Universitat
de les Illes Balears

DOCTORAL THESIS
2016

RECONSTRUCTION PROBLEMS FOR LGT NETWORKS

Joan Carles Pons Mayol

Universitat
de les Illes Balears

DOCTORAL THESIS
2016

Doctoral Programme of Mathematics

RECONSTRUCTION PROBLEMS FOR LGT NETWORKS

Joan Carles Pons Mayol

Thesis Supervisor:
Dr. Gabriel Cardona Juanals

Doctor by the Universitat de les Illes Balears

Statement of Authorship

This thesis has been submitted to the *Escola de Doctorat*, *Universitat de les Illes Balears*, in fulfilment of the requirements for the degree of *Doctor en Matemàtiques*. I hereby declare that, except where specific reference is made to the work of others, the content of this dissertation is entirely my own work, describes my own research and has not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university.

Joan Carles Pons Mayol

Palma, September 2016

Funding

The work reported in this thesis was supported by *Ministerio de Ciencia e Innovación* through grants *Grafos en biología computacional* (MTM2009 07165), *Aplicaciones bioinformáticas en filogenética, metagenómica, biología de sistemas y genómica del cáncer* (DPI2015-67082-P), *Creación de una red temática en computación biomolecular y celular* (TIN2008-04487-E/TIN) and *Renovación y nuevas actividades de la red temática en computación biomelecular y biocelular* (TIN2011-15874-E) and *Obra Social "La Caixa"* through *Programa Pont "La Caixa" per a grups de recerca de la UIB.*

Supervisor's Agreement

I, Gabriel Cardona, Ph.D. in Mathematics and Associate Professor at the *Department of Mathematics and Computer Science, Universitat de les Illes Balears*

ATTEST THAT

this dissertation, titled *Reconstruction Problems for LGT Networks* and submitted by Joan Carles Pons Mayol for obtaining the degree of *Doctor en Matemàtiques*, was carried out under my supervision and contains enough contributions to be considered as a doctoral thesis.

Dr. Gabriel Cardona

Palma, September 2016

Abstract

Phylogenetics is the study of evolutionary history and relationships among species, and in particular, its reconstruction from biological data. It plays an important role in understanding biology because it allows to stablish the relationships between organisms. Based on Darwin's theory, which states that all species have evolved from a common ancestor, evolutionary histories have been represented using trees. However, when non-vertical evolutionary events such as hybridizations, recombinations and lateral transfer of genes occur, the use of phylogenetic networks is indeed more appropriate than trees in order to model those reticulate evolutionary histories.

The main motivation of this thesis is to develop a new model for phylogenetic networks modelling evolutionary histories with lateral gene transfers, as well as computational methods and algorithms for their reconstruction.

The new model we propose, which we call LGT networks, captures the asymmetry of lateral gene transfer events. The model is based on considering a principal tree that represents the main line of evolution of the considered species, and a set of arcs modelling lateral transfer events. Our LGT networks generalizes some other existent models which were designed for a similar purpose.

We solve the well-known phylogenetic network reconstruction problem for the abovementioned LGT networks from induced sets of trees and trinets. Both cases require some topological constraints to be imposed in order to obtain unicity of solutions, which is lost when considering generic LGT networks.

We reconstruct such networks from a set of trees formed by a principal tree and a set of secondary subtrees, being each of these secondary subtrees associated to a specific secondary arc. To do this, we propose a polynomial algorithm, which we applied to real biological data sets in order to predict or discover lateral transfer events. We also study the reconstruction problem from a set of "basic" LGT networks on three leaves and with only one secondary arc. We call such networks tri-lgt-nets, which are similar to the well-known trinets. With this, we contribute to extend the set of possible phylogenetic networks that can be recovered using the previous substructures.

Finally, we extend the framework for the reconciliation problem between gene trees and species trees using LGT networks as the species phylogeny. In order to set up the evolutionary scenario, we allow transfer events via secondary arcs of the network only, as well as duplications and losses. For this model, we present fast computational algorithms addressed to obtain the most parsimonious reconciliation between a gene tree and an LGT network.

Resum

La filogenètica és l'estudi de les històries evolutives i les relacions entre espècies, i en particular la seva reconstrucció a partir de dades biològiques. Aquesta juga un paper important en la comprensió de la biologia, ja que permet determinar les relacions de parentiu entre organismes. Basant-se en la teoria de Darwin, la qual defensa que totes les espècies han evolucionat d'un ancestre comú, les històries evolutives s'han representat emprant arbres. No obstant això, quan ocorren processos evolutius no verticals tals com hibridacions, recombinacions i transferències laterals de gens, l'ús de xarxes filogenètiques és, certament, més apropiat que l'ús d'arbres per a modelar aquestes històries evolutives reticulars.

La principal motivació d'aquesta tesi és desenvolupar un nou model per a xarxes filogenètiques que modelen històries evolutives amb transferències laterals de gens, així com mètodes computacionals i algorismes per a la seva reconstrucció.

El nou model que proposem, i que anomenem xarxes LGT, captura l'asimetria de les transferències laterals de gens. El model es basa en considerar un arbre principal, representant la línia principal d'evolució de les espècies considerades, i un conjunt d'arcs modelant les transferències laterals de gens. Les nostres xarxes LGT generalitzen altres models ja existents que foren dissenyats amb un propòsit similar.

Resolem també el problema ben conegut de reconstrucció de xarxes filogenètiques per a les esmentades xarxes LGT a partir de conjunts induïts d'arbres i de trinets. Ambdós casos requereixen que s'hi imposin algunes restriccions topològiques per obtenir unicitat de solucions, que es perd considerant xarxes LGT genèriques.

Reconstruïm aquestes xarxes a partir d'un conjunt d'arbres format per un arbre principal i un conjunt d'arbres secundaris. Cadascun d'aquests últims està associat a un arc secundari específic. Per fer-ho, proposem un algorisme polinòmic que apliquem a dades biològiques reals per a predir o descobrir processos de transferència lateral de gens. També estudiem el problema de reconstrucció a partir d'un conjunt de xarxes LGT "bàsiques" de només tres fulles i un sol arc secundari. Anomenarem a aquestes xarxes, xarxes tri-lgt-nets que serien similars a les conegudes trinets. Amb això, contribuïm a estendre el conjunt de xarxes filogenètiques que poden ser reconstruïdes emprant les subestructures prèvies.

Finalment, estenem el marc del problema de reconciliació entre arbres de gens i arbres d'espècies emprant les xarxes LGT com a filogènia d'espècies. Per establir l'escenari evolutiu, permetem les transferències laterals de gens, només a través dels arcs secundaris de la xarxa, així com també duplicacions i pèrdues. Per això, presentem algorismes computacionals ràpids adreçats a obtenir la reconciliació més parsimoniosa entre un arbre de gens i una xarxa LGT.

Resumen

La filogenética es el estudio de las historias evolutivas y las relaciones entre especies, y en particular de su reconstrucción a partir de datos biológicos. Esta juega un papel importante en la comprensión de la biología puesto que permite establecer las relaciones de parentesco entre organismos. Bajo la teoría de Darwin, que defiende la procedencia de todas las especies de un ancestro común, las historias evolutivas han sido representadas usando árboles. No obstante, cuando ocurren procesos evolutivos no verticales cómo hibridaciones, recombinaciones o transferencias laterales de genes, el uso de redes filogenéticas es, ciertamente, más apropiado que el uso de árboles para modelar estas historias evolutivas reticulares.

La principal motivación de esta tesis es desarrollar un nuevo modelo para redes filogenéticas que modelan historias evolutivas con transferencia lateral de genes, además de métodos computacionales y algoritmos para su reconstrucción.

El nuevo modelo que proponemos, que llamamos redes LGT, captura la asimetría de las transferencias laterales de genes. El modelo se basa en considerar un árbol principal representando la línea principal de evolución de las especies consideradas y un conjunto de arcos modelando las transferencias laterales de genes. Nuestras redes LGT generalizan otros modelos ya existentes que fueron diseñados con un propósito similar.

Resolvemos el conocido problema de reconstrucción de redes filogenéticas para las mencionadas redes LGT a partir de conjuntos inducidos de árboles y trinets. En ambos casos se requiere la imposición de restricciones topológicas para obtener unicidad en las soluciones, ya que esta se pierde cuando consideramos redes LGT genéricas.

Reconstruimos estas redes a partir de un conjunto de árboles formado por un árbol principal y un conjunto de árboles secundarios. Cada uno de estos últimos está asociado a un arco secundario específico. Para hacerlo, proponemos un algoritmo polinomial que aplicamos sobre datos biológicos reales para predecir o descubrir procesos de transferencia lateral de genes. También estudiamos el problema de reconstrucción a partir de un conjunto de redes LGT "básicas" de unicamente tres hojas y un sólo arco secundario. Llamamos a estas últimas redes tri-lgt-nets, similarares a las conocidas como trinets. Con esto, contribuimos a extender el conjunto de posibles redes filogenéticas que se pueden reconstruir usando estas subestructuras.

Finalmente, extendemos el marco del problema de reconciliación entre árboles de genes y árboles de especies usando las redes LGT cómo filogenia de especies. Con el fin de establecer el escenario de evolución permitimos transferencias laterales de genes, sólo a través de los arcos secundarios de la red, así como también duplicaciones y pérdidas. Para esto, presentamos algoritmos computacionales rápidos dirigidos a obtener la reconciliación más parsimoniosa entre un árbol de genes y una red LGT de especies.

Agraïments

Arribat aquest moment i després del recorregut que m'ha portat fins aquí, no vull deixar passar l'oportunitat d'expressar el meu agraïment:

- Al meu director, Biel Cardona, pel seu suport i seguiment durant la meva carrera universitària des dels meus inicis i fins al dia d'avui. Sense ell, aquest treball de recerca no hauria estat possible i, el meu interès per l'àlgebra, tampoc.
- Al grup de *Biologia Computacional i Bioinformàtica* (BIOCOM) de la *Universitat de les Illes Balears* (UIB), per contribuir a la meva formació, especialment a Cesc Rosselló com a director del grup.
- Al *Departament de Matemàtiques i Informàtica* de la UIB i en representació seva a Ricardo Alberich, director del mateix durant aquest treball de tesi.
- A na Cati Vich, per tot (hi ha massa coses per anomenar-les totes).
- A en Joan Duran i na NN Vich; cadascun en sap els motius particulars però, gràcies per ser-hi, recolzar-me i ajudar-me en tot moment.
- Als amics d'Algaida i altres companys del departament de la UIB, pel seu interès a seguir i conèixer el meu camí.
- À Céline Scornavacca pour le traitement et l'aide qu'elle m'a donné, dans et hors l'université pendant mon séjour de recherche à *Institut des Sciences de l'Evolution, Université Montpellier II.*
- To Krzysztof Bartoszek, Marta Casanellas and Jesús Fernández to open the doors and for the help they have provided me with during my visits in *Uppsala University* and *Departament de Matemàtica Aplicada I, Universitat Politècnica de Catalunya*, respectively.
- Als meus pares, la meva germana i als meus avis, per la seva ajuda i la força que m'han transmès.
- A en Miquel Amengual, al que li dec la meva passió per les matemàtiques des de ben jove.

Contents

Introduction

Computational biology and bioinformatics are disciplines on the border between mathematics, computer science and biology. They aim to solve the algorithmic problems that appear in molecular biology and are dealt with specific computer tools. In the last decades, technology has experienced a vast improvement, which has boosted research in computational biology and bioinformatics, allowing the continuous development of these disciplines.

Such bioinformatics tools have become essential for biologists, since they are of great use in data generation, analysis and applications. Among other fields, computational biology has become of great importance in phylogenetics, which can be defined as the science that classifies living organisms based on their ancestral relationships. This branch of the biology allows us to study the evolutionary history of a group of organisms.

Looking backwards, already early nature researchers tried to classify the different groups of organisms on Earth, starting with Aristotle, who classified such organisms in a *scalae naturae*. In his scale, the simplest organisms were placed at the bottom, and as its complexity increased they had higher position, until the top, where we find the most complex organisms, humans. However, Aristotle's classification did not attract the interest of major experts in natural history until the 17th and 18th centuries, when Leclerc, Bonnet and Linné based their work on Aristotle's idea. The latter of this researchers is the father of the classification of the living beings, best known as taxonomy. In the 19th century, Lamarck proposed an evolutionary theory which explained the evident and gradual change of living beings within the same species; namely that an organism can pass characteristics acquired during his lifetime onto its descendants. He suggested the use/disuse theory, based on the inheritance of acquired characteristics. In the mid 19th century, a different theory was presented by Darwin in his famous work *On the Origin of Species*, where he established that species evolve from generation to generation by natural selection. He also explains the variety of organisms coming from a common ancestor, using a tree-like representation relating such organisms. Such evolutionary idea forms the basis of phylogeny, a concept introduced later by Haeckel.

Following Darwin, the evolutionary history of a group of species, or phylogeny, has been represented using a (phylogenetic) tree. Ultimately, the *Tree of Live* would show the evolution of all extant and non extant species on Earth from a single common ancestor. Such trees can take a wide variety of shapes, being the most commonly used a directed graph with labelled leaves, each of which refers to an organism. The internal nodes of the graph represent common ancestors, the arcs model the lineage persistence across time, and the root the most recent common ancestor of all considered species.

Since the DNA discovery, at the mid-20th century, the use of the aforementioned phylogenetic trees has notably evolved. Nowadays, their use has been extended to also describe relationships between gene families [Mäser et al. (2001)], to explain population histories

[Edwards (2009)], pathogenic dynamics and epidemiology [Grenfell et al. (2004)], cancer research [Campbell et al. (2010)], language evolution [Gray et al. (2009)], classification of metagenomics sequences [Brady and Salzberg (2011)], gene identification [Kellis et al. (2003)], and reconstruction of ancestral genomes [Paten et al. (2008)], among others.

Despite the importance and wide applicability of phylogenetic trees, some relevant evolutionary events, which mainly affect specific groups of organisms, cannot be properly modeled properly using a tree [Martin (2011); Doolittle and Bapteste (2007)]. In such cases, trees simplify too much the evolution scenario since they only account for those evolutionary processes where each organism has a single direct ancestor. In other words, they are only suitable for modelling "vertical" events, such as mutations. There exists evidence of the existence of phylogenies with evolutionary events that cannot be explained using the paradigm of the single direct ancestor, referred to as reticulations or "horizontal" events. They enclose events such as lateral gene transfers, hybridizations and recombinations, to name a few. Lateral gene transfers consists in the transmission of genes from an organism to another one not genealogically related to the former [Chia and Goldenfeld (2011)]. Examples of this process are found mainly in bacteria [Hotopp (2011); Hao and Golding (2004); Polz et al. (2013)], as well as in some plants [Yue et al. (2012); Nikolaidis et al. (2014)]. Hybridization is the creation process of a hybrid organism, which takes place in specific plants and groups of fishes [Mallet (2007); Seehausen et al. (2008)]. Finally, in a recombination event, descendants can have attributes not found in their parents, as happens in recombination events in viruses [Holmes et al. (1999); Martin et al. (2015)].

Thus, we have strong evidence that reticulation events are present in a substantial variety of organisms and hence the quest for the *Tree of Life* is simply vain: *"Molecular phylogenists will have failed to find the 'true tree', not because their methods are inadequate or because they have chosen the wrong genes, but because the history of life cannot properly be represented as a tree"* [Doolittle (1999)]. In order to include such reticulation events in the modelling of evolutionary histories, a new model is required that generalizes phylogenetic trees: phylogenetic networks. As it happens in the case of phylogenetic trees, one can consider *rooted* as well as *unrooted* phylogenetic networks, and for both kinds of networks we find in the literature a large amount of different definitions and subclasses. In this dissertation we will focus on rooted phylogenetic networks, for which there seems to be a consensus in the use, as mathematical model, of rooted directed acyclic graphs with a bijective correspondence between their leaves and the set of extant species under consideration [Semple and Steel (2003); Huson et al. (2010); Morrison (2011); Felsenstein (2004)].

One of the main goals of phylogenetics is the development of methods to reconstruct evolutionary histories. With these methods we obtain phylogenetic trees, as well as networks, that represent the most accurate hypothesis of the ancestral relationships between species. There exist lots of references in the literature about tree inference [De Bruyn et al. (2014) and references therein]; however, the reconstruction problem using phylogenetic networks is nowadays still a challenge. Roughly speaking, this reconstruction problem consists in obtaining an optimum phylogenetic network under certain specific constraints that models the evolution of different organisms. The algorithms to reconstruct phylogenetic networks can have as input data clusters [Van Iersel et al. (2010a)], triples [van Iersel and Kelk (2011)], distances [Francis and Steel (2015a)], trees [Wu (2013)] and trinets [Oldman et al. (2016)], among others. Nevertheless, the reconstruction problem for arbitrary networks is necessarily difficult, since it has been proved to be NP-hard, and hence no polynomial algorithm can solve it (if one assumes that $P \neq NP$) [Kanj et al. (2008); Bordewich and Semple (2007)]. Hence, some topological constraints, either based on biological facts or mathematical conditions, are needed in order to tackle the problem. For further reading and full guide on phylogenetic networks, as well as related software, we refer the reader to Gambette (2010), Huson et al. (2010) and Morrison (2011).

Although in the early days of phylogenetics the reconstruction of the evolutive history of species was based on visible characteristics of the species (that is, their *genotype*), all modern studies are based on the comparison of genetic material (that is, their *genotype* or more generally their full *genome*). Since the comparison of the full genome of species is clearly intractable, in order to study the evolution of a set of species, one usually takes different genes and studies their evolution in the species under consideration, obtaining *gene trees*. The fact that different genes are considered, together with the diversity of reconstruction algorithms, makes that different studies may lead to different evolutive hypothesis.

This discrepancy between different studies leads to different problems. One of them is the yet mentioned reconstruction problem for phylogenetic networks: given different gene trees explaining the evolutive history of a set of genes in different species, try to infer a phylogenetic network that explains the evolution of species. A second problem is the comparison of trees and networks, that can be treated mathematically as finding sound distances in the spaces of phylogenetic trees [Robinson (1971); Hein (1990); Allen and Steel (2001)] and networks [Moret et al. (2004); Cardona et al. (2008c,a, 2009b,e,c); Nakhleh (2010b)]. A third problem is how to harmonize the evolutive history of genes with the evolution of species.

This last problem is commonly known as the reconciliation problem between gene trees and species trees or networks. The solution to this problem is given by a *reconciliation scenario*, where evolutive events at the gene level are mapped to evolutive events at the species level; more technically, it is obtained by means of a mapping that takes each internal node of each gene tree to a node of the species tree or network subject to certain conditions that model mathematically the underlying biological mechanisms. Depending on the events that are considered at gene level (essentially, mutations, duplications, losses and transfers), one obtains different models, such as $\mathbb{DTL}$ models [Doyon et al. (2010), Tofigh et al. (2011), Bansal et al. (2012)], that take into account duplications, transfers and losses, or $\mathbb{DL}$ models [Doyon et al. (2009), To and Scornavacca (2015)], where transfers are not allowed. The reconciliation problem has applications in, for instance, coevolution, [Merkle et al. (2010)] becoming of great importance in parasitology and biogeography, [Ronquist (1995); Page and Charleston (1998); Nieberding et al. (2010); Brooks and Ferrao (2005); Merkle and Middendorf (2005); Charleston and Perkins (2006)].

From an algorithmic point of view, we are thus led to consider the problem of reconstructing combinatorial structures (namely phylogenetic networks and recombination scenarios) from substructures or partial data (gene trees, clusters, subtrees, etc.). The methods developed to solve these reconstruction problems can be divided into two big families, those based on *parsimony* and those based on *maximum likelihood*. In big lines, methods of the first kind are based on Occam's razor: the simplest solution is the best one. An example in reconstruction of networks would be to choose as solution a network with the minimum number of reticulation nodes, or to choose the tree that minimizes the number of mutations on the branches. When working in the reconciliation problem, a parsimonious solution could be the one that minimizes the number of evolutionary events. The second family of methods require the definition of a statistical model in order to evaluate candidate trees, networks or scenarios. In general, methods based on maximum likelihood are much more complex and consume much more computing resources, but the results they give are statistically optimal. Thus, both methods present their own advantages and disadvantages;

however, they both share the same goals: efficiency, which tries to reduce the execution time; consistency, which asks if the phylogenetic tree or network is a reliable reproduction of the given data; and robustness, which measures the sensitivity of a particular method to small changes in the input data.

Contributions of the dissertation

The main subject of this dissertation is the development of models and algorithms for the reconstruction of phylogenetic networks modelling lateral gene transfers and its reconciliation with gene trees.

LGT networks

The first contribution is the development of a model to describe phylogenetic networks where the non-tree-like events are lateral transfers of genes. The existing models for phylogenetic networks did not allow to distinguish between the different species involved in the appearance of a new one, which is appropriate for events like hybridizations or recombinations. However, in lateral gene transfers, there is one (and only one) species that contributes in greatest measure to the genetic material of the formed species. We introduce a new kind of phylogenetic networks, that we call *LGT networks*, that model properly this asymmetry in the contributions of the different parents. We do so by distinguishing the arcs in the network between *principal* and *secondary* arcs; the former describe the main line of evolution and the latter describe lateral gene transfer events. We also define a sound metric on the space of LGT networks that extends the well known Robinson-Foulds distance on trees.

Reconstruction from trees

Our second contribution in this field is the development of a method for the recovery of an LGT network from a set of trees that it induces. More precisely, we consider the *principal* subtree of an LGT network, formed by its principal arcs, and for each secondary arc one *secondary* subtree, that essentially models a gene that has evolved through the main line of evolution except for the chosen arc, through which it has been transferred laterally. In order to solve the associated reconstruction problem, we introduce a technical condition on the networks to consider and give an algorithm that recovers the network from the set of trees (or detects that no such network can exist).

These results have been published in a journal article:

Gabriel Cardona, Joan Carles Pons, Francesc Rosselló. A reconstruction problem for a class of phylogenetic networks. *Algorithms for Molecular Biology*, 2015; 10:28.

Reconstruction from trinets

The third contribution of this dissertation is the study of the reconstruction problem of LGT networks from substructures induced by triplets of leaves. This problem is well known for trees, but for phylogenetic networks our knowledge is limited to some particular cases. The first problem to solve is how can one define a substructure induced by a triplet, like trinets used on some phylogenetic networks, that is more suitable for LGT networks, and then decide whether or not one can recover the full network from this data. For this reason, we introduce what we call *basic tri-lgt-nets*, which are LGT networks with three leaves and at most one secondary arc, and a class of LGT networks that can be singled out by the basic tri-lgt-nets that they contain.

The results we have obtained have led to a preprint (joint with Dr. Gabriel Cardona) that will be submitted to *Journal of Mathematical Biology.*

Reconciliation with gene trees

Finally, the fourth contribution of this dissertation is the development of efficient algorithms for solving the reconciliation problem between gene trees and LGT networks representing the evolution of species. We consider a scenario with duplications, losses and transfers, but restrict transfers to happen through secondary arcs. In this setting we adapt previous results on trees to obtain a polynomial algorithm that gives the most parsimonious reconciliation between a gene tree and an LGT network.

These last results were obtained while a research stay of the author at the *Institut des Sciences de l'Evolution, Université Montpellier II* under the supervision of Dr. Céline Scornavacca and have led to a manuscript (joint with Dr. Céline Scornavacca and Dr. Gabriel Cardona) that has been submitted to *Journal of Theoretical Biology.*

Organization of the text

This dissertation, apart from this introduction, consists of five chapters plus one last small chapter of conclusions.

Chapter 1 provides a review of phylogenetic trees, phylogenetic networks and an overview of methods to infer and reconcile such networks. Moreover, some biological concepts and processes that cause dissimilarities between phylogenies are described.

In Chapter 2 we introduce the model of LGT networks and some notation relative to such networks that is used in later chapters. Also, there is a comparison between our model and other ones designed in a similar flavour, and a metric which allows for its comparison.

In Chapter 3 we define the secondary subtrees, reduced versions of subtrees and the SPR operation that can be used to analyze the distance between the principal and secondary subtrees. Moreover, we present the subclass of restricted LGT networks and also give an algorithm that allows for the reconstruction of such networks from the set of phylogenetic trees corresponding to its principal and secondary subtrees, provided that such a network exists. Finally, in order to test our algorithms, we include two computational experiments using real biological data.

In Chapter 4 we introduce what we call basic tri-lgt-nets, which are LGT networks with 3 leaves and one secondary arc. We define how a secondary arc induces a set of those basic networks and characterize which tri-lgt-nets are represented in a network. This leads to the concept of redundant arcs, arcs whose removal does not change the basic lgt networks represented, and their coverings. Finally we exhibit a class of LGT networks that are determined by the set of tri-lgt-nets that they represent.

Finally, in Chapter 5 we first look into the reconciliation problem between gene trees and species trees, its extension to species network and the barriers to extend the duplication and loss model to the one where also transfer events are allowed. Then, we solve the problem of finding the most parsimonious reconciliation using LGT networks as species networks and provide computationally efficient algorithms for its computation.

Chapter 1

Preliminaries

Contents

In this chapter, we review concepts, definitions and problems relevant to this dissertation. First we introduce basic required nomenclature to work in graph theory and we briefly summarize some biological concepts and processes which are important for phylogenetic evolution. Then, we define phylogenetic trees and networks, some concepts where both structures appear to be linked and different ways to represent them. Mainly focusing on phylogenetic networks, we review their classification and also different ways to compare them. Finally, we introduce the reconstruction problem of phylogenetic networks and the reconciliation problem between gene trees and species trees.

1.1 Graphs

An *undirected graph* is an ordered pair $G = (V, E)$ (see Figure 1.1 (a)) where V is a set of *vertices* or *nodes* and E is a set of *edges*. Each edge e is determined by an unordered pair of nodes $\{u, v\}$ which are called its *end nodes* or simply its *ends*; to simplify notations, we simply write $e = uv$. In this case we say that the node u (or v) and the edge e are *incident* and also that u and v are *adjacent*.

A *directed graph* (see Figure 1.1 (b)) is defined analogously as in the undirected case except that the pair of nodes defining an *arc* (which is the name used for edges in directed graphs) are taken as an ordered pair (u, v). In a directed graph, given an arc (u, v), the node u is called the *source* or *starting node* of the arc and the node v is called its *target* or *destination node*. In this case we also say that v is a *child* of u and that u is a *parent* of v. The pair of nodes that determine an arc are called its *extremes*.

If $G = (V, E)$ is an undirected graph, the *degree* of a node $u \in V$, denoted by $\deg(u)$, is defined as the number of edges incident to u. If u is a node in a directed graph, its *in-degree* (resp. *out-degree*), denoted as $\text{indeg}(u)$ (resp. $\text{outdeg}(u)$), is defined as the number of arcs whose destination node (resp. starting node) is u. We say that a node $u \in V$ is a *root* of G when $\text{indeg}(u) = 0$. We say that a node $u \in V$ is a *leaf* if $\deg(u) = 1$ (in the undirected case) or if $\text{outdeg}(u) = 0$ (in the directed case). The nodes of the graph that are not leaves are called *internal* nodes. The sets of leaves and internal nodes of a graph G are denoted by $\mathcal{L}(G)$ and $\mathcal{I}(G)$, respectively. Sometimes we also consider that some nodes of the graph are *labelled* by a set $L(G)$; that is, one considers a mapping from a certain subset U of V to $L(G)$. Although in a general setting all nodes can be labelled, we hereafter will consider that the labelled nodes are exactly the leaves and that the labelling is injective, that is, no different leaves share the same label. More formally, we consider a fixed bijection between $\mathcal{L}(G)$ and $L(G)$. Also, we will identify, usually without further mention, a leaf with its label.

Given $G = (V, E)$ a directed graph, and $u, v \in V$, a *directed path* P between u and v, denoted by $u \rightsquigarrow v$, is a sequence of nodes $(u = u_0, u_1, \ldots, u_{k-1}, v = u_k)$ with $k \geq 1$ such that $(u_{i-1}, u_i) \in E$ for all $i = 1, \ldots, k$. When $k > 1$, the path is *proper*. A *directed cycle* is a proper directed path which starts and finishes in the same node. A *directed acyclic graph*, or simply a DAG, is a directed graph that does not contain any directed cycle. A DAG is *rooted* when it has only one root, and it is sometimes simply called an rDAG. A rooted directed acyclic graph labelled on a set S is called a S-rDAG (see Figure 1.1(b) or Figure 1.3).

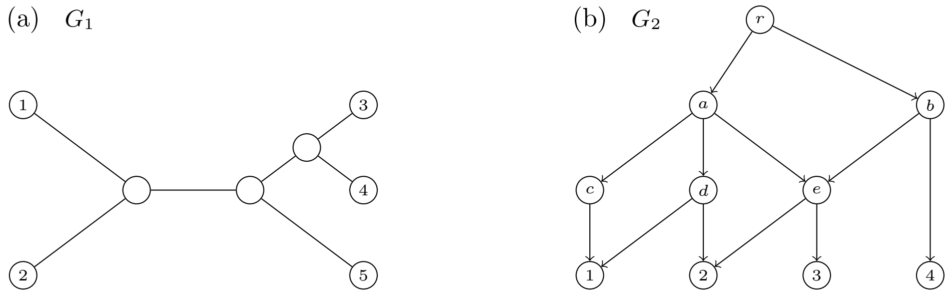

Figure 1.1: (a) The graph G_1 is an undirected graph labelled on $\{1, 2, 3, 4, 5\}$. (b) The graph G_2 is a rooted directed acyclic graph labelled on $S = \{1, 2, 3, 4\}$ (i.e. an S-rDAG). The root of G_2 is r, its leaves are $\mathcal{L}(G_2) = \{1, 2, 3, 4\}$ and its internal nodes are $\mathcal{I}(G_2) = \{r, a, b, c, d, e\}$. The proper directed path $r \rightsquigarrow 2$ can represent any of the paths $(r, a, d, 2)$, $(r, a, c, 2)$, or $(r, b, c, 2)$. The nodes c, d, e are the children of a, hence $\text{outdeg}(a) = 3$. The nodes a i b are the parents of e and hence $\text{indeg}(e) = 2$.

A directed acyclic graph is *connected* if there is an *undirected path*, that is a directed path ignoring arc orientations, between each pair of nodes. A node (arc) of a directed graph is called a *cut node* (*cut arc*) if its removal disconnects the graph. A directed graph is *biconnected* if it contains no cut-nodes. A biconnected subgraph B of a directed graph G is said to be a *biconnected component* if there is no biconnected subgraph $B' \neq B$ of G that contains B. For example, the directed graph depicted in Figure 1.2 has four biconnected components.

Two S-rDAGs on the same set S are *isomorphic* if there exists an isomorphism of DAGs between them that preserves and reflects the respective labellings of the leaves. More

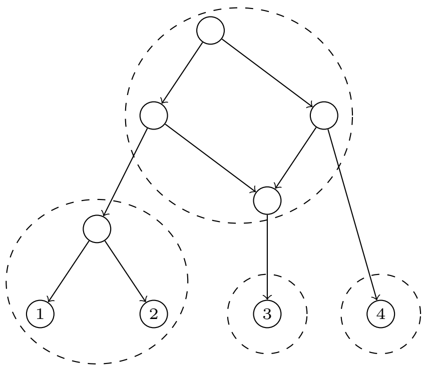

Figure 1.2: A rooted directed acyclic graph which biconnected components are highlighted by circles.

formally, given $G = (V, E)$ and $G' = (V', E')$ two S-rDAGs, an isomorphism between G and G' is a bijection $\phi : V \to V'$ such that:

- $(u, v) \in E$ if, and only if, $(\phi(u), \phi(v)) \in E'$;
- $u \in V$ is a leaf labelled by $s \in S$ if, and only if, $\phi(u) \in V'$ is a leaf labelled by s.

We denote by $G \cong G'$, or even $G = G'$, if two $S-$rDAGs G and G' are isomorphic. For example, the two rooted directed acyclic graphs labelled on $\{1, 2, 3, 4\}$ depicted in Figures 1.2 and 1.3 are isomorphic.

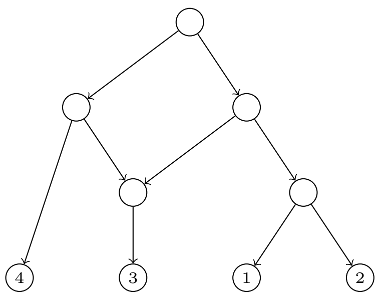

Figure 1.3: A rooted directed acyclic graph isomorphic to the one depicted in Figure 1.2.

1.2 Some biological concepts

In this section we present some basic biological concepts and processes which will appear later in this manuscript and that will help us to understand some of the biological phenomena modelled in bioinformatics. For a broader vision, see Otto and Day (2007), Purves et al. (2003), Nei (1987) or Crow et al. (1986), among others. We will focus on the mechanisms of genomic evolution which give rise to new biological entities.

The functioning of all known living organisms is governed by *genetic* instructions that are encoded in molecules of *Desoxyribo Nucleic Acid* (DNA). This molecules are formed by long chains of *nucleotides* (or *bases*); each of these nucleotides can either be *cytosine* (C), *guanine* (G), *adenine* (A), or *thymine* (T). Thus, a sequence of DNA can be encoded by a string on the alphabet $\{\mathrm{C, G, A, T}\}$. The *genome* of a living organism is the full set of DNA that is present in each cell of the organism and identifies it. This genome is organized in *chromosomes*; for instance, the humane genome, that is formed by approximately 3 billions of bases, is organized in 23 pairs of chromosomes, whose length can vary from approximately 50 millions of bases to about 250 millions of bases. The DNA in each chromosome is divided into different *genes* that can act as instructions for the cell to make proteins and ultimately determine the characteristics of the living being that will be transmitted to its descendants.

Although all living organisms have their own genome, and in principle different organism have different genomes, in order to study characteristics that are common to a group of individuals, they are grouped into *species*. We shall not get into the details of how species defined, that is, what makes two different organisms to be considered as members of the same species or not. In either case, members of the same species share a great amount of genome; for instance, the human genetic variation is only 0.5% (approximately). We can therefore make a simplification and assume that all members of a species have "the same" genome. However, different populations of the same species subject to different conditions may evolve differently, making their genomes to diverge and giving rise to different species in a process called *speciation*.

There are different mechanisms present in the evolution of species. Roughly speaking, we can say that there are two kinds of such mechanisms: *vertical* and *horizontal* events. Vertical events are those where the evolution of a species does not involve the exchange of genetic material with other species, while in horizontal events there is such an exchange. Classically, the studies of the evolution of species have only taken into account vertical events, basically mutations and natural selection, as introduced by Darwin. However, in the last few decades it has become clear that evolution cannot be properly explained without taking into account horizontal events.

This horizontal events include hybridizations, lateral gene transfers and recombinations. *Hybridizations* model the formation of a new species by cross-breeding two members of different species. The new species members, called *hybrids*, acquire equal amount of genetic material from both parent species. In *lateral (or horizontal) gene transfers* (LGT or HGT events, for short) there is a (generically small) transfer of genetic material from one species to an unrelated species. Finally, in *recombinations* different traits of the genome found in either parent are combined making the offspring to have a trait distinct to those found in the parents.

Although most of the genetic evolution studies have centered their attention on vertical events, it is now evident the increasing importance of events like lateral gene transfers in evolution theory [Boto (2010); Daubin and Szöllősi (2016)]. Among other things, lateral gene transfers enable the generation of new genetic combinations as well as the expansion of those with high biological efficiency. There is a high presence of this phenomenon in bacteria and viruses [Ochman et al. (2000); Keen (2012)] but it can also be present in plants and animals [Keeling and Palmer (2008)].

Notice that reticulate evolution may function at different levels. For example, through hybridization at the species level and through recombination below the species level, at population level.

If we focus on evolutionary events at gene level, where some sequences of DNA nucleotides experience alterations, we find the particular case of *duplication* and *loss* events. A duplication takes place when two homologous chromosomes break at different points and are reconnected to the wrong partners. In such cases, one of the resulting molecules will have a missing DNA segment, which has been deleted, and the other one will have two copies of it, hence a duplication. Such duplication processes are important since they permit proteins to adquire new functions or to keep its original function when a mutation affects a gene. When the effect of the duplication for the population is either detrimental or it has a neutral (neither detrimental nor beneficial) effect, this sequence may get lost.

