable only under certain restrictions. If all ancestral taxa are represented by degree-2 nodes, they must necessarily all be an ancestor of at least one leaf (i.e., one of the studied extant taxa). It seems unlikely that this is always the case, and an ancestral taxon (for example as found in the fossil record) may just as well be part of an extinct clade. In such a case, it is not an ancestor of any extant taxon, and must be represented either as a leaf, or as an internal node which is ancestral only to extinxt taxa. The blade cuts on both sides in this case, as

[3] In fact, the results similarly generalize to spaces of networks where a subset of the nodes is non-bijectively labelled, as long as the pre-image of each label consists of nodes that are all of the same type.

by representing an ancestral taxon with a leaf, it can never be an ancestor of any extant taxon in the network.

For the purposes where networks with branch lengths are used, this can easily be solved using these branch lengths. Indeed, we can always represent an ancestral taxon using a leaf, and set the length of its pendant branch to zero to effectively make it a degree-2 node. This immediately changes the moves one can do in the network, as these 'degree-2 nodes' can now be moved, too. For example two adjacent 'degree-2 nodes' can be swapped using one rNNI move, and 'degree-2 nodes' can even be combined with other nodes: if a length-0 arc is moved to another length-0 arc, their pendant leaves are essentially in the same place in the network. Whether this last option is desirable should be determined for each application separately.

While this workaround is useful for methods based on explicit evolutionary models, such as likelihood methods, it does not work for parsimony methods. Consider the tree consisting of a root, one leaf, and one degree-2 node. If the state of the root and the leaf are A, and the degree-2 node has state C, then the parsimony score of this tree is 2. However, if the degree-2 node is replaced by a leaf, where the state of the degree-2 node is moved to the leaf, the parsimony score becomes 1. This shows that the representation of ancestral states needs to be chosen quite carefully.

One solution to all these problems, is to introduce additional moves that can change the labeling of a network more liberally. One can, for example, combine any of the moves studied in this thesis with a move that swaps the labels of two internal nodes, or with a move that attaches a leaf to a degree-2 node and relabels the new leaf with the label of the degree-2 node. Connectedness of the space of networks $\mathcal{N}(n, k, m)$ under either of these extended move types directly follows from the connectedness of the corresponding space on $\mathcal{N}(n, k, m)$. However, the natural set of networks for these moves is not $\mathcal{N}(n, k, m)$, but one where a larger set of permutations of the labels should be allowed, because, for example, leaf-labels can become degree-2 labels and tree node labels can become reticulation labels. For these larger spaces, connectedness likely still follows from the previously mentioned connectedness results, but one needs to be careful and prove that each permutation of the labels can be achieved using the new set of moves.

Multi-rooted networks

In research into rearrangement moves, it is always assumed that the network has one root. This makes sense in most cases, because, generally, a well spread sample of taxa is chosen, and we can assume there is some ancestor of all these taxa that nicely fits in the evolutionary history we want to show. However,

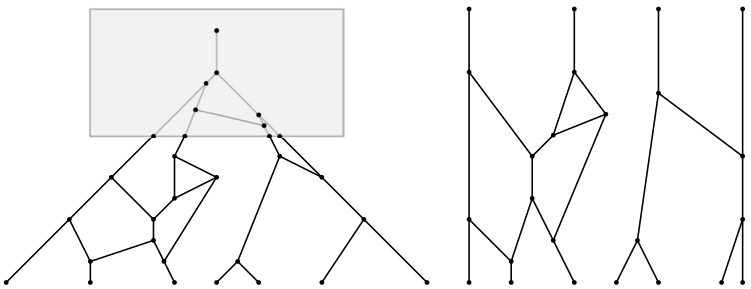


Figure 8.2: Removing the top part (grey box) of a network leaves a multi-rooted network with multiple components, as shown on the right.

it is also common to study small parts of the full network of life, concerning, for example, a set of very related taxa, or two very distantly related groups where we are looking for the most likely HGT event. In such cases, we may want to fix the network structure representing the evolutionary history in the distant past. In that case, rearrangement moves may not change this part of the network, and we could also remove it. This leads to networks with multiple roots (Figure 8.2).

The advantage is that one does not have to make assumptions about how these roots are connected higher up; we do not have to assume the evolutionary history before the existence of these root genes or species [HJH+13]. Additionally, a famous but slightly dated view of the evolutionary history is the net of life by [Doo99], which features multiple roots. A third reason becomes apparent when we take a broader view of phylogenetic networks that includes pedigrees: these often start with multiple individuals that may coalesce in the distant past.

As can be seen in Figure 8.2, removing the top part of the network may create multiple components and roots—these roots are labelled to show where they connect to the fixed part of the network. This prompts the need to consider networks with multiple roots and multiple components. One could argue that we could simply connect all these roots at the top, but this would add no information, and it would appear to represent some history which we are not actually sure of.

There is an additional mathematical reason for studying multi-rooted networks. We have seen that reversing the direction of all arcs in a network with one leaf gives a new network. Each tail move (head move) in the original network becomes a head move (tail move) in the new network. When we allow multiple roots, this works for all networks; each root becomes a leaf and vice versa. Hence, to unify the results for tail moves and head moves, one could consider head or tail moves in multi-rooted networks. Connectedness results for head moves and for tail moves can then easily be related.

Polytomies and non-binary networks

Another assumption we have made for the definitions of networks, is that they are binary. This assumption is quite common in mathematical phylogenetics, where it is argued that non-binary nodes can simply be resolved to obtain a binary network. It is sometimes (tacidly) assumed that the real network or tree must be binary [e.g., in Gra89, WZB14], but this goes against the evidence that hard polytomies exist [RC06]. These are speciation events into three or more lineages, which cannot be represented by a binary node. The existence of these hard polytomies cannot be ignored when using local search heuristics, even for trees [WM10]. This means we should also consider non-binary networks and rearrangement moves for these networks.[4]

It is not directly clear how our moves should generalize to non-binary networks, especially when internal labels are involved. In the absence of internal labels, one option is to allow for the reattachment of endpoints to existing nodes. This means that we should also allow for the pruning of one arc to remove it from a polytomy, and we must suppress nodes after pruning only if this results in a degree-2 node. Using these generalizations, high degree tree nodes can be resolved using tail moves, and high degree reticulations using head moves. Note that the degree of a tree node (resp. reticulation) cannot be changed by a head move (resp. tail move). Another option is to introduce an additional move that can contract arcs or resolve nodes, which makes it possible to change the degree of a tree node or reticulation regardless of the rearrangement move used.Connectedness of such spaces (using the generalized rSPR move, or any move together with the contraction/resolution move) is easily proven: simply use the connectedness results for binary networks after resolving each polytomy.

It is unclear how these generalizations should behave when internal labels are involved as well. For example, what should happen to the labeling when an endpoint moves away from a labeled high-degree node. Should the moving arc always carry the label with it, or only when the node of origin is suppressed, or never, resulting in a labelled degree-2 node? If it does carry a label with it, can we merge two labelled nodes to create a node with higher degree and multiple labels? In these cases, it seems the solution used by [BBDVP19] for trees may work, although their move is not directly related to classic tree moves like rNNI and rSPR moves on binary trees. It would be interesting to explore the connections between their rearrangement move, and the ones used in this thesis in more detail.

[4] The existence of hard polytomies actually only necessitates the introduction of high degree tree nodes, and not high degree reticulations, but it seems natural to consider those as well.

Classes of networks

Finally, we note that most of the adjustments proposed in the previous section generalize the definition of a network to include more types of structures. However, it is also quite common to restrict the structure of the networks. This can be done to make the search space smaller, or simply because one expects a network with a specific structure. This leads to subspaces of networks formed by classes of networks. Bordewich et al. [BLS17] first studied such spaces. An extensive overview of the connectedness of these subspaces of networks for a given class can be found in [Kla20b]. Hence, we do not repeat such an overview here.

8.3 The use of rearrangement moves in software

To see how our results apply to existing methods that reconstruct evolutionary histories, we discuss the use of rearrangement moves in existing software packages. As we have argued in the introduction, it is important that the search spaces of these methods are connected. Nevertheless, most of the papers accompanying such software packages neglect to investigate the search space they use.

In Appendix A, we study the connectedness of search spaces of several software packages, including some functions within the PhyloNet package [TRN08], some BEAST 2.5 packages [BVBS+19], but also other methods like GTmix [Wu20] and RF-Net [MAVE19, ME19, Mar20]. Among these, we found one method (GTmix) where the search space was highly disconnected, and consisted of many components. This shows that rearrangement moves should be chosen carefully when designing new methods for reconstructing phylogenetic networks.

All other methods we have investigated in Appendix A had connected search spaces, which could be explained by the large set of moves they used. These sets often contain some rSPR-like move, where each method uses its own variation. By modifying our results of the connectedness of tiers under rSPR moves, it can be shown that the search spaces of most of these software tools (except for BACTER, which uses a different type of network definition, and GTmix, which has a disconnected search space) have connected tiers.

Additional to these rSPR-like moves, most methods also use vertical moves and other horizontal moves. These horizontal moves change the direction of an arc, swap the endpoints of two arcs, or they move an entire arc. The use of vertical moves generally make the spaces connected via trees and the use of rSPR-like moves make the tiers connected as well.

8.3.1 Move selection for heuristics

The fact that all methods use their own versions of an rSPR move makes it very hard to compare the spaces of networks used in all these methods. Moreover, this myriad of moves seems unnecessary if only the connectedness of the search spaces is of interest. However, these could exist a justification by the fact that the underlying optimization criterion or sequence/tree evolution model is different. Nevertheless, it must be noted that, with the exception of the BACTER paper [VWD+17], none of the papers explaining the methods justify the introduction of new variations. Therefore, there does not seem to be a good reason for the proliferation of rSPR-like moves in heuristics. So, why then is a new rSPR-like move invented for each new heuristic?

Interaction of moves and the model

To design a good set of moves, it is important to keep in mind the interaction between the moves and the optimization criterion or likelihood function. According to [VWD+17], which uses a Bayesian approach, moves "should not generate new states that are too bold (accepted with very low frequency) nor too timid (accepted with very high frequency): both extremes tend to lead to chains with long autocorrelation periods." Hence, they constructed moves with two criteria in mind. The first is that they should minimally affect the likelihood of the network, presumably to ensure that the steps are small in light of the evolution model that is used; and the second is that they should "draw any significant changes from the prior", which is probably so that larger steps through the space are justified by the prior distribution. They forego the strategy of using a mix of timid and bold moves, which is used by most of the software tools we have investigated. This strategy, if correctly calibrated, should on average also generate states which are on average not too bold, nor too timid.

For local search heuristics solving optimization problems, it is also important to understand the interaction between the moves and the optimization criterion. Although Markin et al. have not devised their own moves for the use in their RF-Net software, they have studied the interplay between SNPR moves and clusters, to better understand what local search can achieve in their problem. As the study of rearrangement moves for networks is still young, the interplay between rearrangement moves and network optimization criteria is virtually untouched. For trees, such studies have been done, for example by studying the interaction between NNI moves and parsimony [UFSJ16].

As we have seen above, it is common to use a mix of moves, which may make these interactions even more complicated. There seems to be some hierarchy in

these mixes, as there are clearly moves that can affect the topology much more than others. For example, consider an rNNI move versus an rSPR move. In light of this, it would also be interesting to consider a hierarchy of rearrangement moves, and use them in variable neighbourhood searches, where it is also important to understand the effect of moves on the optimization criterion (see "Intensified shaking" in [HMP10]).

Effects on the running time

Of course, one should not only consider the connectedness of the space when selecting or designing rearrangement moves for a problem. It is also important that the desired solution is found within reasonable time. Indeed, if connectedness were the only condition, one could use the 'move' that simply picks a new network at random from the full set of networks. Of course, for a local search heuristic, this would not work, as it would require the computation of the score of all networks. Hence, it is desirable that the neighbourhood of a network is small, and, if possible, that it primarily contains promising new networks, but it should still be possible to reach an optimal network quickly. Choosing a small neighbourhood so that any network can still be reached quickly is generally possible, as we have shown that the tiers of network space are already connected under tail_1 moves with a quadratic diameter; but choosing a neighbourhood that contains promising networks requires a better understanding of the interplay between moves and optimization criteria, as argued above.

For Bayesian approaches, a similar argument holds: introducing a move that simply picks a new network at random is rather pointless, but moves that cannot make large enough changes would lead to very long convergence times. Here, again, our results show that tail_1 moves could be a reasonable first guess for a good move, as they lead to relatively small neighbourhoods in a connected space of networks with small diameter. Of course, these are not the only desirable characteristics that one should keep in mind, so the effectiveness of tail_1 moves would have to be proven in practice. This can be done either by simply testing methods that use these moves (as is common practice), or by proving facts about the *mixing time* for the Markov chains used in these methods. The mixing time is the number of steps it takes for the infered distribution to get close to the posterior distribution.

As a first approach, the diameter can be used to give a lower bound on the mixing time [LP17, Chapter 7]. Hence, if the diameter is large, the mixing time will necessarily be bad. However, the diameter of a graph does not have a strong connection to upper bounds on the mixing time. In fact, the mixing time may be exponential in the diameter. However, the example to show this is a tree [LP17, Example 7.7], a graph that has low connectivity per definition. There

is some evidence that higher connectivity corresponds to faster mixing. For example, [Sin92] links "canonical paths" to the mixing time, which is interpreted by [MYK10] as a link between connectivity and mixing time. In this light, it would be interesting to study the edge-connectivity of phylogenetic spaces, and keep this in mind when selecting moves as well.