1.3 Trees and networks in phylogenetics

A group of organisms are interrelated by ancestor and descendant relationships derived from "vertical" and "horizontal" processes as those we have seen in the previous section. A *phylogenetic tree* (see also Section 1.10) is a branching diagram illustrating the evolutionary history infered from a set of taxa reflecting vertical events like mutations. The use of phylogenetic trees limits the identification and visualization of more complex evolutionary scenarios. That is, the evolutionary processes like hibridizations, lateral gene transfer or recombinations can not be modelled by a tree structure. For this reason, *phylogenetic networks* were introduced in order to model the evolutionary history of a group of organisms where we can take into account both vertical and horizontal events.

In the field of computational biology, the concepts of phylogenetic trees and networks appear in a multitude of ways depending on many factors [Semple and Steel (2003); Morrison (2011); Huson et al. (2010)]. Generically, they are graphs with certain restrictions that are used to model mathematically some biological problems related to the evolution of species. One of the main differences between different models lies on if one considers trees and networks which are rooted or unrooted. Commonly, rooted trees or networks are called *evolutionary* phylogenetic trees or networks to emphasize the existence of a common ancestor of all considered organisms and that the arcs can be interpreted as the evolution in time between the respective entities (see also Section 1.6).

In this manuscript we will usually consider rooted phylogenetic trees and networks. Hence, the mathematical structure used to design a *phylogenetic network* is, generally, a rooted directed acyclic graph with labelled leaves (labelled rDAG, see Section 1.1). See an example of phylogenetic network depicted in Figure 1.4(a). It is usual to forbid *elementary* nodes, that is nodes with in-degree one (or zero) and out-degree one. In phylogenetic networks, nodes are classified depending on their in-degree; namely, nodes with in-degree one are called *tree nodes* nodes and nodes with in-degree at least two are called *reticulation nodes.* Then, a *phylogenetic tree* is a phylogenetic network without reticulation nodes (see Figure 1.4(c-f)). In this phylogenetic scenario, if there is a directed path $u \rightsquigarrow v$ between two nodes in a (phylogenetic) tree or network, we say that u is an *ancestor* of v, or also v is a *descendant* of u. Given two nodes u, v in a phylogenetic tree T, its *lowest common ancestor* (or LCA, for short), noted as $LCA_T(u, v)$, is their common ancestor that is descendant of every other common ancestor of them. This concept can be extended to consider the LCA of any set of nodes in a tree.

Displayed trees and switching

The definitions of phylogenetic trees and networks as rDAGs are closely related. In fact, a phylogenetic tree is a particular case of a phylogenetic network. The main difference between them is that, in a phylogenetic tree, each pair of nodes are connected by exactly one (undirected) path, while in phylogenetic networks there can be many (undirected) paths connecting two given notes. The ultimate reason for the multiplicity of paths comes from the fact that, in a phylogenetic network, a node (reticulation node) can have different parents. This tree-network relation becomes relevant to identify trees obtained from a network and, in the other way, networks obtained from a set of trees.

A *switching* of a phylogenetic network is obtained by choosing, for each reticulation node in the network, an incoming arc to *switch on* and *switch off* all the others. Once this is done, we also recursively switch off all switched-on arcs whose target node has only switched-off outgoing arcs [Huson et al. (2010); Kelk and Scornavacca (2014)]. After applying all these switchings, if one removes the switched-off arcs one gets a tree. Notice also that some elementary nodes may appear and hence they must be contracted. If a tree can be obtained from this process from a network, we say that the tree is *displayed* by the network. See a complete example in Figure 1.4.

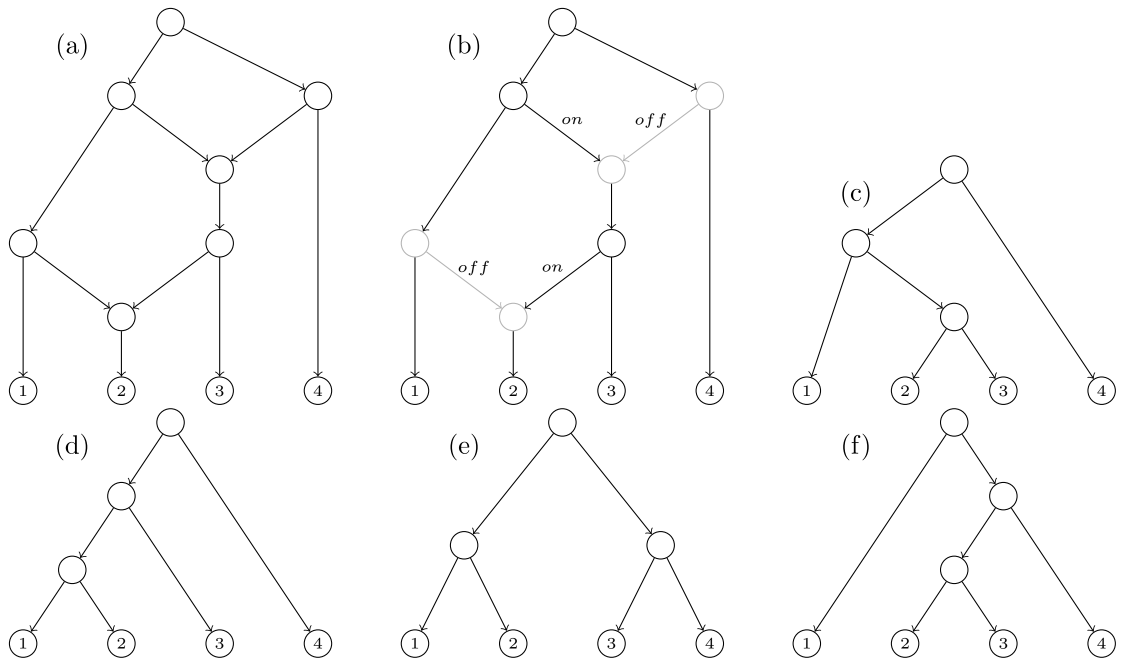

Figure 1.4: (a) A phylogenetic network N defined in $\{1, 2, 3, 4\}$. (b) A switching of N. (c) A tree displayed by N derived from the chosen switching. Note that there are 2 switched-off arcs and 4 nodes which become elementary and, consequently, they are suppressed. (d-f) The rest of trees displayed by N.

1.4 Newick notation for trees and networks

The *Newick* format [Felsenstein (2004)] is a way to represent the topology (and the edge lengths if needed) of a phylogenetic tree using plain text. This is one of the most commonly used formats in software specific to bioinformatics. Later, in Cardona et al. (2008b) it was extended to encode phylogenetic networks.

Newick notation for phylogenetic trees

We consider a rooted tree. To obtain the Newick string encoding the tree we proceed recursively in the following way. Each leaf is encoded by its label. Each internal node is encoded by a string starting with "(" followed by the list of encodings of each of its children (separated by commas) and then closing the parenthesis "`)`"; finally, if the internal node is labelled, this label is added as the last part of the string. The Newick string for the tree is the encoding associated to its root by the procedure above followed by the ending character `;`. For example, the Newick strings that represent the 4 phylogenetic trees in Figure 1.4 are:

- $((1,(2,3)),4);$ for the tree in (c),
- $(((1,2),3),4);$ for the tree in (d),
- $((1,2),(3,4));$ for the tree in (e),
- $(1,((2,3),4));$ for the tree in (f).

Newick notation for phylogenetic networks

The Newick format can be generalized to consider phylogenetic networks. The *extended* Newick (or *eNewick* for short) [Cardona et al. (2008b)] is one of such generalizations. The principal idea of this version relies on transforming the network into a rooted tree (possibly multilabelled and with internal nodes labelled in some very specific way) such that the original network can be reconstructed from it. The Newick string of the obtained tree (with some special rules to encode labelled internal nodes) is the eNewick string of the original network.

More precisely, to obtain the eNewick string which represents a phylogenetic network N we proceed as follows: let $\{H_1,\dots,H_m\}$ be the set of hybrid nodes of N ordered in any fixed way. For each hybrid node $H = H_i$, let $u_1,\dots,u_k$ and $v_1,\dots,v_l$ be its set of parents and children, respectively. Then H is splitted in k different nodes where the first copy has u_1 as its parent and $v_1,\dots,v_l$ as its children; and the other copies have (one for each) $u_2,\dots,u_k$ as their respective parent and have no children. Finally, each of the copies of H are labelled as [`label`]#[`type`]`tag`[: `branch length`] where:

- label (optional): string providing a labelling for the node.
- type (optional): string indicating which is the corresponding event that the node models: hybridization indicated by `H`, or lateral gene transfer indicated by `LGT`.
- tag (mandatory): integer i identifying the node $H = H_i$.
- branch length (optional): number indicating the length of the arc from the considered copy of H to its parent.

For example, the eNewick of the phylogenetic network depicted in Figure 1.4 (a) is

$$(((1,(2)\#H2),((\#H2,3))\#H1),(\#H1,4));$$

where $H1$ and $H2$ represents the above and the up and down reticulation nodes in the network, respectively.

1.5 Decomposition of trees and networks

In this section we describe how a “global object”, describing all the evolutionary relations among a set of species, can be decomposed into “local objects” describing relations among restricted subsets of species. Namely, we will focus on clusters and triples. Such objects only make sense in the setting of rooted trees but they have their unrooted counterparts, splits and quartets. We will also show how these concepts can be extended to networks, where one needs to consider different kinds of clusters and generalize triples to what we call *trinets*.

In the classical case of phylogenetic trees, clusters and triples are enough to encode trees; that is, one can recover the tree from this data. However, for phylogenetic networks, it is no longer true and one needs to impose restrictions on the networks in order to recover them from this “local data” [Cardona et al. (2008c, 2009b,c,e)].

Clusters

Given $T = (V, E)$ a phylogenetic tree labelled on a finite set S, and a node $u \in V$, the *cluster* of u, denoted by $C_T(u)$, is the set of labels of all the descendant leaves of u. The set of clusters *defined* by T, denoted by $\mathcal{C}(T)$, is the set of clusters of all its nodes: $\mathcal{C}(T) = \{C_T(u) : u \in V\}$ (see Figure 1.5). Notice that the clusters of nodes in T satisfies that:

- The cluster of a leaf is the singleton composed of its label.
- The cluster of the root is the set of labels of all leaves in the tree.
- The cluster of a node is the disjoint union of the clusters of its children.

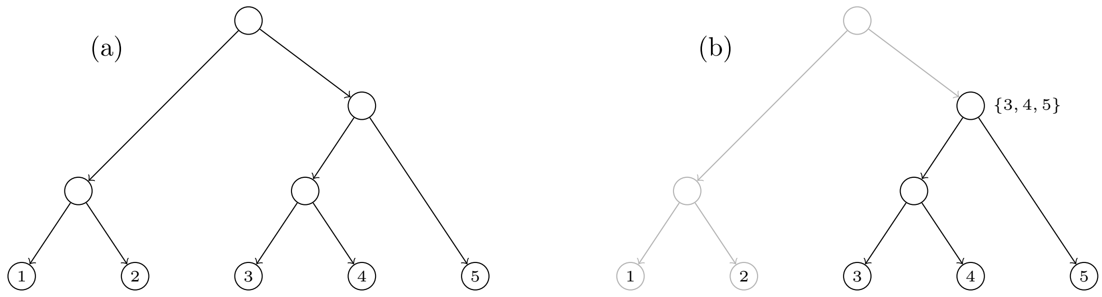

Figure 1.5: (a) A phylogenetic tree T such that $\mathcal{C}(T) = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{1,2\},\{3,4\},\{3,4,5\},\{1,2,3,4,5\}\}$. (b) Depiction of the cluster defined by a specific node in T.

We can also talk of clusters in phylogenetic networks. Notice that in this case there can be more than one path from an internal node to a leaf. This fact produces different kinds of clusters that can be considered. We say that a phylogenetic network N defines or displays a cluster C in *hardwired* sense if C is the set of descendants leaves of a node in N (which is the usual concept of cluster used in trees). On the other hand, we say that N defines or displays C in *softwired* sense if C is the cluster of a node in some tree displayed by N (see Section 1.3). Note that if N displays C in hardwired sense, then it is displayed also in softwired sense. Indeed, let u be a node with hardwired cluster C. A tree displayed by N and where the cluster of u is C can be constructed by considering the following switching. For each reticulation node h, if h is not a descendant of u, keep switched on any incoming arc; if h is a descendant of u, keep switched on any incoming arc (v, h) such that v is also a descendant of u. It follows easily that for each descendant leaf of u there exists a path formed by switched on arcs. Notice, however, that softwired clusters need not be hardwired clusters; see Figure 1.6.

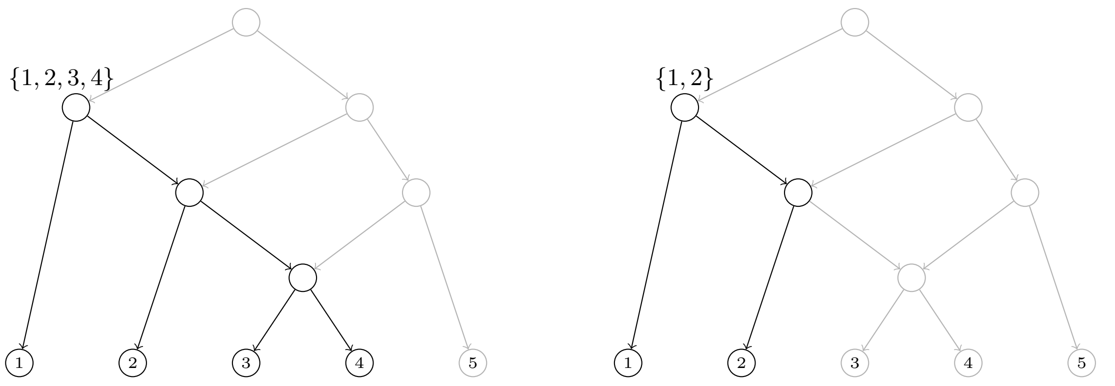

Figure 1.6: A phylogenetic network displays in hardwired sense the cluster $\{1, 2, 3, 4\}$ (left) and in softwired sense the cluster $\{1, 2\}$ (right). Note that, the cluster $\{1, 2\}$ is only displayed by N in softwired sense.

Triples

A *triple* on three different labels $x, y, z \in S$ is a rooted phylogenetic tree on $\{x, y, z\}$. Figure 1.7 depicts the only four possible (non-isomorphic) triples defined on x, y, z, together with their Newick notation (see Section 1.4). If we consider only binary trees, the triple (x, y, z) doesn't have to be taken into consideration. The other ones, $((x, y), z)$, $((y, z), x)$ and $((x, z), y)$ are commonly represented as $xy|z$, $yz|x$ and $xz|y$, respectively.

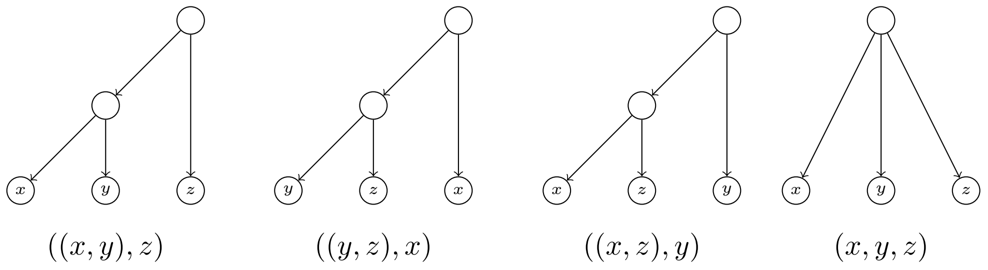

Figure 1.7: All possible triples defined on x, y, z.

We say that a phylogenetic tree T on S *defines* or *displays* a triple on $x, y, z \in S$ if there exists a subgraph of T homeomorphic to the given triple. The triple on $x, y, z \in S$ displayed by T is denoted by $T_{x,y,z}$. The set of triples displayed by T is denoted by $\Gamma(T)$; see Figure 1.8. That is $\Gamma(T) = \{T_{x,y,z} : \{x, y, z\} \subset S\}$.

Analogously as what we did in the case of clusters, we can also define the set of triples defined or displayed by a phylogenetic network. In this case, we need to generalize the concept of triple used on trees and consider *trinets*. This last concept will be analyzed later in Chapter 4.

A set of triples $\mathcal{T}$ defined on a set of taxa S is called *dense* if for each subset of three taxa in S there is at least one triple in $\mathcal{T}$ defined on these three taxa. This notion of dense set

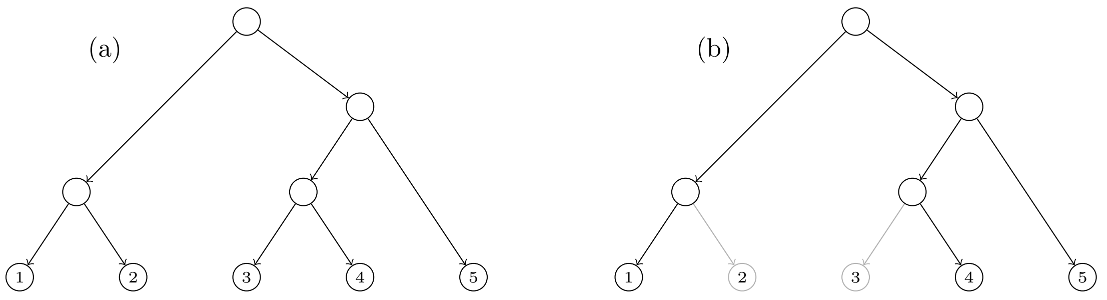

Figure 1.8: (a) Phylogenetic tree, we call T, such that its set of displayed triples are $\Gamma(T) = \{((1,2),3),((1,2),4),((1,2),5),(1,(3,4)),(1,(3,5)),(1,(4,5)),(2,(3,4)),(2,(3,5)),$ $(2,(4,5)),((3,4),5)\}$. (b) Depiction of how the triple $(1,(4,5))$ is displayed by T.

of triples will become significant in the reconstruction problem of phylogenetic networks (see Section 1.9).

Recall that two phylogenetic trees defined on S are isomorphic if, and only if, they have the same set of clusters, and also if, and only if, they define the same set of triples [Theorems 3.5.2 and 6.4.1 in Semple and Steel (2003)]. Then, the set of clusters and triples defines with unicity or *encodes* a phylogenetic tree. Actually, the descriptions of a phylogenetic tree T on S by means of $\mathcal{C}(T)$ and $\Gamma(T)$ are equivalent, through the following result [Lemma 9.1 in Dress et al. (2012)].

Theorem 1.1. *Let T be a phylogenetic tree T on S. For every non-empty subset $C \subset S$, $C \in \mathcal{C}(T)$ if, and only if, $((c,c'),x) \in \Gamma(T)$, for every $c, c' \in C$ and $x \in S \setminus C$.*

1.6 Classification of phylogenetic networks

There are a multitude of definitions for many kinds of phylogenetic networks. We can even find different names to design the same kind of phylogenetic network and, what is worse, different kinds of networks with the same name. The classification of phylogenetic networks may depend on a large amount of factors, as we shall shortly see. Then, it is (at the moment) quite impossible to find a general classification that can be used by all the scientific community. Based on Morrison (2011), Huson et al. (2010), Huson and Scornavacca (2011), we can find two kinds of classifications. The first one divides the whole set of phylogenetic networks into two big blocks:

- Rooted vs. unrooted networks. This classification is based on the underlying graph model that is used (directed vs. undirected) and, biologically, it corresponds to assuming or not that the least common ancestor of a set of species is known.
- Abstract or data-display networks vs. explicit or evolutionary networks. The abstract or data-display networks are those whose edges or arcs cannot be interpreted biologically and they are just used as a visualization tool of possible incompatibilities. On the other hand, the explicit or evolutionary networks describe an evolutionary scenario where the nodes and edges should represent species and evolutionary events between them.

Other possible classifications depend on the biological phenomenon under study (for instance, recombination or hybridization networks), the data used to build the networks (like split, cluster or consensus networks) or the topological constraints they must satisfy (such as galled trees, tree-child, tree-sibling or level-k networks). Next section is devoted to the study of this last group of restrictions.

1.7 Topological restrictions

The problem of reconstructing phylogenetic networks from biological data (or certain substructures like clusters or trees) commonly leads to NP-problems, and even NP-hard ones, when no restriction on the space of phylogenetic networks is imposed. In order to make the reconstruction process feasible, it is usual to introduce topological restrictions on the space of phylogenetic networks.

The close link between the mathematical structure, to be used in solving the problem, and the biological interpretation, that lies behind, appears here with more strength. That is, the main goal is to obtain a mathematical model that, on the one hand, can be analyzed mathematically and, on the other hand, it models accurately the real biological situation.

In the rest of this section, following Morrison (2013), we discuss some of the restrictions that usually appear in the literature related to phylogenetic trees.

Restrictions on degrees

Restrictions on the degrees of the nodes can be imposed either on their indegree, their outdegree, or both.

A typical condition related to degrees is forbidding the existence of elementary nodes, since they cannot be recovered.

Another condition is not allowing nodes to have indegree or outdegree greater than two. The biological motivation behind this assumption is that nodes with indegree or outdegree greater than two are only due to the lack of information, and having more knowledge would *resolve* that *polytomy.* We say that a phylogenetic network is *binary* if its tree nodes have indegree one and outdegree two and its reticulation nodes have indegree two and outdegree one. When this condition is only fulfilled for reticulation nodes, we say that it is *semi-binary.*

Time-consistent networks

Roughly speaking, a phylogenetic network is *time-consistent* if it can be drawn in such a way that arcs leading to a reticulation node are horizontal and arcs leading to tree nodes point downwards (see Figure 1.9(a)). That is, one can assign *times* to the nodes of the network in a way that strictly increases on arcs whose endpoint is a tree node (which should model speciation events, that take time to happen) and is constant on arcs whose endpoint is a reticulation node (which models interactions between coexistent species). Temporal consistency becomes a key factor in the reconciliation problem between gene trees and species trees, as we shall see in Chapter 5.

Formally, a phylogenetic network $N = (V, E)$ is *time-consistent* [Baroni et al. (2006)] if it admits a temporal assignment: a mapping $\tau : V \to \mathbb{N}$ such that:

- $\tau(r) = 0$, where r is the root of N.
- $\tau(u) = \tau(v)$ if $(u, v) \in E$ and v is a reticulation node.
- $\tau(u) < \tau(v)$ if $(u, v) \in E$ and v is a tree node.

Figure 1.9 depicts two phylogenetic networks. The network in (a) is time-consistent and the number close to each node indicates its assignment in a valid mapping. In contrast, the network in (b) is not time-consistent: it is impossible to define a temporal assignment because the two parents of the hybrid node are tree nodes connected by an arc.

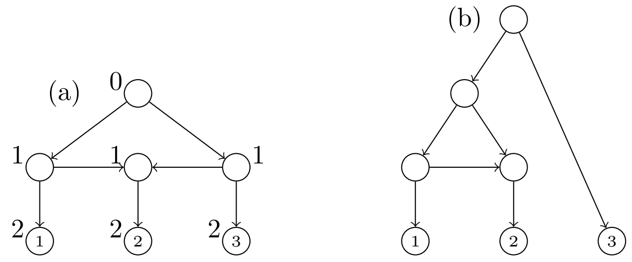

Figure 1.9: A time-consistent phylogenetic network (a) and a non time-consistent phylogenetic network (b).

While the definition above of time-consistency is the most used and well-known, Górecki (2004) uses a slightly different one. First, he considers what he calls *species graphs*, given by a tree together with a set of extra arcs that are added modelling lateral gene transfer. In this setting, the condition on the time assignment is only taken into account for nodes that are adjacent to these extra arcs. A similar definition will be used in Chapters 4 and 5 in our own model of networks.

Tree-child and tree-sibling networks

A phylogenetic network is *tree-child* when every non-extant species has some descendant through mutation [Cardona et al. (2009e)]. Mathematically, this means that each internal node has a child that is a tree node. Both phylogenetic networks depicted in Figure 1.9 are tree-child and the one depicted in Figure 1.10 is not tree-child because it has one node whose two children are hybrid nodes. Some problems, as the *Tree Containment problem* (deciding if a given phylogenetic tree is embedded in a given phylogenetic network), the *reconstruction problem from trinets* or the *reconstruction from path-lengths distances between taxa* [Bordewich and Semple (2015a)], among others, can be solved in polynomial time if the space of solutions is restricted to tree-child networks, while they are NP-hard for generic phylogenetic networks [Cardona et al. (2009e, 2010c)].

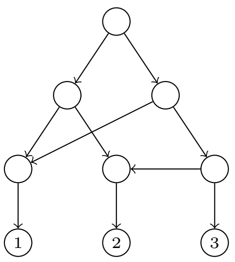

Figure 1.10: A non tree-child phylogenetic network.

A phylogenetic network is *tree-sibling* if at least one of the species involved in a reticulation event has some descendant through mutation [Nakhleh (2004); Cardona et al. (2008a)]. This is, each hybrid node has at least one sibling (a child of one of its parents) that is a tree node. Note that every tree-child network is, in particular, a tree-sibling network. Figure 1.11 shows a tree-sibling network that is, non tree-child (a), and a non tree-sibling network (b); notice that the parent of the leaf 3 is an hybrid node whose two sibling nodes are also hybrid.

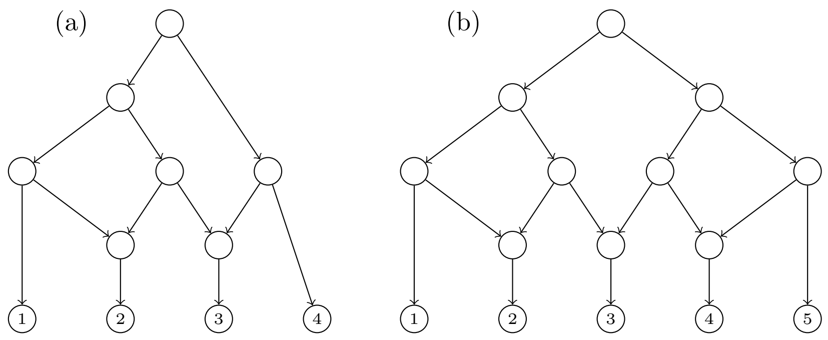

Figure 1.11: (a) A tree-sibling phylogenetic network, (b) a non tree-sibling phylogenetic network.

Stable networks

A node u in a phylogenetic network is *visible* if there exists at least one leaf l such that all paths from the root to l pass through u. In this case, the node u is said to be a *stable* ancestor of l. Then, a phylogenetic network is *reticulation-visible* [Van Iersel et al. (2010b)] or *stable* [Gambette et al. (2015a); Huber et al. (2015a)] if each reticulation node is visible. Using the definition of visible nodes, a tree-child network can be redefined as a network where every node (not only reticulation nodes) is visible. Figure 1.12 shows a reticulation-visible network (a), where each hybrid node is a stable ancestor of a leaf (node a is a stable ancestor of 1, node b is a stable ancestor of 2 and node c is a stable ancestor of 3), and a non reticulation-visible network (b) where one hybrid node, node b, is not visible.

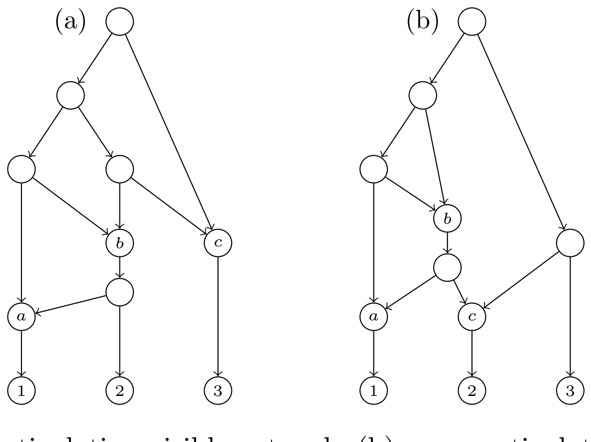

Figure 1.12: (a) A reticulation-visible network, (b) a non reticulation-visible network.

Thanks to the reticulation-visible networks, it is proved in Bordewich and Semple (2015b) that Tree Containment problem becomes tractable. Some other kinds of networks based on the concept of stability are *nearly-stable* networks [Gambette et al. (2015a)], *genetically-*

stable networks [Gambette et al. (2015b)] and *stable-child* networks [Gunawan and Zhang (2015)].

Galled trees and networks

A phylogenetic network is said to be a *galled tree* if each biconnected component has at most one reticulation node [Gusfield et al. (2003)]. It can also defined as one network such that each node belongs to at most one reticulation cycle –pair of paths with same origin and destination nodes and with disjoint intermediate nodes. From a biological point of view, this can be translated as saying that the reticulation events are independent. Galled trees have been extensively used in the reconstruction problem [Jansson and Sung (2006); Huber et al. (2011); Gambette et al. (2015c)].

Galled trees can be generalized in several ways. One of these generalizations, level-k networks, will be analyzed in the next section. Another such generalization is to consider *galled networks* [Huson and Klöpper (2007)], where the following restriction is introduced. For each reticulation node u and each pair of nodes u_i, u_j such that both (u_i, u) and (u_j, u) are arcs of the network, there is a non directed cycle containing u_i, u_j and such that the other nodes of the cycle are tree nodes, see Figure 1.13. This corresponds to a biological scenario in which reticulate events are quite rare [Huson et al. (2009)]. Among others, galled networks permit to solve the *Cluster Containment* problem (deciding if a given subset of leaves is a cluster of some phylogenetic tree embedded in the network) in polynomial time [Huson et al. (2009)], while it is an NP-problem for generic rooted networks [Kanj et al. (2008)].

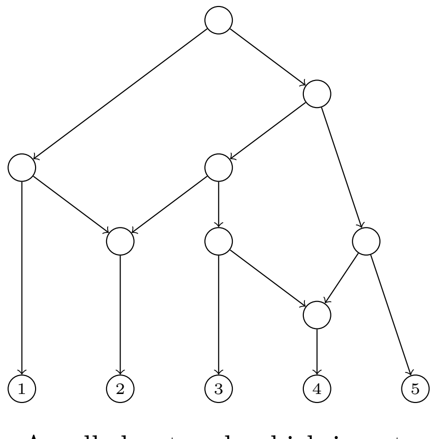

Figure 1.13: A galled network which is not a galled tree.

Level-k networks

The *level* of a network is, roughly speaking, a bound on the number of reticulations that can be mutually dependent [van Iersel and Kelk (2011)]. This is a measure to quantify the complexity of the network by means of biconnected components. More formally, a (binary) phylogenetic network is a *level-k* network if each biconnected component has at most k reticulation nodes [Choy et al. (2005); Jansson and Sung (2006)]. See Figure 1.14. Notice that galled trees are nothing but level-1 networks, and the biological meaning of level-k networks can also be understood as a generalization of those in the sense that the level describes up to which extent the reticulations are dependent. See Gambette et al. (2009), Gambette et al. (2012), Habib and To (2012), Van Iersel et al. (2009a), Van Iersel et al. (2009b) for some reconstruction problems which have can be efficiently solved using the level constraint.

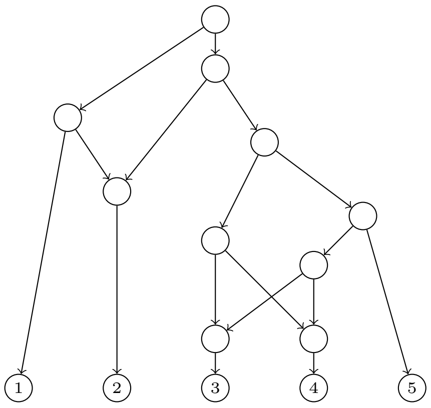

Figure 1.14: A level-2 phylogenetic network.

Tree-based networks

The notion of tree-based networks is important to differentiate networks which should properly be viewed as networks from those that should be viewed as "augmented" trees, i.e. trees with some extra arcs being added between their arcs. This distinction is important from the biological point of view in order to determine which is the evolutionary process under study, especially in some groups of organisms such as the prokaryotes [Dagan and Martin (2006); Doolittle and Bapteste (2007); Martin (2011)].

Although there are similar definitions for these networks, here we will use the one given in Francis and Steel (2015b), where a phylogenetic network is said to be *tree-based* if it can be obtained from a tree with same set of leaves by *inserting* extra arcs between the arcs of the tree. Here, to insert an extra arc between two given arcs, one splits each of these arcs into an elementary path of length two and then connects the introduced elementary nodes with an arc. See also Szöllősi et al. (2013). For example, all phylogenetic networks depicted in this section are tree-based networks. On the other hand, an example of a non-tree based network is presented in Figure 2(iii) of Francis and Steel (2015b) and reproduced in Figure 1.15.

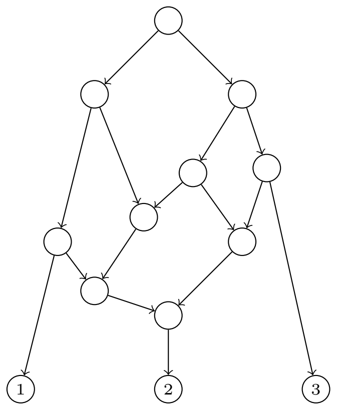

Figure 1.15: A non-tree based phylogenetic network.

Regular and normal networks

A phylogenetic network is *regular* if no two distinct nodes have the same cluster, and the cluster of a node u is contained in the cluster of another one v if, and only if, there is a path from the former to the latter v to u [Baroni et al. (2005); Baroni and Steel (2006)]. In other words, a network is regular if it isomorphic to the Hasse diagram of its set of clusters. A phylogenetic network is said to be *normal* if it is simultaneously tree-child and regular [Willson (2010)].

1.8 Metrics on phylogenetic networks

Metrics to measure the difference between networks play a key role in different tasks such as the detection of incongruences between gene trees and species trees, and the comparison of networks reconstructed from different sets of data or using different algorithms. Hence, these metrics are of great use in determining the degree of success of the reconstruction algorithms and of the recovery of the characteristics of the networks. Based on the idea of comparing a reconstructed network with the *true* evolutionary history, some publications refer to such distances or metrics as *error-distances* or *error-metrics*, see [Moret et al. (2002); Linder et al. (2003)].

Contrarily to phylogenetic trees, networks can present multiple evolutionary paths from internal nodes to leaves, making their comparison much more complex. Since there can be non isomorphic networks whose distance is zero, the distances designed to work on phylogenetic trees do not in general work on phylogenetic networks. Hence, one must either restrict the space of phylogenetic networks to consider or define new metrics.

Let us consider a class $\mathcal{C}$ of phylogenetic networks. A *metric* on $\mathcal{C}$ is a mapping $d : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ such that, for all $A, B, C \in \mathcal{C}$:

(a) Non-negativity: $d(A, B) \geq 0$,

(b) Separation: $d(A, B) = 0$ if and only if, $A \cong B$,

(c) Symmetry: $d(A, B) = d(B, A)$,

(d) Triangular inequality: $d(A, C) \leq d(A, B) + d(B, C)$.

The search of a sound metric on the set of *all* phylogenetic networks that can be computed efficiently seems to be an impossible task. In fact, the isomorphism problem for tree-sibling time-consistent networks (which is a quite restrictive case) is polynomially equivalent to the graph isomorphism problem, hence it is thought to be neither P nor NP-complete [Cardona et al. (2009a)].

Several metrics have been introduced so far in the literature. See [Warnow et al. (2003), Moret et al. (2004), Baroni et al. (2005), Cardona et al. (2008c), Cardona et al. (2008a), Cardona et al. (2009b), Cardona et al. (2009e), Cardona et al. (2009c)]. Next, we review some of them in more detail.

Clusters or Robinson-Foulds metric

The Robinson-Foulds (or clusters or bipartition) metric for phylogenetic (rooted) trees can be generalized in an straightforward way to phylogenetic networks. One associates, to each node of the network, its (hardwired) cluster and takes as distance between two networks the size of the symmetric difference between their respective sets of clusters. Although it does not define a proper metric (the separation axiom fails) on generic enough classes of phylogenetic networks like tree-child or tree-sibling time-consistent; however, in some restricted classes, like tree-child time-consistent networks, it defines a sound metric [Cardona et al. (2009b)].