8.4 Concluding remarks

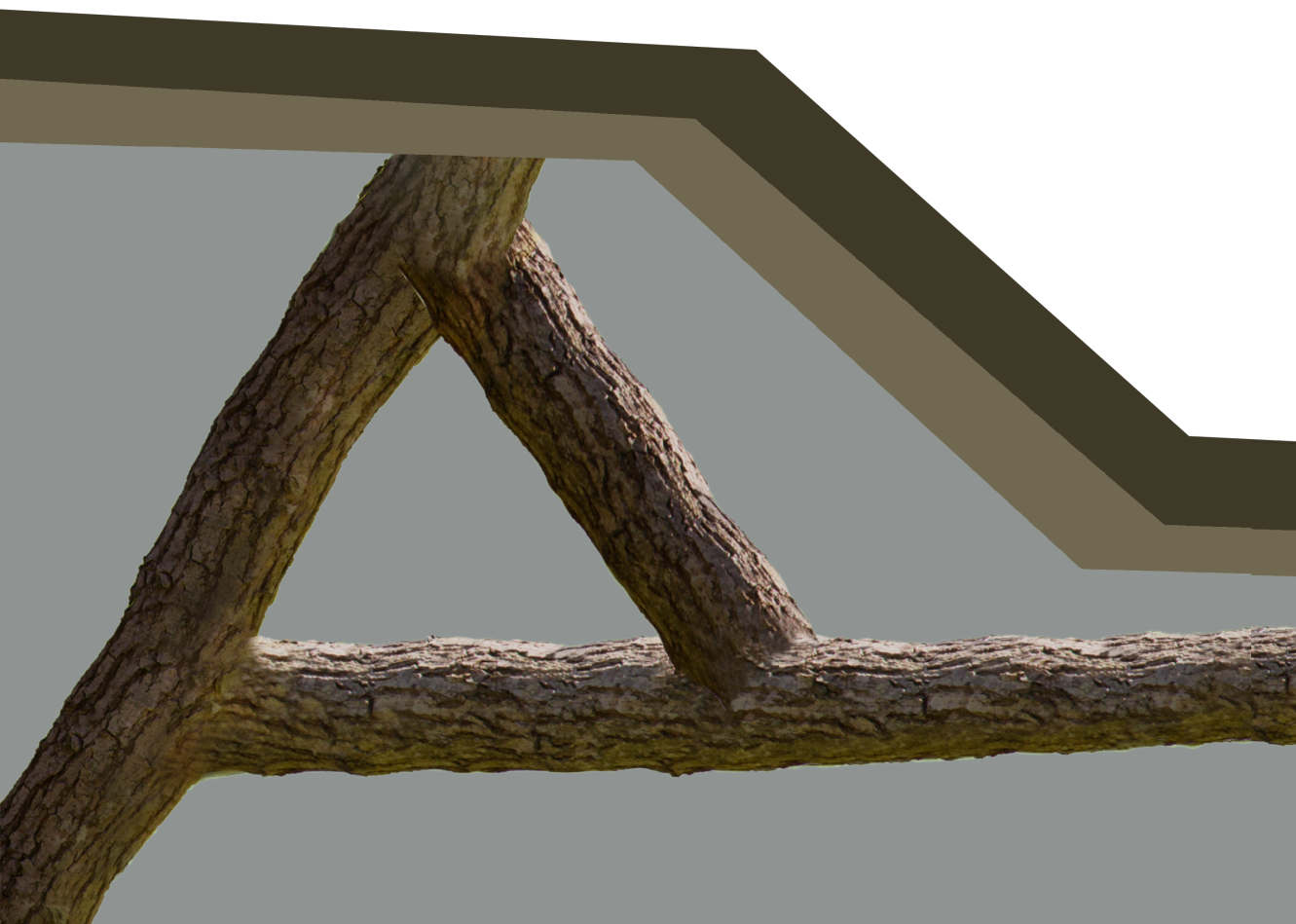
In the introduction, we justified the research in this thesis with the claim that we need to understand spaces of networks for their use in reconstruction methods. However, all the software mentioned in this chapter was tested in practice and seems to function perfectly fine, all without a mathematical proof of the connectedness of these spaces. In fact, in most cases, the moves are chosen so general that, at first sight, it seems unlikely that the corresponding spaces can be disconnected at all. This raises the question of the use of the results in this thesis. They clearly don't make the software more efficient, and they weren't necessary for the methods to be applied to biological data sets.

At first glance, it seems we do not add much to the story, and we have simply checked something that should have been checked by the designers of the software in the first place. In fact, maybe they have, but they did not publish this. Only for the GTmix software we can be reasonably sure that they did not check this, as their space of networks is highly disconnected. This shows that, at the very least, the topic of this thesis is important.

Nevertheless, only focusing on connectedness may be selling the results of this thesis short, and we should consider the results of this thesis in a broader context. As we have argued before, a good understanding of the search space may be important in understanding expected or unexpected output of reconstruction methods. Furthermore, understanding the interaction between properties of networks, such as the sets of embedded trees, and rearrangement moves may help in choosing good rearrangement moves for future applications.

Most importantly, even within the small scope of connectedness results, the results in this thesis are simply necessary. Without these results, many methods used to reconstruct evolutionary histories lacked justification. Hence, this thesis was necessary not for the sake of practicality, but for the sake of scientific rigour.

A



Rearrangement Moves in Software

In this chapter, we investigate the use of rearrangement moves in commonly used software, which are either local search heuristics for optimization problems or Bayesian methods (Section 1.3.1).In particular, we inspect the sets of moves that are employed in their algorithms, paying special attention to the connectedness of their search spaces.

A.1 Move types

To get an idea of the move types used in software, we investigate several (arbitrarily chosen) software packages. These include some functions within the PhyloNet package [TRN08]—interestingly, PhyloNet use different sets of moves in different functions—and the BEAST 2.5 packages [BVBS+19], but we will also consider other methods like GTmix [Wu20] and RF-Net [MAVE19, ME19, Mar20]. We by no means claim to give a full list of network reconstruction methods that use rearrangement moves, as we for example omit SMARTIE [BS10], ClonalOrigin [DLDF10], and AllopNET [JSO13]. None of the publications for the methods mentioned in this paragraph proves the connectedness of the search space, except for the RF-Net software, where connectedness is fully proven.

It will turn out that the connectedness question is easy to answer in many cases, as most of these software packages use vertical moves as well as horizontal moves. Although this makes the connectedness of *tiers* of networks less relevant—connectedness of their spaces follows simply from the facts that each network can be turned into a tree and that tree space is connected [e.g., BLS17]—we still compare their horizontal moves to ours. Because vertical moves seem to be used rarely and a network with fewer reticulations is less likely to explain the input data, the connectedness of tiers may still be important in practice. To validate the methods used in the software packages, we compare the used moves to the moves studied in this thesis.

In most of the software packages, the networks have additional numerical parameters such as the lengths of the arcs and the inheritance probabilities along reticulation arcs. This means these packages also employ *numerical moves* that result in isomorphic networks, where only the numerical parameters have changed. For these moves, unlike for horizontal and vertical moves, we say that they do not change the *topology* of the network. Hence, to check that the space of networks used by the software is connected, it needs to be checked that each network with each set of possible parameters can be reached.

A.2 PhyloNet

A.2.1 MCMC_GT

The first PhyloNet function we consider is MCMC_GT, a Bayesian estimator with gene trees as input data [WYN16]. This methods uses the following seven types of moves: Change-Length, Change-Inheritance, Move-Tail, Move-Head, Flip-Reticulation, Add-Reticulation, and Delete-Reticulation. The first two moves are numerical, Moves 3–5 are horizontal moves, and the last two moves are vertical moves.

It is easy to check that the numerical moves suffice to reach each parameter condition without changing the topology of the network. Hence, we turn to the moves that change the topology, and we check that all networks can be reached. It is important to note that networks in this software tool have a root of outdegree-2, whereas we assume the root has outdegree-1. There is a correspondence between these two representations as sets—simply remove or add this root arc to go between them—which does not necessarily hold for the spaces if the definition of the move is kept the same. We will see that, in this case, the move is defined such that this correspondence also holds for spaces of networks.

The last two types of moves are vertical moves, which are proposed with a probability κ. In the documentation of this software `https://wiki.rice.edu/confluence/display/PHYLONET/MCMC_GT`, there is no mention of a default for κ—it is unclear from the documentation whether this parameter is actually controlled implicitly by the Poisson parameter for the prior for the reticulation number. Hence, it is unclear how big the role of vertical moves is for this method. Therefore, it is unclear how important it is that the tiers of the search space of MCMC_GT are connected. Nevertheless, we now argue why the tiers of network spaces for this software are connected, in case this turns out to be important.

We first note that Move-Tail and Move-Head are equivalent to our tail and head moves. This is non-trivial because of the different types of roots used in MCMC_GT and this thesis: if Move-Tail were defined exactly as the tail move in this thesis, this would result in a different set of neighbours for Move-Tail than for our tail move. However, Move-Tail corrects for the lack of a root arc by providing exceptions to the definition of a move involving the root [WYN16, supplementary material]. A similar inspection shows that Move-Head and our head move are also identical.

The third horizontal move, Flip-Reticulation, is one entirely different from the moves considered in this thesis. A Flip-Reticulation move re-orients an arc (i.e., take an arc (x, y) in the network, and replace it with the arc (y, x)). As

usual, the move is valid precisely when the resulting graph is again a network. If internal labels are used, we have to observe that Flip-Reticulation turns the tree node x into a reticulation and the reticulation y into a tree node. As no internal labels are used in this software tool, this is no issue regarding the connectedness.

In fact, the tiers of the considered search spaces are already connected under Move-Tail and Move-Head moves, so the addition of Flip-Reticulation moves is superfluent for the connectedness of the search spaces. However, this addition may affect distances and diameters. These cannot get much smaller, though, as each Flip-Reticulation of (x, y) can be replaced by a sequence of at most two rSPR moves: head move y up above x, and move it down on the other outgoing arc of x (Figure A.1a). This sequence is valid unless the intermediate network contains parallel arcs (which are not allowed in the software) but, in that case, the Flip-Reticulation can be replaced by at most one rSPR move (Figure A.1b–d).

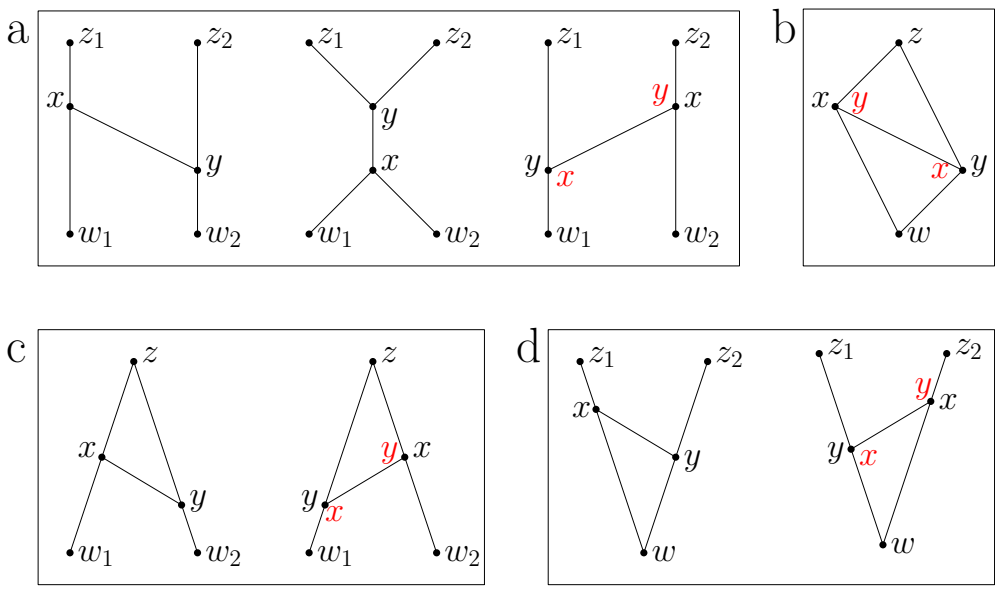


Figure A.1: Replacing a Flip-Reticulation move for (x, y) with at most two rSPR moves. The black labels follow the rSPR moves, and the red labels correspond to the Flip-Reticulation move.

A.2.2 InferNetwork_MP

PhyloNet's InferNetwork_MP is a heuristic for finding a network minimizing the deep coalescence score of a set (or distribution) of trees [YBN13]. These trees are derived from sequences using a parsimony or Bayesian approach, and the score of a network is found by counting the number of extra lineages in an optimal reconciliation of the trees in the network. Because the direct computation of an optimal network is infeasible, a local search heuristic is used to find an optimal network.

The moves used by InferNetwork_MP are quite similar to the ones used in MCMC_GT. The only change is that there is an additional horizontal move: Replace-Reticulation. This move removes a reticulation arc and adds another. This essentially makes it a combination of a tail move and a head move, where the intermediate network is allowed to contain cycles.

Also in this case, the fact that vertical moves are used makes it clear that the space under consideration is connected. However, the search strategy does not allow for going down a tier: all tiers up to a certain reticulation number r are searched in order. First a (locally) optimal tree is found using a hill-climbing approach, then a tier-1 network, and so forth until either a (locally) optimal network is found in $\mathcal{N}(n,r)$ or the (locally) optimal solution in tier-k is at least as good as the score found in tier-$(k+1)$. Because of this strategy, the proof of connectedness via trees is not enough, and we actually need the connectedness of tiers of network space. Therefore, our results show that this search strategy of searching each tier separately may work, although it stays unclear how commonly the hill-climbing heuristic gets stuck in local optima.

For this software, the probabilities of all the move type proposals can be set explicitly as parameters[1]. The default values are $(0.1, 0.1, 0.15, 0.55, 0.15, 0.15)$ for adding a reticulation node, deleting a reticulation node[2], relocating the head of a reticulation edge, relocating the tail of an edge, reversing the direction of a reticulation edge and replacing a reticulation edge respectively. Our results imply it is possible to set these probabilities to zero for the vertical moves, if one desires to search within a tier.

Note that Phylonet's recent package InferNetwork_MP_Allopp [YCLN20] uses the same moves, so our conclusions regarding the search space of InferNetwork_MP can be directly transferred to this method.

[1] See the documentation at `https://wiki.rice.edu/confluence/display/PHYLONET/InferNetwork_MP`.

[2] Why this move is mentioned is unclear, as [YBN13] claims to never reduce the number of reticulations.

A.2.3 InferNetwork_ML and InferNetwork_MPL

The InferNetwork_ML and InferNetwork_MPL methods aim to maximize the likelihood [YDLN14] or pseudo-likelihood [YN15] of a network respectively, given an input of gene trees. They use a hill-climbing heuristic for these problems, which uses vertical moves (removing or adding a reticulation arc) and two types of horizontal moves.

The horizontal moves are head and tail moves. However, the networks have an outdegree-2 root and no correction is mentioned for this like in MCMC_GT (see definition of these moves in the supplementary information of [YDLN14]). Hence, the connectedness of the resulting space does not immediately follow from our results. However, it is easy to see that all head moves and tail moves are (or can be replaced by at most two of these) more restricted tail moves (Figure A.2). Using this obervation, the connectedness of the search spaces in this method follows directly from our connectedness results.

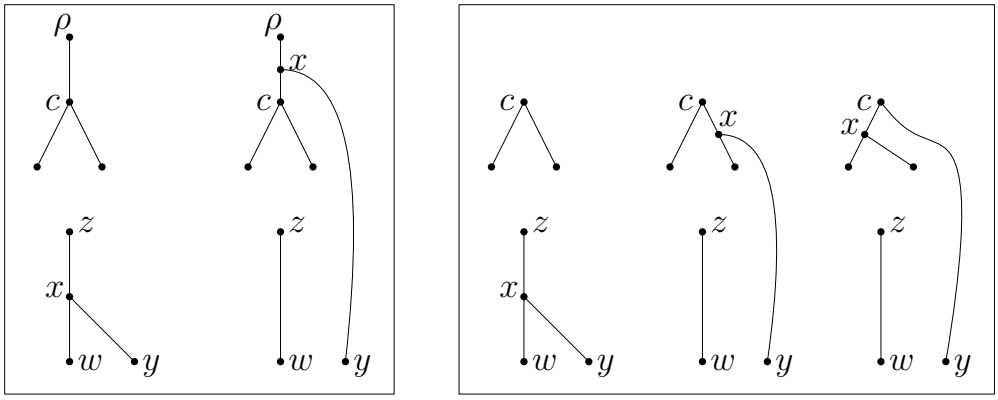


Figure A.2: Tail moves with different types of roots. The tail move on the left is not possible when the root arc is removed to obtain a network with two outgoing arcs as on the right. This move can be simulated using at most two tail moves. Note that if the intermediate network on the right is not valid, then y is the right child of c, but then only one move suffices: $x \xrightarrow{(x,y)} (c, d)$, where d is the left child of c.

Interestingly, the documentation of these methods mention four horizontal moves, and not two.[3] This gives a total of seven moves: adding a reticulation node, deleting a reticulation node, relocating the head of a reticulation edge, relocating the tail of an edge, reversing the direction of a reticulation edge, replac-

[3] `https://wiki.rice.edu/confluence/display/PHYLONET/InferNetwork_ML` and `https://wiki.rice.edu/confluence/display/PHYLONET/InferNetwork_MPL`

ing a reticulation edge and changing branch lengths and inheritance probabilities. The default weights for these moves are $(0.1, 0.1, 0.15, 0.55, 0.15, 0.15, 2.8)$ respectively. Assuming that these methods use the tail and head moves described in the paper, and not another set of moves as mentioned in the documentation, the method indeed uses spaces with connected tiers.

A.2.4 MCMC_SEQ

The last PhyloNet function we consider is MCMC_SEQ, which uses a Bayesian technique to compute a posterior distribution based on sequence data [WN18]. The model of sequence evolution underlying this function takes ILS and hybridization into account in a multi-species network coalescent (*MSNC*) setting.

Like for the previous functions, the paper does mention the used moves, but does not investigate the resulting space of networks. In this case, the software chooses a move from the following twelve types: Scale-PopSize, Change-PopSize, Change-Inheritance, Scale-Time, Change-Time, Swap-Nodes, Flip-Reticulation, Slide-SubNet, Move-Tail, Move-Head, Add-Reticulation, and Delete-Reticulation. The first five of these are numerical moves. Again, it is easy to see that all possible parameter values can be reached for a given network topology, so we turn to the connectedness of the space of networks.

The moves Move-Tail and Move-Head are like the tail and head moves in InferNetwork_ML, in that they use outdegree-2 root nodes. Again, our connectedness results can be leveraged to show the connectedness of the search spaces of this paper. However, we need to be careful, as these Move-Tail and Move-Head take into account the numerical parameters as well: a Move-Head move is a head move $v \xrightarrow{(u,v)} (x, y)$ where $t(y) < t(u)$, that is, if we draw the network with the time on the vertical axis, u must be higher than y.

As all possible numerical parameter values can be reached using the numerical moves, it only remains to prove that there is such a drawing (i.e., set of parameter values) of the network for each head move. This is easy to prove: if $v \xrightarrow{(u,v)} (x, y)$ is valid, then y is not above u, so there exists a total order $t : V \to \mathbb{R}$ such that $t(y) < t(u)$.

This implies the remaining two horizontal moves—Swap-Nodes, which swaps the parents of two randomly selected nodes, and Slide-Subnet, which is a tail move that moves the tail up or down—are not strictly necessary to search the whole space.

Again, the last two moves are vertical moves, that allow for the search to reach trees. Hence, the connectedness is easily proven. Moreover, applying our results about the connectedness of tiers to Move-Tail and Move-Head moves, is is easy to see that a search can be effective even when vertical moves are absent or rare.

A.3 BEAST 2.5

A.3.1 SpeciesNetwork

The BEAST 2.5 package SpeciesNetwork is a Bayesian method based on sequence data that co-estimates a distribution of gene trees as well as a network [ZODS18]. As such, it is much like the PhyloNet function MCMC_SEQ, with four main differences as listed in the paper. Three of these differences concern the evolutionary models and the priors, so they are not directly relevant to this discussion. The fourth difference is that SpeciesNetwork allows for parallel arcs in the networks, whereas MCMC_SEQ does not. Recall from Section 8.2.2 that this is no issue, as parallel arcs can be removed just like parallel paths.

The moves used in this package are also slightly different from the ones used in MCMC_SEQ. In SpeciesNetwork, the moves are: Node Slider, Node Uniform, Inheritance-Probability Uniform, Inheritance-Probability Random-Walk, Relocate Branch, Add-Reticulation, and Delete-Reticulation.

The first four are numerical moves, Relocate Branch is a horizontal move, and the last two are vertical moves. Relocate Branch includes rSPR moves, but also allows for a slightly more general move. This move combines an (invalid) rSPR move with a Flip-Reticulation move. More precisely, if the rSPR move $u \xrightarrow{e} f$ introduces a cycle then this move is followed by a Flip-Reticulation on e. If this results in a network, then this combination of an rSPR and Flip-Reticulation is considered a valid Relocate Branch move. Because this move includes the rSPR move, our results imply that the tiers of the search spaces used by SpeciesNetwork are connected.

A.3.2 BACTER

BACTER [VWD+17] uses a Bayesian approach to find the geneaology of bacterial samples using sequence data and an evolutionary sequence/coalescence model tailored to bacterial taxa. The specific aim is to find a tree-based network, where a given base tree represents the true reproductive genealogy (also called the *clonal frame*) and the reticulation arcs represent recombination.

As the networks are tree based, the types of moves used in the MCMC Bayesian inference are quite different from the moves in this thesis. Instead of moving arcs in the network, they essentially change the base tree, and simultaneously relocate reticulation arcs to conserve the time at which they occurred (See Figure 2 in the paper). Hence, we cannot directly apply our connectedness results to BACTER. Note, however, that BACTER also uses vertical moves, which can turn the network into a tree, and that this tree space is con-

nected simply by tail moves, so it is easy to see that BACTER's search space is connected nevertheless.

A.3.3 CoalRe

CoalRe is a package that uses Bayesian inference to find networks together with embedded trees for genetic segments for viruses [MSD+20]. It tailored to this setting as it uses a coalescent model with reassortment.

In the MCMC process, CoalRe uses the following moves: scale operator, segment diversion operator, add/remove operator, subnetwork slide operator, exchange operator, gibbs operator, empty segment preoperator. Among these, the scale operator and the segment diversion operator are the only moves that cannot change the network topology. The moves in this method are interesting, in that they must jointly change the embedding of the segment trees when the network topology is altered.

For the sake of connectedness, we note that each set of embeddings can be reached in a given network using the segment diversion operator. The question of connectedness then simply comes down to the connectedness of the space of network topologies under the remaining moves. As the add-remove operator is the vertical move used by SNPR moves (see Section 2.4.1), and the exchange operator includes rNNI moves when restricted to trees, it is easy to see this space is connected.

We now check whether tiers of this space are also connected under the given moves. This can easily be seen to be the case, as the subnetwork slide operator is topologically the same as the Slide-SubNet operator in PhyloNet's MCMC_SEQ: it is simply an upward or downward tail move. As all tiers are connected under tail moves (except $\mathcal{N}(2,1)$), and each tail move can be replaced by an upward tail move followed by a downward tail move, each tier is already connected under the subnetwork slide operator alone.

A.4 PhyloNetworks: SNaQ

The SNaQ method (Species Networks applying Quartets) is a maximum pseudo-likelihood method, that computes the pseudo-likelihood of a *semi-directed* network (i.e., only the reticulation arcs are directed) by calculating the likelihoods of all its subnetworks with four leaves [SLBA17].

The search starts in a user specified tree, and then proceeds to search network space using the following moves: 1) move the origin of an existing hybrid edge; 2) move the target of an existing hybrid edge; 3) perform a nearest-neighbor interchange move (NNI) on a tree edge; 4) change the direction of an

existing hybrid edge; and, 5) add a hybridization if the current topology has $k < k_m$, where k_m is the user specified maximal reticulation number considered in the search.

In the first two of these moves, the to-edge is chosen from the 'vicinity' of the current position of the moving endpoint. We assume this means the move is actually an NNI move that moves an endpoint of a reticulation arc. For the third type of move, it is not defined what it means for an NNI move to be performed on a tree edge, but a reference to [HLMW16] is given for a similar definition. There, an NNI on an arc (u, v) is either a tail_1 move $v \xrightarrow{(v,\cdot)} (\cdot, u)$ where u is a tree node, or a tail_1 move $v \xrightarrow{(v,\cdot)} (u, z)$ with $z \neq v$.

If SNaQ uses this definition for NNI moves, then the moves used by SNaQ include our rNNI moves for the following reason. All rNNI moves that move an endpoint of a reticulation arc (either tail or head) are included in the first two types. All other rNNI moves move a tree arc, and must thus be tail_1 moves. The only of these that are included, are of the third type. Clearly, this type of move does not include all rNNI moves as defined in this thesis, as it cannot move a tail down, up to a reticulation arc, or sideways through a reticulation node. However, all these moves result in a network that can also be obtained using different types of moves used by Solís-Lemus and Ané, as can be seen in Figure A.3.

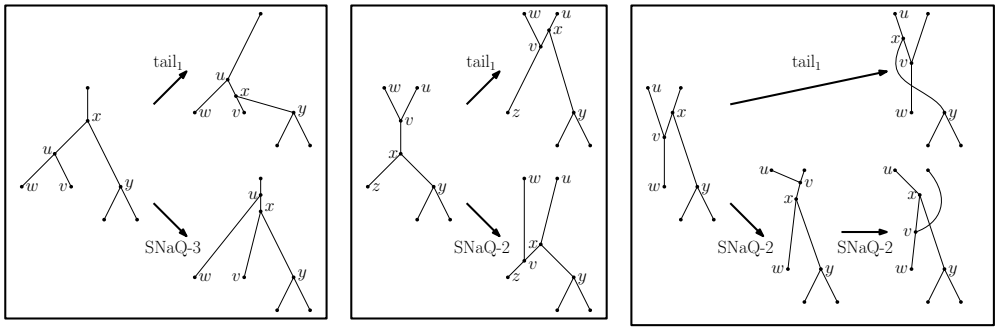


Figure A.3: The rNNI moves not included in the SNaQ moves can all be replaced by moves used by SNaQ. Each upper move is a tail_1 move not explicitly included in the moves used by SNaQ, and the bottom sequence is a simulation of this move fitting within the SNaQ framework.

The direction-changing moves are interesting as well, as they are applied to semi-directed networks. These moves flip the direction of a reticulation arc, so that the head of this arc becomes a tree node and the tail becomes a reticulation node. In a directed network, keeping the directions of all other arcs would fully

define the new network, but for a semi-directed network, the new reticulation node only has one defined reticulation arc after this move. The paper does not specify how the new reticulation arc adjacent to this node is chosen.

The methods restrict to semi-directed level-1 networks (with parallel arcs). For connectedness, the paper mentions Huber et al. [HLMW16], which proves connectedness of the space of undirected level-1 networks under LST moves, a combination of NNI and a vertical move called a *Three-Cycle* operation. According to Solís-Lemus and Ané ([SLBA17]), this suggests that their moves are sufficient. However, this proof of connectedness uses sequences that go via lower tiers (See Fig. 1 in [HLMW16], and observe that the tier-2 networks are only connected to tier-1 networks). Such sequences are unlikely to occur in the search by SNaQ, as the only proposed vertical moves increase the reticulation number. Reticulation arcs are only removed when there is no inheritance of genetic information via that arc (as given by the numerical parameters). Moreover, in Huber et al., networks do not have parallel arcs, so a more complete proof of connectedness of tiers of network space as used by SNaQ is warranted: the results for directed networks in this thesis, together with the fact that a semi-directed network orientable (by the definition in [SLBA17]), imply that the search spaces of SNaQ are connected.

A.5 GTmix

GTmix uses a maximum likelihood approach to find admixture graphs based on local genealogies—which can be thought of as gene trees—in a maximum likelihood setting within a coalescent setting [Wu20]. From a graph theoretical perspective, admixture graphs are the same as phylogenetic networks. The main difference is that admixture graphs are used for smaller scale processes in populations, whereas the same graphs are called phylogenetic networks when they represent similar processes over longer time periods.

Again, directly computing the optimal network is hard, so a hill-climbing heuristic is used. To search the space of networks, GTmix uses rNNI moves of the following type: $v_p \xrightarrow{(v_p,v_c)} (v_q, v_{sp})$, where v_q is a tree node with children v_{sp} and v_p, the latter of which is a tree node with children v_c and v_s. According to the paper, the method only uses these types of moves, and keeps the number of reticulations (admixture nodes) constant.

Interestingly, the resulting spaces of networks are disconnected, with often a large number of components. For example, because leaves cannot move past reticulations (Figure A.4). Although the method is a heuristic and does not claim to find an optimal network, this can still lead to large problems. The paper does claim that the results obtained using GTmix are of a good quality,

and it observes that GTmix is able to work with much larger networks than PhyloNet. Although this is guesswork on my part, this could be a result of using a very limited move, which leads to a small neighbourhood, and a disconnected (and much smaller) search space.

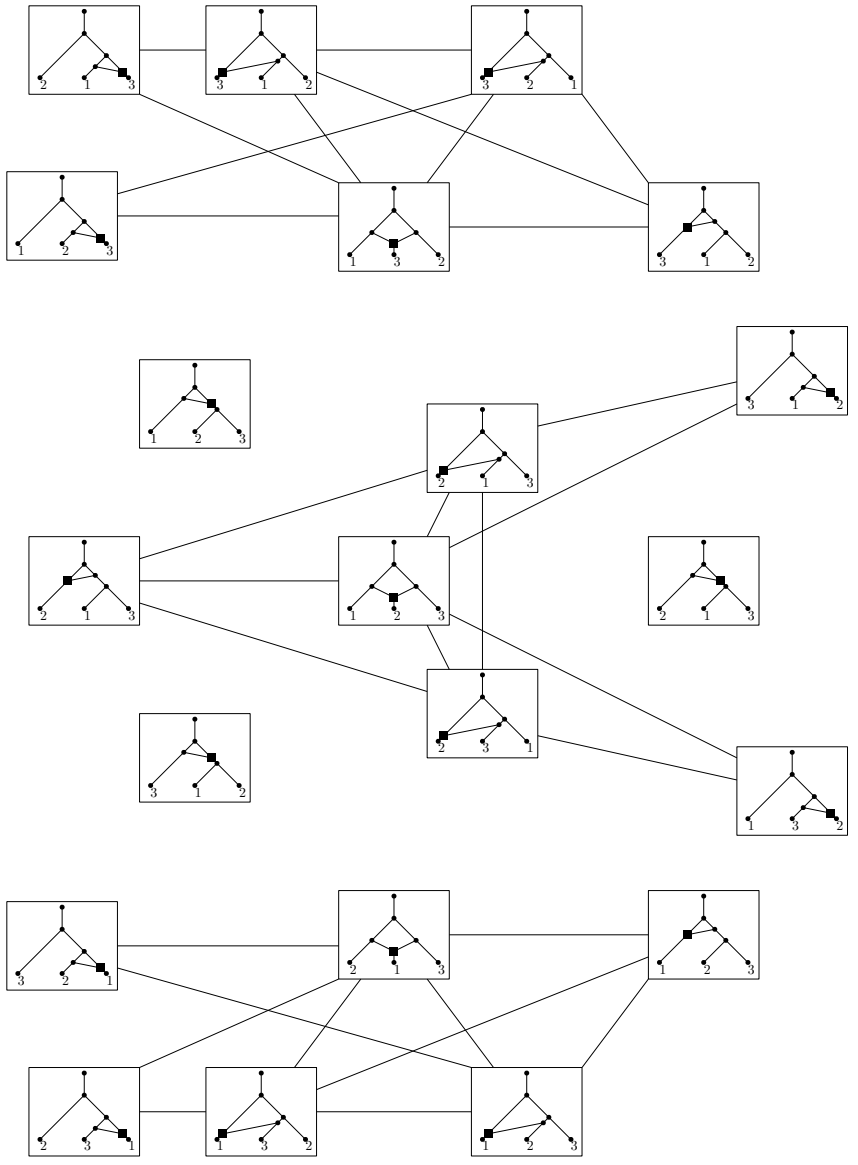


Figure A.4: The space $\mathcal{N}(3,1)$ under the GTmix move is disconnected. Note that the partition of the leaves and tree nodes obtained by removing the reticulations is invariant under the move.