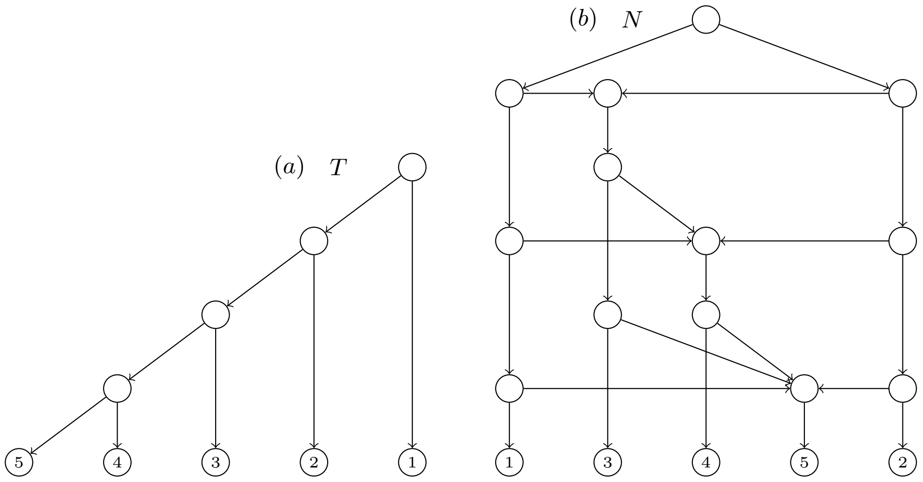

Figure 1.16: (a) A phylogenetic tree T, (b) a tree-child time-consistent network N. Reproduced from Figure 3 in Cardona et al. (2009b).

The cluster representations of the phylogenetic tree T and the phylogenetic network N depicted in Figure 1.16 are:

$$\begin{aligned}
&\mathcal{C}(T) = \big\{\{1\},\{2\},\{3\},\{4\},\{5\},\{4,5\},\{3,4,5\},\\
&\quad \{2,3,4,5\},\{1,2,3,4,5\}\big\}\\
&\mathcal{C}(N) = \big\{\{1\},\{2\},\{3\},\{4\},\{5\},\{1,5\},\{2,5\},\{3,5\},\\
&\quad \{4,5\},\{1,4,5\},\{2,4,5\},\{3,4,5\},\{1,3,4,5\},\\
&\quad \{2,3,4.5\},\{1,2,3,4,5\}\big\}.
\end{aligned}$$

Then, the Robinson-Foulds distance between T and N is 6.

Tripartition metric

The *tripartition* metric is introduced as a generalization of the bipartition metric in the case of *reconstructible networks* in Moret et al. (2004). This metric associates to each node of the network the tripartition of the set of leaves given by its *strict descendants*, its *non-strict descendants* and non-descendants of the considered node. Here, a leaf is a strict descendant of a node u if all paths from the root to the leaf pass through u, and non-strict descendant if the leaf is a descendant of u, but for which there exists some path from the root to it not containing the node u.

Although tripartitions add extra information to clusters up to our knowledge the tripartition distance is not a proper metric for those classes of networks where the Robinson-Foulds distance fails to separate networks [Cardona et al. (2008c)]. Then, for instance, the tripartition distance is not a metric neither on tree-child nor on tree-sibling time-consistent networks, and it is a metric on tree-child time-consistent networks [Cardona et al. (2009b)].

The tripartition representations of the phylogenetic tree T and the phylogenetic network N depicted in Figure 1.16 are:

$$\begin{aligned}
&\theta(T) = \big\{\{\{1\},\emptyset\},\{\{2\},\emptyset\},\{\{3\},\emptyset\},\{\{4\},\emptyset\},\{\{5\},\emptyset\},\\
&\quad\{\{4,5\},\emptyset\},\{\{3,4,5\},\emptyset\},\{\{2,3,4,5\},\emptyset\},\\
&\quad\{\{1,2,3,4,5\},\emptyset\}\big\}\\
&\theta(N) = \big\{\{\{1\},\emptyset\},\{\{2\},\emptyset\},\{\{3\},\emptyset\},\{\{4\},\emptyset\},\{\{5\},\emptyset\},\\
&\quad\{\{1\},\{5\}\},\{\{2\},\{5\}\},\{\{3\},\{5\}\},\{\{4\},\{5\}\},\\
&\quad\{\{1\},\{4,5\}\},\{\{2\},\{4,5\}\},\{\{3\},\{4,5\}\},\\
&\quad\{\{1\},\{3,4,5\}\},\{\{2\},\{3,4,5\}\},\\
&\quad\{\{1,2,3,4,5\},\emptyset\}\big\},
\end{aligned}$$

where for each node we only give the strict and non-strict descendants. Then, the tripartition distance between T and N is 12.

μ-distance

In the *μ-distance*, the data associated to each node generalizes its cluster by counting how many paths connect the node with each of the leaves of the network. This data is summarized in the so-called *μ-vector* of each node, and the distance between networks is defined as the size of the symmetric difference between the sets of μ-vectors of the networks under consideration. The μ-distance defines a metric on the space of tree-child networks and also on the space of semi-binary tree-sibling time-consistent networks [Cardona et al. (2008a, 2009e)].

The $\mu-$representations of the phylogenetic tree T and the phylogenetic network N depicted in Figure 1.16 are:

$$\begin{aligned}
&\mu(T) = \big\{(1,0,0,0,0),(0,1,0,0,0),(0,0,1,0,0),\\
&\quad(0,0,0,1,0),(0,0,0,0,1),(0,0,0,1,1),\\
&\quad(0,0,1,1,1),(0,1,1,1,1),(1,1,1,1,1)\big\}\\
&\mu(N) = \big\{(1,0,0,0,0),(0,1,0,0,0),(0,0,1,0,0),\\
&\quad(0,0,0,1,0),(0,0,0,0,1),(1,0,0,0,1),\\
&\quad(0,1,0,0,1),(0,0,1,0,1),(0,0,0,1,1),\\
&\quad(1,0,0,1,2),(0,1,0,1,2),(0,0,1,1,2),\\
&\quad(1,0,1,2,4),(0,1,1,2,4),(1,1,2,4,8)\big\}.
\end{aligned}$$

Then, the $\mu-$distance between T and N is 12.

Nakhleh's distance m

This distance, introduced in Nakhleh (2010b), is based on the *nested labelling* λ of nodes in a phylogenetic network, defined recursively as follows: If u is a leaf labelled by i, then $\lambda(u) = \{i\}$. If u is an internal node and its children $v_1, \dots, v_k$ have been labelled, then $\lambda(u) = \{\lambda(v_1), \dots, \lambda(v_k)\}$. As before, the comparison of two given networks is defined as the size of the symmetric difference of the sets of the nested labels of their respective nodes. This distance is a true metric on the class of reduced networks, tree-child networks

[Cardona et al. (2009d)] and semi-binary tree-sibling networks [Cardona et al. (2010a)]; however, it does not define a proper metric on tree-sibling time-consistent networks.

The nested label representations of the phylogenetic tree T and the phylogenetic network N depicted in Figure 1.16 are:

$$\begin{aligned}
&\lambda(T) = \big\{\{1\},\{2\},\{3\},\{4\},\{5\},\{\{4\},\{5\}\},\\
&\quad \{\{3\},\{\{4\},\{5\}\}\},\{\{2\},\{\{3\},\{\{4\},\{5\}\}\}\},\\
&\quad \{\{1\},\{\{2\},\{\{3\},\{\{4\},\{5\}\}\}\}\}\big\},\\
&\lambda(N) = \big\{\{1\},\{2\},\{3\},\{4\},\{5\},\{\{1\},\{5\}\},\\
&\quad \{\{2\},\{5\}\},\{\{3\},\{5\}\},\{\{4\},\{5\}\},\{\{5\}\},\\
&\quad \{\{\{4\},\{5\}\}\},\{\{\{1\},\{5\}\},\{\{\{4\},\{5\}\}\}\},\{\{\{3\},\{5\}\},\{\{\{4\},\{5\}\}\}\},\\
&\quad \{\{\{2\},\{5\}\},\{\{\{4\},\{5\}\}\}\},\{\{\{\{3\},\{5\}\},\{\{\{4\},\{5\}\}\}\}\},\\
&\quad \{\{\{\{1\},\{5\}\},\{\{\{4\},\{5\}\}\}\},\{\{\{\{3\},\{5\}\},\{\{\{4\},\{5\}\}\}\}\}\},\\
&\quad \{\{\{\{2\},\{5\}\},\{\{\{4\},\{5\}\}\}\},\{\{\{\{3\},\{5\}\},\{\{\{4\},\{5\}\}\}\}\}\},\\
&\quad \{\{\{\{\{1\},\{5\}\},\{\{\{4\},\{5\}\}\}\},\{\{\{\{3\},\{5\}\},\{\{\{4\},\{5\}\}\}\}\}\},\\
&\quad \{\{\{\{2\},\{5\}\},\{\{\{4\},\{5\}\}\}\},\{\{\{\{3\},\{5\}\},\{\{\{4\},\{5\}\}\}\}\}\}\big\}.
\end{aligned}$$

Then, the $m-$distance between T and N is 17.

Nodal distance

The nodal distance for (binary) trees is defined by comparing the matrices of (undirected) distances between pairs of leaves (from one leaf to the other through their least common ancestor) in each of the networks. It can also be generalized to arbitrary trees by considering the matrix of *splitted distances* between leaves, where the length of the path connecting two given leaves is split between the lengths of the ascending (from one leaf to the LCA) and descending (from the LCA to the other leaf) paths taking from one leaf to the other [Cardona et al. (2010b)]. This last concept of splitted distances allows for the comparison not only of phylogenetic trees but also phylogenetic networks and it is a metric on the class of tree-child time-consistent networks [Cardona et al. (2009c, 2010c)]. Otherwise, the nodal distance does not define a proper metric neither on tree-child nor on tree-sibling time-consistent networks [Cardona et al. (2009c)].

The matrices of distances between pairs of leaves of the phylogenetic tree T and the phylogenetic network N depicted in Figure 1.16 are:

$$\ell(T) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 2 & 0 & 1 & 1 & 1 \\ 3 & 2 & 0 & 1 & 1 \\ 4 & 3 & 2 & 0 & 1 \\ 4 & 3 & 2 & 1 & 0 \end{pmatrix} \qquad \ell(N) = \begin{pmatrix} 0 & 4 & 3 & 2 & 1 \\ 4 & 0 & 3 & 2 & 1 \\ 4 & 4 & 0 & 2 & 1 \\ 3 & 3 & 3 & 0 & 1 \\ 2 & 2 & 2 & 2 & 0 \end{pmatrix}$$

Then, the nodal distance between T and N is 21.

Triplets distance

The *triplet distance* appears as a generalization of the triplets distance used on phylogenetic trees, which compares the subtrees induced by each possible triple of leaves on each of the trees. To generalize this distance to phylogenetic networks one must first define what it means for “induced subnetwork” and find an efficient way to compute and compare them. In Cardona et al. (2009c) a generalization is given that works properly for tree-child time-consistent networks. The triplets distance is a metric on the class of tree-child time-consistent networks but does not define a proper metric neither on tree-child nor on tree-sibling time-consistent networks [Cardona et al. (2009c)].

The triplets representations of the phylogenetic tree T and the phylogenetic network N depicted in Figure 1.16 are:

$$\begin{aligned}\Gamma(T) = \big\{&([2,3])[1,2]{=}[1,3], ([2,4])[1,2]{=}[1,4],\\ &([2,5])[1,2]{=}[1,5], ([3,4])[1,3]{=}[1,4], ([3,5])[1,3]{=}[1,5],\\ &([4,5])[1,4]{=}[1,5], ([3,4])[2,3]{=}[2,4], ([3,5])[2,3]{=}[2,5],\\ &([4,5])[2,4]{=}[2,5], ([4,5])[3,4]{=}[3,5]\big\}\\ \Gamma(N) = \big\{&([1,3],[2,3])[1,2], ([1,4],[2,4])[1,2],\\ &([1,5],[2,5])[1,2], ([1,4],[3,4])[1,3], ([1,5],[3,5])[1,3],\\ &([1,5],[4,5])[1,4], ([2,4],[3,4])[2,3], ([2,5],[3,5])[2,3],\\ &([2,5],[4,5])[2,4], ([3,5],[4,5])[3,4]\big\}.\end{aligned}$$

Then, the triplets distance between T and N is 20.

1.9 Phylogenetic networks reconstruction

Phylogenetic reconstruction tries to infer the evolutionary history of a set of organisms from certain sets of data. In the last decades, the reconstruction problem for phylogenetic trees has been vastly studied and there exist a pretty good number of efficient algorithms for this problem, see De Bruyn et al. (2014) or Felsenstein (2004) for an overview. However, the reconstruction problem for phylogenetic networks is much more recent, and even though some algorithms have been proposed, there is no consensus on which is the best to use in each of the possible scenarios. This problem is more challenging for rooted phylogenetic networks when looking for a twofold purpose: obtaining efficient reconstruction algorithms suitable for real biological data and obtaining sound algorithms for phylogenetic networks as general as possible. Recently, methods based on the maximum likelihood, the pseudo-maximum likelihood and the maximum parsimony have been used to reconstruct phylogenetic networks.

We briefly review here the maximum likelihood criteria for phylogenetic networks reconstruction introduced in Jin et al. (2006). Let N be a phylogenetic network defined on S where the indegree of the reticulation nodes is two. For each reticulation node h_i, let e_l^i and e_r^i be the two incident arcs on h_i which we can interpret as the "left" and the "right" one. For each h_i, we associate a parameter $\gamma_i \in [0,1]$ which denotes the probability of choosing the arc e_l^i (the probability of choosing e_r^i is $(1-\gamma_i)$). That is, γ_i gives the proportion of sites of the sequence data at h_i inherited from each parent. Furthermore, let T be a tree displayed by N. We denote by φ_T the set of switched-on arcs of N which are incident on reticulation nodes and such that the resulting tree associated to this switching is T itself. As the set φ_T could be not unique, let Φ_T denote the set of all possible sets φ_T. Then, the probability of inducing a tree T by N is

$$P(T|N,\lambda^k) = \sum_{\varphi_T\in\Phi_T} \big[\prod_{e_l^i\in\varphi_T} \gamma_i \prod_{e_r^j\in\varphi_T} (1-\gamma_j)\big].$$

where λ^k is a leaf labelling of the set S of taxa and k is the number of sites of the sequences. Then, the *likelihood* of N with a labeling λ^k of S is

$$P(\lambda^k|N,M) = \sum_{T\in\mathcal{D}(N)} \big[P(T|N,\lambda^k)\cdot L(\lambda^k|T,M_T)\big],$$

where $\mathcal{D}(N)$ is the set of trees displayed by N, M is the evolutionary model of sequences with its respective branch lenghts and probabilities γ_i, M_T is the restriction of M to the tree T and $L(\lambda^k|T, M_T)$ is the overall likelihood of the set of aligned sequences. The *maximum likelihood* problem for phylogenetic networks can be defined so as to solve

$$(N^*, M^*) = argmax_{N,M} L(\lambda^k|N, M),$$

where N ranges over all network topologies defined on S, and M ranges over all combinations of branch lenghts, probabilities γ_i and models of sequence evolution.

We refer the reader to Jin et al. (2006); Nakhleh (2010a) for an extensive explanation of the reconstruction methods based on the maximum likelihood criterion, and to Yu and Nakhleh (2015) for the pseudo-maximum likelihood ones.

For our purposes, we focus on the maximum parsimony criterion, trying to infer a phylogenetic network that implies the least number of evolutionary changes, for instance a network with less number of reticulation nodes. Then, in this section, we present different algorithms and sets of data that are used in the reconstruction of rooted phylogenetic networks using methods based on maximum parsimony.

Following Gambette and Huber (2012), a crucial point is to determine if the available data (sets of clusters, distances, trees and triples) enables us to reconstruct the original network, as in the case of trees. Therefore, it will be often necessary to impose some restrictions on the output networks, yet only because some of the intermediate problems to solve, such as the Cluster Containment and the Tree Containment become NP-complete problems for generic networks [Kanj et al. (2008)].

Reconstruction from clusters

Cluster networks were introduced by Huson and Rupp (2008). The authors describe how a phylogenetic network can be reconstructed from a set of clusters, in hardwired sense (see Section 1.5), which are obtained from a family of phylogenetic trees, such as gene trees.

Considering clusters in the softwired sense, see Section 1.5 or Huson et al. (2009), managed to reconstruct specific galled networks. For this type of clusters, one of the most extensively used algorithm for the reconstruction problem is CASS [Van Iersel et al. (2010a)]. This algorithm allows us to combine any set of clusters in a phylogenetic network. Also, CASS usually constructs a network with a significantly lower level and lower number of reticulations compared with other algorithms; therefore, the resulting networks are simpler than those obtained using other algorithms. An improvement of the execution time for the CASS algorithm is proposed in the BIMLR implementation [Wang et al. (2013)].

Hence, finding a network displaying a set of clusters is feasible. Nevertheless, finding the one that minimizes the number of reticulations or the level, becomes a much more complex quest [Van Iersel and Kelk (2011)]. A proof that the reconstruction problem from clusters in the softwired sense is a fixed-parameter tractable problem can be found in Kelk and Scornavacca (2014).

Reconstruction from distances

Contrarily to phylogenetic trees, for each pair of leaves, there can be many "lowest" common ancestors, and for each of them, there can be many different paths to the leaves.

The reconstruction from distances is typically based on computing the average of these distances and using this data as input.

Some of the methods specifically designed to construct phylogenetic networks from distances are the T-Rex algorithm [Makarenkov (2001)], and the NeighborNet algorithm [Bryant and Moulton (2004)]. The first one reconstructs a phylogenetic network by starting with a phylogenetic tree, which provides the initial fit for the given distance matrix. Then, the algorithm adds new branches into the growing reticulogram. The NeighborNet algorithm is an agglomerative method where taxa are combined into progressively larger and larger overlapping clusters in a similar way to the Neighbor Joining (NJ) algorithm used for trees [Saitou and Nei (1987)].

In another attempt to solve the problem, Willson (2013) proposed an algorithm that requires some conditions on the distances to be used as input in order to reconstruct a network. Some improvements of this algorithm can be found in Francis and Steel (2015a) and Bordewich and Semple (2015a), where the authors do not impose those restrictions on the distances but restrict to binary tree-child networks.

Reconstruction from trees

In the case of reconstruction from trees, the input data is a set of trees and the goal is to find a phylogenetic network that displays (see Section 1.3) each of them. A further requirement for the output network is that it has the minimum possible number of reticulation nodes (parsimony principle). In general it is an NP-hard problem [Bordewich and Semple (2007)], but there are algorithms that work reasonably well when the number of reticulation nodes is small, like PIRN$_C$ [Wu (2013)], HybrydInterleave [Collins et al. (2011)], HybridNet [Chen and Wang (2010)], as well as improvements of them in van Iersel et al. (2014) and Albrecht (2015). The problem can also be solved efficiently when restricted to *hybridization networks* [Semple (2006)] or when only two phylogenetic trees are considered [Bordewich et al. (2007)].

Reconstruction from triples and trinets

Another possibility for the set of input data to be used for reconstructing phylogenetic networks are triplets. As it happens in the case of the reconstruction from clusters or trees, the problem of finding a phylogenetic network that represents all the given triplets (or as many ones as possible) using the least possible reticulation notes is hard [Jansson et al. (2006); Van Iersel et al. (2009b)]. Hence some restrictions on the input data (having a dense set of triples, for instance) or the output network (level-k, for instance) are considered in order to get efficient algorithms and implementations, like LEV1ATHAN [Huber et al. (2011)], SIMPLISTIC [van Iersel and Kelk (2011)] and [To and Habib (2009)].

Further details about the reconstruction from trinets will be given in Chapter 4.

1.10 Reconciliation between gene trees and species trees

In this section we consider gene trees and species trees. We will see the differences between these concepts and how the inconsistencies between them leads to the reconciliation problem. For a detailed view on this topic, we refer the reader to Maddison (1997), Nakhleh et al. (2009), Nakhleh (2013) and Doyon et al. (2011). A more detailed treatment of the reconciliation problem between gene trees and species trees (or networks) will be presented in Chapter 5.

Gene trees vs. species trees

A *species tree* or *phylogeny* is a way of representing a hypothesis on the evolutionary relationships or evolutionary history for a group of species over time. In this representation all species share a common ancestor, represented by the root of the tree, and the internal nodes represent speciation events. Hence, it is a representation of the shared and separate history of any pair of species.

On the other hand, a *gene tree* is a representation of the evolutionary history of a family of genes (or even a single one), considering evolutionary events that simultaneously concern the whole set of genes (like speciation events), as well as other events for specific genes (such as duplications, transfers and losses). In a gene tree, the leaves represent the contemporary sequences of the genes under consideration and the internal nodes represent ancestral sequences.

The reconciliation problem

The *reconciliation problem* aims to describe how a species tree and a set of gene trees can be made compatible, that is, how biological processes that affect the genes (duplications, losses, transfers) reconcile with events described in terms of species. Mathematically, such a reconciliation is a mapping between nodes in a gene tree to nodes or arcs in a species tree subject to certain restrictions. These restrictions, together with the costs associated to discrepancies between mapped nodes, are described in terms of *reconciliation models*.

There exist different reconciliation models depending on the biological processes to be taken into account. For example, some partial models only consider duplication ($\mathbb{D}$) and loss events ($\mathbb{L}$), and are called $\mathbb{DL}$ models [Goodman et al. (1979); Page (1994); Doyon et al. (2009)]. Some other models consider lateral transfer events ($\mathbb{T}$) and losses [Abby et al. (2010); Beiko and Hamilton (2006), Nakhleh et al. (2005)]. Finally, the more generic ones consider the three evolutionary events, duplications, transfers and losses, and are known as $\mathbb{DTL}$ models [Tofigh et al. (2011); Scornavacca et al. (2013); Libeskind-Hadas et al. (2014)].

Once the reconciliation model is fixed, together with the associated costs for discrepancies, some method has to be used to solve the problem. There are mainly two different approaches to solving this problem: parsimony and likelihood. Parsimony methods are usually faster but less realistic, since they are based on a discretization process (the cost for each event is knowed in advance and time in the species tree is discretized), while a solution based on likelihood approaches involves the solution of optimization problems on continuous probabilistic models.

In this manuscript we focus on parsimony methods to solve the reconciliation problem. These methods attempt to obtain the optimal (in terms of the cost of the evolutionary events) reconciliation scenario between a gene tree and a species trees to explain the incongruences between them. More formally, the *Most Parsimonious Reconciliation problem*, MPR problem for short, consists on: given a species tree, a gene tree and respective costs for duplication, transfer and loss events, compute a reconciliation that has a minimum cost.

The $\mathbb{DL}$ and the $\mathbb{DTL}$ models

The $\mathbb{DL}$ model assumes that the only evolutionary events of genes are, apart from speciation, duplications and losses. A parsimonious solution (with respect to the number of duplications and losses) to the MPR problem using the $\mathbb{DL}$ model can be easily computed by means of the *least common ancestor mapping*, LCA mapping for short. Given G a gene tree and S a species tree, the *LCA* maps each leaf of G into a leaf of S with the same species name and each internal node u of G into the most recent species x of S such that each descendant gene of u in G resides in a descendant species of x in S. See Figures 1.17 and 1.18. For instance, in Figure 1.17, the node t of G is mapped into the node z in S because z is the most recent species of S such that both a and b (the descendants genes of t) reside in the descendant species A and B of z.

We shall go into further details on the *LCA reconciliation* in Section 5.4.

In the $\mathbb{DTL}$ model, apart from duplications and losses one also allows lateral gene transfers. This change in the model makes the problem much more difficult. In order to obtain computationally tractable scenarios, one must impose restrictions in terms of time-consistency. Furthermore, unlike the $\mathbb{DL}$ model, the LCA mapping does not always give a solution for the MPR problem. In Figure 1.19, the depicted reconciliation scenario is more parsimonious (considering an unitary cost for each kind of event) than the one that respects the LCA mapping. For instance, the nodes r, v and t are located below its LCA mapping (nodes x, x and z of S, respectively).

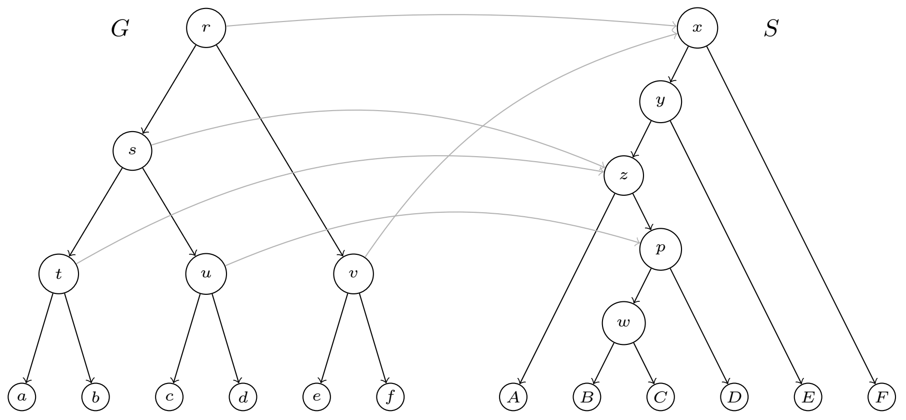

Figure 1.17: LCA mapping between a gene tree G and a species tree S.

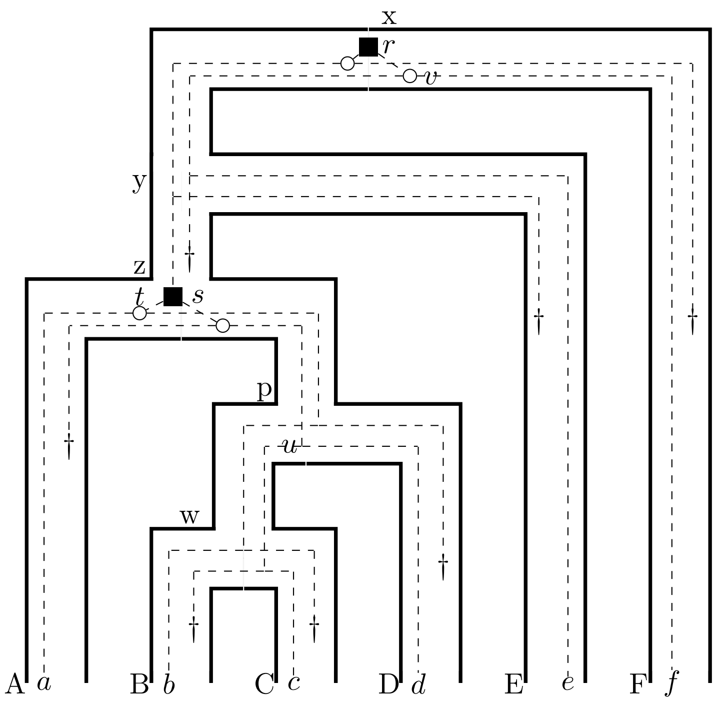

Figure 1.18: A $\mathbb{DL}$ reconciliation between the gene tree G and the species tree S depicted in Figure 1.17, according to the LCA mapping between them. The dashed lines represent the gene evolution. The $\mathbb{D}$ and $\mathbb{L}$ events are represented as filled square and cross dagger, respectively.

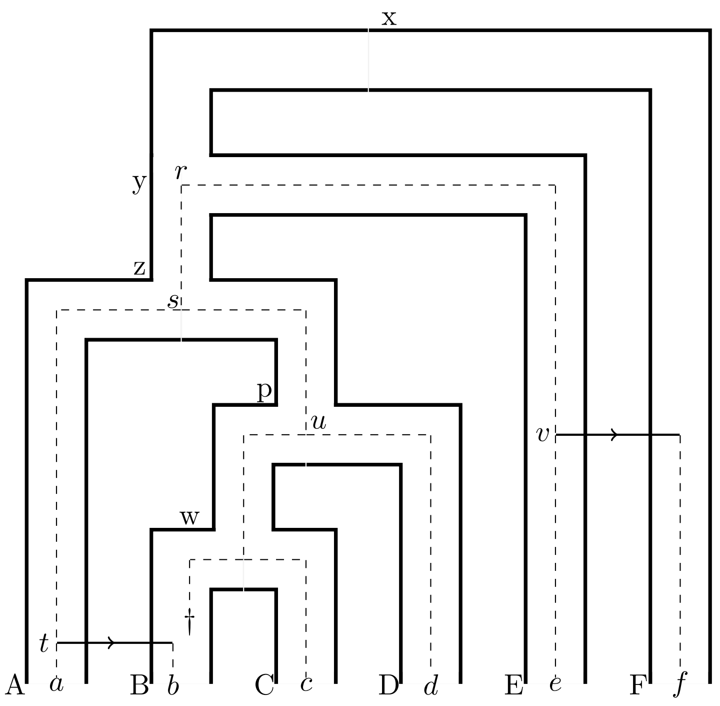

Figure 1.19: A $\mathbb{DTL}$ reconciliation between the gene tree G and the species tree S depicted in Figure 1.17. The dashed lines represent the gene evolution. The $\mathbb{T}$ and $\mathbb{L}$ events (notice that no duplication takes place) are represented by arrows and cross daggers, respectively.

Chapter 2

LGT networks

Contents

2.1 Introduction

As we have already seen in the introduction, in order to model many relevant evolutionary processes that cannot be represented by a tree [Martin (2011); Doolittle and Bapteste (2007)], more general models of (evolutionary) phylogenetic networks have been progressively introduced [Huson et al. (2010); Morrison (2011)]. One specific type of non tree-like events are Lateral, or Horizontal, Gene Transfers: transfers of genetic material from one species to a different and, usually, taxonomically distant one [Boto (2010)] which are frequent and important not only among unicellular species [McDaniel et al. (2010)] but also, for instance, among plants [Yue et al. (2012)] or from parasites to hosts [Gilbert et al. (2010)]. Many papers are devoted to modelling or inferring reticulation events in phylogenetic networks, such as lateral gene transfer events from, for example, incongruent gene trees [see Tofigh et al. (2011); Bansal et al. (2013); Abby et al. (2010), among others].

Although lateral gene transfers are commonly modeled as arcs added to a tree, and hence the resulting phylogenetic networks are tree-based in the sense of Francis and Steel (2015b), in most cases the mathematical model under consideration makes no reference to the base tree and all parents of a node are treated symmetrically. This is not accurate, because in lateral gene transfers, the resulting species acquires its DNA mostly from one, and only one, of its parents, which should be understood as its "principal" parent, in contrast to the other parents which contribute in a much lesser way and should be considered as "secondary" parents. This asymmetry is usually emphasized in graphical representations of phylogenetic networks with lateral gene transfers, like for instance those depicted in Figure 3 of Benveniste and Todaro (1974) (which, according to Morrison (2014) are the first published in the literature), but again seldom in the mathematical model. Actually, and up to our knowledge, the only types of phylogenetic networks that explicitly distinguish between the primary, tree-like, line of evolution and the secondary, due to lateral gene transfers, that have been studied in the literature are those in Francis and Steel (2015b) and those in Górecki (2004).

In this chapter we introduce a general model of phylogenetic network with lateral gene transfers similar to the species graphs' approach introduced in Górecki (2004): *LGT networks*, which consist roughly of a principal rooted tree with its leaves labelled on a set of taxa (and possibly with elementary, that is, out-degree 1, nodes) and a set of secondary arcs between nodes in this tree, representing lateral gene transfers, such that the resulting directed graph turns out to be rooted, acyclic, with its leaves labelled and its internal nodes unlabelled. Any such LGT network gives rise to a principal phylogenetic subtree (by suppressing out-degree 1 nodes in the principal subtree) which can be interpreted as representing the primary line of evolution.

Furthermore, in this chapter we analyze some previous models of phylogenetic networks mainly focusing on which lateral transfer events are represented. Finally, we introduce a metric distance on the space of all LGT networks.

2.2 LGT networks

In a DAG, we say that a node u is an *elementary* node if $\text{outdeg}(u) = 1$. Notice that this definition is slightly different from the commonly used elementary node definition which impose that both indegree and outdegree of the node are exactly one. Given S a finite non-empty set, we consider that a *phylogenetic network* on the set of labels S is an $S-$rDAG without elementary nodes. Notice, in particular, that we forbid in our phylogenetic networks the existence of reticulations with out-degree 1. The reason to this decision will be explained later. When talking about trees, we distinguish between *S-trees* and *phylogenetic trees*. Both of them are S-rDAGs without reticulation nodes, but S-trees can have elementary nodes, while phylogenetic trees cannot have any such node. Note that every S-tree gives rise to a phylogenetic tree on S by suppressing all its elementary nodes. To suppress an elementary node u in a tree T, we perform the following operation: if u is the root, we remove it together with its incident arc; if, otherwise, u has parents $w_1, \ldots, w_k$ and child v, we remove u together with the arcs incident to it, and we replace them by the arcs $(w_1, v), \ldots, (w_k, v)$.

In Górecki (2004, 2010), and Górecki and Tiuryn (2012), the authors define a species graph on a set of labels S as an S-tree endowed with a set of extra arcs, representing lateral gene transfers, that satisfies a set of restrictions motivated by their use in the representation of common evolutionary histories of species and genes (see Section 2.3). Here we introduce phylogenetic networks with lateral gene transfers more general than species graphs (we compare both models in Section 2.3), by imposing only that the graph obtained by adding arcs to the tree is a phylogenetic network.

Definition 2.1. *An* LGT network *on a set S is a phylogenetic network $N = (V, E)$ on S together with a partition $E = E_p \sqcup E_s$ of its set of arcs such that $T_0(N) = (V, E_p)$ is an S-tree. The arcs in E_p are called* principal*, and those in E_s,* secondary*. We shall call $T_0(N)$ the* principal subtree *of N.*

Figure 2.1 depicts an LGT network and Figure 2.2 depicts another LGT network with its principal subtree $T_0(N)$. Henceforth, in graphical representations of LGT networks, we shall use the following conventions: principal arcs are represented by continuous arrows and secondary arcs by dashed arrows.

Let N be an LGT network. Since $T_0(N) = (V, E_p)$ is an S-tree, every arc in N ending in a tree node is principal and the set of arcs ending in each reticulation h contains exactly one principal arc: we call its origin the *principal parent* of h, and its other parents, *secondary parents*. To ease the notations, we shall also say that the single parent of a tree node is its *principal parent*. We also split the children of every node v into *principal* and *secondary*, depending on the type of the arcs going from v to them. For instance, in Figure 2.2, the node a is the principal parent of h, and the nodes c and d are its secondary parents; also, the leaf 4 is the principal child of c and the nodes h and k are its secondary children.

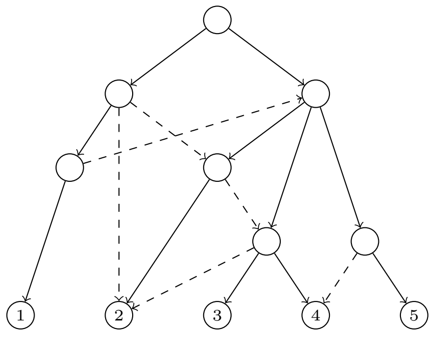

Figure 2.1: An LGT network defined on $\{1, 2, 3, 4, 5\}$.