A.6 RF-Net

RF-Net is a method that optimizes a Robinson-Foulds based criterion [MAVE19, ME19, Mar20]. This criterion is

$$\sum_{T \in \mathcal{I}} \min_{T' \in \mathcal{T}(N)} RF(T, T'),$$

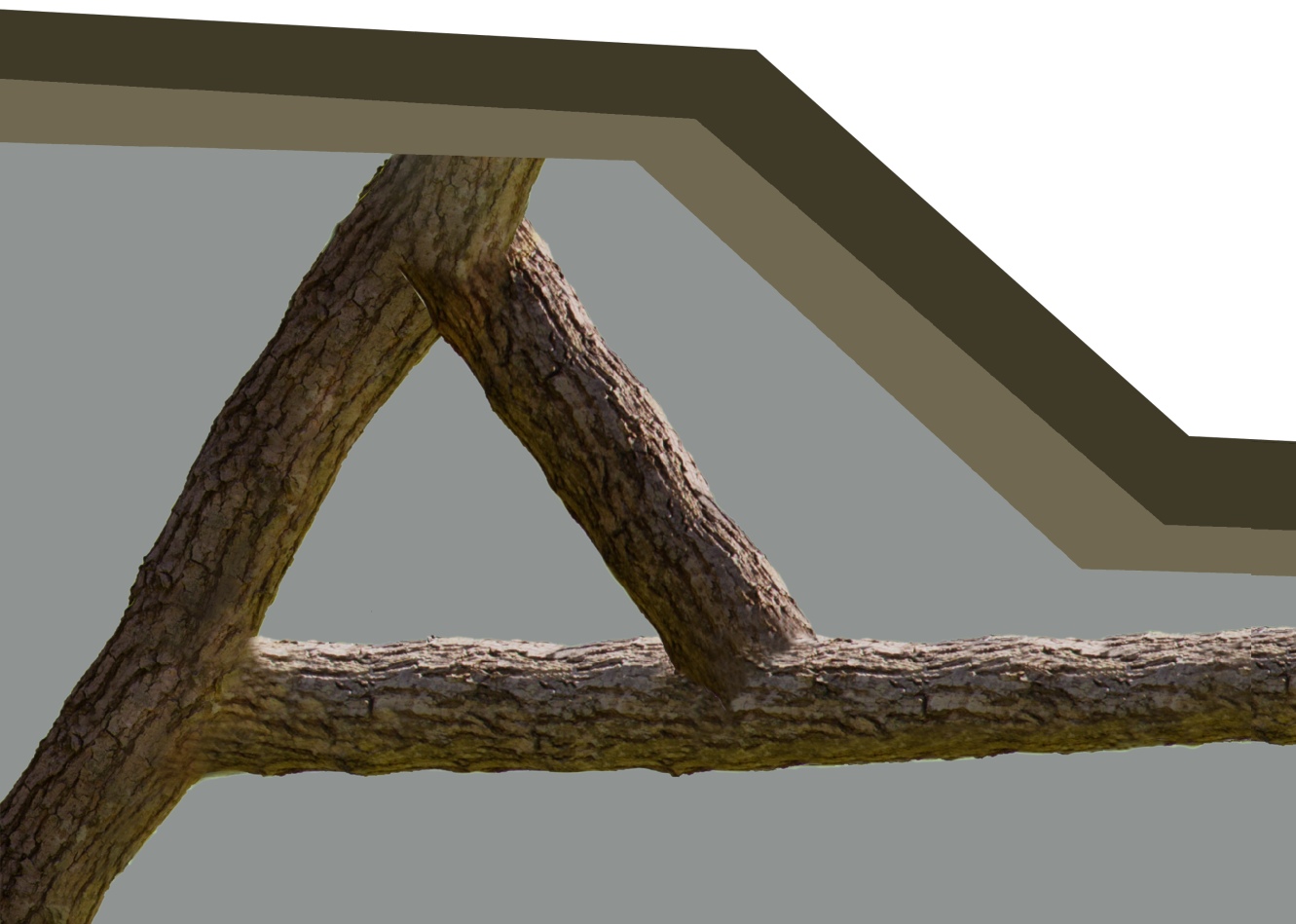
where $\mathcal{I}$ is an input set of trees, and $RF(T, T')$ is the Robinson-Foulds distance between the trees T and T'. In other words, the aim is to find a network that explains the clusters of the input trees the best. Computing the optimal network is NP-hard, so a local search heuristic is used.

The moves used by the local search are tail moves.[4] The paper also mentions distance-1 moves, but it seems these are only used as a conceptual tool, and not as a rearrangement move in the algorithms. All searches are performed within a tier $\mathcal{N}(n, k)$, or in the restriction of $\mathcal{N}(n, k)$ to tree-child networks.

Note that this paper is the only one to fully address the connectedness of the search spaces. This was done by citing [JJE+18] and [BLS17] for the connectedness of $\mathcal{N}(n, k)$ and the tree-child restricted space respectively. This makes this the only paper we found that fully justifies their choice of moves with regards to connectedness of the search space.

[4] In the paper, the moves are called SNPR moves, but only horizontal moves are used.

B



Open Problems

In the discussion of this thesis, we have already mentioned several open problems regarding the study of rearrangement moves. As these were introduced piecemeal, we will now give a more complete overview of such questions. This list is only partly inspired by questions regarding the efficiency of moves in heuristics and Bayesian inferences. A larger part of these questions are based on similar questions for trees, or on open directions following from queries in this thesis.

B.1 Gaps in this thesis

B.1.1 Connectedness

We have fully characterized the connectedness of spaces under tail_1, tail, head, head_2, rSPR, rNNI, SPR, and NNI moves, except for the spaces $\dot{\mathcal{N}}_{\text{head}_2}(n, k, m)$ with $m > 0$. As mentioned in Section 4.4.2, we know these spaces are disconnected, but we do not know to what extent. There are two open problems to this regard.

- Is there a simple characterization for the connected components of head_2 spaces?
- Are all spaces $\dot{\mathcal{N}}(n, k, m)$ connected under head_{m+2} moves?

The reason for the disconnectedness is similar to that for head_1 moves, where a head cannot be moved past its tail. As head_1 moves are a more logical choice for a local move than head_2 moves, it is worth investigating the components of head_1 space as well.

- Is there a simple characterization for the connected components of head_1 spaces?

B.1.2 Diameter bounds

Besides connectedness, the other main focus of this thesis were the diameters of all studied spaces of networks (Tables 8.1, 8.2, and 8.3). Many of these bounds are asymptotically tight—where the asymptotic bound is given in Θ-notation in the tables—even though the lower and upper bounds may be quite far apart. Indeed, we know that our bounds for networks without internal labels are asymptotically tight for all move types except tail_1 and head_2 moves.

For networks with internal labels, the only tight bounds are for rSPR and SPR moves; and for networks with degree-2 nodes, none of our bounds have

been proven tight, as no known bounds include the number of degree-2 nodes. This leads to the following problem regarding asymptotic bounds for the diameters of network spaces, where all problems are still open except for the ones just mentioned.

- Determine asymptotically tight bounds for the diameters $\mathrm{diam}_M(n,k)$, $\mathrm{diam}_M(n,k,0)$, and $\mathrm{diam}_M(n,k,m)$.

We conjecture that $\mathrm{diam}_{\mathrm{tail}_1}(n,k) = \Theta(n\log n + k\log k)$ which may be provable using a technique similar to the one used for rNNI moves. The main steps to reprove are that any network can efficiently be turned into a tree-based network, and that collecting the heads at one pendant arc can be done efficiently.

For internally labeled networks, we conjecture that the diameter bounds for tail and head moves can be improved to linear. This seems likely for the following reason. For tail moves, the quadratic number of moves is a result of the permutation of the reticulations nodes, which can be done using a linear number of head moves. We have proven that each head move can be replaced by a constant number of tail moves, but we have not considered the internal labels in that proof (Theorem 5.16). If it turns out each head move can be replaced by a constant number of tail moves for internally labeled networks as well, this directly gives a linear upper bound for $\mathrm{diam}_{\mathrm{tail}}(n,k,0)$. Similarly, if we can replace a tail move by a constant number of head moves respecting the internal labels, we get a linear upper bound for $\mathrm{diam}_{\mathrm{head}}(n,k,0)$.

- Can each tail move be replaced by a constant number of head moves, and vice versa, in an internally labeled network?

- Does a similar relation exist between local moves?

After determining asymptotic bounds, it would be interesting to see if we can calculate the exact diameters of these spaces, or at least find lower and upper bounds that are less far apart. The direct use of this is less apparent, but it is interesting from a pure mathematical perspective. To improve the bounds, it may be worth to try to improve the techniques that were used for the current bounds. Many of the proofs for upper bounds use inefficient sequences for the sake of simpler proofs. The lower bounds can possibly already be improved by using more recent bounds on the number of phylogenetic networks in each tier, which are not based on a quick estimation counting like for the Echidna networks, for example as in [FGM20, Man20, CZ20].

- Identify unnecessarily long sequences in proofs of upper bounds for the diameters.

- Use better estimates of the number of networks to improve lower bounds for the diameters.

B.1.3 Computational complexity

We have shown it is hard to compute the distance between two networks without internal labels for several types of moves. For the version of the problem where we do not fix the tier (i.e., M Distance), most problems are NP-hard because they are already NP-hard for trees. This argument only fails for head moves, so we have proven NP-hardness separately for Head Distance. This only leaves the complexity of Head_2 Distance open.

- Determine the complexity of Head_2 Distance.

For, M Distance Tier-k, the problem restricted to a fixed tier, we have only been able to prove hardness for tail moves and rSPR moves.

- Determine the complexity of M Distance Tier-k for $M \in \{\text{rNNI}, \text{NNI}, \text{tail}_1, \text{head}_2, \text{SPR}, \text{head}\}$.

The complexity of this problem for $M \in \{\text{rNNI}, \text{NNI}, \text{tail}_1\}$ may be easy to solve: these problems seem to be NP-hard by a reduction that adds a blob with k reticulations to each tree separated by a long chain. This may force the distance to be entirely determined by the distance between the trees, which is NP-hard to compute. For SPR, a technique similar to the one used for rSPR moves can possibly be used. Only for head_2 moves and head moves, there is no obvious method for proving hardness. The problem Head Distance Tier-1 is interesting in particular, because of it is much like finding a weighted rSPR distance between the embedded trees (Section 7.4).

In these problems, we rather arbitrarily fix one of the parameters (n and k) for our network spaces. Of course, we could also fix the number of leaves, and ask whether the problem is hard for a fixed number of leaves as well. As the blobs in such networks can have almost arbitrary structure, it seems that these problems should be hard in most cases as well.

- Determine the complexity of M Distance when the input is restricted to networks with a fixed number of leaves.

Lastly, note that we have only considered networks without internal labels for all these computational problems, which begs the question of the complexity of all the distance problems for networks with internal labels and degree-2 nodes. For the link-and-cut distance on fully labelled trees, the distance computation problem is polynomial time computable [BBDVP19], but when no

internal labels are given, the NNI or SPR distance between two trees is NP-hard to find.

- Determine the complexity of M Distance for inputs with internal labels and degree-2 nodes.

B.1.4 Improved algorithms

Although most of these problems are hard, we would still like to have efficient (approximation or parameterized) algorithms for computing the distances. As we have shown, a naive breadth first search is not efficicient enough to find larger distances, or even small distances for moderately large networks. Hence, to find these distances, we need more efficient search algorithms or reduction rules. Alternatively, it may be possible to improve the upper and lower bounds (quality as well as running time), so that they can be used in an A* search. This leads to the following open problems, which are open for all move types M if a move type is mentioned.

- Determine reduction rules that work for M Distance.
- Is M Distance FPT?
- Is there an approximation algorithm for M Distance?
- Improve upper bounds using the heuristics from this thesis.
- Improve lower bounds (e.g., for use in A* search).

To find approximation or parameterized algorithms for M Distance, new characterizations of this distance may be useful. This could be in the form of agreement graphs, such as for networks defined by [KL18, Kla18a, Kla20a]. These characterizations are approximate characterizations for SNPR moves (which also use vertical moves). Therefore, for full use in this setting, they would have to be generalized to rSPR moves (only horizontal moves), and preferably also be made exact instead of approximate.

The heuristic upper bounds in this thesis can easily be improved for use in A* search. First, it should be rather straightforward to improve the running time of the rSPR and tail heuristics. This can be done by keeping a list of the lowest nodes above the isomorphism, and of the tree and reticulation nodes above the isomorphism. In fact, a large part of the running time is a result of looping over all nodes of the network to find these nodes, so keeping track of these nodes and efficiently updating these sets should greatly improve the running time. Another way to improve the running time and the quality at

the same time, is by finding better rules for picking the nodes that determine the sequence. As mentioned, Husanovic [Hus20] and Versendaal [Ver20] have shown promising results in this direction.

- Improve the running time and quality of the heuristics.

It seems unlikely that these improvements will be helpful for the head move heuristic. Indeed, the results for the head move heuristic are quite bad as a result of the isomorphism being built top-down. A better head move heuristic may work bottom-up, like the rSPR and the tail move heuristics. It should be possible to write such a bottom-up heuristic for head moves, as each tail move in the rSPR heuristic can be replaced by a constant sequence of head moves.

- Find a bottom-up heuristic for the head move distance.

As mentioned in the discussion of Chapter 7, it may also be worth to reconsider the test sets for the heuristics. Ideally, the test sets should consist of realistic networks, but it is unclear what a realistic network would look like.

- How can one generate realistic phylogenetic networks?

B.2 Alternative network definitions

In this thesis, we have chosen a particular definition for a phylogenetic network. As discussed above, this definition is not the only meaningful definition. Hence, all questions that we have answered for our type of network can also be answered for other definitions. This leads to the following variations on the connectedness questions, some of which are simply mathematical variations, without clear biological interpretations. Note that we give partial or non-rigorous answers to some of these questions in Chapter 8 and in Chapter A.

- How does using a root with outdegree-2 affect the connectedness of spaces? (See Section A.2.3)
- How does allowing parallel arcs affect the connectedness of spaces? (See Section 8.2.2)
- Are spaces of reduced (non-binary) networks [PS15] connected?
- Are the spaces of multi-rooted networks connected? That is, networks are allowed to have multiple roots, and we can optionally require that a network is connected.

- How does allowing cycles affect the connectedness and diameters of directed network spaces?

- Are spaces of undirected network with one or no leaves connected? That is, change the definition of undirected networks to leaf labeled $\{1, 2, 3\}$-graphs without restriction on the number of leaves.

As mentioned in Section 8.2.2, the question about parallel arcs is partly answered by the lemmas about removing parallel paths. For multi-rooted networks that are allowed to be disconnected, we conjecture that tail, head, and rSPR spaces are all still connected. For local moves, these spaces are obviously disconnected, as nodes cannot be moved between different components of the network. In these cases, we can modify the question so that multi-rooted networks need to stay connected as well.

The last question may be relevant in more abstract mathematics, studying polytopes corresponding to graphs [FdPRAR20]. Or it may just be interesting per se, as similar problems have been studied [Tsu96, Tsu98]. Note that [Tsu96] formed the basis of the first proof of connectedness under NNI in [HMW16].

B.2.1 Extra structure

Besides these small changes within the definition of a network, we can also impose extra structure on the networks. An example of this would be the addition of time. This could be in the form of an order or time-label for the nodes ([MNW+04]), such as in temporal networks ([Bar04]) or as an additional structure for tree-child networks ([BLS20]). Time could also be added in the form of branch lengths.

- Study spaces of networks with time added as node labels.

- Study spaces of networks with branch lengths.

For both these questions, the moves studied in this thesis have to be extended to handle the extra structure of the network as well. As we have seen in Chapter A, there are many options for this for branch lengths.

Regarding the first question, a study of trees with ordered node labels has recently been put online [CEF+19]. Interestingly, they show that some distance questions become polynomial under certain conditions. It would be interesting to see if a similar result holds for networks.

B.2.2 Classes of networks

As mentioned earlier, these extra structures can also lead to subspaces of networks corresponding to a class of networks. Again, as Klawitter gives an extensive overview of the connectedness of these subspaces of networks [Kla20b], we do not go into detail about the related open problems here. Although it is worth mentioning that the isomorphism building techniques have not been used to find good diameter bounds for these subspaces yet. This seems to particularly promising for tree-child networks and orchard networks, as these have structures that are excellent for bottom-up treatment [LS19, JM20b, ESS19].

- Can the bottom-up heuristics be applied to the spaces of tree-child networks and orchard networks?

B.3 Rearrangement moves in reconstruction

For the use of rearrangement moves in reconstruction methods, it is important the spaces of networks are understood well. This means we do not only need to know that the spaces are connected, but also how connected they are, and how the methods interact with the spaces. One could, for example, study the following.

- What is the edge and vertex connectivity of network spaces?

A partial answer can easily be given for trees. Indeed, it is easy to prove that spaces of trees have high connectivity, for example, by noticing that for each tree T on n leaves, there is a K_{2n-3} in $\mathcal{N}_{\text{tail}}(n,0)$ for each leaf l of T. This K_{2n-3} consists of the trees that can be obtained by moving only this leaf. As for each pair of networks, there is a sequence of moves between them that moves only one leaf at a time, this implies the space $\mathcal{N}_{\text{tail}}(n,0)$ is $(n-1)$-connected. This also raises the question whether there are more interesting subgraphs of $\mathcal{N}_{\text{tail}}(n,0)$ or $\mathcal{N}_{\text{tail}_1}(n,0)$. In the tail_1 space, we do not get a copy of K_{2n-3} for each leaf in a tree, but the line-graph of that tree with the moving leaf removed (a bunch of connected triangles). A proof of biconnectedness of (unrooted) tree space under NNI moves already exists in [GFJ13], where it is proven that the space of trees under NNI moves is Hamiltonian. As $\mathcal{N}(n,0) \simeq \mathcal{U}(n+1,0)$ (Lemma 2.64), this also proves Hamiltonicity of rooted NNI tree spaces.

For this question, it is also worth revisiting the definition of a network space. In our definition, it is a simple graph, where two networks are connected if there is a move transforming one into the other. In many applications, however, a neighbour is chosen by trying a random move (somewhat uniformly), and not by chosing a neighbour uniformly at random. To reflect this, it may make more

sense to view spaces of networks as multigraphs, where each edge represents a move.

- What are the minimum and maximum neighbourhood sizes?
- What are the minimum and maximum number of valid moves for a network in a given space? In other words, what are the minimal and maximal degrees of the nodes in network spaces when viewed as multigraphs?

Neighbourhood sizes have been studied in detail for tree-child networks [Kla18b]. No extensive study has been done into neighbourhood sizes for general networks, however. For these networks, we do have some trivial observations about the maximal neighbourhood sizes, but they do not reflect the variation in possible neighbourhoods for different networks.

- How are subsequent spaces related? In other words, compare $\mathcal{N}_{\mathrm{rSPR}}(n,k)$ to $\mathcal{N}_{\mathrm{rSPR}}(n+1,k)$ and $\mathcal{N}_{\mathrm{rSPR}}(n,k+1)$.

For each network in $\mathcal{N}_{\mathrm{rSPR}}(n,k)$, we can add an extra leaf to find multiple networks in $\mathcal{N}_{\mathrm{rSPR}}(n+1,k)$, and all these new networks are quite similar. Hence, there must be some relation between $\mathcal{N}_{\mathrm{rSPR}}(n,k)$ and $\mathcal{N}_{\mathrm{rSPR}}(n+1,k)$, but it is not clear to what extent we can find $\mathcal{N}_{\mathrm{rSPR}}(n,k)$ within $\mathcal{N}_{\mathrm{rSPR}}(n+1,k)$. A similar observation holds for $\mathcal{N}_{\mathrm{rSPR}}(n,k)$ and $\mathcal{N}_{\mathrm{rSPR}}(n,k+1)$. To get an idea of such relations, it might be interesting to fully describe $\mathcal{N}_{\mathrm{rSPR}}(n,k)$ for small n and k. For trees, there has already been some interest in fully describing these spaces [WMI18].

B.3.1 Interaction with reconstruction methods

Besides knowing more about the space as a graph on itself, we can try to understand the interplay between the methods and this graph. In particular, we should try to understand the behaviour of random walks (i.e., Markov chains) on network spaces, and the interaction of optimization criteria with rearrangement moves. The former leads, among others, to the following problems.

- Understand random walks on network spaces.
- Determine the Ricci–Oliver curvature of network spaces.

The second of these is tightly related to the first, and has been studied for tree spaces [WMI17]. It is unlikely we will understand these random walks through mathematical studies alone, and practical experiments will be needed as well to determine the best combinations of moves for different underlying models.

For optimization problems, we will need to understand the interplay of moves and objective functions. This means we have to answer questions like the following for all optimization criteria, such as maximum parsimony, maximum likelihood, and deep coalescence.

- Is the subspace of optimal networks connected?
- Can local optima be arbitrarily bad compared to the global optimum?
- How much can the objective function change after one move?

These types of questions have been investigated sparingly, but there has been some interest in these problems for trees. For example, the interaction was investigated for likelihood in [MW12], and for parsimony in [UFSJ16]. Moreover, among the methods we have surveyed in Chapter A, only Markin and Eulenstein [ME19] study a version of the third question.

B.3.2 Comparing networks

Finally, the rearrangement distance can also be used as a way to compare networks or find an average or consensus network. As the rearrangement distances are hard to compute, and several other distances are easy to compute [e.g., JMRS19], it is important to know whether the rearrangement distances actually add information.

- How do the rearrangement distances compare to other distances such as the Robinson-Foulds distance?

Such a focus on pairwise distances is less useful when we need to compare a set of networks. In that case, we may need to compute the *radius* of such a set $\{N_i\}_{i=1}^{l}$ of networks (i.e., the minimal distance d so that there exists a network N with $d(N, N_i) \leq d$ for all i). Computing this is at least as hard as M Distance, by a reduction where the set consists simply of two networks. This central network can possibly be used as a *consensus network*.

- Study the radius of a set of networks in network space.
- How many networks can there be that realise the radius of a set of networks?
- Are these networks useful as consensus networks?

Interestingly, it seems our tail move and rSPR heuristics can be used to find upper bounds on the radius of a set of networks as well. For example, in rSPR

and tail move heuristics of this thesis, we can simply pick a lowest node in one of the networks and apply the subroutines to modify all other networks until so that a node can be added to the isomorphisms between all networks.

- Modify the heuristics so they compute the radius of a set of networks.

This does not work for head moves, as the networks only become non-labeled isomorphic after the top-down part of the algorithm. This is another reason to find a new bottom-up heuristic for head moves.

Curriculum Vitæ

Remie Janssen

10-03-1992	Born in Gendt, The Netherlands.
2004–2010	Gymnasium (Grammar School) Stedelijk Gymnasium Nijmegen, Nijmegen
2010–2014	B.Sc. Biology Utrecht University *Thesis:* Markovketens in de Immunologie en Epidemiologie *Supervisor 1:* Christiaan H. van Dorp, M.Sc. *Supervisor 2:* Prof. Dr. R.J. de Boer
2012–2014	B.Sc. Mathematics Utrecht University *Thesis:* Dynamical Properties of Cellular Automata *Supervisor:* Dr. Siamak Taati
2014–2017	M.Sc. Mathematical Sciences Utrecht University *Thesis:* Random Geometric Complexes *Supervisor:* Dr. Tobias Müller
2017–2021	Ph.D. Research Delft University of Technology *Thesis:* Rearranging Phylogenetic Networks *Promotor:* Prof. dr. K. Aardal *Copromotor:* Dr. ir. L.J.J. van Iersel

List of Publications

Journal Articles

1. **Remie Janssen**, Mark Jones, Péter L Erdős, Leo van Iersel, and Celine Scornavacca. Exploring the tiers of rooted phylogenetic network space using tail moves. *Bulletin of mathematical biology*, 80(8):2177–2208, 2018

2. Yukihiro Murakami, Leo van Iersel, **Remie Janssen**, Mark Jones, and Vincent Moulton. Reconstructing tree-child networks from reticulate-edge-deleted subnetworks. *Bulletin of mathematical biology*, 81(10):3823–3863, 2019

3. **Remie Janssen**, Mark Jones, Steven Kelk, Georgios Stamoulis, and Taoyang Wu. Treewidth of display graphs: Bounds, brambles and applications. *Journal of Graph Algorithms and Applications*, 23(4), 2019

4. **Remie Janssen** and Jonathan Klawitter. Rearrangement operations on unrooted phylogenetic networks. *Theory and Applications of Graphs*, 6(2), 2019

5. Leo van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. Polynomial-time algorithms for phylogenetic inference problems involving duplication and reticulation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(1):14–26, 2019

6. **Remie Janssen**. The burning number of directed graphs: Bounds and computational complexity. *Theory and Applications of Graphs*, 7(1):8, 2020

7. **Remie Janssen** and Yukihiro Murakami. On cherry-picking and network containment. *Theoretical Computer Science*, 2020

8. **Remie Janssen**. Heading in the right direction? using head moves to traverse phylogenetic network space. *Journal of Graph Algorithms and Applications*, 25(1):263–310, 2021

Conference Papers

1. Leo Van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. Polynomial-time algorithms for phylogenetic inference problems. In *International Conference on Algorithms for Computational Biology*, pages LNBI 10849: 37–49. Springer, 2018

2. **Remie Janssen**, Mark Jones, and Yukihiro Murakami. Combining networks using cherry picking sequences. In *International Conference on Algorithms for Computational Biology*, pages 77–92. Springer, 2020

3. **Remie Janssen** and Yukihiro Murakami. Linear time algorithm for tree-child network containment. In *International Conference on Algorithms for Computational Biology*, pages 93–107. Springer, 2020

Preprints

1. Katharina T Huber, Leo van Iersel, **Remie Janssen**, Mark Jones, Vincent Moulton, Yukihiro Murakami, and Charles Semple. Rooting for phylogenetic networks. *arXiv preprint arXiv:1906.07430*, 2019

2. Leo van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. A practical fixed-parameter algorithm for constructing tree-child networks from multiple binary trees. *arXiv preprint arXiv:1907.08474*, 2019

3. Mark Jones, Philippe Gambette, Leo van Iersel, **Remie Janssen**, Steven Kelk, Fabio Pardi, and Celine Scornavacca. Cutting an alignment with ockham's razor. *arXiv preprint arXiv:1910.11041*, 2019

4. Leo van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. A unifying characterization of tree-based networks and orchard networks using cherry covers. *arXiv preprint arXiv:2004.07677*, 2020

5. Elizabeth Gross, Leo van Iersel, **Remie Janssen**, Mark Jones, Colby Long, and Yukihiro Murakami. Distinguishing level-1 phylogenetic networks on the basis of data generated by markov processes. *arXiv preprint arXiv:2007.08782*, 2020

6. Katharina T Huber, Leo van Iersel, **Remie Janssen**, Mark Jones, Vincent Moulton, and Yukihiro Murakami. Reconstructibility of level-2 unrooted phylogenetic networks from shortest distances. *arXiv preprint arXiv:2101.08580*, 2021

7. **Remie Janssen** and Leonie van Steijn. Connectedness of unit distance subgraphs induced by closed convex sets. *arXiv preprint arXiv:2102.12815*, 2021

Bibliography

[AAA+13] Richard Abbott, Dirk Albach, Stephen Ansell, Jan W Arntzen, Stuart JE Baird, Nicolas Bierne, J Boughman, Alan Brelsford, C Alex Buerkle, Richard Buggs, et al. Hybridization and speciation. *Journal of Evolutionary Biology*, 26(2):229–246, 2013.

[Ald96] David Aldous. Probability distributions on cladograms. In *Random discrete structures*, pages 1–18. Springer, 1996.

[AM19] Ross Atkins and Colin McDiarmid. Extremal distances for subtree transfer operations in binary trees. *Annals of Combinatorics*, 23(1):1–26, 2019.

[Arc14] J David Archibald. *Aristotle's ladder, Darwin's tree: The evolution of visual metaphors for biological order*. Columbia University Press, 2014.

[Ata83] Mikhail J Atallah. A graph orientation problem. 1983.

[ATD18] Heval Atas, Nurcan Tuncbag, and Tunca Doğan. Phylogenetic and other conservation-based approaches to predict protein functional sites. In *Computational Drug Discovery and Design*, pages 51–69. Springer, 2018.

[BA20] Christopher Blair and Cécile Ané. Phylogenetic trees and networks can serve as powerful and complementary approaches for analysis of genomic data. *Systematic Biology*, 69(3):593–601, 2020.

[Bar04] Mihaela Carmen Baroni. Hybrid phylogenies: a graph-based approach to represent reticulate evolution. 2004.

[BBDVP19] Giulia Bernardini, Paola Bonizzoni, Gianluca Della Vedova, and Murray Patterson. A rearrangement distance for fully-labelled trees. *arXiv preprint arXiv:1904.01321*, 2019.

[BCWK18] Paul D Blischak, Julia Chifman, Andrea D Wolfe, and Laura S Kubatko. Hyde: a python package for genome-scale hybridization detection. *Systematic Biology*, 67(5):821–829, 2018.

[BDEM+20] Samuel Briand, Christophe Dessimoz, Nadia El-Mabrouk, Manuel Lafond, and Gabriela Lobinska. A generalized robinson-foulds distance for labeled trees. In *Proceedings of APBC*, 2020.

[BF06] Michael GB Blum and Olivier François. Which random processes describe the tree of life? a large-scale study of phylogenetic tree imbalance. *Systematic Biology*, 55(4):685–691, 2006.

[BFLS17] Christopher Bryant, Mareike Fischer, Simone Linz, and Charles Semple. On the quirks of maximum parsimony and likelihood on phylogenetic networks. *Journal of theoretical biology*, 417:100–108, 2017.

[BG18] Frank T Burbrink and Marcelo Gehara. The biogeography of deep time phylogenetic reticulation. *Systematic biology*, 67(5):743–755, 2018.