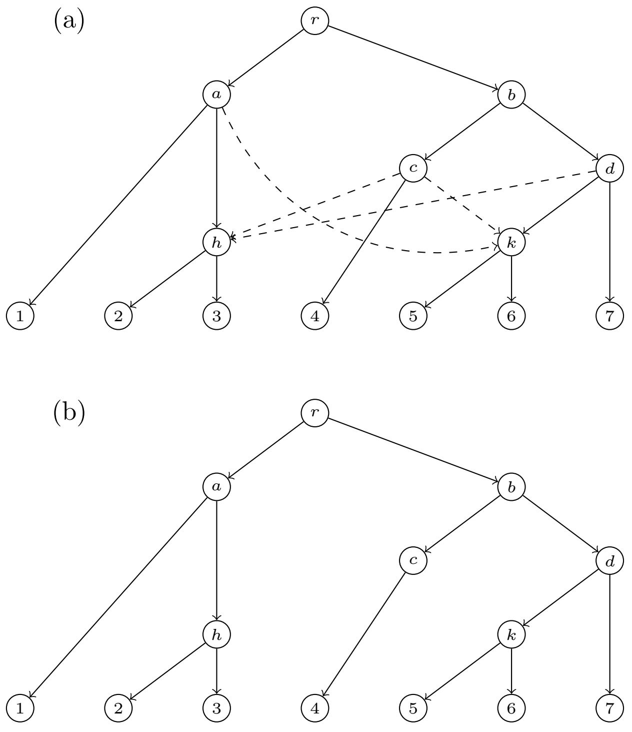

Figure 2.2: An LGT network (a) and its principal subtree (b).

Biologically speaking, the nodes of an LGT network represent species. The principal subtree represents the main line of evolution of these species; that is, the genetic material of a species comes mainly from its principal parent, possibly including mutations, while its secondary parents have introduced some genes in the species through lateral gene transfers. In this way, a secondary arc models one or more lateral gene transfer from its source to its target.

As we have already mentioned, our model is different from other commonly used models in the literature [Huson et al. (2010); Baroni et al. (2005, 2006); Moret et al. (2004)], where it is imposed that reticulations have outdegree 1, while now we impose that outdegree > 1. The reason to do that is twofold. First, for other authors, reticulation nodes modeled the hybridization or recombination process and its single child the resulting species. In our model we collapse these two nodes into a single node representing both the process and the resulting species. Hence, in our model all nodes with outdegree one must be understood as elementary, while in the previous model only tree nodes with outdegree one should be considered elementary. In either case, elementary nodes must be forbidden if one wishes to solve the reconstruction problem, since those node never can be reconstructed. Second, some of the technical results used in Chapter 3 are more easily stated using the model we have chosen.

According to this difference in the interpretation of reticulation nodes, LGT networks can be defined in two different ways, depending on whether you impose or forbid reticulations with outdegree 1. These two models are mathematically equivalent, simply by collapsing the arcs that begin in a reticulation node (resp. expanding them). This fact makes us think that using one model or the other would not strengthen (or weaken) the forthcoming results. The first model (reticulations with outdegree > 1) is used in Chapters 3 and 4; and the second one (reticulations with outdegree 1) is used in Chapter 5. In Figure 2.3 there is depicted the equivalent LGT network imposing reticulation nodes with outdegree 1 to the LGT network depicted in Figure 2.2(a).

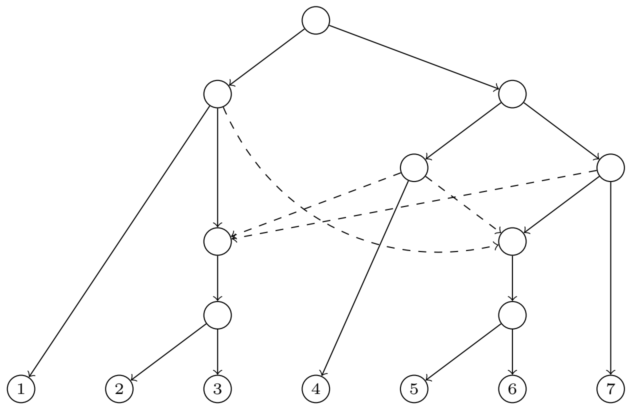

Figure 2.3: LGT network equivalent to the one depicted in Figure 2.2.

Notice that, using some other notations that appear in the literature, we have that the principal subtree $T_0(N)$ is a *switching* of the LGT network N (see Kelk and Scornavacca (2014) or Section 1.3); this is, take, for each reticulation node, the single principal arc reaching it as switched-on, and all the secondary as switched-off. Then, in particular, $T_0(N)$ is also (up to homeomorphism) a tree *displayed* by N (see Huson et al. (2010) or Section 1.3). Moreover, N is also *tree-based*, where $T_0(N)$ is (up to homeomorphism) the distinguished base tree (see Francis and Steel (2015b) or Section 1.7).

An *isomorphism* of LGT networks is an isomorphism of S-rDAG (see Section 1.1) that preserves and reflects the partitions of the sets of arcs into principal and secondary. More formally, given two LGT networks $N = (V, E)$ and $N' = (V', E')$, an isomorphism from N to N' is a bijection $\phi : V \to V'$ such that:

- (u, v) is a principal arc in N if, and only if, $(\phi(u), \phi(v))$ is a principal arc in N';
- (u, v) is a secondary arc in N if, and only if, $(\phi(u), \phi(v))$ is a secondary arc in N';
- $u \in V$ is a leaf labelled with $s \in S$ if, and only if, $\phi(u)$ is a leaf labelled with s.

The isomorphism of LGT networks can be easily checked in linear time in their sizes. Indeed, two LGT networks N and N' are isomorphic (or, *equal*) if, and only if, $T_0(N) = T_0(N')$ –which can be checked in linear time in the number of principal arcs of the networks– and this isomorphism preserves and reflects the sets of secondary arcs.

2.3 LGT networks generalize species graphs

A *species graph* [Górecki (2004, 2010); Górecki and Tiuryn (2012)] is defined as a pair (T, H) where $T = (V, E)$ is an S-tree whose internal nodes have outdegree ≤ 2 and $H \subseteq V \times V$ is a set of arcs satisfying the following conditions:

(HR1) If $(u, h) \in H$, then u and h are not connected by a path in T.

(HR2) If $(u, h) \in H$, then u and h are elementary in T, and every elementary node in T is incident to some arc in H.

(HR3) If $(u, h), (u', h') \in H$ and $(u, h) \neq (u', h')$, then $\{u, h\} \cap \{u', h'\} = \emptyset$.

(HR4) There exists a *time stamp* function $\delta : V \to \mathbb{N}$ that strictly increases from source to target in arcs $e \in E$, that remains constant from source to target in arcs $e \in H$, and that takes different values on nodes incident to different arcs in H.

The tree $T = (V, E)$ and the arcs in H in a species graph are the equivalent to principal subtree and the secondary arcs in an LGT network, respectively. Note that it is easy to check that any species graph $((V, E), H)$ defines an LGT network with set of nodes V, set of principal arcs E and set of secondary arcs H. In our model, we consider phylogenetic networks with lateral gene transfers more general than species graphs, by imposing only that $(V, E \sqcup H)$ is a phylogenetic network.

Comparing species graphs to our LGT network model, there exist some similarities and some differences. Even though we do not impose any restrictions similar to HR1 and HR4 in our general LGT model, we will use some conditions analogous to them in a latter stage, when working in certain subclasses of LGT networks. More precisely, in Chapter 3 we will revisit condition HR1, and in Chapters 4 and 5 we will find conditions similar to HR4. Condition HR2 states that, in a species graph, nodes with both indegree and outdegree equal to 1 are not allowed. This is also the case for LGT networks where any such node is the extreme of an arc in H.

In order to keep the binarity in a species graph, the starting node of each "secondary" arc (an arc in H) is created by splitting an arc of T. These restrictions are more relaxed in the LGT model since the starting node of a secondary arc can also be a node of the principal subtree. Finally, restriction HR3 is also relaxed in our model, since we do allow that two different secondary arcs have the same starting or destination nodes, or even that the starting node of one of them is the destination node of the other (see Figure 2.1).

2.4 Other models for LGT events

In this section we review some other approaches to the modelling of lateral gene transfer events.

We first outline the *lateral transfer scheme* proposed in Hallett and Lagergren (2001). Such model simplifies the evolutionary process by allowing lateral gene transfers, whose main goal is to achieve the reconciliation scenario between a gene tree and a species tree (see Section 1.10) via lateral gene transfers. A lateral transfer scheme for a species tree S is defined as a pair (S', A') where S' is a subdivision of S (that is, S' is the tree S with some of its arcs split by an intermediate node) and a set of arcs $A' \subseteq \{(x, y) : x, y \in V(S') \setminus V(S), x \neq y\}$ such that:

- The mixed graph $S' \cup E(A')$, where $E(A')$ denotes the underlying undirected edges of A', does not contain a directed mixed cycle. That is, a cycle that may contain arcs as well as undirected edges, and where the cycle can be traversed in a direction that respects the arcs but edges may be traversed in either direction.
- The source of each arc in A' has in-degree 1 and out-degree 2 in $S' \cup A'$,
- The target of each arc in A' has in-degree 2 and out-degree 1 in $S' \cup A'$.

A lateral transfer scheme shows where the lateral transfers have occurred during evolution. The arcs in A' represent the set of lateral transfers. The partial order of evolution induced by the species tree S is respected in the scheme since directed mixed cycles are not allowed. A lateral gene transfer is an event that causes some portion of the evolution represented by an arc in the gene tree to occur along one arc in the species tree, and the remaining portion of evolution to occur along another arc of the species tree. A lateral transfer scheme is meaningful when combined to the notion of scenario (see Section 1.10). A scenario where a transfer occurs in one direction between two genomes is not necessarily equivalent to a scenario where a transfer that occurs in the opposite one. Then, in this sense, the transfers are directed. However, in the definition of lateral transfer scheme, due to the first condition, this transfer direction is disabled. Then, the intuition is that a transfer can be traversed in either direction, and it represents a "moment in evolution" where the two ancestral genomes co-existed [Addario-Berry et al. (2003)].

Similarly to the previous model, we find in Bansal et al. (2013) a method in which non directed horizontal edges are defined in order to detect *highways*, which compile a great number of lateral gene transfers taking place between pairs of species in either direction. That method of highway detection takes as input a species tree and a set of gene trees, and computes, for each gene tree, the lateral gene transfer events affecting that gene on the species tree. The lateral transfer events that are inferred for a significant fraction of the gene trees are postulated as the highways. Therefore, given a rooted tree T, a *horizontal edge* on T is a pair of nodes $\{u, v\}$, such that u and v are different nodes and none of them is descendant of the other. Horizontal edges represent potential lateral gene transfers events. A directed arc (u, v) represents a lateral gene transfer from the species represented by the arc with destination node u to the species represented by the arc with destination node v. Thus, the undirected horizontal edge $\{u, v\}$ represents the lateral gene transfer events (u, v) and (v, u).

Both models were created to be used in a reconciliation framework and not also as a evolutionary phylogenetic networks by itself, as is our model of LGT networks. Moreover, in both cases the underlying trees that appear are binary trees and the arcs used to model the lateral transfer events are undirected against our model which allows outdegree greater than two for internal nodes and where the direction of the transfer events is specified.

2.5 An extension of Robinson-Foulds metric for LGT networks

In this section we present a distance metric on the space of LGT networks over a given set of taxa that generalizes, in some sense, Robinson-Foulds' metric for phylogenetic trees and that can be easily computed in polynomial time on the number of leaves.

Let $N = (V, E)$ be an LGT network and $T_0(N) = (V, E_p)$ its principal subtree. Since it can have elementary nodes, each node u is singled out in $T_0(N)$ by means of the pair $\ell_{T_0(N)}(u) = (C_{T_0(N)}(u), \delta_{T_0(N)}(u))$, where $C_{T_0(N)}(u)$ is the cluster of u in $T_0(N)$ and $\delta_{T_0(N)}(u)$ stands for the length, that is the number of arcs, of the longest path in $T_0(N)$ from u to a node with its same cluster as a node in $T_0(N)$. Let

$$\Upsilon_p(N) = \{\ell_{T_0(N)}(u) : u \in V\}.$$

On the other hand, let

$$\Upsilon_s(N) = \big\{(\ell_{T_0(N)}(u), \ell_{T_0(N)}(v)) : (u, v) \in E_s\big\}.$$

And finally, let

$$\Upsilon(N) = (\Upsilon_p(N), \Upsilon_s(N)).$$

Given two LGT networks N_1, N_2, we define their *LGT distance* $d_{LGT}(N_1, N_2)$ as

$$d_{LGT}(N_1, N_2) = \frac{1}{2}\big(|\Upsilon_p(N_1) \bigtriangleup \Upsilon_p(N_2)| + |\Upsilon_s(N_1) \bigtriangleup \Upsilon_s(N_2)|\big)$$

where $\bigtriangleup$ denotes the symmetric difference between two sets, that is the set of elements which are in either of the sets and not in their intersection.

Example 2.1. *Let N be the LGT network depicted in Figure 2.2(a). Table 2.1 gives $\ell_{T_0(N)}(u)$ for every node u.*

Table 2.1: $\ell_{T_0(N)}(u)$ for every node u in the LGT network N in Figure 2.2(a).

u	$\ell_{T_0(N)}(u)$
1	$(\{1\}, 0)$
2	$(\{2\}, 0)$
3	$(\{3\}, 0)$
4	$(\{4\}, 0)$
5	$(\{5\}, 0)$
6	$(\{6\}, 0)$
7	$(\{7\}, 0)$
h	$(\{2, 3\}, 0)$
k	$(\{5, 6\}, 0)$
d	$(\{5, 6, 7\}, 0)$
c	$(\{4\}, 1)$
b	$(\{4, 5, 6, 7\}, 0)$
a	$(\{1, 2, 3\}, 0)$
r	$(\{1, 2, 3, 4, 5, 6, 7\}, 0)$

Then, $\Upsilon_p(N)$ is the set consisting of these ordered pairs:

$$\begin{aligned}\Upsilon_p(N) = \ & \big\{(\{1\}, 0), (\{2\}, 0), (\{3\}, 0), (\{4\}, 0), (\{5\}, 0), (\{6\}, 0), (\{7\}, 0),\\ & (\{2, 3\}, 0), (\{5, 6\}, 0), (\{5, 6, 7\}, 0), (\{4\}, 1), (\{4, 5, 6, 7\}, 0),\\ & (\{1, 2, 3\}, 0), (\{1, 2, 3, 4, 5, 6, 7\}, 0)\big\}\end{aligned}$$

Since $(c,h),(d,h),(c,k)$ and (a,k) are the secondary arcs in N, we get

$$\Upsilon_s(N) = \big\{((\{4\},1),(\{2,3\},0)),((\{5,6,7\},0),(\{2,3\},0)),((\{4\},1),(\{5,6\},0)), ((\{1,2,3\},0),(\{5,6\},0))\big\}$$

Example 2.2. *Let N_1 and N_2 be the LGT networks represented in Figure 2.2(a) and Figure 2.4 respectively. Since*

$$\Upsilon_p(N_1) \triangle \Upsilon_p(N_2) = \big\{(\{4,5,6,7\},0),(\{1,2,3,4\},0), (\{5,6\},1),(\{7\},1)\big\}$$
$$\Upsilon_s(N_1) \triangle \Upsilon_s(N_2) = \big\{((\{1,2,3\},0),(\{5,6\},0)), ((\{5,6,7\},0),(\{2,3\},0)),((\{5,6\},1),(\{2,3\},0)), ((\{7\},1),(\{5,6\},0))\big\}$$

the LGT distance between these networks is

$$d_{LGT}(N_1,N_2) = \frac{1}{2}(4+4) = 4.$$

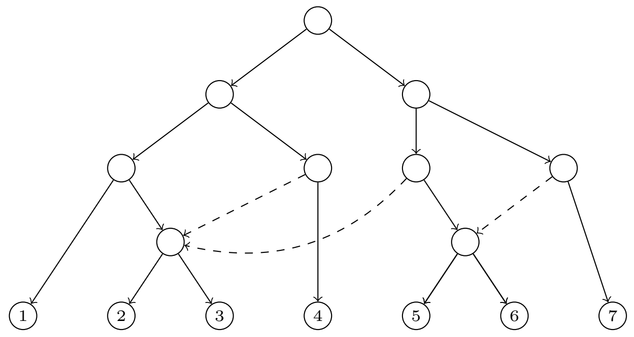

Figure 2.4: The LGT network N_2 defined on $\{1,\dots,7\}$ used in Example 2.2.

We prove now that one can reconstruct, up to isomorphisms, an LGT network N from $\Upsilon(N)$.

Proposition 2.1. *Let $N = (V,E)$ be an LGT network on a set S of taxa. Then it is possible to reconstruct from $\Upsilon(N)$ an LGT network $N' = (V',E')$ such that $N \cong N'$.*

Proof. We construct $N' = (V',E')$, with $E' = E'_p \cup E'_s$, as follows:

- $V' = \Upsilon_p(N)$,
- $(\ell_{T_0(N)}(u), \ell_{T_0(N)}(v)) \in E'_p$ if, and only if, one of the following conditions holds
  - $C_{T_0(N)}(u) = C_{T_0(N)}(v)$ and $\delta_{T_0(N)}(u) = \delta_{T_0(N)}(v)+1$
  - $C_{T_0(N)}(u) \supsetneq C_{T_0(N)}(v)$, and if $C_{T_0(N)}(u) \supset C_{T_0(N)}(w) \supset C_{T_0(N)}(v)$ for some node w, then either $C_{T_0(N)}(u) = C_{T_0(N)}(w)$ and $\delta_{T_0}(u) < \delta_{T_0}(w)$ or $C_{T_0(N)}(v) = C_{T_0}(N)(w)$ and $\delta_{T_0}(v) > \delta_{T_0}(w)$
- $(\ell_{T_0(N)}(u), \ell_{T_0(N)}(v)) \in E'_s$ if, and only if, $(\ell_{T_0(N)}(u), \ell_{T_0(N)}(v)) \in \Upsilon_s(N)$

It is straightforward to check that $N \cong N'$. □

Corollary 2.2. *d_{LGT} is a metric on any class of LGT networks over a fixed set of taxa S.*

Proof. All conditions that define a metric (see Section 1.8) follow trivially from the properties of the symmetric difference of sets, except the separation axiom

$$\Upsilon(N_1) = \Upsilon(N_2) \Rightarrow N_1 \cong N_2,$$

which is a direct consequence of Proposition 2.1. □

Once we have defined our model for LGT networks, we are ready to study the reconstruction problem from a set of trees that they induce and from substructures induced by triplets of leaves.

Chapter 3

A reconstruction problem for LGT networks based on trees

Contents

3.1 Introduction

As we have seen in the previous chapter, any LGT network gives rise to a principal phylogenetic subtree representing the primary line of evolution of the involved species. Similarly, any LGT network gives rise to a set of secondary phylogenetic subtrees, each one of them obtained by replacing one arc in the principal subtree by one secondary arc with the same target node (and then recursively removing non-labelled leaves and out-degree 1 nodes). These phylogenetic subtrees can be seen as the secondary histories, involving just one lateral gene transfer event.

This set of subtrees displayed by an LGT network suggests us to propose a reconstruction problem of phylogenies using them as the input data. However, it is not guaranteed that the principal and secondary phylogenetic subtrees of an LGT network are pairwise different and that these trees determine, up to isomorphism, the LGT network. In order to avoid these incompatibilities, in this chapter we introduce a subclass of *restricted* LGT networks, which are characterized by a set of topological restrictions that allow us to solve the reconstruction problem. The use of restricted LGT networks in order to infer with unicity an LGT network from a set of trees is a similar approach to those presented in Section 1.9.

More precisely we give an algorithm that solves the corresponding reconstruction problem from incongruent trees: given a set of pairwise different phylogenetic trees $T_0, T_1, \dots, T_k$ on the same set of taxa, find, when it exists, the unique restricted LGT network such that its principal phylogenetic tree is T_0 and its secondary phylogenetic trees are $T_1, \ldots, T_k$. Then, we include two computational experiments to test the introduced models and algorithms. The first experiment uses the database of phylogenetic trees in Beiko et al. (2011), which are based on real biological data, in order to find some subtrees to be used as input for our algorithms. In the second one we fix an LGT network and simulate DNA strands at the nodes using a statistical model of evolution that takes into account both mutations (through principal arcs) and lateral gene transfers (through secondary arcs); then, we try to infer the network from the DNA at the leaves.

The Results in this chapter have been published in Cardona et al. (2015).

3.2 Secondary and reduced subtrees

Let N be an LGT network and $e = (u, h)$ a secondary arc in N. The *secondary subtree* of N associated to e, $T_e(N)$, is the tree obtained from $T_0(N)$ by removing the principal arc ending in h and replacing it by e, see Figure 3.1.

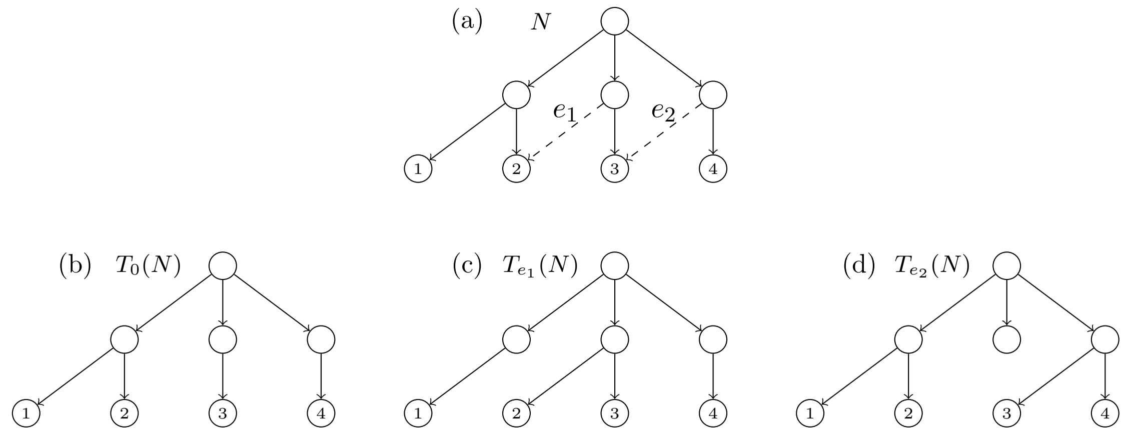

Figure 3.1: An LGT network (a), its principal subtree (b) and its secondary subtrees (c) and (d).

Although $T_0(N)$ is always an S-tree, a secondary subtree of N may have non-labelled leaves: we shall say that it is *partially leaf-labelled* in S. To obtain phylogenetic trees on S from the principal and secondary subtrees of N, we *reduce* them. That is, we recursively remove (in secondary subtrees) all their non labelled leaves together with the arcs ending in them, and then we recursively suppress all their elementary nodes. We shall generically denote by $\widetilde{T}$ the *reduced phylogenetic tree* on S obtained by reducing a partially leaf-labelled tree T on S. Notice that $\widetilde{T}$ is an *homeomorphic subtree* of T, in the sense that they have the same set of labels, the set of nodes of $\widetilde{T}$ is contained in the set of nodes of T, this inclusion preserves the leaf labelling, and every arc in $\widetilde{T}$ corresponds to a path in T. The construction of the reduced principal and secondary subtrees of an LGT network is illustrated by Figures 3.1 and 3.2.

It is important to stress the close conection between the reduced subtrees of an LGT network and their switchings. Let N be an LGT network. As we have already pointed out in Section 2.2, $T_0(N)$ is a switching of N. In addition, if $e = (u, h)$ is a secondary arc of N, $T_e(N)$ is also a switching of N, where this switching can be obtained from the one associated to $T_0(N)$ by switching-off the principal arc ending in h and switching-on the arc e. However, since $T_0(N)$ and $T_e(N)$ may have elementary nodes, they are not, in general, trees displayed by N. Indeed, we need the reduced versions $\widetilde{T}_0(N)$ and $\widetilde{T}_e(N)$ to get trees displayed by N, which are the phylogenetic trees associated to the above mentioned switchings.

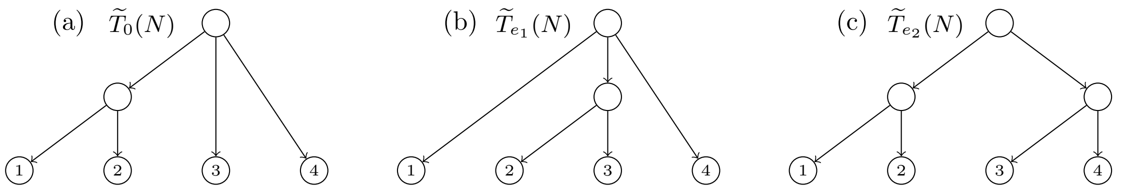

Figure 3.2: The reduced principal subtree (a) and the reduced secondary subtrees (b) and (c) of the LGT network N depicted in Figure 3.1.

3.3 Subtree prune and regraft on LGT networks

A method to compute a lower bound for the number of reticulation events which cause the inconsistency between a pair of trees (for instance, two different gene trees) is the number of *rooted subtree prune and regraft*, *rSPR* for short, operations to transform one tree into the other. Roughly speaking, an SPR operation applied to a rooted phylogenetic tree cuts, or prunes, a subtree and attaches, or regrafts, it from its root to another arc of the remaining tree. Given any pair T, T' of phylogenetic trees on the same set of labels, their *rSPR distance* $d_{rSPR}(T, T')$ is defined as the minimum number of rSPR operations that transform T into T'. This distance has been extensively used in phylogenetics, where many papers have been written applying and investigating its properties [Hein (1990); Allen and Steel (2001); Bordewich and Semple (2005), among others].

Intuitively, the difference between the reduced principal subtree $\widetilde{T}_0(N)$ and any reduced secondary subtree $\widetilde{T}_e(N)$ is that some rooted subtree of the former is pruned (by removing the principal arc whose end is also the end of e) and regrafted (through the secondary arc e) into the latter. This fact motivates to consider rSPR operations to analyze the differences between the reduced principal subtree of an LGT network and its reduced secondary subtrees. However, since these trees may not be binary, we slightly generalize the rSPR operations defined in Bordewich and Semple (2005) to allow for the pruned subtree to be regrafted not only to an arc but also to a node.

More precisely, we define an rSPR operation of a tree T as the following procedure:

1) Choose an arc $e = (u, v)$ of T.
2) Remove e from T.
3) Choose a node w that is not a descendant of v.
4) If w is an internal node other than u, then apply either (4a) or (4b) below. If w is a leaf or $w = u$, apply (4b).
   4a) Add an arc (w, v).
   4b) Add a new node $\widetilde{w}$ and new arcs $(\widetilde{w}, v)$ and $(\widetilde{w}, w)$. If w was not the root of T and w' was its parent, then remove the arc (w', w) and add a new arc $(w', \widetilde{w})$. If w was the root, then $\widetilde{w}$ becomes the root of the resulting tree.
5) Suppress u if it has become elementary.

We shall denote such an rSPR operation by $v \overset{node}{\longleftarrow} w$ (a *node rSPR operation*) if step (4a) is applied, and $v \overset{arc}{\longleftarrow} w$ (an *arc rSPR operation*) if step (4b) is applied; see Figure 3.3. In graphical representations of trees, paths are represented by snaked arrows. When it is not necessary to specify whether it is a node or an arc rSPR operation, we shall denote it by $v \overset{spr}{\longleftarrow} w$.

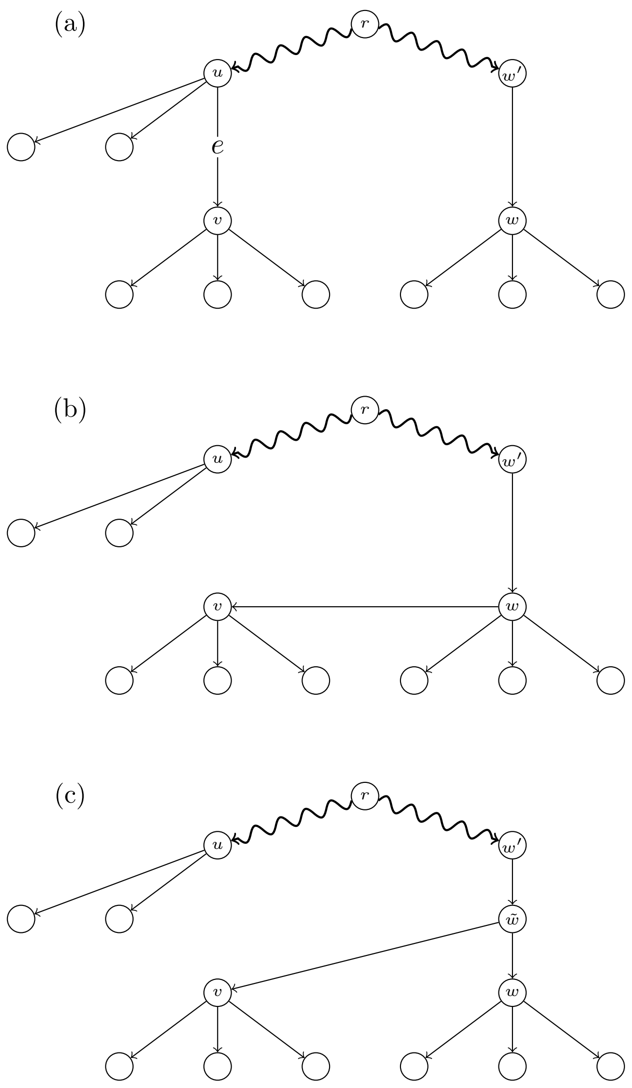

Figure 3.3: The original tree (a), the tree obtained via node rSPR operation $v \overset{node}{\longleftarrow} w$ (b), and the tree obtained via arc rSPR operation $v \overset{arc}{\longleftarrow} w$ (c).

For example, the two phylogenetic trees T and T' depicted in Figure 3.4 have $d_{rSPR}(T, T') = 2$ where two arc rSPR operations has been applied to obtain T' from T.

Particularly, since a reduced secondary subtree $\widetilde{T}_e(N)$ of an LGT network is obtained from its reduced principal subtree $\widetilde{T}_0(N)$ by means of an rSPR operation, we have that $d_{rSPR}(\widetilde{T}_0(N), \widetilde{T}_e(N)) \leq 1$, and $d_{rSPR}(\widetilde{T}_0(N), \widetilde{T}_e(N)) = 1$ if, and only if, $\widetilde{T}_0(N) \neq \widetilde{T}_e(N)$.

3.4 LGT network reconstruction problem

In this section we consider the problem of reconstructing an LGT network from its reduced principal subtree T_0 and its set of reduced secondary subtrees $T_1, \ldots, T_k$. We shall take

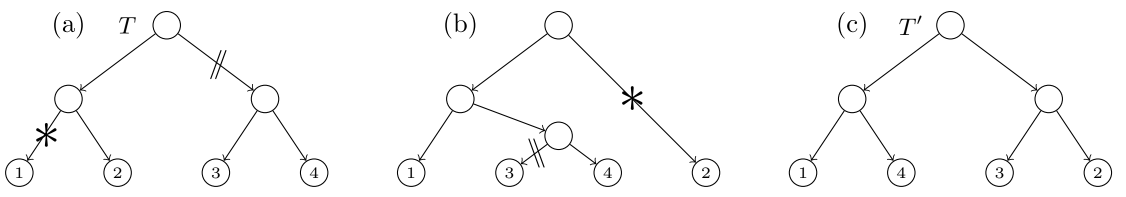

Figure 3.4: The rSPR distance between the tree T in (a) and the tree T' in (c) is 2. The tree in (b) is the intermediate one in the process to obtain T' from T. The slash and the star symbols on the arcs in (a) and (b) indicate the arc which is pruned and the arc on which is regrafted, respectively; in order to obtain the tree located at its right-hand side.

into account only the case when $T_1, \ldots, T_k$ are pairwise different, because if $T_i = T_j$, they can be defined by the same secondary arc. Moreover, we shall restrict ourselves to the case when $T_0 \neq T_i$ for every $i = 1, \ldots, k$, because when a reduced secondary subtree is equal to the reduced principal subtree, it only means that we are not able to "distinguish" the secondary line of evolution from the principal one. This leads us to the following general problem:

Problem 3.1 *LGT network Reconstruction*

Input: A family of pairwise different phylogenetic trees $T_0, T_1, \ldots, T_k$, on the same set of labels S, such that $d_{rSPR}(T_0, T_i) = 1$ for every $i = 1, \ldots, k$.

Output: An LGT network N on S with secondary arcs $e_1, \ldots, e_k$ such that $\widetilde{T}_0(N) = T_0$ and $\widetilde{T}_{e_i}(N) = T_i$, for every $i = 1, \ldots, k$, if it exists.

Note that, the LGT Network Reconstruction Problem, is focused in pairs of input trees (T_0, T_i), such that $d_{rSPR}(T_0, T_i) = 1$. That is, we only get input trees (for instance, gene trees) with one lateral transfers each. Then, the problem is not suitable for situations where, for instance, one tree shows the vertical signal and the other one with at least two lateral transfers events. Although these scenarios could be adopted, for example decomposing a tree with $n \geq 2$ lateral transfer events in n trees with only one transfer, we focus in this chapter in the case where $d_{rSPR}(T_0, T_i) = 1$.

Notice that the number of secondary subtrees displayed by an LGT network is at most the number of secondary arcs, since one gets one secondary tree for each secondary arc, but different arcs may give the same secondary tree. Hence, a network N that is a solution of the LGT Network Reconstruction Problem has at least k secondary arcs, and one solution with exactly k secondary arcs will necessarily be optimal.

The LGT Network Reconstruction Problem may have no solution for certain input trees. Consider, for instance, the trees T_0, T_1, T_2 depicted in Figure 3.5. A simple inspection shows that if there exists an LGT network N with reduced principal subtree T_0 and two secondary arcs e_1, e_2 such that $\widetilde{T}_{e_1}(N) = T_1$ and $\widetilde{T}_{e_2}(N) = T_2$, then e_1 must go from an elementary node added in the arc ending in 4 to a (or to an elementary node added in the arc ending in a), and e_2 must go from an elementary node added in the arc ending in 3 to c (or to an elementary node added in the arc ending in c). But then, the resulting directed graph contains a directed cycle: see, for instance, the graph N in Figure 3.5.

On the other hand, as it was already hinted in the discussion above, if the LGT network reconstruction problem has a solution for a specific input, it need not be unique: see, for instance, Figure 3.6. And, as we mentioned at the beginning of this section, there may be repetitions in the family of reduced principal and secondary subtrees of a general LGT network, and therefore not every LGT network can be obtained as an output of this

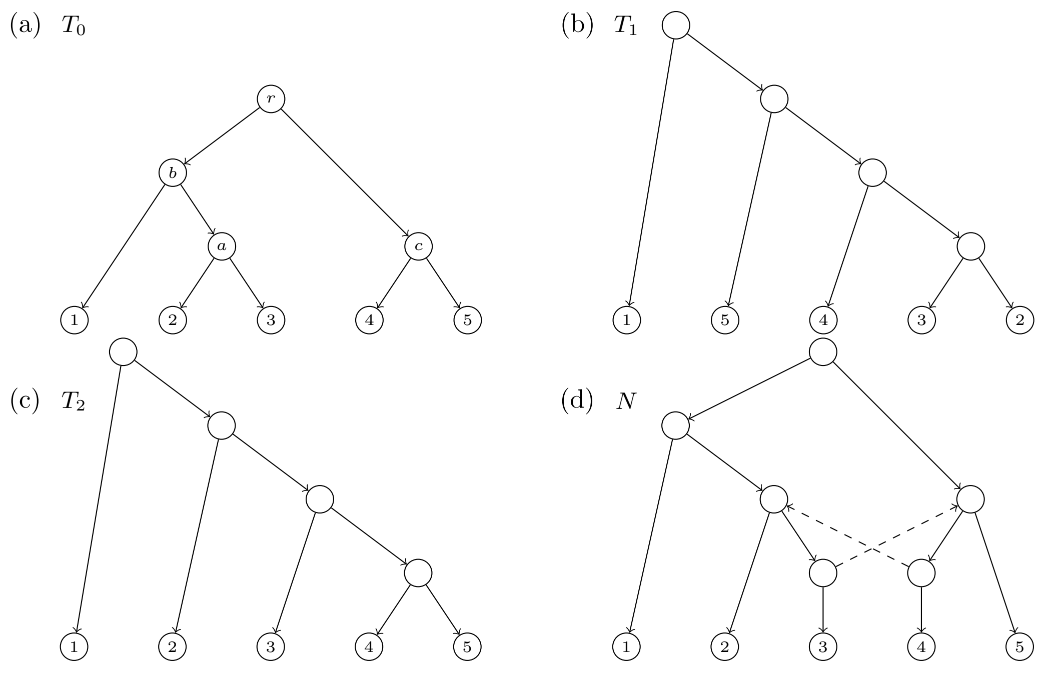

Figure 3.5: Any "LGT network" with reduced principal subtree T_0 and reduced secondary subtrees T_1, T_2 would contain a cycle.

problem.

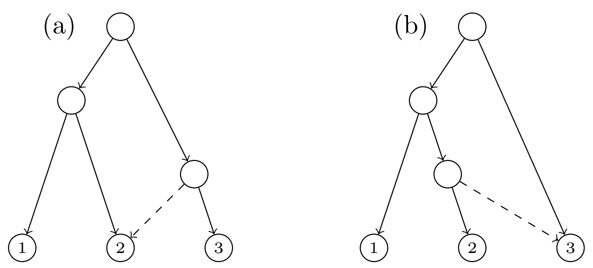

Figure 3.6: Two LGT networks with the same reduced principal and secondary subtrees.

3.5 Restricted LGT networks

As we have shown in the previous section, where we have exhibited two different LGT networks with the same reduced principal and secondary subtrees, the LGT Network Reconstruction Problem cannot be solved with unicity for generic LGT networks. This motivates us to restrict ourselves to a class of LGT networks satisfying a set of conditions that guarantee, on the one hand, that their reduced principal and secondary subtrees are pairwise different and, on the other hand, that there are no two different networks in that class with the same reduced principal and secondary subtrees. The drawback of restricting to a subclass of LGT networks is that, in some instances, the reconstruction problem might have a solution but outside our class, which we cannot find.

Before giving the definition of this subclass of LGT networks, we introduce some notation relative to nodes and paths in an LGT network N. Since the principal subtree $T_0(N)$ of

N is an S-tree, every internal node of N has some principal child. We say that a node v is *principally elementary* when it has exactly one principal child, i.e., when it is elementary in $T_0(N)$. Since N cannot contain elementary nodes, this implies that every principally elementary node is the source of some secondary arc. We say that a path in N is *principal* when it consists only of principal arcs. In graphical representations of LGT networks, principal paths are represented by snaked arrows. Given u and v two nodes in N, if v can be reached from u through a principal path, we say that v is a *principal descendant* of u, and that u is a *principal ascendant* of v. A path $u \rightsquigarrow v$ is *elementary* when all its nodes, except at most v (but including its origin u), are elementary. A *principally elementary path* in N is an elementary path in $T_0(N)$.

Definition 3.1. *An LGT network is* restricted *when it satisfies the following properties:*

(a) No principal child of a principally elementary node is principally elementary.

(b) The target of a secondary arc is never principally elementary.

(c) If (u, h) is a secondary arc, then there exists no principal path $u \rightsquigarrow h$.

(d) If (u, h) is a secondary arc and $z = LCA_{T_0(N)}(u, h)$, then the principal path $z \rightsquigarrow h$ contains some non principally elementary intermediate nodes.

Conditions (a) and (b) are necessary to guarantee the uniqueness of the solutions:

- We suppose that condition (a) fails in an LGT network N. That is, N contains a principal arc (u, u') with both u, u' principally elementary: then (since N cannot contain elementary nodes) both u, u' must be sources of secondary arcs, say $e = (u, h)$ and $e' = (u', h')$. If $h = h'$, these arcs define the same reduced secondary subtree. If $h \neq h'$, then, if we replace e and e' by $\bar{e} = (u, h')$ and $\bar{e}' = (u', h)$, we obtain a new LGT network with the same reduced principal and secondary subtrees as N. Then, condition (a) is a necessary condition. See Figure 3.7.
- We suppose that condition (b) fails in an LGT network N. That is, N contains a secondary arc $e = (u, h)$ with h principally elementary. Let h' be the principal child of h. We shall assume that N does not contain the secondary arc $e' = (u, h')$, because otherwise $\widetilde{T}_e(N) = \widetilde{T}_{e'}(N)$. Then, if we replace the secondary arc (u, h) by a secondary arc (u, h'), we obtain a new LGT network with the same reduced principal and secondary subtrees as N. Then, condition (b) is a necessary condition. See Figure 3.8.

Conditions (c) and (d) are needed to avoid that the reduced principal subtree is isomorphic to some reduced secondary subtree, or that two different reduced secondary subtrees are isomorphic. These two conditions are topological constraints that are biologically meaningful, similarly to some other constraints that we have seen in Section 1.7. More precisely, condition (c) prevents the existence of a lateral gene transfer from a species to a principal descendant of it, and similarly, condition (d) requires to have at least one speciation event between the LCA of both species involved in a lateral gene transfer and the species that receives the transference.

In Figure 3.7, there are depicted two (different) phylogenetic networks N_1 and N_2 which do not satisfy condition (a) and both networks display the same set of reduced principal and secondary subtrees. The same happens in the networks depicted in Figure 3.8, where condition (b) fails. Moreover, in Figures 3.9 and 3.10 there are depicted phylogenetic networks where conditions (c) and (d) fail, respectively. In the right side of both figures we can observe that the reduced principal subtree is isomorphic to the reduced secondary subtrees.

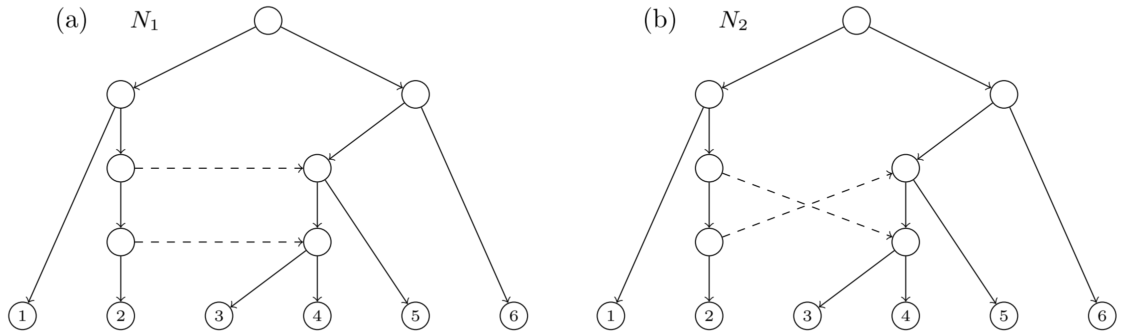

Figure 3.7: Two phylogenetic networks where condition (a) in Definition 3.1 fails and both networks have the same reduced principal subtree and also the same reduced secondary subtrees.

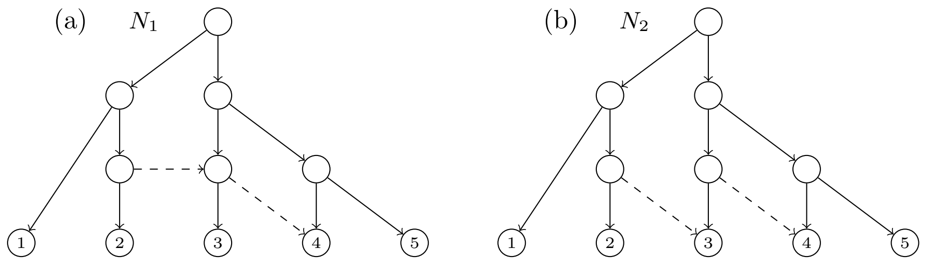

Figure 3.8: Two phylogenetic networks where condition (b) in Definition 3.1 fails and both networks have the same reduced principal subtree and also the same reduced secondary subtrees.

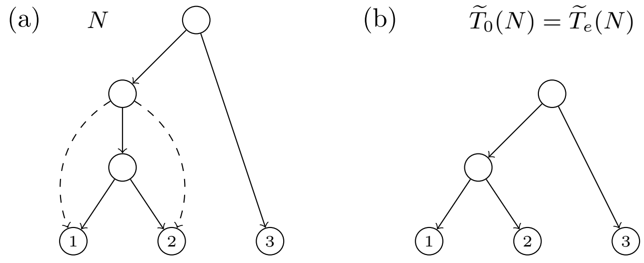

Figure 3.9: (a) A phylogenetic network where condition (c) in Definition 3.1 fails and (b) the reduced principal subtree which is isomorphic to the two reduced secondary subtrees.

As we have previously mentioned in Section 2.3, except for condition (c), which is shared by both definitions, the conditions that define our restricted LGT networks are independent of those defining species graphs [Górecki (2004)]. Note that the principal subtree of a restricted LGT network is not necessarily a semi-binary tree and the source of a secondary arc is not necessarily principally elementary; we explicitly consider the target of a secondary arc being non principally elementary; we also explicitly consider two principally elementary source nodes of secondary arcs being not connected by a principal arc; we allow two secondary arcs to be adjacent; and we impose no extra condition on the relative position of secondary arcs, except for the fact that the resulting graph must be acyclic. On the other hand, no condition similar to (d) is imposed on species graphs.

Notice that conditions (a) and (b) in the definition of restricted LGT networks allow us

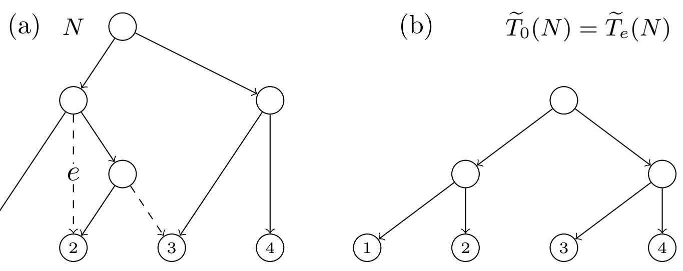

Figure 3.10: (a) A phylogenetic network where condition (d) in Definition 3.1 is not satisfied and (b) the reduced principal subtree which is isomorphic to the reduced secondary subtrees of the arc e.

to construct any such network by "adding" arcs to its reduced principal subtree. More precisely, let $T = (V, E)$ be a phylogenetic tree and $F \subset V \times V \cup E \times V$ a set whose elements are pairs formed by either two nodes of the tree or an arc and a node of the tree. We can define an LGT network, that we will denote by $N = T + F$, as follows:

1. For each $e \in E$, if there is some node h with $(e, h) \in F$, split the arc e by inserting an intermediate elementary node v_e. More formally, add a node v_e, remove the arc e, and add arcs from the origin of e to u and from u to the end node of e.
2. For each element $(e, h) \in F$ with $e \in E$, add a secondary arc from v_e to h
3. For each element $(u, h) \in F$ with $u \in V$, add a secondary arc from u to h.

It is straightforward to check that any restricted LGT network can be obtained by means of this procedure, but not every choice for F gives a restricted LGT network. Strictly speaking, this process generates directed graphs whose arcs can be principal o secondary, but since this procedure can even generate cycles, there is in principle no guarantee that it produces LGT networks for an arbitrary choice of F. However, if this procedure gives a restricted LGT network, then its reduced principal subtree will be the tree T used as input of the procedure. Notice also that if two different restricted networks have isomorphic reduced principal subtrees, then the nodes and arcs of both trees can be identified. For instance, each node is identified by its cluster, which is an invariant of the isomorphism class. More formally, both trees are canonically isomorphic and hence we can safely identify them and say that they are equal. A consequence of the construction above is that we can also say when two secondary arcs in two different networks are "the same", provided that they have the same reduced principal tree. Indeed, each secondary arc is constructed from a unique pair formed by an arc and a node or two nodes of the common reduced principal subtree, and secondary arcs in different networks can be compared by comparing this data that determines them and depends only on the reduced principal subtree.

We shall prove now that the reduced principal and secondary subtrees of a restricted LGT network form a family of pairwise different phylogenetic trees.

Notice that, if $T = (V, E)$ is an $S-$tree and $\widetilde{T}$ is the reduced version of T, for every node v in $\widetilde{T}$, $C_T(v) = C_{\widetilde{T}}(v)$. We shall often use this equality without any further mention. Given N an LGT network, we say that the *principal cluster* of a node u is the set $C_{T_0(N)}(u)$ of leaves that are principal descendants of u. The following result is a direct consequence of the fact that the set of triples defined by a phylogenetic tree characterizes it, and that the triple defined on a set of three labels by a partially leaf-labelled tree with, possibly, elementary nodes, is the same as the triple defined by its reduction. For a review for some definitions, notation and results relative to clusters and triples, see Section 1.5.

Lemma 3.1. *Let T_1, T_2 be two partially leaf-labelled trees on a set S. Then, $\widetilde{T_1} = \widetilde{T_2}$ if, and only if, T_1 and T_2 define the same triple on each set of three different labels of S.*

The two following results prove that the reduced principal and secondary subtrees of a restricted LGT network are pairwise different.

Proposition 3.2. *If N is a restricted LGT network and e is a secondary arc in it, then $\widetilde{T_0}(N) \neq \widetilde{T_e}(N)$.*

Proof. Let $e = (u, h) \in E_s$; to simplify the notations, we shall denote $T_0(N)$ and $T_e(N)$ by T_0 and T_e, respectively. We shall prove that these trees define different sets of triples; by Lemma 3.1, this will imply that $\widetilde{T_0} \neq \widetilde{T_e}$.

By condition (c) in Definition 3.1, there exists no principal path connecting u and h, and therefore $C_{T_0}(h) \cap C_{T_0}(u) = \emptyset$. Let $x_1 \in C_{T_0}(u)$ and $x_2 \in C_{T_0}(h)$. Moreover, if $z = LCA_{T_0}(u, h)$, condition (d) in Definition 3.1 implies that the principal path $z \rightsquigarrow h$ contains some intermediate node w with a principal child w_1 outside this path; let $x_3 \in C_{T_0}(w_1)$ (see Figure 3.11). It is straightforward to check now that T_0 defines the triple $((x_2, x_3), x_1)$ and T_e defines the triple $((x_1, x_2), x_3)$. Therefore, $\Gamma(T_0) \neq \Gamma(T_e)$, as we claimed. ☐

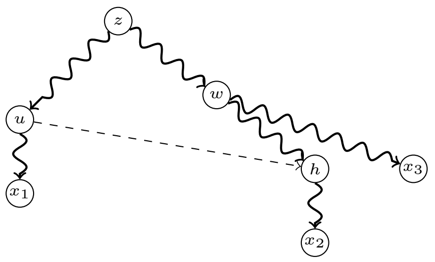

Figure 3.11: The structure of N involving arc e in the proof of Proposition 3.2.

Proposition 3.3. *If N is a restricted LGT network and e, e' are two different secondary arcs in it, then $\widetilde{T_e}(N) \neq \widetilde{T_{e'}}(N)$.*

The proof of this proposition is similar to that of Proposition 3.2, but much longer because we must distinguish many cases, depending on the relative positions of the source and the target nodes of e and e' in $T_0(N)$. Therefore, for the sake of clarity, and in order not to lose the pace of the chapter, we give it in Section 3.7, containing some proofs of technical results.

The problem we want to solve (and effectively do in some cases) in this section is, then, the following special case of the LGT Network Reconstruction Problem:

PROBLEM 3.2 *Restricted LGT network Reconstruction*

INPUT: A family of pairwise different phylogenetic trees $T_0, T_1, \ldots, T_k$, on the same set of labels S, such that $d_{rSPR}(T_0, T_i) = 1$ for every $i = 1, \ldots, k$.

OUTPUT: A *restricted* LGT network N on S with secondary arcs $e_1, \ldots, e_k$ such that $\widetilde{T_0}(N) = T_0$ and $\widetilde{T_{e_i}}(N) = T_i$, for every $i = 1, \ldots, k$, if it exists.

Our next goal is now to determine a set of necessary and sufficient conditions for the existence of a restricted LGT network N with a given principal subtree T and a given secondary subtree T'. First, we give these conditions in terms of rSPR operations. Next, we translate the resulting conditions in terms of triples and clusters.

Proposition 3.4. *Let T, T' be two phylogenetic trees on the same set of labels. There exists a restricted LGT network N with a secondary arc e such that $T = \widetilde{T}_0(N)$ and $T' = \widetilde{T}_e(N)$ if, and only if:*

(1) $d_{rSPR}(T, T') = 1$,

(2) if $h \overset{spr}{\longleftarrow} w$ is an rSPR operation that produces T' from T, then, in T, w is neither an ancestor of h nor a descendant of the parent of h.

Proof. As far as the necessity of conditions (1) and (2) goes, recall from Section 3.3 that, if N is an LGT network and $e = (u, h)$ a secondary arc in it, then $\widetilde{T}_e(N)$ is obtained from $\widetilde{T}_0(N)$ by means of either a node rSPR operation $h \overset{node}{\longleftarrow} u$, when u is not principally elementary in N, or an arc rSPR operation $h \overset{arc}{\longleftarrow} u^*$, with u^* the only principal child of u in N, when it is principally elementary. Since, moreover, $\widetilde{T}_e(N) \neq \widetilde{T}_0(N)$ by Proposition 3.2, this entails that $d_{rSPR}(T, T') = 1$. On the other hand, u (or u^*, in the second case) can be neither a principal ancestor of h, because of condition (c) in Definition 3.1, nor a proper principal descendant of the parent v of h in $\widetilde{T}_0(N)$, because this would imply that $v = LCA_{T_0}(u, h)$, against condition (d) in Definition 3.1.

Let us prove now the sufficiency of conditions (1) and (2). If T' is obtained from T by means of a node rSPR operation $h \overset{node}{\longleftarrow} w$, let N be the LGT network obtained by adding to T the secondary arc (w, h). If T' is obtained by means of an arc rSPR operation $h \overset{arc}{\longleftarrow} w$, then, since h is not a descendant of w in T, the latter cannot be the root; in this case, if v is its parent in T, split the arc (v, w) by adding an intermediate node u in it, and add a secondary arc $e = (u, h)$; let N be the resulting LGT network.

In both cases, it is clear by construction that $\widetilde{T}_0(N) = T$ and $\widetilde{T}_e(N) = T'$. Moreover, N clearly satisfies condition (a) (because N has at most one principally elementary node), (b) (because h is not elementary in T), (c) (because h is not a descendant of w in T), and (d) (because, since w is not a descendant in T of the parent h_0 of h, the path $LCA_T(w, h) \rightsquigarrow h$ in $T_0(N)$ contains h_0 as intermediate node, and it is not elementary in T) in the definition of restricted LGT network. □

We rewrite the characterization provided by the previous proposition in terms of triples (Proposition 3.5) using the following definition.

Definition 3.2. *Let T, T' be two trees defined on the same set of labels S. Let $\{T_{x,y,z} : \{x, y, z\} \subset S\}$ and $\{T'_{x,y,z} : \{x, y, z\} \subset S\}$ be their respective set of triples. Then, we say that T and T' satisfy the* principal-secondary condition on triples *if there exists $k, l, m \geq 1$ and a partition of S*

$$A_1, \ldots, A_k, B, C_1, \ldots, C_{l-1}, C_{l,1}, \ldots, C_{l,m}$$

(and to ease notations, let $C_l = \bigcup_{i=1}^{m} C_{l,i}$) such that, for every $x, y, z \in S$:

(1) If $x \in \bigcup_{i=1}^{k} A_i$, $y \in B$, and $z \in \bigcup_{i=1}^{l} C_i$, then $T_{x,y,z} = ((x, y), z)$ and $T'_{x,y,z} = ((y, z), x)$.

(2) If $x \in B$, $y \in A_j$ and $z \in A_i$, for some $1 \leq i < j \leq k$, then $T_{x,y,z} = ((x, y), z)$ and $T'_{x,y,z} = ((y, z), x)$.

(3) *If $x \in C_i$, $y \in C_j$ and $z \in B$, for some $1 \leq i < j \leq l$, then $T_{x,y,z} = ((x, y), z)$ and $T'_{x,y,z} = ((y, z), x)$.*

(4) *If $x \in C_{l,i}$, $y \in C_{l,j}$ and $z \in B$, for some $1 \leq i < j \leq m$, then $T_{x,y,z} = ((x, y), z)$ and $T'_{x,y,z} = (x, y, z)$.*

(5) *If x, y, z do not satisfy any of the previous conditions, then $T_{x,y,z} = T'_{x,y,z}$.*

See Figure 3.12 for a depiction of these sets.

Proposition 3.5. *Let T, T' be two phylogenetic trees on the same set of labels. There exists a restricted LGT network N with a secondary arc e such that $T = \widetilde{T}_0(N)$ and $T' = \widetilde{T}_e(N)$ if, and only if, they satisfy the principal-secondary condition on triples.*

Proof. As far as the "only if" implication goes, assume that $e = (w, h)$ and let $v = LCA_{T_0(N)}(w, h) = LCA_{\widetilde{T}_0(N)}(w, h)$. Let $\tilde{w} \in \widetilde{T}_0(N)$ be the first non principally elementary principal descendant of w: that is, $\tilde{w} = w$ if w is not principally elementary, and its principal child otherwise. Now:

- let $v \to u_1 \to \cdots \to u_k \to h$ be the path $v \rightsquigarrow h$ in $\widetilde{T}_0(N)$ (where $k \geq 1$ by condition (d) in Definition 3.1);
- let $v \to w_1 \to \cdots \to w_{l-1} \to w_l = \tilde{w}$ be the path $v \rightsquigarrow \tilde{w}$ in $\widetilde{T}_0(N)$ (where $l \geq 1$ because condition (c) in Definition 3.1 implies that $w \neq v$);
- for every $i = 1, \ldots, k-1$, let $A_i = C_{T_0(N)}(u_i) \setminus C_{T_0(N)}(u_{i+1})$;
- let $A_k = C_{T_0(N)}(u_k) \setminus C_{T_0(N)}(h)$;
- let $B = C_{T_0(N)}(h)$;
- for every $i = 1, \ldots, l-1$, let $C_i = C_{T_0(N)}(w_i) \setminus C_{T_0(N)}(w_{i+1})$;
- if $\tilde{w} = w$, let $x_1, \ldots, x_m$ be its children in $\widetilde{T}_0(N)$, and let $C_{l,i} = C_{T_0(N)}(x_i)$, for $i = 1, \ldots, m$; if w is principally elementary in N, let $C_l = C_{l,1} = C_{\widetilde{T}_0(N)}(\tilde{w}) = C_{T_0(N)}(w)$.

See Figure 3.12. It is straightforward to check that the triples defined by $T_0(N)$ and $T_e(N)$ are the same except for those in the statement.

Let us consider now the "if" implication. We shall outline here the proof, and fill in the details in a series of Claims proved in the Section 3.7.

Assuming that the symmetric difference $\Gamma(T) \triangle \Gamma(T')$ consists of those triples described in the statement, we have that B is a cluster of both T and T' (Claim 1 in Section 3.7, where it is proved). Since every triple in $\Gamma(T) \triangle \Gamma(T')$ involves one, and only one, leaf in B, it is clear that $\Gamma(T|_B) = \Gamma(T'|_B)$ and $\Gamma(T|_{S\setminus B}) = \Gamma(T'|_{S\setminus B})$ and hence $T|_B = T'|_B$ and $T|_{S\setminus B} = T'|_{S\setminus B}$. So, $T|_B$ and $T|_{S\setminus B}$ form a maximum-agreement forest for T and T' in the sense of Hein et al. (1996), which implies that $d_{rSPR}(T, T') = 1$ [Theorem 2.1 Bordewich and Semple (2005)]. Then, the rSPR operation that transforms T into T' must have the form $h \overset{spr}{\longleftarrow} x$, with h the root of $T|_B$, that is, the node in T with $C_T(h) = B$. In order to prove that this rSPR operation satisfies condition (2) in Proposition 3.4, we must identify the node x and the type of rSPR operation. To do that, we use that each $C_{l,i}$ is a cluster in T and T' (Claim 2 in Section 3.7) and that $B \cup C_l$ is a cluster in T' but not in T (Claim 3 in Section 3.7). Then:

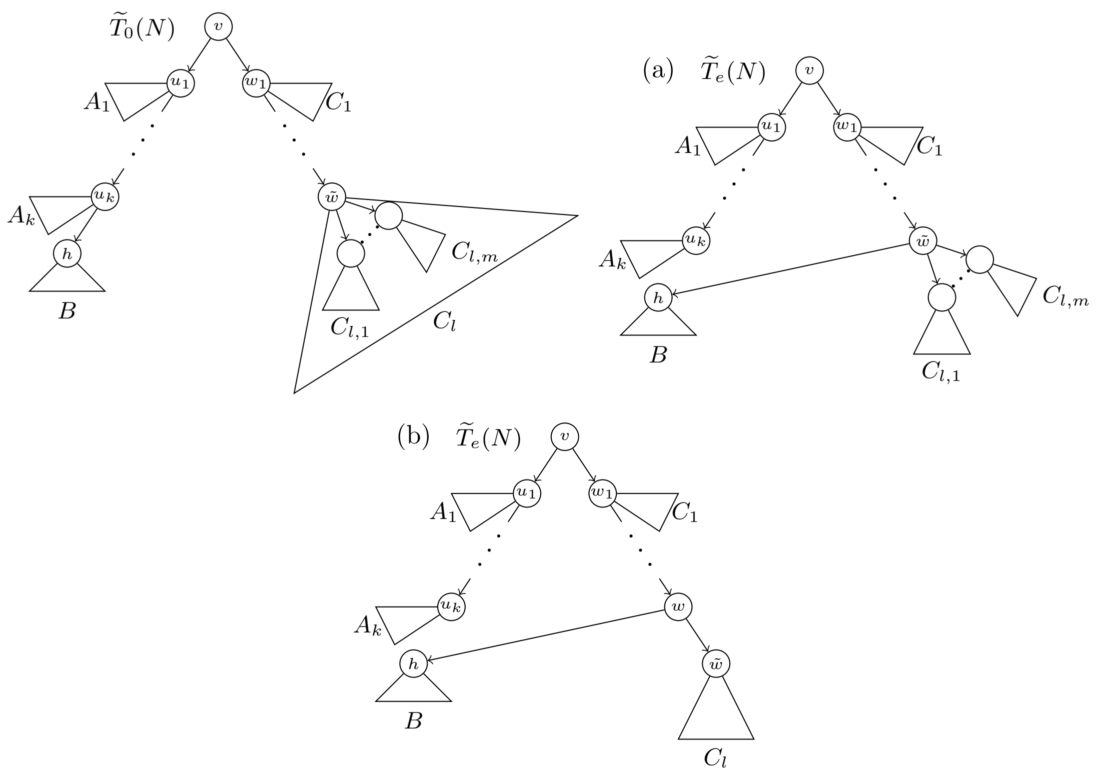

Figure 3.12: The local structure of $\widetilde{T}_0(N)$ and $\widetilde{T}_e(N)$ around a secondary arc $e = (w, h)$, when w is not principally elementary (a) and when it is principally elementary (b).

- If $m = 1$, so that $C_l = C_{l,1} \in C(T) \cap C(T')$, this entails that the nodes with clusters B and C_l are sibling in T' but not in T, and therefore that x is the node in T with cluster C_l and that the rSPR operation is of type arc.
- If $m > 1$, since C_l is a cluster in T but not in T' (Claim 4 in Section 3.7) and $B \cup C_{l,i_1} \cup \cdots \cup C_{l,i_k} \notin C(T')$ for every $\emptyset \neq \{i_1, \ldots, i_k\} \subsetneq \{1, \ldots, m\}$ (Claim 5 Section 3.7), we have that the nodes with clusters $B, C_{l,1}, \ldots, C_{l,m}$ are sibling in T' but not in T, and therefore that x is the node in T with cluster C_l and that the rSPR operation is of type node.

In both cases, it is easy to see that x is not connected in T with h (because $B \cap C_l = \emptyset$) and that $LCA_T(x, h)$ is not the parent of h (because if $a \in A_1$, $b \in B$ and $c \in C_l$, then $((a, b), c) \in \Gamma(T)$). □

Corollary 3.6. *Let N and N' be two restricted LGT networks on the same set of labels S, each with a single secondary arc: say, e and e', respectively. If $\widetilde{T}_0(N) = \widetilde{T}_0(N')$ and $\widetilde{T}_e(N) = \widetilde{T}_{e'}(N')$, then $N = N'$.*

Proof. Let us denote $\widetilde{T}_0(N) = \widetilde{T}_0(N')$ simply by T. Since both N and N' are restricted LGT networks with reduced principal subtree T, their arcs and nodes can be identified through arcs and nodes in T. The proof of Proposition 3.5 allow us that if $\widetilde{T}_e(N) = \widetilde{T}_{e'}(N')$ both arcs e and e' can be determined as follows. With the notations therein, let e^* be either the arc ending in the node (in T) with cluster C_l (if $m = 1$) or the node (in T) with cluster C_l (if $m > 1$), and let h be the node (in T) with cluster B. Since the secondary arc determined by the pair (e^*, h) can be identified through nodes and arcs in T, adding the secondary arc in T by $T + (e^*, h)$, we can conclude that $N = N'$. □

Corollary 3.7. *Let N and N' be two restricted LGT networks on the same set of labels S. If $\widetilde{T}_0(N) = \widetilde{T}_0(N')$ and $\{\widetilde{T}_e(N) \mid e \in E_s(N)\} = \{\widetilde{T}_{e'}(N') \mid e' \in E_s(N')\}$, then $N = N'$.*

Proof. Let us denote $\widetilde{T}_0(N) = \widetilde{T}_0(N')$ simply by T. Let e be a secondary arc of N; the equality of reduced secondary trees of N and N' implies the existence of a secondary arc e' of N' such that $\widetilde{T}_e(N) = \widetilde{T}_{e'}(N')$. The same argument as in the previous corollary proves that e and e' have the same source and target and hence are the same. This proves that every secondary arc of N is also a secondary arc of N', and the symmetrical argument proves the other inclusion. Then, N and N' have the same principal subtree and the same secondary arcs, and hence they are equal. □

Notice that the naïve implementation of the procedure given by Proposition 3.5, that computes and writes all the $O(n^3)$ triples defined by T and T', and then checks whether the symmetric difference of the corresponding sets of triples has the form described therein, takes at least $O(n^4)$ time, due to there is at least $O(n)$ possible partitions. Although this cost can possibly be reduced by using the strategy in Brodal et al. (2013), we found it simpler to translate this condition on triples into an equivalent condition on clusters that is faster to check. That is, we rewrite now the characterization provided by Proposition 3.4 in terms of clusters (Proposition 3.8). Before, we introduce some definitions in the context of partially ordered sets (posets) in order to be considered on clusters, since for a given tree T, $(C(T), \subseteq)$ is a poset. If $(P, \leq)$ is a poset, a *segment* in $(P, \leq)$ is a chain such that every element in the poset lying between the ends of the chain also belongs to the chain. More formally, a chain $A_1 \leq A_2 \leq \cdots \leq A_n$ is a segment if for each $B \in P$ such that $A_1 \leq B \leq A_n$, then $B = A_i$ for some $i = 1, \ldots, n$.

Definition 3.3. *Let T, T' be two trees defined on the same set of labels S. Let $C(T)$ and $C(T')$ be their respective set of clusters. Then, we say that T and T' satisfy the* principal-secondary condition on clusters *if:*

(a) The symmetric difference of the clusters of T and T' can be written as follows: there exist $k, l \geq 1$ such that:

- *$C(T) \setminus C(T')$ consists (at most) of two maximal disjoint segments in $C(T)$*
$$U_k \subsetneq \cdots \subsetneq U_1, \quad W_{l_0} \subsetneq \cdots \subsetneq W_1,$$
with $l - 1 \leq l_0 \leq l$.
- *$C(T') \setminus C(T)$ consists (at most) of two maximal disjoint segments in $C(T')$*
$$U'_{k_0} \subsetneq \cdots \subsetneq U'_1, \quad W'_l \subsetneq \cdots \subsetneq W'_1,$$
with $k - 1 \leq k_0 \leq k$.
- *If $l = 1$ and $l_0 = l - 1$, (respectively, if $k = 1$ and $k_0 = k - 1$), the chain $W_{l_0} \subsetneq \cdots \subsetneq W_1$ (respectively, $U'_{k_0} \subsetneq \cdots \subsetneq U'_1$) does not exist, and then $C(T) \setminus C(T')$ (respectively, $C(T') \setminus C(T)$) consists only of the other segment.*
- *If $C(T) \setminus C(T')$ (respectively, $C(T') \setminus C(T)$) consists of two maximal disjoint segments of clusters, then $U_1 \cap W_1 = \emptyset$ (respectively, $U'_1 \cap W'_1 = \emptyset$).*

(b) The minimal elements in the chains above satisfy that $U_k \cap W'_l \in C(T) \cap C(T')$. Let B denote this cluster.

(c) The difference between the first element in the first segment and the common cluster B, say $A_k = U_k \setminus B$ satisfies:

- *$A_k \in C(T')$;*
- *if $k_0 = k-1$, then $A_k \in C(T)$;*
- *if $k_0 = k$, then $U'_k = A_k \notin C(T)$.*

(d) Analogously, the difference between the first element in the last segment and the common cluster B, say $C_l = W'_l \setminus B$ satisfies:

- *$C_l \in C(T)$;*
- *if $l_0 = l-1$, then $C_l \in C(T')$;*
- *if $l_0 = l$, then $W_l = C_l \notin C(T')$.*

(e) If $k > 1$, the differences between consecutive sets in the segments above satisfy:

- *$A_k \subsetneq U'_{k-1}$;*
- *setting (even when $k_0 = k-1$) $U'_k = A_k$, we have that $U_i \setminus U_{i+1} = U'_i \setminus U'_{i+1}$ for every $i = 1, \dots, k-1$.*

(f) And analogously, if $l > 1$, then:

- *$C_l \subsetneq W_{l-1}$;*
- *setting (even when $l_0 = l-1$) $W_l = C_l$, we have that $W_i \setminus W_{i+1} = W'_i \setminus W'_{i+1}$ for every $i = 1, \dots, l-1$.*

Proposition 3.8. *Let T, T' be two different phylogenetic trees on the same set of labels. There exists a restricted LGT network N with a secondary arc e such that $T = \widetilde{T}_0(N)$ and $T' = \widetilde{T}_e(N)$ if, and only if they satisfy the principal-secondary condition on clusters.*

The principal-secondary condition on clusters can be checked in $O(n^2)$ time. Indeed, conditions (b) to (f) can be checked in linear time, since they only involve testing if certain sets are clusters of the trees or subsets of some specific sets of leaves. As for condition (a), one only needs to compute all the clusters of both trees, which can be done in $O(n^2)$ time, and then computing the symmetric difference of those sets and arranging this symmetric difference in chains, which can be done in linear time in the size of the clusters.

Proposition 3.8 allows us to detect easily the secondary arc that must be added to T in order to obtain T' as the corresponding reduced secondary tree, when it exists, using Algorithm 1.

To ease notations, we will denote by $(\omega_{T,T'}, h_{T,T'})$ the output that the algorithm returns and $N(T,T')$ the LGT network $T + \{(\omega_{T,T'}, h_{T,T'})\}$.

It turns out that $N(T,T')$ is contained in every restricted LGT network with reduced principal subtree T and having T' as a reduced secondary subtree.

Proposition 3.9. *Let N be a restricted LGT network such that $\widetilde{T}_0(N) = T$ and $\widetilde{T}_e(N) = T'$, for some secondary arc e. Let N' be the LGT network obtained by removing from N all secondary arcs except e and then suppressing elementary nodes. Then, $N' = N(T,T')$.*

Proof. In this situation, N' is also a restricted LGT network with $\widetilde{T}_0(N') = T$ and $\widetilde{T}_e(N') = T'$, and then Corollary 3.6 applies. □

Algorithm 1

Input: Two phylogenetic trees $T = (V, E)$ and T' on the same set of taxa S.
Output: A pair $(\omega, h) \in V \times V \cup E \times V$ such that $N = T + \{(\omega, h)\}$ is a restricted LGT network with reduced principal subtree T and reduced secondary subtree T', or False if no such network exists.

1. Check that T and T' satisfy the principal-secondary condition on clusters. If they do not satisfy the condition, return False.

   With the notations of that condition, detect the clusters U_k and W'_l and whether $l_0 = l$ or $l_0 = l - 1$.