[BGMS05] Mihaela Baroni, Stefan Grünewald, Vincent Moulton, and Charles Semple. Bounding the number of hybridisation events for a consistent evolutionary history. *Journal of mathematical biology*, 51(2):171–182, 2005.

[BHMS18] Magnus Bordewich, Katharina T Huber, Vincent Moulton, and Charles Semple. Recovering normal networks from shortest inter-taxa distance information. *Journal of mathematical biology*, 77(3):571–594, 2018.

[BLS17] Magnus Bordewich, Simone Linz, and Charles Semple. Lost in space? Generalising subtree prune and regraft to spaces of phylogenetic networks. *Journal of theoretical biology*, 423:1–12, 2017.

[BLS20] François Bienvenu, Amaury Lambert, and Mike Steel. Combinatorial and stochastic properties of ranked tree-child networks. *arXiv preprint arXiv:2007.09701*, 2020.

[BS05] Magnus Bordewich and Charles Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of combinatorics*, 8(4):409–423, 2005.

[BS10] Erik W Bloomquist and Marc A Suchard. Unifying vertical and nonvertical evolution: a stochastic arg-based framework. *Systematic biology*, 59(1):27–41, 2010.

[BS16] Magnus Bordewich and Charles Semple. Determining phylogenetic networks from inter-taxa distances. *Journal of mathematical biology*, 73(2):283–303, 2016.

[BSP+19] Cameron Browne, Dennis JNJ Soemers, Éric Piette, Matthew Stephenson, Michael Conrad, Walter Crist, Thierry Depaulis, Eddie Duggan, Fred Horn, Steven Kelk, et al. Foundations of digital archaeoludology. *arXiv preprint arXiv:1905.13516*, 2019.

[BSTW17] Magnus Bordewich, Celine Scornavacca, Nihan Tokac, and Mathias Weller. On the fixed parameter tractability of agreement-based phylogenetic distances. *Journal of Mathematical Biology*, 74(1-2):239–257, 2017.

[BVBS+19] Remco Bouckaert, Timothy G Vaughan, Joëlle Barido-Sottani, Sebastián Duchêne, Mathieu Fourment, Alexandra Gavryushkina, Joseph Heled, Graham Jones, Denise Kühnert, Nicola De Maio, et al. Beast 2.5: An advanced software platform for bayesian evolutionary analysis. *PLoS computational biology*, 15(4):e1006650, 2019.

[Car14] Cosimo Cardellicchio. Evolution for games. *Board Game Studies Journal*, pages 1–2, 2014.

[CEF+19] Lena Collienne, Kieran Elmes, Mareike Fischer, David Bryant, and Alex Gavryushkin. Geometry of ranked nearest neighbour interchange space of phylogenetic trees. *bioRxiv*, 2019.

[CHW17] Zhi-Zhong Chen, Youta Harada, and Lusheng Wang. A new 2-approximation algorithm for rspr distance. In *International Symposium on Bioinformatics Research and Applications*, pages 128–139. Springer, 2017.

[CPR15] Gabriel Cardona, Joan Carles Pons, and Francesc Rosselló. A reconstruction problem for a class of phylogenetic networks with lateral gene transfers. *Algorithms for Molecular Biology*, 10(1):1–15, 2015.

[CT06] Benny Chor and Tamir Tuller. Finding a maximum likelihood tree is hard. *Journal of the ACM (JACM)*, 53(5):722–744, 2006.

[CZ20] Gabriel Cardona and Louxin Zhang. Counting and enumerating tree-child networks and their subclasses. *Journal of Computer and System Sciences*, 2020.

[DGH11] Yang Ding, Stefan Grünewald, and Peter J Humphries. On agreement forests. *Journal of Combinatorial Theory, Series A*, 118(7):2059–2065, 2011.

[Die12] Reinhard Diestel. *Graph Theory: Springer Graduate Text GTM 173*, volume 173. Reinhard Diestel, 2012.

[DLDF10] Xavier Didelot, Daniel Lawson, Aaron Darling, and Daniel Falush. Inference of homologous recombination in bacteria using whole-genome sequences. *Genetics*, 186(4):1435–1449, 2010.

[DM06] Tal Dagan and William Martin. The tree of one percent. *Genome biology*, 7(10):118, 2006.

[Doo99] W Ford Doolittle. Phylogenetic classification and the universal tree. *Science*, 284(5423):2124–2128, 1999.

[DPRS11] Eric Y Durand, Nick Patterson, David Reich, and Montgomery Slatkin. Testing for ancient admixture between closely related populations. *Molecular biology and evolution*, 28(8):2239–2252, 2011.

[Dun15] Michael Dunn. Language phylogenies. *The Routledge handbook of historical linguistics*, pages 190–211, 2015.

[DW04] Charles C Davis and Kenneth J Wurdack. Host-to-parasite gene transfer in flowering plants: phylogenetic evidence from malpighiales. *Science*, 305(5684):676–678, 2004.

[EFM20] Péter L Erdős, Andrew Francis, and Tamás Róbert Mezei. Rooted nni moves on tree-based phylogenetic networks. *arXiv preprint arXiv:2003.07283*, 2020.

[EOZN19] RA Leo Elworth, Huw A Ogilvie, Jiafan Zhu, and Luay Nakhleh. Advances in computational methods for phylogenetic networks in the presence of hybridization. In *Bioinformatics and Phylogenetics*, pages 317–360. Springer, 2019.

[Ere10] Marc Ereshefsky. Microbiology and the species problem. *Biology & Philosophy*, 25(4):553–568, 2010.

[ESS19] Péter L Erdős, Charles Semple, and Mike Steel. A class of phylogenetic networks reconstructable from ancestral profiles. *Mathematical biosciences*, 313:33–40, 2019.

[ET76] Shimon Even and Robert Endre Tarjan. Computing an st-numbering. *Theoretical Computer Science*, 2(3):339–344, 1976.

[FBL20] David Fernández-Baca and Lei Liu. Testing the agreement of trees with internal labels. *arXiv preprint arXiv:2002.09725*, 2020.

[FdPRAR20] Cristina G Fernandes, José C de Pina, Jorge Luis Ramírez Alfonsín, and Sinai Robins. Cubic graphs, their ehrhart quasi-polynomials, and a scissors congruence phenomenon. *Discrete & Computational Geometry*, pages 1–17, 2020.

[Fel04] Joseph Felsenstein. *Inferring Phylogenies*, volume 2. Sinauer associates Sunderland, MA, 2004.

[FF20] Mareike Fischer and Andrew Francis. The space of tree-based phylogenetic networks. *Bulletin of Mathematical Biology*, 82(6):70–70, 2020.

[FFRF20] Peter Forster, Lucy Forster, Colin Renfrew, and Michael Forster. Phylogenetic network analysis of sars-cov-2 genomes. *Proceedings of the National Academy of Sciences*, 117(17):9241–9243, 2020.

[FG82] Les R Foulds and Ronald L Graham. The steiner problem in phylogeny is np-complete. *Advances in Applied mathematics*, 3(1):43–49, 1982.

[FGM20] Michael Fuchs, Bernhard Gittenberger, and Marefatollah Mansouri. Counting phylogenetic networks with few reticulation vertices: Exact enumeration and corrections. *arXiv preprint arXiv:2006.15784*, 2020.

[FHMW17] Andrew Francis, Katharina T. Huber, Vincent Moulton, and Taoyang Wu. Bounds for phylogenetic network space metrics. *Journal of Mathematical Biology*, Aug 2017.

[Fit71] Walter M Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[FS15] Andrew R Francis and Mike Steel. Which phylogenetic networks are merely trees with additional arcs? *Systematic biology*, 64(5):768–777, 2015.

[FvIKS15] Mareike Fischer, Leo van Iersel, Steven Kelk, and Celine Scornavacca. On computing the maximum parsimony score of a phylogenetic network. *SIAM Journal on Discrete Mathematics*, 29(1):559–585, 2015.

[GANA+17] Christophe Guyeux, Bashar Al-Nuaimi, Bassam AlKindy, Jean-François Couchot, and Michel Salomon. On the ability to reconstruct ancestral genomes from mycobacterium genus. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 642–658. Springer, 2017.

[GBR+99] Feng Gao, Elizabeth Bailes, David L Robertson, Yalu Chen, Cynthia M Rodenburg, Scott F Michael, Larry B Cummins, Larry O Arthur, Martine Peeters, George M Shaw, et al. Origin of HIV-1 in the chimpanzee pan troglodytes troglodytes. *Nature*, 397(6718):436, 1999.

[GFJ13] Kevaughn Gordon, Eric Ford, and Katherine St John. Hamiltonian walks of phylogenetic treespaces. *IEEE/ACM transactions on computational biology and bioinformatics*, 10(4):1076–1079, 2013.

[GL18] Elizabeth Gross and Colby Long. Distinguishing phylogenetic networks. *SIAM Journal on Applied Algebra and Geometry*, 2(1):72–93, 2018.

[Gra89] Alan Grafen. The phylogenetic regression. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 326(1233):119–157, 1989.

[GvIJ+17a] Philippe Gambette, Leo van Iersel, Mark Jones, Manuel Lafond, Fabio Pardi, and Celine Scornavacca. Rearrangement moves on rooted phylogenetic networks. *Accepted for publication in PLOS Computational Biology*, 2017.

[GvIJ+17b] Philippe Gambette, Leo van Iersel, Mark Jones, Manuel Lafond, Fabio Pardi, and Celine Scornavacca. Rearrangement moves on rooted phylogenetic networks. *PLoS computational biology*, 13(8):e1005611, 2017.

[GvIJ+20] Elizabeth Gross, Leo van Iersel, **Remie Janssen**, Mark Jones, Colby Long, and Yukihiro Murakami. Distinguishing level-1 phylogenetic networks on the basis of data generated by markov processes. *arXiv preprint arXiv:2007.08782*, 2020.

[GWSD14] Alexandra Gavryushkina, David Welch, Tanja Stadler, and Alexei J Drummond. Bayesian inference of sampled ancestor trees for epidemiology and fossil calibration. *PLoS Comput Biol*, 10(12):e1003919, 2014.

[HDRCB08] Glenn Hickey, Frank Dehne, Andrew Rau-Chaplin, and Christian Blouin. Spr distance computation for unrooted trees. *Evolutionary Bioinformatics*, 4:EBO–S419, 2008.

[Hey01] Jody Hey. The mind of the species problem. *Trends in Ecology & Evolution*, 16(7):326–329, 2001.

[HF17] Lina Herbst and Mareike Fischer. Ancestral sequence reconstruction with maximum parsimony. *Bulletin of Mathematical Biology*, 79(12):2865–2886, 2017.

[HJH+13] Leanne S Haggerty, Pierre-Alain Jachiet, William P Hanage, David A Fitzpatrick, Philippe Lopez, Mary J O'Connell, Davide Pisani, Mark Wilkinson, Eric Bapteste, and James O McInerney. A pluralistic account of homology: adapting the models to the data. *Molecular biology and evolution*, 31(3):501–516, 2013.

[HLMW16] Katharina T Huber, Simone Linz, Vincent Moulton, and Taoyang Wu. Spaces of phylogenetic networks from generalized nearest-neighbor interchange operations. *Journal of Mathematical Biology*, 72(3):699–725, 2016.

[HMP10] Pierre Hansen, Nenad Mladenović, and José A Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[HMW16] Katharina T Huber, Vincent Moulton, and Taoyang Wu. Transforming phylogenetic networks: Moving beyond tree space. *Journal of Theoretical Biology*, 404:30–39, 2016.

[Hou12] Wybo Houkes. Tales of tools and trees: Phylogenetic analysis and explanation in evolutionary archaeology. In *EPSA Philosophy of Science: Amsterdam 2009*, pages 89–100. Springer, 2012.

[HRS10] Daniel H Huson, Regula Rupp, and Celine Scornavacca. *Phylogenetic Networks: Concepts, Algorithms and Applications*. Cambridge University Press, 2010.

[Hus20] Selma Husanovic. Tail move rearrangement algorithm for rooted binary phylogenetic networks. Bachelor's thesis, Delft University of Technology, 2020.

[HvIJ+19] Katharina T Huber, Leo van Iersel, **Remie Janssen**, Mark Jones, Vincent Moulton, Yukihiro Murakami, and Charles Semple. Rooting for phylogenetic networks. *arXiv preprint arXiv:1906.07430*, 2019.

[HvIJ+21] Katharina T Huber, Leo van Iersel, **Remie Janssen**, Mark Jones, Vincent Moulton, and Yukihiro Murakami. Reconstructibility of level-2 unrooted phylogenetic networks from shortest distances. *arXiv preprint arXiv:2101.08580*, 2021.

[HZKH08] Tracy A Heath, Derrick J Zwickl, Junhyong Kim, and David M Hillis. Taxon sampling affects inferences of macroevolutionary processes from phylogenetic trees. *Systematic biology*, 57(1):160–166, 2008.

[vanIJJ+18] Leo van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. Polynomial-time algorithms for phylogenetic inference problems. In *International Conference on Algorithms for Computational Biology*, pages LNBI 10849: 37–49. Springer, 2018.

[vanIJS17] Leo van Iersel, Mark Jones, and Celine Scornavacca. Improved maximum parsimony models for phylogenetic networks. *Systematic biology*, 67(3):518–542, 2017.

[Jan20] **Remie Janssen**. The burning number of directed graphs: Bounds and computational complexity. *Theory and Applications of Graphs*, 7(1):8, 2020.

[Jan21] **Remie Janssen**. Heading in the right direction? using head moves to traverse phylogenetic network space. *Journal of Graph Algorithms and Applications*, 25(1):263–310, 2021.

[JBZ20] Katharina Jahn, Niko Beerenwinkel, and Louxin Zhang. The bourque distances for mutation trees of cancers. *bioRxiv*, 2020.

[JGvI+19] Mark Jones, Philippe Gambette, Leo van Iersel, **Remie Janssen**, Steven Kelk, Fabio Pardi, and Celine Scornavacca. Cutting an alignment with ockham's razor. *arXiv preprint arXiv:1910.11041*, 2019.

[JJE+18] **Remie Janssen**, Mark Jones, Péter L Erdős, Leo van Iersel, and Celine Scornavacca. Exploring the tiers of rooted phylogenetic network space using tail moves. *Bulletin of mathematical biology*, 80(8):2177–2208, 2018.

[JJK+19] **Remie Janssen**, Mark Jones, Steven Kelk, Georgios Stamoulis, and Taoyang Wu. Treewidth of display graphs: Bounds, brambles and applications. *Journal of Graph Algorithms and Applications*, 23(4), 2019.

[JJM20] **Remie Janssen**, Mark Jones, and Yukihiro Murakami. Combining networks using cherry picking sequences. In *International Conference on Algorithms for Computational Biology*, pages 77–92. Springer, 2020.

[JK19] **Remie Janssen** and Jonathan Klawitter. Rearrangement operations on unrooted phylogenetic networks. *Theory and Applications of Graphs*, 6(2), 2019.

[JL19] Guillaume Jacques and Johann-Mattis List. Save the trees. *Journal of Historical Linguistics*, 9(1):128–166, 2019.