2. Take the nodes h and u^* in T with $C_T(h) = U_k \cap W'_l$ and $C_T(u^*) = W'_l \setminus (U_k \cap W'_l)$.

3. Now:

   3.1 If $l_0 = l - 1$, let e be the single arc of T ending in u^*, and return (e, h).

   3.2 If $l_0 = l$, return (u^*, h).

Algorithm 2

Input: A family of phylogenetic trees $T = (V, E), T'_1, \dots, T'_m$ on the same set of taxa S.
Output: A set of pairs $F \subset V \times V \cup E \times V$ such that $N = T + F$ is a restricted LGT network with reduced principal subtree T and reduced secondary subtrees $T'_1, \dots, T'_m$, or False if no such network exists.

1. Check that each pair (T, T'_i), $i = 1, \dots, m$, satisfies the principal-secondary condition on clusters. If the condition fails for any of these pairs, return False.

2. For each $i = 1, \dots, m$, let (ω_i, h_i) be the output returned by Algorithm 1 when its input is T, T'_i.

3. Set $F = \{(\omega_i, h_i) \mid i = 1, \dots, m\}$.

4. Let $N = T + F$ and check if it contains cycles.

   - If N contains cycles, return False.
   - Otherwise, return F.

Now, Algorithm 2 solves the RESTRICTED LGT NETWORK RECONSTRUCTION problem, as the following theorem proves.

Theorem 3.10. *Let $T, T'_1, \dots, T'_m$ be a family of pairwise different phylogenetic trees on S such that each pair (T, T'_i), $i = 1, \dots, m$, satisfies the principal-secondary condition on clusters. If there exists some restricted LGT network $\bar{N}$ with reduced principal subtree T and reduced secondary subtrees $T'_1, \dots, T'_m$, then the graph N defined in step 4 of Algorithm 2 applied to $T, T'_1, \dots, T'_m$ is equal to $\bar{N}$ (up to isomorphisms of LGT networks).*

Proof. In this situation, N is also a restricted LGT network with the same reduced principal subtree and reduced secondary subtrees as $\bar{N}$, and then Corollary 3.7 applies. □

This theorem entails, on the one hand, that if there exists some restricted LGT network with reduced principal subtree T and reduced secondary subtrees $T'_1, \dots, T'_m$, then it is unique (up to isomorphisms), and, on the other hand, that Algorithm 2 is correct (and also independent of the ordering of the trees $T'_1, \dots, T'_k$), in the sense that such a restricted LGT network exists if, and only if, the algorithm finds it: notice that if the algorithm detects a cycle in step 4, then this theorem implies that no restricted LGT network can have T and $T'_1, \dots, T'_m$ as reduced principal and reduced secondary subtrees. Another consequence is the stability of the network reconstructed: if some new tree is added to the input of the algorithm, then a new secondary arc is added to the network, without altering the other secondary arcs (notice, however, that this last secondary arc could create a cycle in the network and hence the problem would have no solution).

We have implemented the algorithms in this chapter using Python. The program can be downloaded from the url `http://bit.do/LGTnetworksReconstruction`, and the only requirements are the libraries `networkx` and `pyparsing`, which are included in most of the standard distributions of python for scientific computation (e.g. anaconda). The zip file contains a README file with specific instructions on how to use the program.

Some examples

The following examples show simple applications of Algorithm 2.

Example 3.1. *Consider the trees depicted in Figure 3.13.*

- $C(T) \setminus C(T'_1) = \{\{1,2\}\}$ *and* $C(T'_1) \setminus C(T) = \{\{2,3,4,5\}\}$*. Then, with the notations of Algorithm 2,* $k = l = 1$, $k_0 = l_0 = 0$, $U_k = \{1,2\}$, $W'_l = \{2,3,4,5\}$, $B = \{2\}$, $C_l = \{3,4,5\}$, $u^*_1 = b$*, and* $h_1 = 2$*. So, we add a new principally elementary node in the middle of the arc* (r, b) *and a secondary arc* e_1 *from it to* 2.

- $C(T) \setminus C(T'_2) = \{\{1,2\},\{3,4\},\{3,4,5\}\}$ *and* $C(T'_2) \setminus C(T) = \{\{2,3\},\{1,2,3\},\{4,5\}\}$*. Then,* $k = l = 2$, $k_0 = l_0 = 1$, $U_k = \{3,4\}$, $W'_l = \{2,3\}$, $B = \{3\}$, $C_l = \{2\}$, $u^*_2 = 2$ *and* $h = 3$*. So, we add a new principally elementary node in the middle of the arc* $(a, 2)$ *and a secondary arc* e_2 *from it to* 3.

- $C(T) \setminus C(T'_3) = \{\{3,4,5\}\}$ *and* $C(T'_3) \setminus C(T) = \{\{1,2,3,4\}\}$*. Then,* $k = l = 1$, $k_0 = l_0 = 0$, $U_k = \{3,4,5\}$, $W'_l = \{1,2,3,4\}$, $B = \{3,4\}$, $C_l = \{1,2\}$, $u^*_3 = a$ *and* $h_3 = c$*. So, we add a new principally elementary node in the middle of the arc* (r, a) *and a secondary arc* e_3 *from it to* c*.*

We obtain the directed graph depicted in Figure 3.14., which is acyclic and therefore a restricted LGT network with reduced principal subtree T and reduced secondary subtrees T_1', T_2', T_3'.

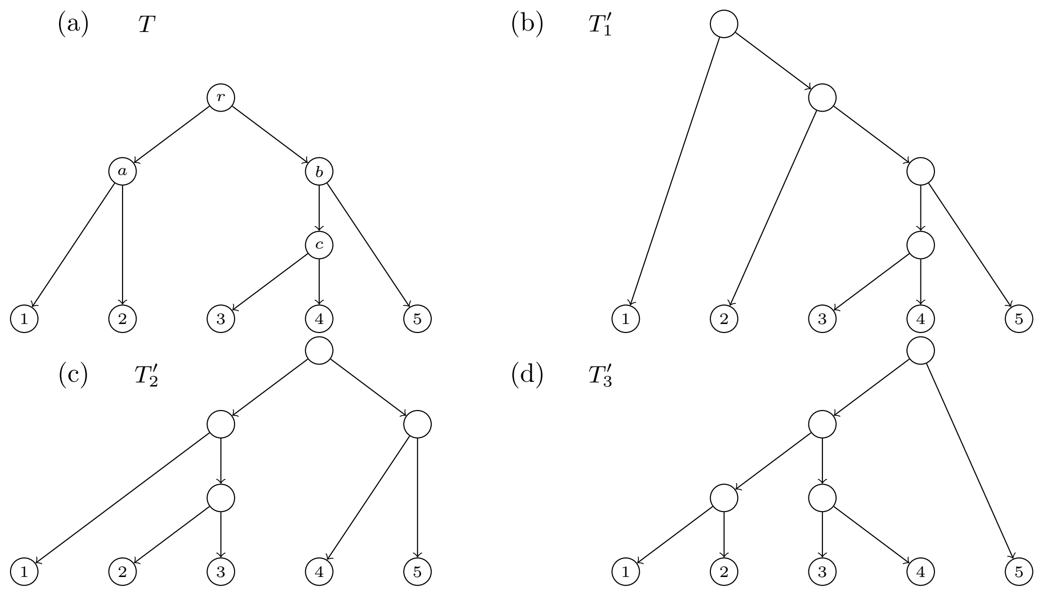

Figure 3.13: The phylogenetic trees used as input in Example 3.1.

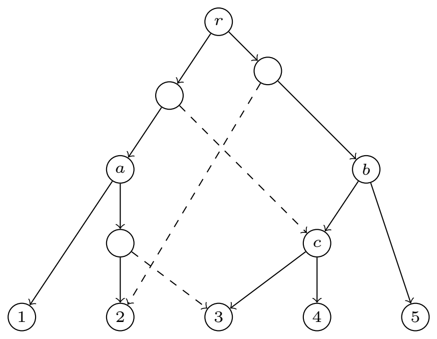

Figure 3.14: The graph obtained as output when applying Algorithm 2 to the trees T, T_1', T_2', T_3' in Figure 3.13.

Example 3.2. *Consider the trees depicted in Figure 3.15.*

- $C(T)\setminus C(T_1') = \big\{\{1,2\},\{1,2,3\},\{4,5,6\}\big\}$ *and* $C(T_1')\setminus C(T) = \big\{\{1,5,6\},\{1,2,5,6\},\{1,2,3,5,6\}\big\}$. *Then,* $k=1$, $l=3$, $k_0=0$, $l_0=2$, $U_k=\{4,5,6\}$, $W_l'=\{1,5,6\}$, $B=\{5,6\}$, $C_l=\{1\}$, $u_1^*=1$ *and* $h_1=d$. *So, we add a new principally elementary node in the middle of the arc* $(c,1)$ *and a secondary arc* e_1 *from it to* d.
- $C(T)\setminus C(T_2') = \big\{\{1,2,3\},\{5,6\},\{4,5,6\}\big\}$ *and* $C(T_2')\setminus C(T) = \big\{\{1,2,6\}, \{1,2,5,6\},\{1,2,4,5,6\}\big\}$. *Then,* $k=1$, $l=3$, $k_0=0$, $l_0=2$, $U_k=\{1,2,3\}$, $W_l'=\{1,2,6\}$, $B=\{1,2\}$, $C_l=\{6\}$, $u_2^*=6$ *and* $h_2=c$. *So, we add a new principally elementary node in the middle of the arc* $(d,6)$ *and a secondary arc* e_2 *from it to* c.

We obtain the directed graph depicted in Figure 3.16, which contains a directed cycle. Therefore, there does not exist any restricted LGT network with T as reduced principal subtree and T_1', T_2' as reduced secondary subtrees.

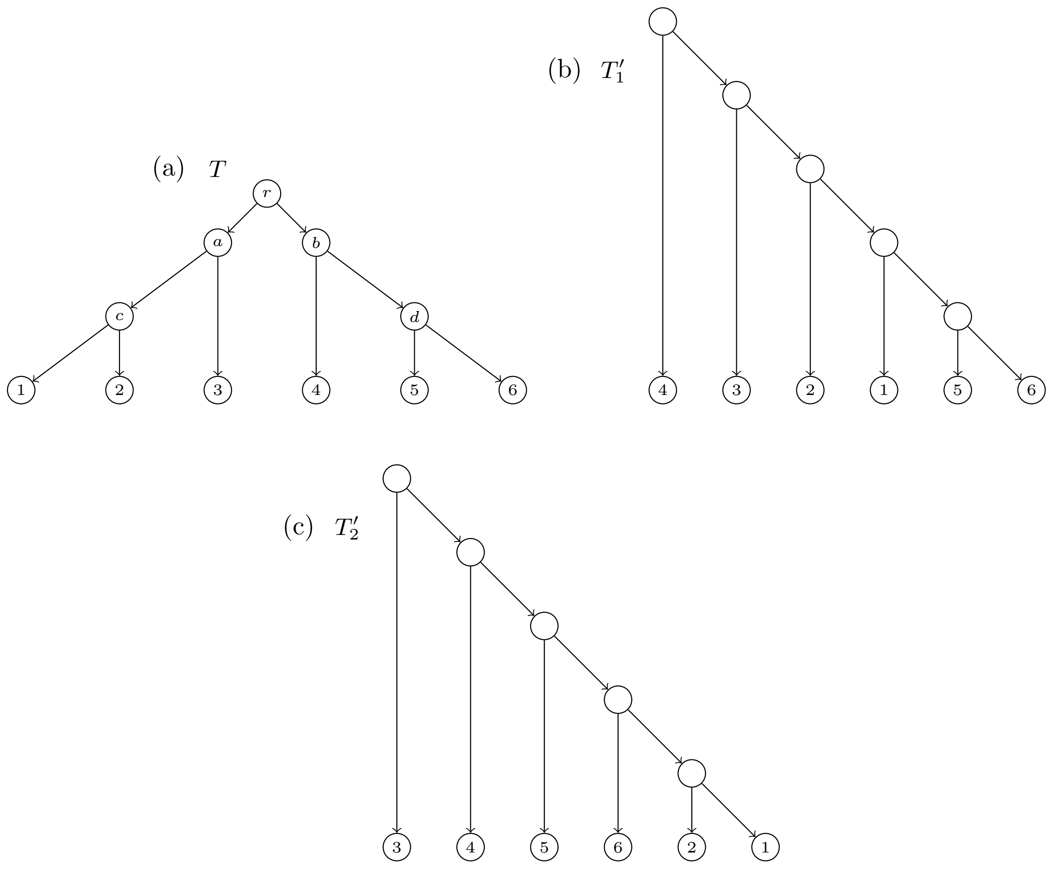

Figure 3.15: The phylogenetic trees used as input in Example 3.1.

Of course, it is possible that, on a given input, the LGT NETWORK RECONSTRUCTION PROBLEM has a solution and the RESTRICTED LGT NETWORK RECONSTRUCTION PROBLEM does not, as the following example shows.

Example 3.3. *Consider the trees T, T_1' depicted in Figure 3.17. Then, $C(T) \setminus C(T_1') = \{\{3,4,5\},\{2,3,4,5\}\}$ and $C(T_1') \setminus C(T) = \{\{2,3\},\{2,3,6\}\}$, and therefore these trees do not satisfy the principal-secondary condition on clusters: from $C(T) \setminus C(T_1')$ we have that $k = 2$, and from $C(T_1') \setminus C(T)$ that $l = 2$, but then both differences should consist of a pair of segments, instead of a single segment. This means that there does not exist any restricted LGT network with reduced principal subtree T and reduced secondary subtree T_1'. But there actually exists an LGT network with reduced principal subtree T and reduced secondary subtree T_1': the network N depicted in the same figure, which is not restricted because condition (c) in the definition of restricted LGT networks fails.*

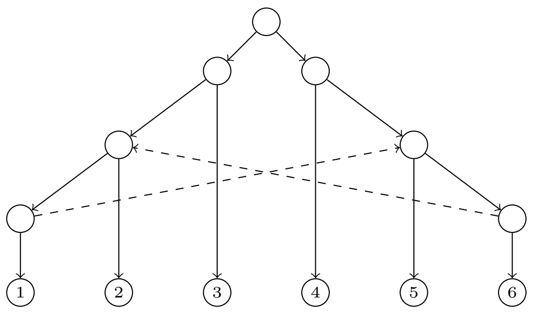

Figure 3.16: The graph obtained as output when Algorithm 2 to the trees T, T_1', T_2' in Figure 3.15.

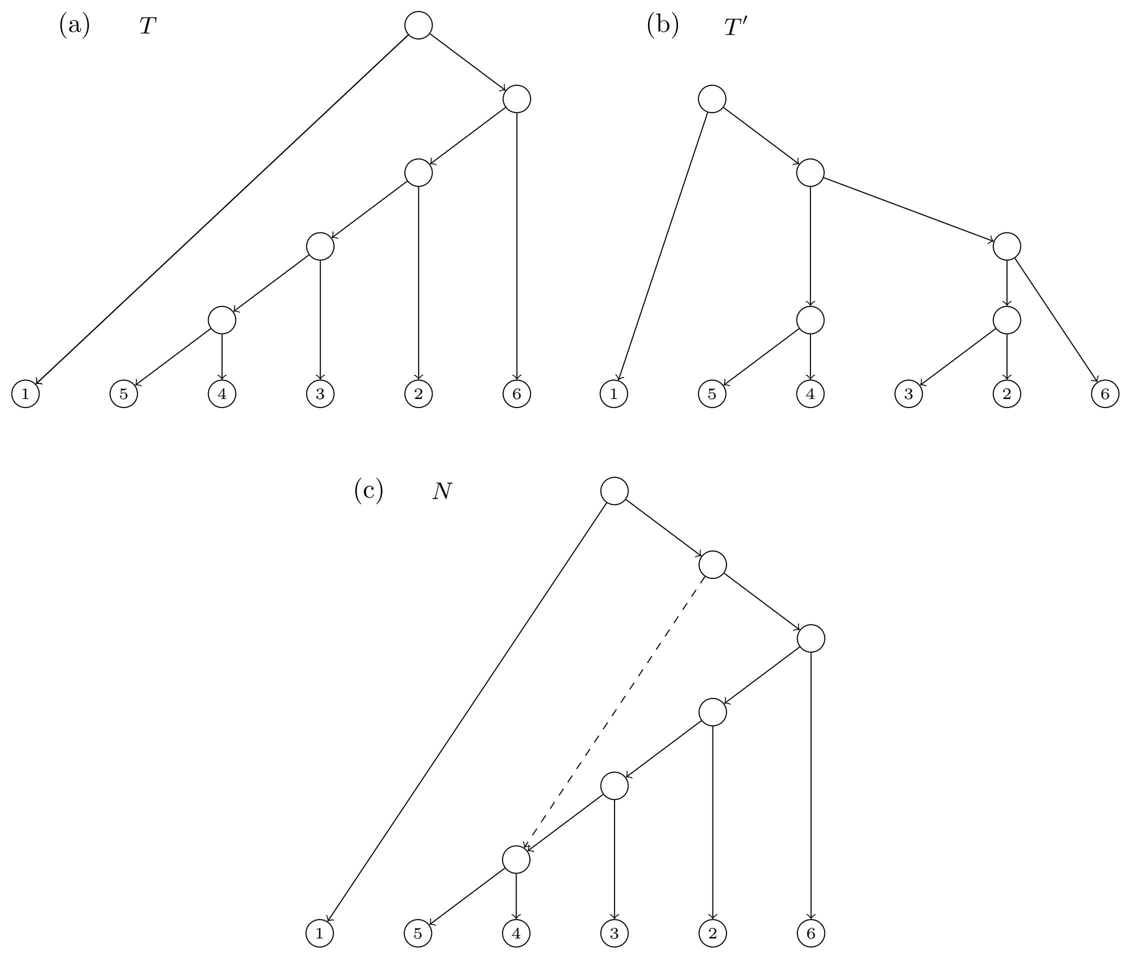

Figure 3.17: The phylogenetic trees used as input in Example 3.3, and an LGT network (c) that has them as reduced principal (a) and secondary (b) subtrees, respectively.

3.6 Computational experiments

In this section we report on two different experiments we have performed in order to test our results and algorithms.

An application using real biological data

In the first experiment, our goal is to find a set of trees based on true biological data and where our algorithms can be applied. First, we want to stress that our research is more fundamental than applied and hence it has been difficult to find such input data; we hope that further research on this topic will allow us to treat more generic input data. More precisely, the condition that secondary subtrees are at distance one from the principal subtree can be inderstood as if a single gene transfer has taken place, which is a rather restrictive condition. Hence, our strategy was to start with a database with many trees on many species and try to find a small subset of trees and species such that the subtrees of the chosen trees restricted to the chosen set of species meet the required condition. In particular, even when the networks we have found are consistent with LGTs reported in the literature, we do not claim that our results are biologically significant.

The general strategy for this search is as follows: We first choose a database with many phylogenetic trees; among these trees we exhaustively search a "central" tree sharing many leaves with a large set of "companion" trees in the database. Then, we exhaustively look for pairs formed by a subtree of this central tree and a companion tree such that their topological restrictions to their common set of leaves satisfy the principal-secondary condition on clusters. With all pairs satisfying this condition we look for a maximal example: with as many leaves as possible and as many secondary trees as possible. Finally, this maximal set of trees is used as an input to Algorithm 2.

We have taken as our datasource the database of phylogenetic trees in Beiko et al. (2011). That database contains 159.905 phylogenetic trees, but in order to make the computations feasible we have restricted our experiment to a random sample of 15.000 trees. Within this sample, we have found a "central" tree T with 100 leaves and 200 other "companion" trees sharing at least 30 labels with T. We have then kept these 201 trees and discarded the others. More precisely, the selection of the central and companion trees from the set of 15.000 trees has been done in the following way. We extract from the set the tree that shares the largest number of taxa with the largest number of trees possible in the set. The obtained tree has 100 leaves. Given the dissimilarity between the taxa of the trees, a large number of trees do not share any taxa with the central tree. Next, to maximize the possible similarity between the studied trees, we take the 200 trees that share more than 30 taxa with the pivot tree from the data set of 15.000 trees.

Next, for each subtree T_0 of T with at least 4 leaves and for each tree T' in the remaining set of 200 trees, we have computed the topological restriction of both T_0 and T' to their common set of leaves and checked whether they satisfy the principal-secondary condition on clusters. With this search, we have found the subtree T_0 of T described by the Newick string

$$(((((9,8),7),6),5),((4,3),(1,2)));$$

where the numbers correspond to the organisms given in Table 3.1, and the following three subtrees of some of the remaining set of 200 trees:

T'_1: (((((9,8),7),6),5),(((2,3),1),4));
T'_2: (((((9,8),7),6),5),(((1,3),2),4));
T'_3: ((((((9,8),7),6),5),4),((3,(1,2))));

such that each pair of trees (T_0, T'_i), $i = 1, 2, 3$, satisfies the conditions in Proposition 3.8. Applying Algorithm 2 to T_0, T'_1, T'_2, T'_3, we obtain the restricted LGT network depicted in Figure 3.18, that contains T_0 as reduced principal subtree and T'_1, T'_2, T'_3 as reduced secondary subtrees. This network suggests the existence of three lateral gene transfer events that explain the differences between T_0 and T'_1, T'_2, T'_3. Although there is no reference in the literature to these specific events, several lateral gene transfer events involving *Rhodobacter sp.*, *Ruegeria pom.* and *Ruegeria sp.* have been reported in the literature [Frank et al. (2005); Poggio et al. (2007); Todd et al. (2012)].

An application using simulated data

In this second experiment we want to test if we can take an LGT network and recover it from simulated DNA strands at its leaves. More precisely, we proceed as follows: We fix an LGT network and we introduce a statistical model of evolution of DNA sequences associated to its nodes that takes into account both mutations and lateral gene transfers;

Identifier	Organism
1	`Roseobacter_denitrificans_OCh_114`
2	`Ruegeria_pomeroyi_DSS-3`
3	`Ruegeria_sp._TM1040`
4	`Dinoroseobacter_shibae_DFL_12`
5	`Paracoccus_denitrificans_PD1222`
6	`Rhodobacter_sphaeroides_ATCC_17025`
7	`Rhodobacter_sphaeroides_KD131`
8	`Rhodobacter_sphaeroides_ATCC_17029`
9	`Rhodobacter_sphaeroides_2.4.1`

Table 3.1: The organisms involved in the phylogenetic trees T_0, T_1', T_2', T_3' given in the first experiment in Section 3.6.

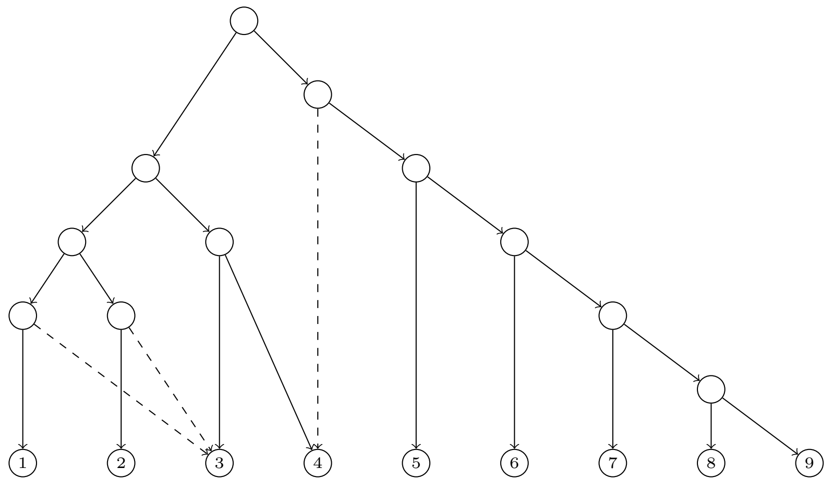

Figure 3.18: Restricted LGT network obtained in the first experiment in Section 3.6.

then, from different observed DNA sequences at the leaves, we reconstruct the phylogenetic trees that best illustrate these sequences; finally, we analyze these inferred trees in order to recover the original network with our reconstruction algorithm.

More precisely, we have considered the LGT network N depicted in Figure 3.19. For each node, we have simulated 1000 sequences of DNA, each of them of length 100 using the following model:

- The sequence at the root is fixed, but arbitrary. Given the symmetry of the considered evolutionary model, the selection of the initial sequence is not relevant, and we take the one where all nucleotides are equal to A.

- We associate to each arc a substitution matrix, which gives the probability of mutation of a nucleotide through the considered arc. For simplicity, we use the same matrix for each arc, corresponding to the Jukes-Cantor model [Jukes and Cantor (1969)] with the same α parameter:

$$M = \begin{pmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \alpha & 1-3\alpha & \alpha & \alpha \\ \alpha & \alpha & 1-3\alpha & \alpha \\ \alpha & \alpha & \alpha & 1-3\alpha \end{pmatrix}$$

We have run the experiment using different values for α. These values are summarized in Table 3.2.

- For each reticulation node h, we consider that, with a given probability p_h, the DNA subsequence of h is inherited from its secondary parent (modeling an LGT event) and with probability $1 - p_h$ it is inherited from its principal parent. For simplicity, we have assumed that all reticulation nodes have the same p_h and made the same experiment with different values, that are summarized in Table 3.2.

Note that the DNA sequence at the root of the network evolves through principal and secondary arcs simulating mutations and lateral gene transfer events until it arrives at the leaves. Each of the 1000 sequences at the leaves has been used as input of the `dnapars` program, from the `PHYLIP` package [Plotree and Plotgram (1989)] in order to infer the most parsimonious phylogenetic tree. As a result, different inferred topologies are obtained. Some of these trees are inferred much more times than other ones. This difference allows us to predict which are the candidates to be the reduced principal subtree and the reduced secondary subtrees of N, respectively. Figure 3.20 shows the reduced principal subtree $\widetilde{T}_0(N)$ and the two reduced secondary subtrees $\widetilde{T}_{e_1}(N)$ and $\widetilde{T}_{e_2}(N)$ of the network N.

Table 3.2 shows a summary of the results we have obtained. The columns labelled by $\#T_0$, $\#T_{e_1}$ and $\#T_{e_2}$ give the number of appearances in the simulation of the reduced principal subtree and of both of the reduced secondary subtrees, respectively. The columns labeled $\#T_{next}$ and T_{next} give, among all the inferred trees different from the aforementioned ones, how many times did the most frequent one appear and which is its Newick string.

Analyzing the results, we can observe that the reduced principal subtree is always the most inferred one, for any of the choices of the parameters α and p_h. This number ranges from 550 to 850, approximately, and the largest values correspond to lower values of p_h, as expected. Indeed, a lower value for p_h means that each DNA sequence in a node mainly comes from its principal parent. Similarly, the two reduced secondary subtrees are in the second and third position of the most inferred trees, except for the case where $\alpha = 0.01$ and $p_h = 0.05$. The reason for this erroneous behaviour is twofold: First, small value of p_h make that the secondary trees we want to find are less likely to be found , since rarely the simulated flow of DNA will pass through Furthermore, in most cases, the fourth tree in the list of inferred trees T_{next} appears less than a half of times than the third one. Thereby we can differentiate clearly which of the inferred trees corresponds to the reduced principal subtree, which correspond to the reduced secondary subtrees and which can be considered as spurious trees. The number of these spurious trees (trees which are different from T_0, T_{e_1} and T_{e_2}) range from $100 - 170$ (when $\alpha = 0.02, 0.03, 0.05$) to around 330 (for $\alpha = 0.01$). From this identification of principal and secondary trees, and applying our algorithm we can recover the original LGT network.

Finally, we remark that when we take high values of p_h, the fourth most frequently inferred tree is that with Newick string $(((2, 3), 6),((4, 5), 1))$. This tree is precisely the subtree obtained by switching-on the two secondary arcs, which could be expected since a large number of DNA sequences have been inherited from the secondary parents of both hybrid nodes instead of from the principal ones.

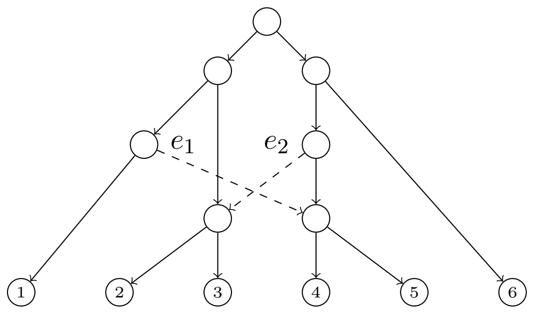

Figure 3.19: The LGT network N used in the second experiment in Section 3.6.

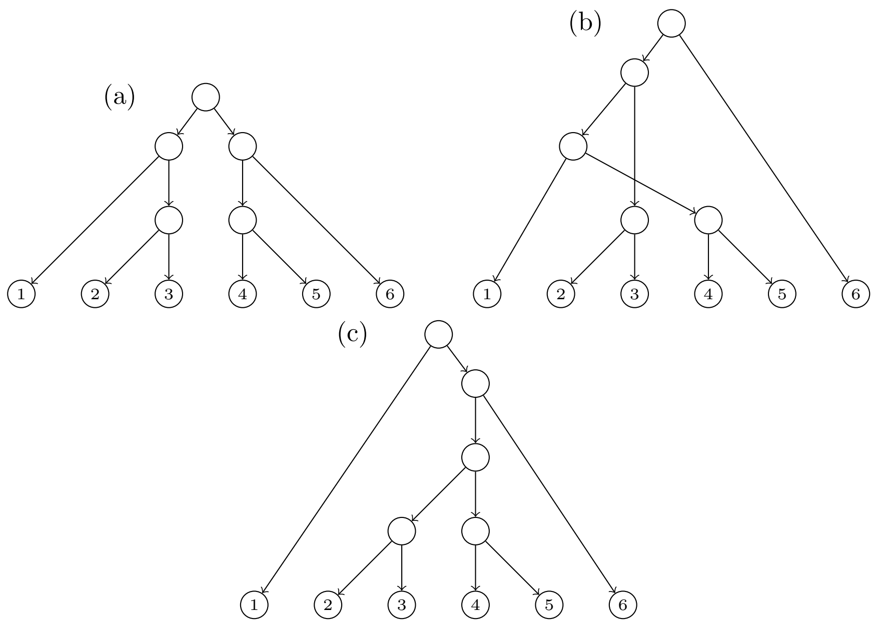

Figure 3.20: The reduced principal subtree $T_0(N)$ (a), and the two reduced secondary subtrees $\widetilde{T}_{e_1}(N)$ (b) and $\widetilde{T}_{e_2}(N)$ (c) of the LGT network depicted in Figure 3.19.

3.7 Some technical proofs

Proof of Proposition 3.3

Let $e = (u, h), e' = (u', h') \in E_s$; to simplify the notations, we shall denote $T_0(N)$, $T_e(N)$ and $T_{e'}(N)$ by T_0, T_e and $T_{e'}$, respectively. We shall prove that $\Gamma(T_e) \neq \Gamma(T_{e'})$, which will imply, by Lemma 3.1, that $\widetilde{T}_e \neq \widetilde{T}_{e'}$. To do that, we shall distinguish three main cases, depending on the relationship between u and u' in T_0, and in each case (and its subcases, when necessary) we shall show the existence of three labels on which T_e and $T_{e'}$ define different triples.

A) Consider first the case when $u = u'$. By condition (c) in Definition 3.1, neither h nor h' are principal descendants of u, and therefore $C_{T_0}(u) \cap C_{T_0}(h) = C_{T_0}(u) \cap C_{T_0}(h') = \emptyset$. This implies the existence of a leaf $x_3 \in C_{T_0}(u)$ that does not belong to $C_{T_0}(h) \cup C_{T_0}(h')$.

Since there cannot exist simultaneously two principal paths $h \rightsquigarrow h'$ and $h' \rightsquigarrow h$, we shall assume without any loss of generality that the latter, $h' \rightsquigarrow h$, does not exist. If there exists a principal path $h \rightsquigarrow h'$, then, since h is not principally elementary by condition (b) in Definition 3.1, it has a principal child v outside this principal path and then any leaf

Table 3.2: Simulation study of LGT network depicted in Figure 3.19.

Parameters	$\#T_0$	$\#T_{e_1}$	$\#T_{e_2}$	$\#T_{next}$	T_{next}
$\alpha = 0.01$, $p_h = 0.05$	707	42	52	56	(((4,5),6),(2,3),1);
$\alpha = 0.01$, $p_h = 0.10$	672	62	63	44	(((2,3),1),(4,5),6);
$\alpha = 0.01$, $p_h = 0.15$	571	98	106	48	(((2,3),1),(4,5),6);
$\alpha = 0.01$, $p_h = 0.20$	527	147	115	39	(((4,5),6),(1,2,3));
$\alpha = 0.02$, $p_h = 0.05$	874	49	37	27	(((1,3),2),((4,5),6));
$\alpha = 0.02$, $p_h = 0.10$	786	84	93	19	((((2,3),1),(4,5)),6);
$\alpha = 0.02$, $p_h = 0.15$	684	129	127	29	(((2,3),6),((4,5),1));
$\alpha = 0.02$, $p_h = 0.20$	629	173	131	39	(((2,3),6),((4,5),1));
$\alpha = 0.03$, $p_h = 0.05$	893	41	43	22	((((2,3),1),(4,5)),6);
$\alpha = 0.03$, $p_h = 0.10$	800	91	80	15	(((1,2),3),((4,5),6));
$\alpha = 0.03$, $p_h = 0.15$	689	140	132	22	((((2,3),1),(4,5)),6);
$\alpha = 0.03$, $p_h = 0.20$	627	149	169	40	(((2,3),6),((4,5),1));
$\alpha = 0.05$, $p_h = 0.05$	855	56	50	34	((((2,3),1),(4,5)),6);
$\alpha = 0.05$, $p_h = 0.10$	761	101	89	29	((((2,3),1),(4,5)),6);
$\alpha = 0.05$, $p_h = 0.15$	690	122	120	34	((((2,3),1),(4,5)),6);
$\alpha = 0.05$, $p_h = 0.20$	610	162	140	49	(((2,3),6),((4,5),1));

$x_1 \in C_{T_0}(v)$ belongs to $C_{T_0}(h)$ but not to $C_{T_0}(h')$. If, on the contrary, no principal path connects h with h', then $C_{T_0}(h) \cap C_{T_0}(h') = \emptyset$ and no $x_1 \in C_{T_0}(h)$ belongs to $C_{T_0}(h')$. So, in both cases, there exists some leaf $x_1 \in C_{T_0}(h) \setminus C_{T_0}(h')$. Take, finally, $x_2 \in C_{T_0}(h')$; see Figure 3.21. Notice that $x_1, x_2 \notin C_{T_0}(u)$.

Now, on the one hand, in $T_{e'}$ we have that $u = LCA_{T_{e'}}(x_2, x_3)$ and $x_1 \notin C_{T_{e'}}(u) = C_{T_0}(h') \cup C_{T_0}(u)$. This implies that $T_{e'}$ defines the triple $((x_2, x_3), x_1)$. On the other hand, in T_e, the principal path $u \rightsquigarrow x_3$ survives because h is not a principal descendant of u, and therefore $u = LCA_{T_{e'}}(x_1, x_3)$, and moreover, since h' is nor a principal descendant of u, there cannot be any path in T_e from an intermediate node in the principal path $u \rightsquigarrow x_3$ to x_2. This makes it impossible that T_e defines the triple $((x_2, x_3), x_1)$.

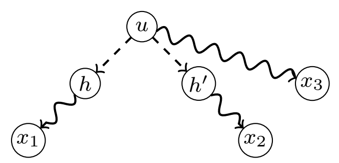

Figure 3.21: The structure of N involving e and e' in case (A) in the proof of Prop. 3.3.

B) Consider now the case when u and u' are connected by a proper principal path, say $u \rightsquigarrow u'$. By condition (c) in Definition 3.1, there do not exist principal paths connecting u with h or u' with either h' or h.

Assume first that the principal path $u \rightsquigarrow u'$ is not principally elementary. Let v be the last node in this path that is not principally elementary (v can be u, if every intermediate node in the path $u \rightsquigarrow u'$ is principally elementary), and let v' be any principal child of v outside this path $u \rightsquigarrow u'$. If h' is not a principal descendant of v', take $x_3 \in C_{T_0}(v')$. If, on the contrary, h' is a principal descendant of v', then, by condition (d) in Definition 3.1, the principal path $v \to v' \rightsquigarrow h'$ must contain some intermediate node w with a principal child w' outside this path; in this case, take $x_3 \in C_{T_0}(w')$. In this way, we always obtain a leaf $x_3 \in C_{T_0}(v) \setminus (C_{T_0}(u') \cup C_{T_0}(h'))$. Let, moreover, $x_1 \in C_{T_0}(u')$ and $x_2 \in C_{T_0}(h')$; notice that $x_1 \neq x_2$ because $C_{T_0}(u') \cap C_{T_0}(h') = \emptyset$. The situation is summarized in Figure

3.22.(a).

Since h' does not belong to the principal paths $v \rightsquigarrow x_3$ or $v \rightsquigarrow u' \rightsquigarrow x_1$, it is clear that $T_{e'}$ defines the triple $((x_1, x_2), x_3)$. Let us prove now that T_e cannot define this triple. Indeed, notice that, since h is not a principal descendant of u, it does not belong to the principal paths $v \rightsquigarrow x_3$ or $v \rightsquigarrow u' \rightsquigarrow x_1$. This implies that these paths survive in T_e and hence that $LCA_{T_e}(x_1, x_3) = v$. Therefore, should T_e define the triple $((x_1, x_2), x_3)$, this would imply that T_e contains some path $u' \rightsquigarrow x_2$ (recall that every intermediate node in the principal path $v \rightsquigarrow u'$ is principally elementary). But since u cannot be a descendant of u', this path could not contain the secondary arc $e = (u, h)$ and therefore it would be principal, implying the existence of a principal path connecting u' and h', which does not exist. This leads to a contradiction, showing that, as we claimed, T_e does not define the triple $((x_1, x_2), x_3)$.

Assume now that the principal path $u \rightsquigarrow u'$ is principally elementary. Since T_0 cannot contain two consecutive elementary nodes, this implies that u' is not principally elementary (and that $(u, u') \in E_p$). Let u'_1, u'_2 be two principal children of it and let $x_1 \in C_{T_0}(u'_1)$ and $x_2 \in C_{T_0}(u'_2)$, and let $x_3 \in C_{T_0}(h')$; see Figure 3.22.(b). Then $T_{e'}$ defines the triple (x_1, x_2, x_3). Now, since h is not a principal descendant of u, the bifurcating principal paths $u' \rightsquigarrow x_1$ and $u' \rightsquigarrow x_2$ survive in T_e and hence $LCA_{T_e}(x_1, x_2) = u'$, but h' is not a descendant of u' in T_e (because neither h' nor u are principal descendants of u') and therefore T_e defines the triple $((x_1, x_2), x_3)$.

In both cases, $\Gamma(T_e) \neq \Gamma(T_{e'})$.

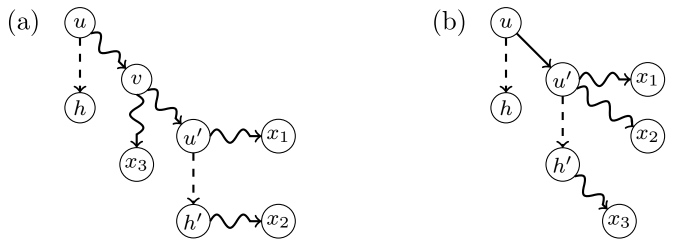

Figure 3.22: The two possible structures of N involving e and e' in case (B) in the proof of Prop. 3.3.

C) Assume finally that u and u' are different and not connected by any principal path, that is, $C_{T_0}(u) \cap C_{T_0}(u') = \emptyset$. Recall that, by condition (c) in Definition 3.1, $C_{T_0}(u) \cap C_{T_0}(h) = C_{T_0}(u') \cap C_{T_0}(h') = \emptyset$, too. We shall consider now several subcases, up to interchanging e and e'.

C.1) Assume that there exist principal paths $u \rightsquigarrow h'$ and $h \rightsquigarrow u'$. Then, since $C_{T_0}(h') \subseteq C_{T_0}(u)$ and $C_{T_0}(u) \cap C_{T_0}(h) = \emptyset$, we have that $C_{T_0}(h) \cap C_{T_0}(h') = \emptyset$. Moreover, since h is not principally elementary, there exists some leaf $x_1 \in C_{T_0}(h) \setminus C_{T_0}(u')$. Take any $x_2 \in C_{T_0}(h')$ and $x_3 \in C_{T_0}(u')$ (see Figure 3.23, where $v = LCA_{T_0}(u, h)$). It is easy to check that T_e defines the triple $((x_1, x_3), x_2)$ and $T_{e'}$ defines the triple $((x_2, x_3), x_1)$.

C.2) Assume that there exist principal paths $u \rightsquigarrow h'$ and $u' \rightsquigarrow h$. As in (C.1), this implies that $C_{T_0}(h) \cap C_{T_0}(h') = \emptyset$. Let $v = LCA_{T_0}(u, u')$. By condition (d) in Definition 3.1, the principal path $v \rightsquigarrow u \rightsquigarrow h'$ contains some intermediate node w with a principal child w_1 outside this path, and the principal path $v \rightsquigarrow u' \rightsquigarrow h$ contains some intermediate node w' with a principal child w'_1 outside this path.

Assume first that, up to interchanging u and u', the node w belongs the path $u \rightsquigarrow h'$ (this

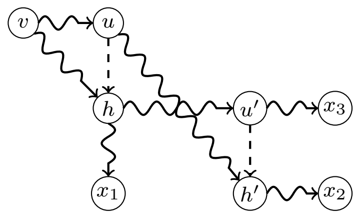

Figure 3.23: The structure of N involving e and e' in case (C.1) in the proof of Prop. 3.3.

includes the case $w = u$). Let $x_1 \in C_{T_0}(w_1)$, $x_2 \in C_{T_0}(h)$ and $x_3 \in C_{T_0}(h')$; see Figure 3.24.(a) (where w can be u; we have not distinguished this possibility in the figure). In this case, T_e defines the triple $((x_1, x_3), x_2)$ and $T_{e'}$ defines the triple $((x_2, x_3), x_1)$.

Assume now that both principal paths $u \rightsquigarrow h'$ and $u' \rightsquigarrow h$ are principally elementary; that is, principal arcs with u and u' principally elementary. In this case, w is intermediate in the principal path $v \rightsquigarrow u$ and w' is intermediate in the principal path $v \rightsquigarrow u'$. Let $x_1 \in C_{T_0}(w_1)$, $x_2 \in C_{T_0}(w'_1)$ and $x_3 \in C_{T_0}(h)$; see Figure 3.24.(b). Then, T_e defines the triple $((x_1, x_3), x_2)$ and $T_{e'}$ defines the triple $((x_2, x_3), x_1)$.

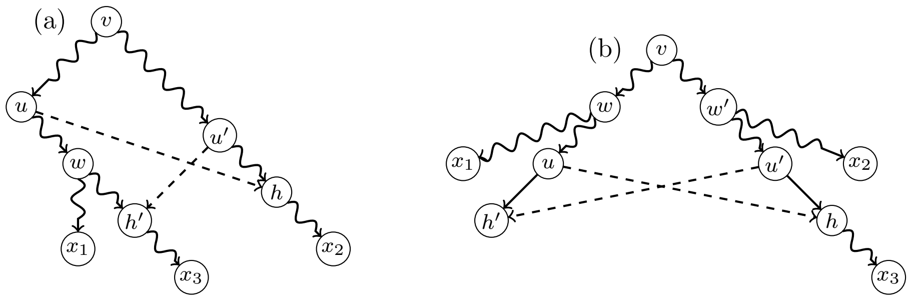

Figure 3.24: The structure of N involving e and e' in case (C.2) in the proof of Prop. 3.3.

C.3) Assume that there exist a principal path $u \rightsquigarrow h'$ but no principal path connecting u' and h. Since u and h are not connected by any principal path, neither are h and h'. Let $v = LCA_{T_0}(u, u')$, $x_1 \in C_{T_0}(h)$, $x_2 \in C_{T_0}(h')$ and $x_3 \in C_{T_0}(u')$ (see Figure 3.25). In $T_{e'}$, we have that $u' = LCA_{T_{e'}}(x_2, x_3)$ and, since there exists no principal path connecting h with u' or h', $x_1 \notin C_{T_{e'}}(u')$. This implies that $T_{e'}$ defines the triple $((x_2, x_3), x_1)$. Now, again because h is not connected in T_0 with u' or h', the principal paths $u \rightsquigarrow h' \rightsquigarrow x_2$ and $v \rightsquigarrow u' \rightsquigarrow x_3$ survive in T_e, and hence this tree defines the triple $((x_1, x_2), x_3)$.

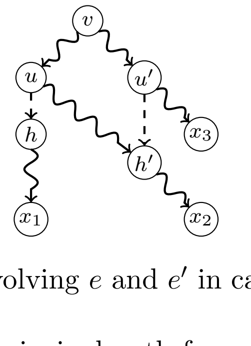

Figure 3.25: The structure of N involving e and e' in case (C.3) in the proof of Prop. 3.3.

C.4) Assume that there exists no principal path from $\{u, u'\}$ to $\{h, h'\}$, but there exists a principal path $h \rightsquigarrow u'$. Let $v = LCA_{T_0}(u, h) = LCA_{T_0}(u, u')$.

By condition (d) in Definition 3.1, the principal path $v \rightsquigarrow h$ contains some intermediate node w with a principal child w_1 outside this path. If h' is not a principal descendant of w_1, take $x_1 \in C_{T_0}(w_1)$. If h' is a principal descendant of w_1, then $w = LCA_{T_0}(u', h')$, and the path $w \to w_1 \rightsquigarrow h'$ contains some intermediate node w' with a principal child w'_1 outside this path: in this case, take $x_1 \in C_{T_0}(w'_1)$. In both cases, $x_1 \in C_{T_0}(w)$ and $x_1 \notin C_{T_0}(h) \cup C_{T_0}(h')$. Let moreover $x_2 \in C_{T_0}(u)$ and $x_3 \in C_{T_0}(u')$: see Figure 3.26.

It is clear then that T_e defines the triple $((x_2, x_3), x_1)$. Now, h' does not belong to the principal paths $v \rightsquigarrow h \rightsquigarrow u' \rightsquigarrow x_3$, $w \rightsquigarrow x_1$ or $v \rightsquigarrow u \rightsquigarrow x_2$ (as far as this last path goes, notice that u cannot be a principal descendant of h', because h' is a descendant of u in N, and that h' is not a principal descendant of u by assumption). Therefore, these principal paths survive in $T_{e'}$ and this tree defines the triple $((x_1, x_3), x_2)$.

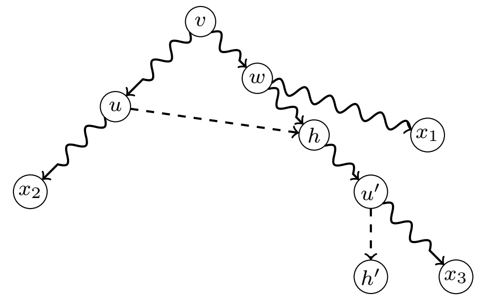

Figure 3.26: The structure of N involving e and e' in case (C.4) in the proof of Prop. 3.3.

C.5) Assume now that there exists no principal path connecting a node in $\{u, u'\}$ and a node in $\{h, h'\}$, but that h and h' are connected by a principal path, say $h \rightsquigarrow h'$ (this includes the case $h = h'$). Let $v = LCA_{T_0}(u, u')$, $x_1 \in C_{T_0}(u)$, $x_2 \in C_{T_0}(u')$, and $x_3 \in C_{T_0}(h')$; see Figure 3.27. Then, T_e defines the triple $((x_1, x_3), x_2)$ and $T_{e'}$ defines the triple $((x_2, x_3), x_1)$.

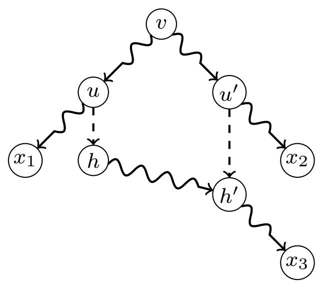

Figure 3.27: The structure of N involving e and e' in case (C.5) in the proof of Prop. 3.3. The path $h \rightsquigarrow h'$ need not be proper.

C.6) Assume finally that there exists no principal path connecting any pair of nodes $\{u, u', h, h'\}$. Let $v = LCA_{T_0}(u, u')$. We shall split this case into two subcases, up to symmetry.

Assume first that $w = LCA_{T_0}(u, h)$ is not a proper descendant of v: therefore, it is an ancestor of it. In this case, let $x_1 \in C_{T_0}(u)$, $x_2 \in C_{T_0}(h)$, and $x_3 \in C_{T_0}(u')$; see Figure 3.28.(a) (where w can be v; we have not distinguished this possibility in the figure). Since h does not belong to the principal path $v \rightsquigarrow u' \rightsquigarrow x_3$, T_e defines the triple $((x_1, x_2), x_3)$, and since h' does not belong to the principal paths $w \rightsquigarrow v \rightsquigarrow u \rightsquigarrow x_1$, $w \rightsquigarrow h \rightsquigarrow x_2$ and $w \rightsquigarrow v \rightsquigarrow u' \rightsquigarrow x_3$, they survive in $T_{e'}$ and then it defines either the triple $((x_1, x_3), x_2)$ (if $v \neq w$) or (x_1, x_2, x_3) (if $v = w$).

Assume now that $w = LCA_{T_0}(u, h)$ and $w' = LCA_{T_0}(u', h')$ are both proper descendants of v and therefore they are intermediate nodes in the principal paths $v \rightsquigarrow u$ and $v \rightsquigarrow u'$, respectively; in particular, $v = LCA_{T_0}(h, h')$. By condition (d) in Definition 3.1, the path

$w \rightsquigarrow h$ must contain some intermediate node w_0 with some principal child w_1 outside this path. Let $x_3 \in C_{T_0}(w_1)$, $x_1 \in C_{T_0}(u)$, and $x_2 \in C_{T_0}(h)$: see Figure 3.28.(b).

In this situation, T_e defines the triple $((x_1, x_2), x_3)$ and (since h' cannot belong to the principal path $w_1 \rightsquigarrow x_3$, because $LCA_{T_0}(h, h') = v$) $T_{e'}$ defines the same triple on x_1, x_2, x_3 as T_0, namely $((x_2, x_3), x_1)$.

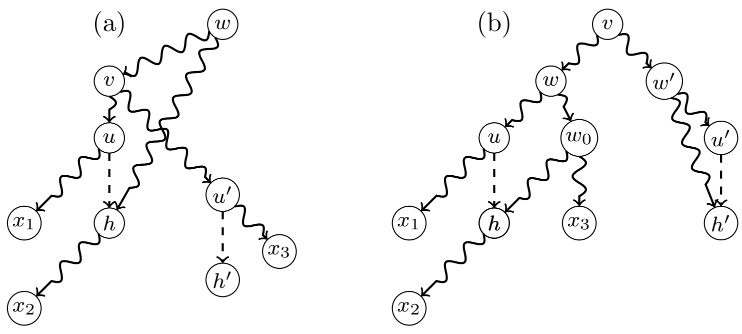

Figure 3.28: The structure of N involving e and e' in case (C.6) in the proof of Prop. 3.3.

So, $\Gamma(T_e) \neq \Gamma(T_{e'})$ in all subcases in which we have divided (C). This finishes the proof that T_e and $T_{e'}$ always define different sets of triples. □

Proof of the Claims in Proposition 3.5

We first establish an easy auxiliary lemma on triples calculus, which will allow us to avoid repeating the same argument several times:

Lemma 3.11. *Let T a phylogenetic tree on S, and let $x, y, z, t \in S$.*

(1) If $((x, y), z), ((x, t), z) \in \Gamma(T)$, then $((y, t), z) \in \Gamma(T)$.

(2) If $((x, y), z), ((z, t), x) \in \Gamma(T)$, then $((x, y), t) \in \Gamma(T)$.

(3) If $((x, y), z) \in \Gamma(T)$ and $((x, y), t) \notin \Gamma(T)$, then $((x, t), z) \in \Gamma(T)$.

Proof. Assertions (1) and (2) are proved in Corollary 9.3 Dress et al. (2012); although the trees considered therein are binary, it is easy to check that the proof is valid in the arbitrary setting. As far as (3) goes, if $((x, y), z) \in \Gamma(T)$, then $LCA_T(x, y)$ is a proper descendant of $LCA_T(x, z)$. Now, if moreover $((x, y), t) \notin \Gamma(T)$, then t is a descendant of $LCA_T(x, y)$ and hence $LCA_T(x, t)$ is a descendant of $LCA_T(x, y)$ and *a fortiori* a proper descendant of $LCA_T(x, z)$, which implies that $((x, t), z) \in \Gamma(T)$. □

Let us proceed with the proofs of the claims. Assume in the rest of this section that $\Gamma(T) \triangle \Gamma(T')$ consists of those triples described in the statement. To simplify the notations, set $A = \bigcup_{i=1}^{k} A_i$, $C_l = \bigcup_{i=1}^{m} C_{l,i}$ and $C = \bigcup_{i=1}^{l} C_i$.

Claim 1: $B \in C(T) \cap C(T')$.

We shall prove that, for every $b, b' \in B$ and $x \notin B$, $((b, b'), x) \in \Gamma(T) \cap \Gamma(T')$, which implies that $B \in C(T) \cap C(T')$. Since, by the explicit description of $\Gamma(T) \triangle \Gamma(T')$ given in the statement, $T_{b,b',x} = T'_{b,b',x}$, it is enough to prove that $((b, b'), x) \in \Gamma(T) \cup \Gamma(T')$. To do that, we consider three different cases, depending on x.

(1.1) If $x \in A$, let $c \in C$. Since $((b,c),x),((b',c),x) \in \Gamma(T')$, by Lemma 3.11.(1) we have that $((b,b'),x) \in \Gamma(T')$.

(1.2) If $x \in C$, let $a \in A$. Since $((a,b),x),((a,b'),x) \in \Gamma(T)$, again by Lemma 3.11.(1) we have that $((b,b'),x) \in \Gamma(T)$.

(1.3) If $x \notin A \cup B \cup C$, let $a \in A$ and $c \in C$. We shall assume that $((b,b'),x) \notin \Gamma(T) \cup \Gamma(T')$ and we shall reach a contradiction. Indeed, we know from (1.1) that $((b,b'),a) \in \Gamma(T')$. Then, if $((b,b'),x) \notin \Gamma(T')$, by Lemma 3.11.(3), we have that $((b,x),a) \in \Gamma(T')$, and since $((b,c),a) \in \Gamma(T')$, Lemma 3.11.(1) implies that $((x,c),a) \in \Gamma(T')$. On the other hand, we know from (1.2) that $((b,b'),c) \in \Gamma(T)$. Then, if $((b,b'),x) \notin \Gamma(T)$, by Lemma 3.11.(3) we have that $((b,x),c) \in \Gamma(T)$, and since $((a,b),c) \in \Gamma(T)$, by Lemma 3.11.(1) we have that $((a,x),c) \in \Gamma(T)$. But if $x \notin B$, $T_{a,c,x} = T'_{a,c,x}$, which leads to the announced contradiction.

This finishes the proof of Claim 1.

Claim 2. $C_{l,i} \in C(T) \cap C(T')$, for every $i = 1,\ldots,m$.

We shall prove that, for every $c,c' \in C_{l,i}$ and $x \notin C_{l,i}$, $((c,c'),x) \in \Gamma(T) \cap \Gamma(T')$. As in the previous Claim, it is enough to prove that $((c,c'),x) \in \Gamma(T) \cup \Gamma(T')$; and, also as in the previous claim, we consider different possibilities for x.

(2.1) If $x \in A \cup \bigcup\limits_{i<l} C_i$, let $b \in B$. Since $((b,c),x),((b,c'),x) \in \Gamma(T')$, by Lemma 3.11.(1) we have that $((c,c'),x) \in \Gamma(T')$.

(2.2) If $x \in B$, let $a \in A$. As we have just seen, $((c,c'),a) \in \Gamma(T')$ and therefore $((c,c'),a) \in \Gamma(T)$. Then, since $((a,x),c) \in \Gamma(T)$, by Lemma 3.11.(2) we have that $((c,c'),x) \in \Gamma(T)$.

(2.3) If $x \in C_{l,j}$ for some $j \neq i$, let $b \in B$. By (2.2), $((c,c'),b) \in \Gamma(T)$ and hence $((c,c'),b) \in \Gamma(T')$, too; moreover, by assumption, $(c,x,b),(c',x,b) \in \Gamma(T')$. Then, $LCA_{T'}(x,c) = LCA_{T'}(b,c) = LCA_{T'}(b,x) = LCA_{T'}(b,c') = LCA_{T'}(x,c')$ and it is a proper ancestor of $LCA_{T'}(c,c')$, which implies that $((c,c'),x) \in \Gamma(T')$.

(2.4) If $x \notin A \cup B \cup C$, let $a \in A$ and $b \in B$. We shall assume that $((c,c'),x) \notin \Gamma(T) \cup \Gamma(T')$ and we shall reach a contradiction. Indeed, by (2.1) we know that $((c,c'),a) \in \Gamma(T')$, and therefore, if $((c,c'),x) \notin \Gamma(T')$, Lemma 3.11.(3) implies that $((c,x),a) \in \Gamma(T')$. Moreover, $((b,c),a) \in \Gamma(T')$ and then, by Lemma 3.11.(1), $((b,x),a) \in \Gamma(T')$. Since T and T' define the same triples on $\{a,c,c'\},\{a,c,x\},\{a,b,x\}$, we have that $((c,c'),a),((c,x),a),((b,x),a) \in \Gamma(T)$. But $\Gamma(T)$ also contains $((a,b),c)$, and this is impossible, because these four triples are incompatible: the Aho graph they define on $\{a,b,c,c',x\}$ is connected [Aho et al. (1981)].

This finishes the proof of Claim 2.

Claim 3. $B \cup C_l \in C(T') \setminus C(T)$.

Since, for every $a \in A$, $b \in B$ and $c \in C_l$, $T_{a,b,c} = ((a,b),c)$, it is clear that $B \cup C_l$ is not a cluster of T. Now, to prove that $B \cup C_l \in C(T')$, since we already know that $B \in C(T')$, we shall prove that:

(i) For every $c,c' \in C_l$ and $x \notin B \cup C_l$, T' defines the triple $((c,c'),x)$.

(ii) For every $b \in B$, $c \in C_l$ and $x \notin B \cup C_l$, T' defines the triple $((b,c),x)$.

As far as (i) goes, we distinguish the following cases:

(3.1) If $x \in A \cup \bigcup_{i<l} C_i$, the same argument as in (2.1) proves that $((c,c'),x) \in \Gamma(T')$.

(3.2) If $x \notin A \cup B \cup C$, the same argument as in (2.4) proves that $((c,c'),x) \in \Gamma(T')$.

As far as (ii) goes, we distinguish the following cases:

(3.3) If $x \in A \cup \bigcup_{i<l} C_i$, we already know by assumption that $((b,c),x) \in \Gamma(T')$.

(3.4) If $x \notin A \cup B \cup C$, let $a \in A$; we shall assume that $((b,c),x) \notin \Gamma(T')$ and we shall reach a contradiction. Indeed, if this is the case, since $((b,c),a) \in \Gamma(T')$, by Lemma 3.11.(3) we obtain that $((b,x),a),((c,x),a) \in \Gamma(T')$. Since $T_{a,b,x} = T'_{a,b,x}$ and $T_{a,c,x} = T'_{a,c,x}$, this implies that $((b,x),a),((c,x),a) \in \Gamma(T)$, but then, by Lemma 3.11.(1), this implies that $((b,c),a) \in \Gamma(T)$, which contradicts the assumption that $((a,b),c) \in \Gamma(T)$.

This finishes the proof of Claim 3.

Claim 4. If $m > 1$, $C_l \in C(T) \setminus C(T')$.

C_l is not a cluster in T' when $m > 1$, because if $c \in C_{l,1}$, $c' \in C_{l,2}$ and $b \in B$, then $T'_{c,c',b} = (c,c',b)$ and therefore $((c,c'),b) \notin \Gamma(T')$. To prove that $C_l \in C(T)$, it is enough to check that, for every $c, c' \in C_l$ and $x \notin C_l$, $((c,c'),x) \in \Gamma(T)$. Now:

(4.1) If $x \in A \cup \bigcup_{i<l} C_i$, by (3.1) we know that $((c,c'),x) \in \Gamma(T')$, and, by assumption, in this case $T'_{c,c',x} = T_{c,c',x}$.

(4.2) If $x \in B$, the same argument as in (2.2) proves that $((c,c'),x) \in \Gamma(T)$.

(4.3) If $x \notin A \cup B \cup C$, by (3.2) we know that $((c,c'),x) \in \Gamma(T')$, and, by assumption, in this case $T'_{c,c',x} = T_{c,c',x}$, too.

This finishes the proof of Claim 4.

Claim 5. If $m > 1$, $B \cup C_{l,i_1} \cup \cdots \cup C_{l,i_k} \notin C(T')$ for every $\emptyset \neq \{i_1,\ldots,i_k\} \subsetneq \{1,\ldots,m\}$

Let $c \in \bigcup_{j=1}^{k} C_{l,i_j}$, $c' \in C_l \setminus \bigcup_{j=1}^{k} C_{l,i_j}$ and $b \in B$. Then, by assumption, $T'_{c,c',b} = (c,c',b)$ and therefore $((c,b),c') \notin \Gamma(T')$.

This finishes the proof of Proposition 3.5. □

Proof of Proposition 3.8

As far as the "only if" implication goes, assume that $e = (w,h)$ and let $T = \widetilde{T}_0(N)$ and $T' = \widetilde{T}_e(N)$. Consider the situation of the corresponding implication in Proposition 3.5, depicted in Figure 3.12: we shall use the same notations as therein.

- For every $i = 1, \ldots, k$, let $U_i = C_T(u_i)$.
- For every $i = 1, \ldots, k-1$, let $U'_i = C_{T'}(u_i)$; if, moreover, $\text{outdeg}_T(u_k) > 2$, let $U'_k = A_k$, which then belongs to $C(T') \setminus C(T)$.
- For every $i = 1, \ldots, l$, let $W'_i = C_{T'}(w_i)$ (and recall that $w_l = \tilde{w}$, the first non principally elementary principal descendant of w).
- For every $i = 1, \ldots, l-1$, let $W_i = C_T(w_i)$; if, moreover, $\tilde{w} = w$ (that is, if $\text{outdeg}_{T_0(N)}(w) \geq 2$), let $W_l = C_l$, which then belongs to $C(T) \setminus C(T')$.

It is straightforward to check that conditions (a) to (f) in the statement are satisfied.

In order to prove the "if" implication, assume that $C(T)$ and $C(T')$ satisfy conditions (a) to (f) in the statement. Let B, A_k, C_l be as defined in conditions (b)–(d), let $U'_k = A_k$ and $W_l = C_l$ even when $k_0 = k-1$ or $l_0 = l-1$ (the only difference is that, in these cases, they belong to $C(T) \cap C(T')$), and let

$$\begin{aligned} A_i &:= U_i \setminus U_{i+1} = U'_i \setminus U'_{i+1}, \text{ for every } i = 1, \ldots, k-1 \\ C_i &:= W_i \setminus W_{i+1} = W'_i \setminus W'_{i+1}, \text{ for every } i = 1, \ldots, l-1 \end{aligned}$$

It is easy to check, then, that

$$\begin{aligned} U_i &= A_i \sqcup \cdots \sqcup A_k \sqcup B \text{ for every } i = 1, \ldots, k \\ U'_i &= A_i \sqcup \cdots \sqcup A_k \text{ for every } i = 1, \ldots, k \\ W_i &= C_i \sqcup \cdots \sqcup C_l \text{ for every } i = 1, \ldots, l \\ W'_i &= C_i \sqcup \cdots \sqcup C_l \sqcup B \text{ for every } i = 1, \ldots, l \end{aligned}$$

so that, in particular, $U'_i = U_i \setminus B$, for every $i = 1, \ldots, k$, and $W_i = W'_i \setminus B$, for every $i = 1, \ldots, l$.

Let h' the node in T' with cluster B. If $l_0 = l$ (that is, if $C_l \notin C(T')$), let w' be the parent of h', let $x'_1, \ldots, x'_m$ be the other children of w' and let $C_{l,i} = C_{T'}(x'_i)$, for every $i = 1, \ldots, m$.

It turns out that, with these notations, the symmetric difference $\Gamma(T) \triangle \Gamma(T')$ consists exactly of those triples described in the statement of Proposition 3.5. To prove it, we shall describe explicitly the structures of T and T':

- Let h be the node in T with $C_T(h) = B$; recall that h' is the node in T' with $C_{T'}(h') = B$.
- For every $i = 1, \ldots, k$, let u_i be the node in T such that $U_i = C_T(u_i)$ and, for every $i = 1, \ldots, k-1$, let u'_i be the node in T' such that $U'_i = C_{T'}(u'_i)$. By (a), u_{i+1} is a child of u_i for every $i = 1, \ldots, k-1$, and u'_{i+1} is a child of u'_i for every $i = 1, \ldots, k-2$.
- Let u'_k be the node in T' such that $C_{T'}(u'_k) = U'_k = A_k$, which exists by (c), and assume that $k > 1$. Then, since $A_k \subsetneq U'_{k-1}$ by (e), u'_k is a descendant of u'_{k-1}. It turns out that u'_k is a child of u'_{k-1}.

  Indeed, if $k_0 = k$, then it is a direct consequence of (a). Assume now that $k_0 = k-1$, so that the cluster of every proper descendant of u'_{k-1} also belongs to $C(T)$. If u'_k is not a child of u'_{k-1}, the path $u'_{k-1} \rightsquigarrow u'_k$ contains an intermediate node $\bar{u}$ with $C_{T'}(\bar{u}) = A_k \sqcup \bar{A}$, with $\emptyset \neq \bar{A} \subsetneq A_{k-1}$. Then, $A_k \sqcup \bar{A} \in C(T)$ and $U_k \cap (A_k \sqcup \bar{A}) = (A_k \sqcup B) \cap (A_k \sqcup \bar{A}) \neq \emptyset$. This implies, by the compatibility of clusters in phylogenetic trees, that either $B \subseteq \bar{A}$ or $\bar{A} \subseteq B$ and hence that $\emptyset \neq B \cap A_{k-1}$, which is false. This contradiction implies that $\bar{u}$ cannot exist, and therefore that u'_k is a child of u'_{k-1}, as we claimed.

- For every $i = 1, \ldots, l-1$, let w_i be the node in T such that $W_i = C_T(w_i)$ and, for every $i = 1, \ldots, l$, let w'_i be the node in T' such that $W'_i = C_{T'}(w'_i)$. Again by (a), w_{i+1} is a child of w_i for every $i = 1, \ldots, l-2$, and w'_{i+1} is a child of w'_i for every $i = 1, \ldots, l-1$.

- Let w_l be the node in T such that $C_T(w_l) = W_l = C_l$, which exists by (d). A similar argument as the one used to prove that u'_k is a child of u'_{k-1} also proves that, if $l > 1$, w_l is a child of w_{l-1}.

- Let us prove now that h is a child of u_k. If $k_0 = k-1$, it is a direct consequence of the fact that $A_k \in C(T)$ by (c): if, in this case, a_k is the node in T such that $C_T(a_k) = A_k$, then the equality $U_k = A_k \sqcup B$ implies that a_k and h are the only children of u_k.

  Assume now that $k_0 = k$. Since $B \subsetneq U_k$, h is a proper descendant of u_k. Assume that the path $u_k \rightsquigarrow h$ contains some intermediate node $\bar{u}$, with $C_T(\bar{u}) = B \sqcup \bar{A}$ with $\emptyset \neq \bar{A} \subsetneq A_k$. By (a), the cluster of every proper descendant of u_k also belongs to $C(T')$, and therefore $B \sqcup \bar{A} \in C(T')$. But then, $W'_l = B \sqcup C_l \in C(T')$ by (d) and $(B \sqcup \bar{A}) \cap (B \sqcup C_l) \neq \emptyset$, which implies that $\bar{A} \subseteq C_l$ or $C_l \subseteq \bar{A}$ and hence $C_l \cap \bar{A} \neq \emptyset$, which is impossible because $C_l \cap \bar{A} \subseteq C_l \cap A_k \subseteq W'_1 \cap U'_1 = \emptyset$. This contradiction implies that $\bar{u}$ cannot exist, and hence that h is a child of u_k, as we claimed.

- A similar argument shows that h' is a child of w'_l. In particular, $w' = w'_l$.

- The equalities $U_1 \cap W_1 = W'_1 \cap U'_1 = \emptyset$ always hold. Indeed
  - if $k, l > 1$ or $k_0 = k = 1$ or $l_0 = l = 1$, these intersections are empty by (a);
  - if $k > 1$, $l = 1$ and $l_0 = l-1$, then $W'_1 \cap U'_1 = \emptyset$ by (a) and $U_1 \cap W_1 = (U'_1 \sqcup B) \cap C_1 = (U'_1 \sqcup B) \cap (W'_1 \setminus B) = \emptyset$;
  - if $l > 1$, $k = 1$ and $k_0 = k-1$, a symmetrical argument applies;
  - if $k = l = 1$, $k_0 = k-1$, then $C(T) \setminus C(T') = \{U_1\}$ with $U_1 = A_1 \sqcup B$ and $C(T') \setminus C(T) = \{W'_1\}$ with $W'_1 = C_1 \sqcup B$. Now, in this case, $U_1 \cap W_1 = W'_1 \cap U'_1 = A_1 \cap C_1 = \emptyset$, because, by (b), $(A_1 \sqcup B) \cap (C_1 \sqcup B) = U_1 \cap W'_1 = B$.

- $U_1 \cup W_1 = W'_1 \cup U'_1$: actually, in all cases,

$$U_1 \cup W_1 = A_1 \sqcup \cdots \sqcup A_k \sqcup B \sqcup C_1 \sqcup \cdots \sqcup C_l = W'_1 \sqcup U'_1.$$

- Let us check that, if $l > 1$ or $l_0 = l = 1$, the nodes u_1 and w_1 are sibling in T. To do that, recall that, by (a), every cluster in T strictly containing U_1 or W_1 is also a cluster in T'.

  Let v be any proper ancestor of u_1. Then, $C_T(v) = U_1 \sqcup X$ with $X \neq \emptyset$, and, thus, $U_1 \sqcup X \in C(T')$. Now, since $(U_1 \sqcup X) \cap U'_1 \neq \emptyset$ and $(U_1 \sqcup X) \cap W'_1 \neq \emptyset$ (because $U_1 = U'_1 \sqcup B$ and $B \subseteq W'_1$) and $U'_1 \cap W'_1 = \emptyset$, we conclude that $U'_1 \cup W'_1 \subseteq U_1 \sqcup X$ and in particular $W_1 \subseteq W'_1 \subseteq U_1 \sqcup X$. This entails that every proper ancestor of u_1 is also an ancestor of w_1.

  Let now v be any proper ancestor of w_1. Then, $C_T(v) = W_1 \sqcup X$ with $X \neq \emptyset$ and, hence, $W_1 \sqcup X \in C(T')$. Now, since $(W_1 \sqcup X) \cap W'_1 \neq \emptyset$ and $W'_1 = W_1 \sqcup B$, we conclude that $X \cap B \neq \emptyset$ and hence $(W_1 \sqcup X) \cap U_1 \neq \emptyset$. Since $W_1 \cap U_1 = \emptyset$, this implies that $U_1 \subseteq X$ and, hence, that u_1 is a descendant of v. Therefore, every proper ancestor of w_1 is also an ancestor of u_1. This finishes the proof that u_1 and v_1 are sibling.

- A similar argument shows that, if $k > 1$ or $k_0 = k = 1$, the nodes u'_1 and w'_1 are sibling in T'.