[JLM+16] Jeffrey B Joy, Richard H Liang, Rosemary M McCloskey, T Nguyen, and Art FY Poon. Ancestral reconstruction. *PLoS computational biology*, 12(7):e1004763, 2016.

[JLTZ00] Bhaskar DasGupta Xin He Tao Jiang, Ming Li, John Tromp, and Louxin Zhang. On computing the nearest neighbor interchange distance. In *Discrete Mathematical Problems with Medical Applications: DIMACS Workshop Discrete Mathematical Problems with Medical Applications, December 8-10, 1999, DIMACS Center*, volume 55, page 125. American Mathematical Soc., 2000.

[JM20a] **Remie Janssen** and Yukihiro Murakami. Linear time algorithm for tree-child network containment. In *International Conference on Algorithms for Computational Biology*, pages 93–107. Springer, 2020.

[JM20b] **Remie Janssen** and Yukihiro Murakami. On cherry-picking and network containment. *Theoretical Computer Science*, 2020.

[JMRS19] Jesper Jansson, Konstantinos Mampentzidis, Ramesh Rajaby, and Wing-Kin Sung. Computing the rooted triplet distance between phylogenetic networks. In *International Workshop on Combinatorial Algorithms*, pages 290–303. Springer, 2019.

[JSO13] Graham Jones, Serik Sagitov, and Bengt Oxelman. Statistical inference of allopolyploid species networks in the presence of incomplete lineage sorting. *Systematic Biology*, 62(3):467–478, 2013.

[JST15] Vinicio D Armijos Jaramillo, Serenella A Sukno, and Michael R Thon. Identification of horizontally transferred genes in the genus colletotrichum reveals a steady tempo of bacterial to fungal gene transfer. *BMC genomics*, 16(1):2, 2015.

[JVD+14] Wei Jiao, Shankar Vembu, Amit G Deshwar, Lincoln Stein, and Quaid Morris. Inferring clonal evolution of tumors from single nucleotide somatic mutations. *BMC bioinformatics*, 15(1):35, 2014.

[JvS21] **Remie Janssen** and Leonie van Steijn. Connectedness of unit distance subgraphs induced by closed convex sets. *arXiv preprint arXiv:2102.12815*, 2021.

[KGDO05] Victor Kunin, Leon Goldovsky, Nikos Darzentas, and Christos A Ouzounis. The net of life: reconstructing the microbial phylogenetic network. *Genome Research*, 15(7):954–959, 2005.

[KL18] J. Klawitter and S. Linz. On the Subnet Prune and Regraft Distance. *ArXiv e-prints*, (1805.07839), May 2018.

[Kla18a] J. Klawitter. The agreement distance of rooted phylogenetic networks. *ArXiv e-prints*, (1806.05800), June 2018.

[Kla18b] Jonathan Klawitter. The SNPR neighbourhood of tree-child networks. *Journal of Graph Algorithms and Applications*, 22(2):329–355, 2018.

[Kla20a] Jonathan Klawitter. The agreement distance of unrooted phylogenetic networks. *Discrete Mathematics & Theoretical Computer Science*, vol. 22 no. 1, July 2020.

[Kla20b] Jonathan Klawitter. *Spaces of phylogenetic networks*. PhD thesis, PhD thesis, University of Auckland, 2020.

[KMC+07] Carole Knibbe, Olivier Mazet, Fabien Chaudier, Jean-Michel Fayard, and Guillaume Beslon. Evolutionary coupling between the deleteriousness of gene mutations and the amount of non-coding sequences. *Journal of Theoretical Biology*, 244(4):621–630, 2007.

[KNTX08] Iyad A Kanj, Luay Nakhleh, Cuong Than, and Ge Xia. Seeing the trees and their branches in the network is hard. *Theoretical Computer Science*, 401(1-3):153–164, 2008.

[KP08] Patrick J Keeling and Jeffrey D Palmer. Horizontal gene transfer in eukaryotic evolution. *Nature Reviews Genetics*, 9(8):605–618, 2008.

[Kra00] Alex R Kraaijeveld. Origin of chess–a phylogenetic perspective. *Board Games Studies*, 3:39–50, 2000.

[KST93] Johannes Kobler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity*. Springer Science & Business Media, 1993.

[Kwo11] Rex Bing Hung Kwok. Phylogeny, genealogy and the linnaean hierarchy: a logical analysis. *Journal of mathematical biology*, 63(1):73–108, 2011.

[KYT20] Paschalia Kapli, Ziheng Yang, and Maximilian J Telford. Phylogenetic tree building in the genomic age. *Nature Reviews Genetics*, pages 1–17, 2020.

[KZNL02] Henrik Kaessmann, Sebastian Zöllner, Anton Nekrutenko, and Wen-Hsiung Li. Signatures of domain shuffling in the human genome. *Genome Research*, 12(11):1642–1650, 2002.

[Lar05] Bret Larget. Introduction to markov chain monte carlo methods in molecular evolution. In *Statistical methods in molecular evolution*, pages 45–62. Springer, 2005.

[Lar20] Nicolas Lartillot. The bayesian approach to molecular phylogeny, 2020.

[LCK+16] Justin Lessler, Lelia H Chaisson, Lauren M Kucirka, Qifang Bi, Kyra Grantz, Henrik Salje, Andrea C Carcelen, Cassandra T Ott, Jeanne S Sheffield, Neil M Ferguson, et al. Assessing the global threat from zika virus. *Science*, 353(6300):aaf8160, 2016.

[LEC67] Abraham Lempel, Shimon Even, and Israel Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs: International Symposium*, volume 67, pages 215–232. Gordon and Breach, New York, 1967.

[LP17] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[LRM+15] Fay-Wei Li, Carl J Rothfels, Michael Melkonian, Juan C Villarreal, Dennis W Stevenson, Sean W Graham, Gane K-S Wong, Sarah Mathews, and Kathleen M Pryer. The origin and evolution of phototropins. *Frontiers in Plant Science*, 6, 2015.

[LS19] Simone Linz and Charles Semple. Attaching leaves and picking cherries to characterise the hybridisation number for a set of phylogenies. *Advances in Applied Mathematics*, 105:102–129, 2019.

[LS20] Johann-Mattis List and Nathanael Erik Schweikhard. Modeling word trees in historical linguistics: Preliminary ideas for the reconciliation of word trees and language trees. 2020. Unrevised preprint, submitted for the Sektionsband Historische Linguistik.

[LSV09] Philippe Lemey, Marco Salemi, and Anne-Mieke Vandamme. *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press, 2009.

[LTZ96] Ming Li, John Tromp, and Louxin Zhang. On the nearest neighbour interchange distance between evolutionary trees. *Journal of Theoretical Biology*, 182(4):463–467, 1996.

[Luk82] Eugene M Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of computer and system sciences*, 25(1):42–65, 1982.

[LVDMH+08] Clemens Lakner, Paul Van Der Mark, John P Huelsenbeck, Bret Larget, and Fredrik Ronquist. Efficiency of markov chain monte carlo tree proposals in bayesian phylogenetics. *Systematic biology*, 57(1):86–103, 2008.

[LVK+14] Fay-Wei Li, Juan Carlos Villarreal, Steven Kelly, Carl J Rothfels, Michael Melkonian, Eftychios Frangedakis, Markus Ruhsam, Erin M Sigel, Joshua P Der, Jarmila Pittermann, et al. Horizontal transfer of an adaptive chimeric photoreceptor from bryophytes to ferns. *Proceedings of the National Academy of Sciences*, 111(18):6672–6677, 2014.

[Mag13] Anne E Magurran. *Measuring biological diversity*. John Wiley & Sons, 2013.

[Man20] Marefatollah Mansouri. Counting general phylogenetic networks. *arXiv preprint arXiv:2005.14547*, 2020.

[Mar20] Alexey Markin. Phylogenetic comparison measurements and their application towards the accurate inference of evolutionary histories. 2020.

[MAVE19] Alexey Markin, Tavis K Anderson, Venkata SKT Vadali, and Oliver Eulenstein. Robinson-foulds reticulation networks. *bioRxiv*, page 642793, 2019.

[May97] Richard L Mayden. A hierarchy of species concepts: the denouement in the saga of the species problem. 1997.

[ME19] Alexey Markin and Oliver Eulenstein. Consensus clusters in robinson-foulds reticulation networks. In *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[MNW+04] Bernard ME Moret, Luay Nakhleh, Tandy Warnow, C Randal Linder, Anna Tholse, Anneke Padolina, Jerry Sun, and Ruth Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):13–23, 2004.

[Mor11] DA Morrison. *Introduction to Phylogenetic Networks*. RJR Productions, 2011.

[Mor13] David Morrison. False analogies between anthropology and biology. `https://phylonetworks.blogspot.com/2013/01/false-analogies-between-anthropology.html`, 2013. Accessed: October 5, 2020.

[MR12] Adria Alcala Mena and Francesc Rosselló. Ternary graph isomorphism in polynomial time, after luks. *arXiv preprint arXiv:1209.0871*, 2012.

[MSD+20] Nicola F Müller, Ugnė Stolz, Gytis Dudas, Tanja Stadler, and Timothy G Vaughan. Bayesian inference of reassortment networks reveals fitness benefits of reassortment in human influenza viruses. *Proceedings of the National Academy of Sciences*, 117(29):17104–17111, 2020.

[MvIJ+19] Yukihiro Murakami, Leo van Iersel, **Remie Janssen**, Mark Jones, and Vincent Moulton. Reconstructing tree-child networks from reticulate-edge-deleted subnetworks. *Bulletin of mathematical biology*, 81(10):3823–3863, 2019.

[MW12] Daniel Money and Simon Whelan. Characterizing the phylogenetic tree-search problem. *Systematic biology*, 61(2):228, 2012.

[MYK10] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 383–389, 2010.

[NSvHM14] Lam-Tung Nguyen, Heiko A Schmidt, Arndt von Haeseler, and Bui Quang Minh. Iq-tree: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular biology and evolution*, 32(1):268–274, 2014.

[NW05] Luay Nakhleh and Li-San Wang. Phylogenetic networks, trees, and clusters. In *International Conference on Computational Science*, pages 919–926. Springer, 2005.

[OBB+14] Michael J OâĂŹBrien, Matthew T Boulanger, Briggs Buchanan, Mark Collard, R Lee Lyman, and John Darwent. Innovation and cultural transmission in the american paleolithic: phylogenetic analysis of eastern paleoindian projectile-point classes. *Journal of Anthropological Archaeology*, 34:100–119, 2014.

[Pre19] Anna Marie Prentiss. *Handbook of evolutionary research in archaeology*. Springer, 2019.

[PS15] Fabio Pardi and Celine Scornavacca. Reconstructible phylogenetic networks: do not distinguish the indistinguishable. *PLoS Comput Biol*, 11(4):e1004135, 2015.

[PSC19] Joan Carles Pons, Celine Scornavacca, and Gabriel Cardona. Generation of level-k lgt networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(1):158–164, 2019.

[Rag09] Mark A Ragan. Trees and networks before and after darwin. *Biology direct*, 4(1):43, 2009.

[RC06] Antonis Rokas and Sean B Carroll. Bushes in the tree of life. *PLoS Biol*, 4(11):e352, 2006.

[Roc06] Sebastien Roch. A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(1):92–94, 2006.

[SDG20] Celine Scornavacca, Frédéric Delsuc, and Nicolas Galtier, editors. *Phylogenetics in the Genomic Era*. No commercial publisher | Authors open access book, 2020.

[SFvN19] Thomas Sakoparnig, Chris Field, and Erik van Nimwegen. Whole genome phylogenies reflect long-tailed distributions of recombination rates in many bacterial species. *BioRxiv*, page 601914, 2019.

[SGF15] David Strait, Frederick E Grine, and John G Fleagle. Analyzing hominin phylogeny: cladistic approach. *Handbook of paleoanthropology*, pages 1989–2014, 2015.

[Sin92] Alistair Sinclair. Improved bounds for mixing rates of markov chains and multicommodity flow. *Combinatorics, probability and Computing*, 1(4):351–370, 1992.

[SJ17] Katherine St. John. The shape of phylogenetic treespace. *Systematic biology*, 66(1):e83–e94, 2017.

[SKS94] IN Shindyalov, NA Kolchanov, and CHRIS Sander. Can three-dimensional contacts in protein structures be predicted by analysis of correlated mutations? *Protein Engineering, Design and Selection*, 7(3):349–358, 1994.

[SLA16] Claudia Solís-Lemus and Cécile Ané. Inferring phylogenetic networks with maximum pseudolikelihood under incomplete lineage sorting. *PLoS genetics*, 12(3):e1005896, 2016.

[SLBA17] Claudia Solís-Lemus, Paul Bastide, and Cécile Ané. Phylonetworks: a package for phylogenetic networks. *Molecular biology and evolution*, 34(12):3292–3298, 2017.

[SLCA20] Claudia Solis-Lemus, Arrigo Coen, and Cecile Ane. On the identifiability of phylogenetic networks under a pseudolikelihood model. *arXiv preprint arXiv:2010.01758*, 2020.

[SPKPS+20] Santiago J Sánchez-Pacheco, Sungsik Kong, Paola Pulido-Santacruz, Robert W Murphy, and Laura Kubatko. Median-joining network analysis of sars-cov-2 genomes is neither phylogenetic nor evolutionary. *Proceedings of the National Academy of Sciences*, 117(23):12518–12519, 2020.

[SS03] Charles Semple and Mike A Steel. *Phylogenetics*, volume 24. Oxford University Press on Demand, 2003.

[SS19] Lena Schlipf and Jens M Schmidt. Simple computation of st-edge- and st-numberings from ear decompositions. *Information Processing Letters*, 145:58–63, 2019.

[Ste16] Mike Steel. *Phylogeny: discrete and random processes in evolution.* SIAM, 2016.

[Str19] Larissa Mendoza Straffon. The uses of cultural phylogenetics in archaeology. In *Handbook of Evolutionary Research in Archaeology*, pages 149–160. Springer, 2019.

[TRN08] Cuong Than, Derek Ruths, and Luay Nakhleh. Phylonet: a software package for analyzing and reconstructing reticulate evolutionary relationships. *BMC Bioinformatics*, 9(1):322, 2008.

[Tsu96] Yasuyuki Tsukui. Transformations of cubic graphs. *Journal of the Franklin Institute*, 333(4):565–575, 1996.

[Tsu98] Yasuyuki Tsukui. Transformations of edge-coloured cubic graphs. *Discrete mathematics*, 184(1-3):183–194, 1998.

[UFSJ16] Ellen Urheim, Eric Ford, and Katherine St. John. Characterizing local optima for maximum parsimony. *Bulletin of mathematical biology*, 78(5):1058–1075, 2016.

[VB15] Séverine Vuilleumier and Sebastian Bonhoeffer. Contribution of recombination to the evolutionary history of hiv. *Current Opinion in HIV and AIDS*, 10(2):84–89, 2015.

[vDMCH19] Bram van Dijk, Jeroen Meijer, Thomas D Cuypers, and Paulien Hogeweg. Trusting the hand that feeds: microbes evolve to anticipate a serial transfer protocol as individuals or collectives. *BMC evolutionary biology*, 19(1):201, 2019.

[Ver20] Bryan Versendaal. Improving algorithms in phylogenetics using machine learning. Master's thesis, Delft University of Technology, 2020. `http://resolver.tudelft.nl/uuid:83690896-2783-4340-8b02-11a23d82e9f9`.

[vIJJ+19a] Leo van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. Polynomial-time algorithms for phylogenetic inference problems involving duplication and reticulation.

IEEE/ACM Transactions on Computational Biology and Bioinformatics, 17(1):14–26, 2019.

[vIJJ+19b] Leo van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. A practical fixed-parameter algorithm for constructing tree-child networks from multiple binary trees. *arXiv preprint arXiv:1907.08474*, 2019.

[vIJJ+20] Leo van Iersel, **Remie Janssen**, Mark Jones, Yukihiro Murakami, and Norbert Zeh. A unifying characterization of tree-based networks and orchard networks using cherry covers. *arXiv preprint arXiv:2004.07677*, 2020.

[vIMM20] Leo van Iersel, Vincent Moulton, and Yukihiro Murakami. Reconstructibility of unrooted level-k phylogenetic networks from distances. *Advances in Applied Mathematics*, 120:102075, 2020.

[VMM+14] Logan Volkmann, Iain Martyn, Vincent Moulton, Andreas Spillner, and Arne O Mooers. Prioritizing populations for conservation using phylogenetic networks. *PLoS One*, 9(2), 2014.

[VTPL05] Christine Vogel, Sarah A Teichmann, and Jose Pereira-Leal. The relationship between domain duplication and recombination. *Journal of molecular biology*, 346(1):355–365, 2005.

[VWD+17] Timothy G Vaughan, David Welch, Alexei J Drummond, Patrick J Biggs, Tessy George, and Nigel P French. Inferring ancestral recombination graphs from bacterial genomic data. *Genetics*, 205(2):857–870, 2017.

[WBZ13] Chris Whidden, Robert G Beiko, and Norbert Zeh. Fixed-parameter algorithms for maximum agreement forests. *SIAM Journal on Computing*, 42(4):1431–1466, 2013.

[WF17] Kristina Wicke and Mareike Fischer. Comparing the rankings obtained from two biodiversity indices: the fair proportion index and the shapley value. *Journal of theoretical biology*, 430:207–214, 2017.

[WF18] Kristina Wicke and Mareike Fischer. Phylogenetic diversity and biodiversity indices on phylogenetic networks. *Mathematical biosciences*, 298:80–90, 2018.

[WM10] Simon Whelan and Daniel Money. The prevalence of multifurcations in tree-space and their implications for tree-search. *Molecular biology and evolution*, 27(12):2674–2677, 2010.

[WM18] Chris Whidden and Frederick A Matsen. Calculating the unrooted subtree prune-and-regraft distance. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(3):898–911, 2018.

[WMI17] Chris Whidden and Frederick A Matsen IV. Ricci–ollivier curvature of the rooted phylogenetic subtree–prune–regraft graph. *Theoretical Computer Science*, 699:1–20, 2017.

[WMI18] Chris Whidden and Frederick A Matsen IV. Efficiently inferring pairwise subtree prune-and-regraft adjacencies between phylogenetic trees. In *2018 Proceedings of the Fifteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 77–91. SIAM, 2018.

[WN18] Dingqiao Wen and Luay Nakhleh. Coestimating reticulate phylogenies and gene trees from multilocus sequence data. *Systematic Biology*, 67(3):439–457, 2018.

[WPR19] Matthew J Walsh, Anna Marie Prentiss, and Felix Riede. Introduction to cultural microevolutionary research in anthropology and archaeology. In *Handbook of Evolutionary Research in Archaeology*, pages 25–47. Springer, 2019.

[Wu20] Yufeng Wu. Inference of population admixture network from local gene genealogies: a coalescent-based maximum likelihood approach. *Bioinformatics*, 36:i326–i334, 07 2020.

[WYHN16] Dingqiao Wen, Yun Yu, Matthew W Hahn, and Luay Nakhleh. Reticulate evolutionary history and extensive introgression in mosquito species revealed by phylogenetic network analysis. *Molecular ecology*, 25(11):2361–2372, 2016.

[WYN16] Dingqiao Wen, Yun Yu, and Luay Nakhleh. Bayesian inference of reticulate phylogenies under the multispecies network coalescent. *PLoS genetics*, 12(5):e1006006, 2016.

[WZ09] Chris Whidden and Norbert Zeh. A unifying view on approximation and fpt of agreement forests. In *International Workshop on Algorithms in Bioinformatics*, pages 390–402. Springer, 2009.

[WZB14] Christopher Whidden, Norbert Zeh, and Robert G Beiko. Supertrees based on the subtree prune-and-regraft distance. *Systematic biology*, 63(4):566–581, 2014.

[YBN13] Yun Yu, R Matthew Barnett, and Luay Nakhleh. Parsimonious inference of hybridization in the presence of incomplete lineage sorting. *Systematic biology*, 62(5):738–751, 2013.

[YCLN20] Zhi Yan, Zhen Cao, Yushu Liu, and Luay Nakhleh. Maximum parsimony inference of phylogenetic networks in the presence of polyploid complexes. *bioRxiv*, 2020.

[YDLN14] Yun Yu, Jianrong Dong, Kevin J Liu, and Luay Nakhleh. Maximum likelihood inference of reticulate evolutionary histories. *Proceedings of the National Academy of Sciences*, 111(46):16448–16453, 2014.

[YHBH15] Yan Yu, Alan J Harris, Christopher Blair, and Xingjin He. Rasp (reconstruct ancestral state in phylogenies): a tool for historical biogeography. *Molecular phylogenetics and evolution*, 87:46–49, 2015.

[YN15] Yun Yu and Luay Nakhleh. A maximum pseudo-likelihood approach for phylogenetic networks. *BMC genomics*, 16(S10):S10, 2015.

[YvBW13] Rolf JF Ypma, W Marijn van Ballegooijen, and Jacco Wallinga. Relating phylogenetic trees to transmission trees of infectious disease outbreaks. *Genetics*, 195(3):1055–1062, 2013.

[Zal40] Adam ZalužanskážŔ. *Methodi herbariae libri tres.* e Collegio Paltheniano, 1940.

[ZD11] Olga Zhaxybayeva and W Ford Doolittle. Lateral gene transfer. *Current Biology*, 21(7):R242–R246, 2011.

[Zha16] Louxin Zhang. On tree-based phylogenetic networks. *Journal of Computational Biology*, 23(7):553–565, 2016.

[ZODS18] Chi Zhang, Huw A Ogilvie, Alexei J Drummond, and Tanja Stadler. Bayesian inference of species networks from multilocus sequence data. *Molecular biology and evolution*, 35(2):504–517, 2018.

Symbol Index

Basics

iff	if and only if
w.l.o.g.	without loss of generality
$\sim$	An equivalence relation
ln	logarithm base e
log	logarithm base 2
$\log_6$	logarithm base 6
$\lfloor x \rfloor$	the floor of a real number x
$\mathbb{R}$	the real numbers
$\mathbb{Z}$	the integers
$\mathbb{Z}_{\geq n}$	the integers greater or equal to n
$\mathbb{N}$	the natural numbers $\{0, 1, 2, \dots\}$
$[n]$	the set of n numbers $\{1, 2, \dots, n\}$
$\lvert X \rvert$	the cardinality of a set X
$A \subseteq B$	A is a (not necessarily proper) subset of B
$A \subsetneq B$	A is a proper subset of B
$P(X)$	the powerset $\{Y : Y \subseteq X\}$ of a set X
$\binom{X}{n}$	the set of n-element subsets of X
$\mathrm{dom}(f)$	the domain of the function f
$\mathrm{im}(f)$	the image of the function f
$f^{-1}(y)$	the pre-image $\{x \in \mathrm{dom}(f) : f(x) = y\}$ of y under f
$f^{-1}(y)$	the pre-image x of y for the bijective function f

Graphs

$G = (V, E)$	an undirected graph with nodes V and edges E
$D = (V, A)$	a directed graph with nodes V and arcs A
$V(G)$	the set of nodes of a (di)graph G
$E(G)$	the set of edges of a graph G
$A(D)$	the set of arcs of a digraph D
$d_G(u, v)$	the distance between u and v
$\mathrm{diam}(G)$	the diameter of G

$G[X]$	the subgraph of G induced by $X \subseteq V(G)$
$G/\sim$	the quotient of G for the equivalence relation $\sim$
$G \setminus X$	the subgraph $G[V(G) \setminus X]$ of G
$\deg^-(v)$	the indegree of a node v of a digraph D
$\deg^+(v)$	the outdegree of a node v of a digraph D
$G \simeq G'$	G is isomorphic to G'
$G \overset{\phi}{\simeq} G'$	ϕ is an isomorphism between G and G'
(G, l)	the labeled graph with labels $l(w)$ for $w \in W \subseteq V(G)$
$(G, l) \simeq_X (G', l')$	G is labeled isomorphic to G' for the label set $X \subseteq l(V(G) \cap V(G'))$
$(G, l) \overset{\phi}{\simeq}_X (G', l')$	ϕ is a labeled isomorphism between G and G' for the label set $X \subseteq l(V(G) \cap V(G'))$
$(G, l) \equiv (G', l')$	$S(G) \simeq_X S(G')$
$\|G\|$	the number of nodes of G, i.e., $\|V(G)\|$
$(\cdot, x)$	the (unique) incoming arc of a tree node or degree-2 node x
$(x, \cdot)$	the (unique) outgoing arc of a reticulation or degree-2 node x

Phylogenetic networks

$T = (V, A, l)$	a directed phylogenetic tree on X, with labels $l : W \to X$
$T = (V, E, l)$	an undirected phylogenetic tree on X, with labels $l : W \to X$
$N = (V, A, l)$	a directed phylogenetic network on X, with labels $l : W \to X$
$U = (V, E, l)$	an undirected phylogenetic network on X, with labels $l : W \to X$
$\dot{N}$	a directed internally labeled phylogenetic network
$\dot{U}$	an undirected internally labeled phylogenetic network
$\ddot{N}$	a directed subdivided internally labeled phylogenetic network
$\ddot{U}$	an undirected internally labeled subdivided phylogenetic network
$c(v)$	a child of the node v
$p(v)$	a parent of the node v
$l(v)$	the label of a node v of a network
$r(N)$	the reticulation number of a network
v_k^n	the number of nodes in an undirected network in $\mathcal{U}(n, k)$
$L(N)$	the set of leaves of an (un)directed network N
$R(N)$	the set of reticulations of a directed network N

$X(N)$	the labels $l(V(N))$ of a network N, often simply X
X^l	the leaf labels of an internally labeled network $\dot{N}$ or $\dot{U}$ labeled with X
X^t	the tree node labels of an internally labeled network $\dot{N}$ labeled with X
X^r	the reticulation labels of an internally labeled network $\dot{N}$ labeled with X
X^i	the degree-3 (internal) node labels of an internally labeled network $\dot{U}$ labeled with X
$X^{(2)}$	the degree-2 node labels of an internally labeled network $\ddot{N}$ labeled with X
$T \downarrow v$	the pendant tree of T with outdegree-2 node v
T^+	the tree obtained from T by attaching a new leaf to the root arc
$\mathrm{LCA}(u, v)$	the set of LCAs of two nodes u and v
$S(\ddot{N})$	the supressed version of a subdivided network
$H_k(T, x, y)$	the handcuffed tree with base tree T and handcuffs between the incoming arcs of x and y
$U(N)$	the underlying undirected network of a directed network N
$\mathcal{T}(N)$	the set of embedded trees of N
$N[Y]$	the labeled digraph obtained by restricting N to $Y \subseteq V(N)$
N_A	a network with two leaves and one reticulation
$\perp\!\!\!\perp$	the upside-down version of a tree T
$C(X)$	the caterpillar on the ordered set of leaves X
$B(X)$	the balanced tree on the ordered set of leaves X
$\langle x, y, z \rangle$	a triangle with nodes x, y, z and long side (x, z)
$\langle x, y, z \rangle_t$	a triangle with nodes x, y, z and long side (x, z) where y is a tree node
$\langle x, y, z \rangle_r$	a triangle with nodes x, y, z and long side (x, z) where y is a reticulation
$\langle \cdot x, y, z \cdot \rangle$	a subdivided triangle with degree-3 nodes x, y, z and long path (x, z)

Network spaces

$(p,u,c) \xrightarrow{(u,v)} (p',c')$	the tail move which moves the u-end of (u,v) from (p,c) to (p',c')
$u \xrightarrow{(u,v)} (p',c')$	the tail move which moves the u-end of (u,v) to (p',c')
$(p,v,c) \xrightarrow{(u,v)} (p',c')$	the head move which moves the v-end of (u,v) from (p,c) to (p',c')
$v \xrightarrow{(u,v)} (p',c')$	the head move which moves the v-end of (u,v) to (p',c')
$\{x,u,y\} \xrightarrow{\{u,v\}} \{x',y'\}$	the SPR move which moves the-u end of $\{u,v\}$ from $\{x,y\}$ to $\{x',y'\}$
$u \xrightarrow{\{u,v\}} (x',y')$	the SPR move which moves the u-end of $\{u,v\}$ to $\{x',y'\}$
$\mathcal{N}(n,k)$	the set of directed networks with n leaves and k reticulations
$\mathcal{N}_M(n,k)$	the space of networks on $\mathcal{N}(n,k)$ using move type M
$\dot{\mathcal{N}}(n,k)$	the set of internally labeled directed networks with n leaves and k reticulations
$\dot{\mathcal{N}}_M(n,k)$	the space of internally labeled networks on $\dot{\mathcal{N}}(n,k)$ using move type M
$\dot{\mathcal{N}}(n,k,m)$	the set of subdivided internally labeled directed networks with n leaves, k reticulations, and m degree two nodes
$\dot{\mathcal{N}}_M(n,k,m)$	the space of subdivided internally labeled networks on $\dot{\mathcal{N}}(n,k,m)$ using move type M
$d_M(N,N')$	the graph distance between N and N' in $\mathcal{N}_M(n,k)$ or $\dot{\mathcal{N}}_M(n,k)$
$\mathcal{U}(n,k)$	the set of undirected networks with n leaves and k reticulations
$\mathcal{U}_M(n,k)$	the space of networks on $\mathcal{U}(n,k)$ using move type M
$\dot{\mathcal{U}}(n,k)$	the set of internally labeled undirected networks with n leaves and k reticulations
$\dot{\mathcal{U}}_M(n,k)$	the space of internally labeled undirected networks on $\dot{\mathcal{N}}(n,k)$ using move type M

$\dot{\mathcal{U}}(n,k,m)$	the set of subdivided internally labeled undirected networks with n leaves, k reticulations, and m degree two nodes
$\dot{\mathcal{U}}_M(n,k,m)$	the space of subdivided internally labeled undirected networks on $\dot{\mathcal{N}}(n,k,m)$ using move type M
$d_M(U,U')$	the graph distance between U and U' in $\mathcal{U}_M(n,k)$ or $\dot{\mathcal{U}}_M(n,k)$
$M \subseteq M'$	each move of type M is a move of type M'
$\mathrm{diam}_M(n,k)$	the diameter of $\mathcal{N}_M(n,k)$ or $\mathcal{U}_M(n,k)$
$\mathrm{diam}_M(n,k,m)$	the diameter of $\dot{\mathcal{N}}_M(n,k,m)$ or $\dot{\mathcal{U}}_M(n,k,m)$

Index