- Assume now that $l_0 = l$; as we have seen, the parent w' of h' is w'_l. In this case, by (d), $C_{T'}(w'_l) = W'_l = C_l \sqcup B$ and $C_l \notin C(T)$, which means that w'_l has more than one child other than h'. Consider the clusters $C_{l,1}, \dots, C_{l,m} \in C(T')$, with $m > 1$, of the children of w'_l other than h', so that $C_l = C_{l,1} \sqcup \cdots \sqcup C_{l,m}$. Since they are strictly contained in W'_l, by (a) they also belong to $C(T)$. Let $x_1, \dots, x_m$ be the nodes in T with $C_T(x_i) = C_{l,i}$, for $i = 1, \dots, m$. It turns out that each x_i is a child of w_l.

  Indeed, since $C_{l,i} \subsetneq C_l = W_l$, x_i is a proper descendant of w_l. If it is not its child, then the path $w_l \rightsquigarrow x_{l,i}$ contains an intermediate node $\bar{x}$ with cluster $C_{l,i} \sqcup X$, with $\emptyset \neq X \subsetneq C_l \setminus C_{l,i}$. Then, $C_{l,i} \sqcup X$ is also a cluster in T'. But in T' we have that x'_i, with cluster $C_{l,i}$, is a child of w'_l, with cluster $B \sqcup C_l$, and this leads to a contradiction, because $C_{l,i} \subsetneq C_{l,i} \sqcup X \subsetneq B \sqcup C_l$.

In summary, the structures of T and T' are those described in Figure 3.29. It is straightforward to check that the sets $A_1, \dots, A_k, B, C_1, \dots, C_{l-1}, C_{l,1}, \dots, C_{l,m}$ satisfy the conditions in Proposition 3.5. □

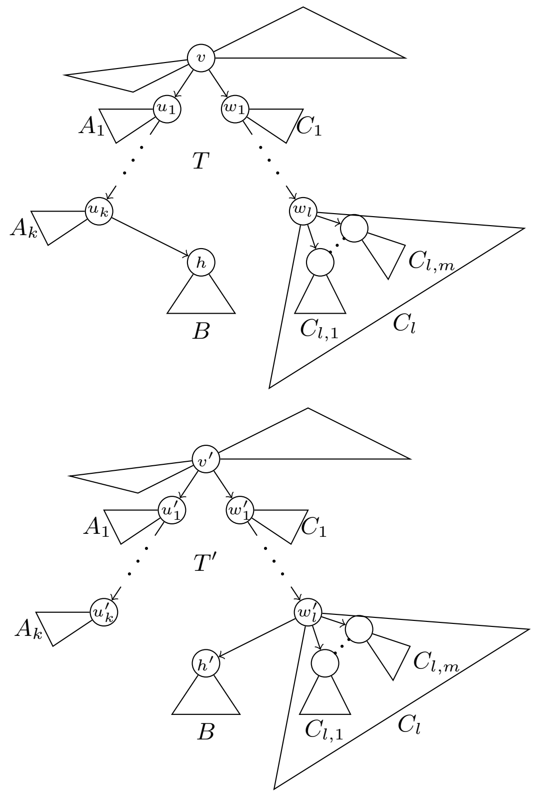

Figure 3.29: The structures of T and T' when they satisfy the conditions in Proposition 3.8.

In this chapter we have seen that the LGT network reconstruction problem using as input data the set of reduced principal and secondary subtrees can not be solved, with unicity, for generic LGT networks. Then, we have focused on the subclass of restricted LGT networks, where the reconstruction problem has been solved by providing an algorithm which has also allowed us to make some computational experiments. In the next chapter, we still consider the LGT network reconstruction problem, but we focus on a different input data for the problem, using an adaptation of trinets to LGT networks.

Chapter 4

A reconstruction problem for LGT networks based on tri-lgt-nets

Contents

4.1 Introduction

One of the strategies used to reconstruct rooted phylogenetic trees is by joining substructures such as triples, clusters and other trees. For the inference of phylogenetic networks, similar methods are also applied, see Section 1.9. However, even though such substructures allow us to reconstruct with unicity phylogenetic trees, the reconstruction is not unique in the case of networks: different phylogenetic networks, representing different evolutionary histories, can have the same set of triples, clusters or trees [Gambette and Huber (2012); Willson (2011)]. This is true also for the full set of displayed subnetworks [Huber et al. (2014)]. Because of that, an important point to take into account is which topological constraints have to be imposed in order to be able to reconstruct networks in an unique way.

In an analogous way that one can reconstruct phylgenetic trees using triples, certain subclass of phylogenetic networks can be reconstructed using *trinets*, which are three-leaved networks introduced by Huber and Moulton (2013). For instance, Huber and Moulton (2013) uses trinets to reconstruct level-1 networks. Similarly, van Iersel and Moulton (2014) reconstruct via trinets both binary tree-child networks and level-2 phylogenetic networks. Also, in Oldman et al. (2016), the authors introduce the `TriLoNet` algorithm to construct level-1 networks from level-1 trinets. Finally, Huber et al. (2015b) demonstrates that given an arbitrary set of trinets, the problem of deciding whether or not there exists a level-1 network displaying trinets is a NP problem. However, the authors give a polynomial tractable algorithm for some specific cases.

It is worth to mention that in van Iersel and Moulton (2014), the authors point out that extending their results to more general networks is not straightforward. For instance, passing from level-2 networks to arbitrary level-k networks, or from tree-child networks to reticulation-visible networks, would require completely new techniques.

In this chapter we investigate the reconstruction of phylogenetic networks using trinets in the specific case of LGT networks. We adapt the concept of trinets to LGT networks, in what we call *tri-lgt networks* or *tri-lgt-nets* for short, LGT networks with three leaves and with at most one secondary arc. As we have mentioned earlier, we must impose some constraints so that trinets can encode phylogenetic networks. We will consider a subclass of LGT networks, characterized by having a binary reduced principal subtree and such that every secondary arc has its source on an elementary node of the principal subtree and its target in a non-elementary node. We call this networks *arc-node LGT networks.* We focus on the decomposition of this subclass of networks using tri-lgt-nets. First, we show how some secondary arcs are irrelevant, in the sense that the tri-lgt-nets that they induce are already induced by some other arc. Finally, by imposing temporal consistency, we reach the conclusion that tri-lgt-nets encode binary arc-node LGT networks as long as the latter do not present redundant arcs.

4.2 Decomposition of a binary arc-node LGT network

In this section we introduce a subclass of LGT networks suitable for the problem of reconstructing networks from the set of tri-lgt-nets that are represented in them.

Definition 4.1. *An* arc-node *LGT network is an LGT network satisfying the following conditions:*

(a) No principal child of a principally elementary node is principally elementary.

(b) For each secondary arc (u, h), the node u is principally elementary and h is not.

(c) If (u, h) is a secondary arc, then there exists no principal path from u to h.

(d) Its reduced principal subtree is binary.

Note that some conditions imposed on arc-node LGT networks, namely (a) and (c), are common to restricted LGT networks, introduced in Section 3.5. Condition (b) here is strengthened by imposing conditions not only on the target node of secondary arcs but also on their source. The binarity condition imposed here is not present in restricted networks. However, as in restricted LGT networks, any arc-node LGT network can be obtained by the operation $T + F$ defined in Section 3.5 where T is a binary phylogenetic tree and $F \subset E \times V$ is a set whose elements are pairs formed by an arc and a node of the tree. Particularly, we can compare (principal and secondary) arcs in different arc-node LGT networks if both networks have the same reduced principal subtree.

See Figure 4.1 for an example of an arc-node LGT network obtained through the operation $T + F$ described above.

Through the rest of this chapter, whenever we talk about LGT networks, we will mean arc-node LGT networks.

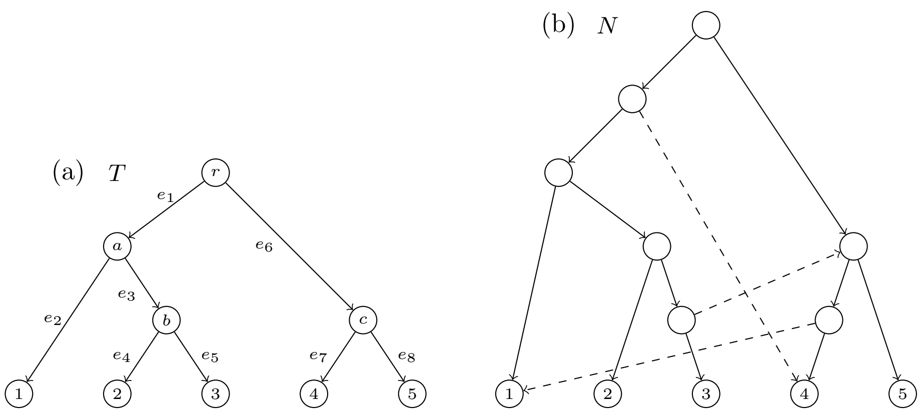

Figure 4.1: An arc-node LGT network N (b) obtained by the operation $N = T + F$ where T is the tree depicted in (a), which is the reduced principal subtree of N, and F is the set of pairs $\{(e_1, 4), (e_5, c), (e_7, 1)\}$.

Basic tri-lgt networks

We call *tri-lgt network*, or *tri-lgt-net* for short, an LGT network with exactly three leaves. There are exactly six non-isomorphic (up to permutations of taxa) tri-lgt-nets with at most one secondary arc. We call this six tri-lgt-nets the *basic* tri-lgt-nets. Given a set of taxa $S = \{x, y, z\}$, there is only one binary rooted tree defined on S (up to permutations of x, y, z), defined by the Newick string $(x, (y, z))$;. We denote this basic tri-lgt-net as $B^0(x, y, z)$. The remaining basic tri-lgt-nets are obtained by adding a secondary arc to $B^0(x, y, z)$ maintaining compliance with restrictions (a)–(d) of Definition 4.1. From condition (b), the extremes of the added secondary arc on $B^0(x, y, z)$ must satisfy that (1) its source is a new node splitting an arc of $B^0(x, y, z)$ and (2) its target is an existing node of $B^0(x, y, z)$. Then, from the set of all possible networks obtained by this procedure, dismissing those not satisfying constraints (a) and (c), we get five new basic tri-lgt-nets, which are denoted as $B^i(x, y, z)$ $(i = 1, \dots, 5)$ and are depicted in Figure 4.2.

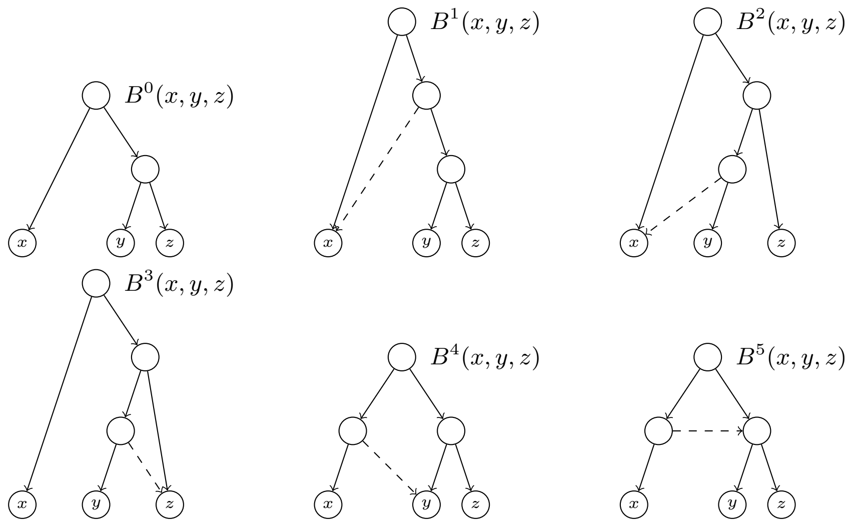

Figure 4.2: The basic tri-lgt-nets: $B^0(x, y, z)$, $B^1(x, y, z)$, $B^2(x, y, z)$, $B^3(x, y, z)$, $B^4(x, y, z)$, and $B^5(x, y, z)$.

The eNewick strings (see Section 1.4) identifying the different basic tri-lgt-nets are:

$$\begin{array}{rcl}
B^0(x,y,z) & = & (x,(y,z)); \\
B^1(x,y,z) & = & (x\#1,(x\#LGT1,(y,z))); \\
B^2(x,y,z) & = & (x\#1,((x\#LGT1,y),z)); \\
B^3(x,y,z) & = & (x,((y,z\#LGT1),z\#1)); \\
B^4(x,y,z) & = & ((x,y\#LGT1),(y\#1,z)); \\
B^5(x,y,z) & = & ((x,(y,z)\#LGT1),(y,z)\#1);
\end{array}$$

Note that if $i = 0, 1, 5$, both nodes y and z play the same role in the respective basic tri-lgt-net, then $B^i(x,y,z) = B^i(x,z,y)$. In all other cases, any permutation of the taxa x, y, z yields a non-isomorphic basic tri-lgt-net.

Basic tri-lgt networks defined by secondary arcs

Given N an LGT network defined on S, a 3-tuple $\{x,y,z\} \subseteq S$ and a secondary arc $e = (u,v)$ in N, we define the tri-lgt network $N^e_{x,y,z}$, as the LGT network obtained from N by applying the following operations:

1. Delete all secondary arcs, except e,
2. Delete all nodes that are not principal ascendants of at least one of the leaves x, y, z.
3. If e has not been deleted in the previous step, and its target v is principally elementary in the resulting network, then redirect the target of e to the first principal descendant node of v which is not principally elementary.
4. Suppress all the elementary nodes.

Example 4.1. *Figure 4.3 shows the steps followed to obtain the network $N^{e_1}_{1,4,5}$ from an LGT network N defined on $S = \{1,\dots,8\}$.*

In general, different secondary arcs $e \neq e'$ yield different networks, $N^e_{x,y,z} \neq N^{e'}_{x,y,z}$. However, some positions of the extremes of e and e' in N may result in getting isomorphic networks, $N^e_{x,y,z} = N^{e'}_{x,y,z}$. For example, let $e = (u,v)$ and $e' = (u',v')$; if there are principal paths $u \rightsquigarrow u' \rightsquigarrow t$ and $v \rightsquigarrow v' \rightsquigarrow s$, with $t, s \in \{x,y,z\}$, such that all descendant leaves of an intermediate node of the paths $u \rightsquigarrow u'$ and $v \rightsquigarrow v'$ are not any of the x, y, z, then in the third operation, the targets of the arcs e and e' are redirected to the same node: the first principal descendant node that is not principally elementary is the same for both target nodes. Finally, as per the fourth operation, all nodes in the path $u \rightsquigarrow u'$ collapse into a single node. Consequently, $N^e_{x,y,z} = N^{e'}_{x,y,z}$. The same result is obtained either by changing the role of the nodes u and u' in $u \rightsquigarrow u'$, or changing the role of v and v' in $v \rightsquigarrow v'$. See Example 4.2.

Example 4.2. *Figure 4.4 shows an LGT network N and the tri-lgt-nets $N^e_{1,4,5}$ associated to all secondary arcs of N: e_1, e_2, e_3 and e_4. Note that $N^{e_1}_{1,4,5} = N^{e_2}_{1,4,5} = N^{e_3}_{1,4,5} \neq N^{e_4}_{1,4,5}$.*

Given an LGT network N defined on a set of taxa S and given e a secondary arc in N, for each set $\{x,y,z\} \subseteq S$, there is an $i \in \{0,1,2,3,4,5\}$ such that $N^e_{x,y,z} = B^i(x,y,z)$ choosing a convenient order of x, y, z. In such case, we say that the tri-lgt-net $N^e_{x,y,z}$ is of *type i*, and, also, that the basic tri-lgt-net $B^i(x,y,z)$ is *displayed* by N.

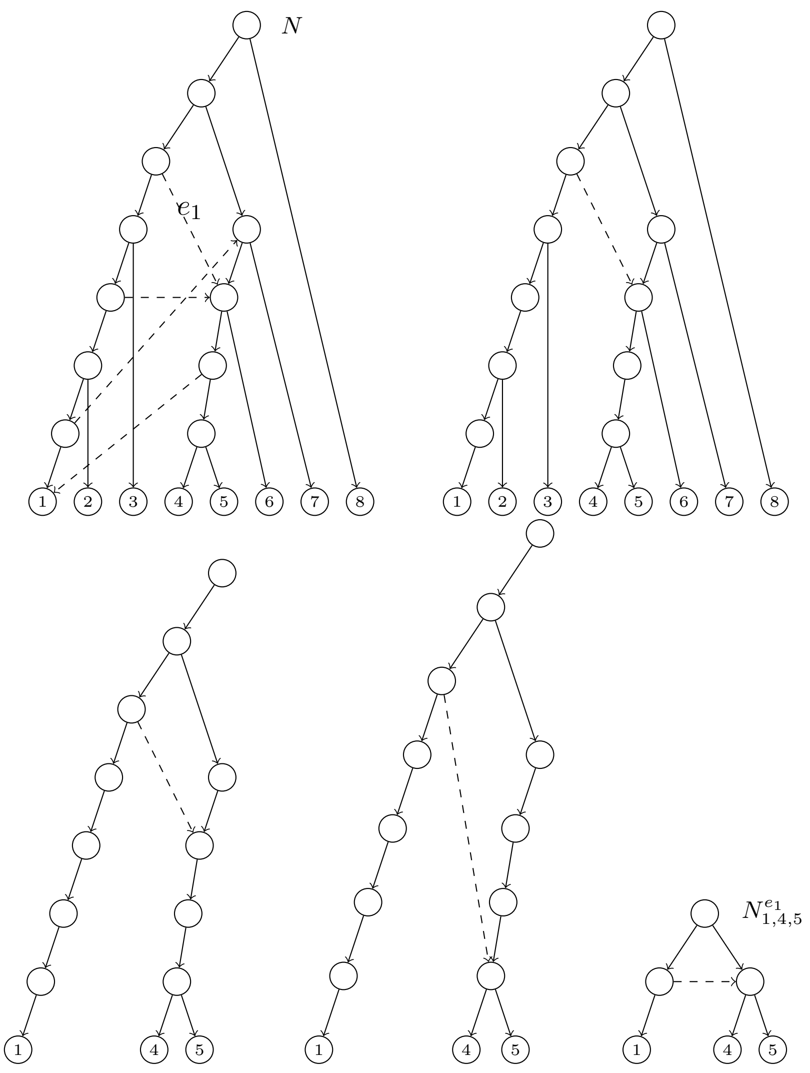

Figure 4.3: Depiction of the steps needed to obtain the tri-lgt-net $N^{e_1}_{1,4,5}$ from an LGT network N. The top two figures represent, from left to right, the network N with a highlighted secondary arc e_1, and the resulting network after applying the first operation, respectively. In the bottom figures, from left to right, there are depicted the resulting networks after applying the second, the third and the fourth operations, respectively.

Notice that, the definition of the tri-lgt-net $N^e_{x,y,z}$ depends on both the secondary arc e and the 3-taxa x, y, z. We denote by $N^0_{x,y,z}$ the triple on x, y, z displayed by $\widetilde{T}_0(N)$ which does not depend on any secondary arc. Notice that $N^0_{x,y,z}$ is, particularly, a basic tri-lgt of type 0 of N.

Example 4.3. *In the LGT network N depicted in Figure 4.4, we have that: $N^{e_1}_{1,2,8} = B^0(8, 1, 2)$, $N^{e_4}_{1,4,5} = B^1(1, 4, 5)$, $N^{e_3}_{1,2,7} = B^2(7, 1, 2)$, $N^{e_2}_{2,6,8} = B^3(8, 2, 6)$, $N^{e_4}_{1,3,4} = B^4(4, 1, 3)$ and $N^{e_1}_{3,4,6} = B^5(3, 4, 6)$, among others. Consequently, we can say that, for example, the LGT network $N^{e_4}_{1,4,5}$ is of type 1, and also that $B^1(1, 4, 5)$ is displayed by N.*

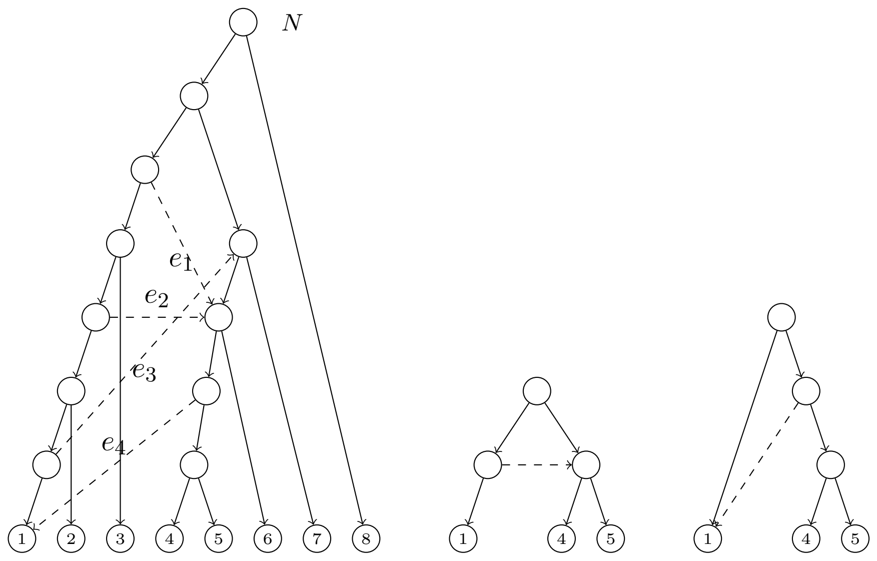

Figure 4.4: An LGT network N (left), the isomorphic LGT networks $N^{e_1}_{1,4,5}, N^{e_2}_{1,4,5}$ and $N^{e_3}_{1,4,5}$ (middle) and $N^{e_4}_{1,4,5}$ (right).

Decomposition in basic tri-lgt networks

Let N be an LGT network defined on S with at least three leaves. For a fixed secondary arc e in N, we denote by $\Gamma_e(N)$ the set of all tri-lgt-nets $N^e_{x,y,z}$ with $\{x, y, z\} \subseteq S$; more formally:

$$\Gamma_e(N) = \{N^e_{x,y,z} : \{x, y, z\} \subseteq S\}.$$

Moreover, we denote

$$\Gamma_0(N) = \{N^0_{x,y,z} : \{x, y, z\} \subseteq S\}.$$

Then, we call the *tri-lgt decomposition* of N, denoted by $\Gamma(N)$, the set of all basic tri-lgt-nets displayed by N jointly with the set $\Gamma_0(N)$. This is

$$\Gamma(N) = \left(\bigcup_{e\in E_s(N)} \Gamma_e(N) \right) \cup \Gamma_0(N).$$

Similarly, in order to obtain alternative definitions of $\Gamma(N)$, we denote by $\Gamma^i(N)$ and $\Gamma^i_e(N)$ the set of tri-lgt-nets in $\Gamma(N)$ and, respectively, in $\Gamma_e(N)$, that are of type i. Then, the tri-lgt decomposition of N can be written as

$$\Gamma(N) = \bigcup_{i=0}^{5} \Gamma^i(N),$$

and also as

$$\Gamma(N) = \left(\bigcup_{e\in E_s(N)} \bigcup_{i=1}^{5} \Gamma^i_e(N) \right) \cup \Gamma_0(N).$$

In order to refer to the set of basic tri-lgt-nets displayed by a secondary arc but that are not displayed by the principal subtree of N, we denote $\Gamma_*(N) = \Gamma(N) \setminus \Gamma_0(N) = \bigcup_{i=1}^{5} \Gamma^i(N)$.

Remark 4.1. *The reduced principal subtree $\widetilde{T}_0(N)$ of an LGT network N is, in particular, an LGT network itself. Its tri-lgt decomposition can be obtained from that of N in a straightforward way:*

$$\Gamma(\widetilde{T}_0(N)) = \Gamma_0(N).$$

Now, we fully characterize the set of tri-lgt networks of $\Gamma_e(N)$ in terms of the relative position of the extremes of the secondary arc e in N. This allows us to compute the tri-lgt decomposition of the network in an alternative way in terms of clusters. In order to do this, we first define some sets of leaves of the network relative to e (see Figure 4.5). Let $e = (u, v)$ be a secondary arc in N, r its root and $l = LCA_{T_0(N)}(u, v)$. Considering the principal paths $l \to w_u \rightsquigarrow u$ and $l \to w_v \rightsquigarrow v$, we define:

- $A_e = Cl_{T_0(N)}(u)$,
- $B_e = Cl_{T_0(N)}(v)$,
- $C_e = Cl_{T_0(N)}(w_u) \setminus Cl_{T_0(N)}(u)$,
- $D_e = Cl_{T_0(N)}(w_v) \setminus Cl_{T_0(N)}(v)$,
- $E_e = Cl_{T_0(N)}(r) \setminus Cl_{T_0(N)}(l)$.

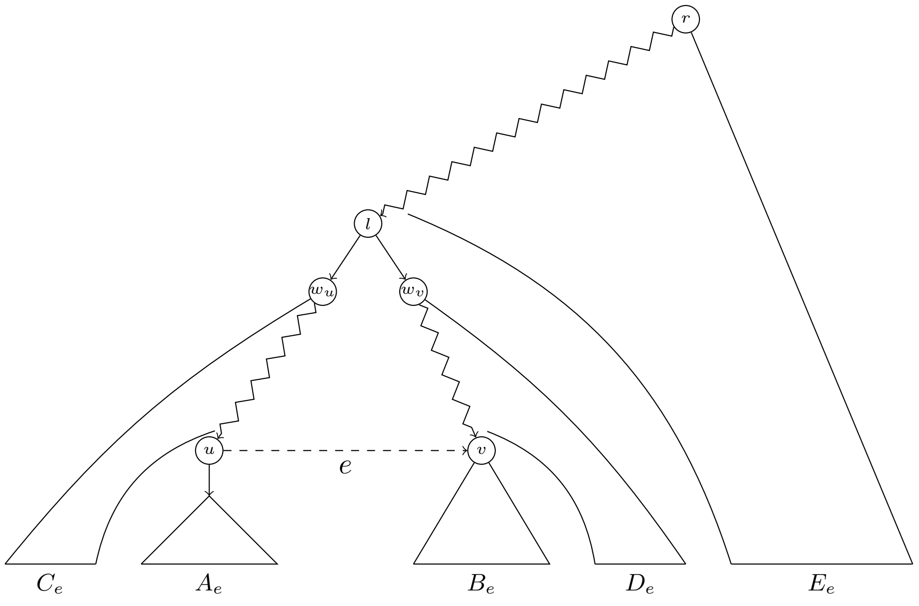

Figure 4.5: Representative scheme of the different set of leaves relative to the secondary arc $e = (u, v)$.

Lemma 4.2. *Given an LGT network N defined on S, a secondary arc e in N, and three taxa $x, y, z \in S$ sorted in such a way that the tree with Newick string $(x, (y, z))$; is in $\Gamma_0(N)$, then the tri-lgt-net $N^e_{x,y,z}$ is*

- *of type 1 if, and only if, $x \in B_e$ and $y, z \in A_e$,*
- *of type 2 if, and only if, $x \in B_e$, $y \in A_e$ and $z \in C_e$,*
- *of type 3 if, and only if, $x \in E_e$, $y \in A_e$ and $z \in B_e$,*

- *of type* 4 *if, and only if,* $x \in A_e$*,* $y \in B_e$ *and* $z \in D_e$*.*
- *of type* 5 *if, and only if,* $x \in A_e$ *and* $y, z \in B_e$*.*

Proof. Let $e = (u, v)$ be the considered secondary arc in N, r the root of N and $l = LCA_{T_0(N)}(u, v)$. We prove each case separately.

- Let $N^e_{x,y,z}$ be a tri-lgt-net of type 1. We need to prove that x, y, z satisfy that $x \in B_e$ and $y, z \in A_e$. If $N^e_{x,y,z}$ is of type 1, then u is an intermediate node in the principal path that starts in l and ends in the least common ancestor of both y and z. Then, $y, z \in Cl_{T_0(N)}(u) = A_e$. Moreover, v is an intermediate node in the principal path from l to x (or $x = v$), consequently, $x \in Cl_{T_0(N)}(v) = B_e$. To prove the converse implication, given $N^e_{x,y,z}$, with $x \in B_e$ and $y, z \in A_e$, we need to proof that $N^e_{x,y,z}$ is of type 1. If $y, z \in A_e$, both nodes are principal descendant nodes of u, then u is an intermediate node in the principal path from l to the least common ancestor of y and z. Similarly, x is a principal descendant node of v, and then $N^e_{x,y,z}$ is of type 1.
- Let $N^e_{x,y,z}$ be a tri-lgt-net of type 2; we have to prove that x, y, z satisfy that $x \in B_e$, $y \in A_e$ and $z \in C_e$. If $N^e_{x,y,z}$ is of type 2, then u is an intermediate node in the principal path that starts in the least common ancestor of y and z (which is a descendant of l) and it concludes in y. Also, v is an intermediate node in the principal path from l to x. Then, $y \in A_e$ and $x \in B_e$. The leaf z is a descendant of l, but it is not a descendant of u; then, $z \in Cl_{T_0(N)}(l) \setminus Cl_{T_0(N)}(u) = C_e$. To prove the converse implication, given $N^e_{x,y,z}$ such that $x \in B_e$, $y \in A_e$ and $z \in C_e$, we prove that $N^e_{x,y,z}$ is of type 2. If $y \in A_e$ then y is a principal descendant of u and, if $z \in C_e$ then u is not a node in the principal path from the least common ancestor of y and z (we call this node w_{yz}) to z; consequently, u is an intermediate node in the principal path from w_{yz} to y. Also, if $x \in B_e$ then x is a principal descendant of v; then $N^e_{x,y,z}$ is of type 2.
- Let $N^e_{x,y,z}$ be a tri-lgt-net of type 3. We prove that $x \in E_e$, $y \in A_e$ and $z \in B_e$. If $N^e_{x,y,z}$ is of type 3, then u and v are intermediate nodes in the principal path from the least common ancestor of y and z to y and to z, respectively. Then $y \in A_e$ and $z \in B_e$. The leaf x is a descendant of r, but it is not a descendant of l; then, $x \in Cl_{T_0(N)}(r) \setminus Cl_{T_0(N)}(l) = E_e$. To prove the converse implication, given $N^e_{x,y,z}$ such that $x \in E_e$, $y \in A_e$ and $z \in B_e$ we prove that $N^e_{x,y,z}$ is of type 3. If $y \in A_e$, then y is a principal descendant of u; if $x \in E_e$, x is not a principal descendant of l; and if $z \in B_e$, z is a principal descendant of v. Hence, $N^e_{x,y,z}$ is of type 3.
- Let $N^e_{x,y,z}$ be a tri-lgt-net of type 4. We prove that $x \in A_e$, $y \in B_e$ and $z \in D_e$. If $N^e_{x,y,z}$ is of type 4, then v is an intermediate node in the principal path from the least common ancestor of y and z to y. Also, u is an intermediate node in the principal path from l to x. Then, $x \in A_e$ and $y \in B_e$. The leaf z is a principal descendant of l, but it is not a descendant of v, then $z \in Cl_{T_0(N)}(l) \setminus Cl_{T_0(N)}(v) = D_e$. To prove the converse implication, given $N^e_{x,y,z}$ such that $x \in A_e$, $y \in B_e$ and $z \in D_e$ we prove that $N^e_{x,y,z}$ is of type 4. If $y \in B_e$ then y is a principal descendant of v and if $z \in D_e$ then v is not a node in the principal path from the least common ancestor of y and z (we call this node w_{yz}) to z. Then, v is intermediate node in the principal path from w_{yz} to y. Moreover, if $x \in A_e$, x is a principal descendant of u; then $N^e_{x,y,z}$ is of type 4.
- Let $N^e_{x,y,z}$ be a tri-lgt-net of type 5. We prove that $x \in A_e$ and $y, z \in B_e$. If $N^e_{x,y,z}$ is of type 5, then v is an intermediate node in the principal path from l to the

least common ancestor of y and z. Then, $y, z \in Cl_{T_0(N)}(v) = B_e$. Also, u is an intermediate node in the principal path from l to x; then, $x \in Cl_{T_0(N)}(u) = A_e$. To prove the converse implication, given $N^e_{x,y,z}$ such that $x \in A_e$ and $y, z \in B_e$ we prove that $N^e_{x,y,z}$ is of type 5. If $y, z \in B_e$, both nodes are principal descendants of v; then v is an intermediate node in the principal path from l to the least common ancestor of y and z. Similarly, x is a principal descendant of u; hence, $N^e_{x,y,z}$ is of type 5. □

As a direct consequence of the Lemma 4.2, we have the following proposition.

Proposition 4.3. *Let e be a secondary arc in an LGT network N. Then $\Gamma_e(N)$ can be computed from the sets A_e, B_e, C_e, D_e, E_e, as follows:*

- $\Gamma^1_e(N) = \{B^1(x, y, z) : x \in B_e \textit{ and } y, z \in A_e\}$,
- $\Gamma^2_e(N) = \{B^2(x, y, z) : x \in B_e, y \in A_e \textit{ and } z \in C_e\}$,
- $\Gamma^3_e(N) = \{B^3(x, y, z) : x \in E_e, y \in A_e \textit{ and } z \in B_e\}$,
- $\Gamma^4_e(N) = \{B^4(x, y, z) : x \in A_e, y \in B_e \textit{ and } z \in D_e\}$,
- $\Gamma^5_e(N) = \{B^5(x, y, z) : x \in A_e \textit{ and } y, z \in B_e\}$.

Note that, knowing $\Gamma_0(N)$ and the sets A_e, B_e, C_e, D_e, E_e for each secondary arc e in an LGT network N, Proposition 4.3 allows us to obtain the tri-lgt decomposition of the network. In the following example we illustrate this fact with an LGT network.

Example 4.4. *Let N be the LGT network depicted in Figure 4.6. For each secondary arc e of N, we compute the sets of leaves A_e, B_e, C_e, D_e, E_e. Using Proposition 4.3, we compute for each secondary arc e the set $\Gamma^i_e(N)$ for $i = 1, \dots, 5$. We also compute $\Gamma_0(N)$. Consequently, we obtain the tri-lgt decomposition of N, $\Gamma(N)$.*

For the secondary arc e_1, we have that

$$A_{e_1} = \{1, 2\}, B_{e_1} = \{4, 5\}, C_{e_1} = \{3\}, D_{e_1} = \{6\}, E_{e_1} = \{7\}.$$

Then, applying Proposition 4.3:

- $\Gamma^1_{e_1}(N) = \{B^1(4, 1, 2), B^1(5, 1, 2)\}$,
- $\Gamma^2_{e_1}(N) = \{B^2(4, 1, 3), B^2(4, 2, 3), B^2(5, 1, 3), B^2(5, 2, 3)\}$,
- $\Gamma^3_{e_1}(N) = \{B^3(7, 1, 4), B^3(7, 2, 4), B^3(7, 1, 5), B^3(7, 2, 5)\}$,
- $\Gamma^4_{e_1}(N) = \{B^4(1, 4, 6), B^4(1, 5, 6), B^4(2, 4, 6), B^4(2, 5, 6)\}$,
- $\Gamma^5_{e_1}(N) = \{B^5(1, 4, 5), B^5(2, 4, 5)\}$.

For the secondary arc e_2, we have that

$$A_{e_2} = \{2\}, B_{e_2} = \{4, 5, 6\}, C_{e_2} = \{1, 3\}, D_{e_2} = \emptyset, E_{e_2} = \{7\}.$$

Then, using Proposition 4.3:

- $\Gamma^1_{e_2}(N) = \emptyset$,
- $\Gamma^2_{e_2}(N) = \{B^2(4,2,1), B^2(4,2,3), B^2(5,2,1), B^2(5,2,3), B^2(6,2,1), B^2(6,2,3)\}$,
- $\Gamma^3_{e_2}(N) = \{B^3(7,2,4), B^3(7,2,5), B^3(7,2,6)\}$,
- $\Gamma^4_{e_2}(N) = \emptyset$,
- $\Gamma^5_{e_2}(N) = \{B^5(2,4,5), B^5(2,4,6), B^5(2,5,6)\}$,

For the secondary arc e_3, we have that

$$A_{e_3} = \{6\}, B_{e_3} = \{4\}, C_{e_3} = \emptyset, D_{e_3} = \{5\}, E_{e_3} = \{1,2,3,7\}.$$

Then, using Proposition 4.3:

- $\Gamma^1_{e_3}(N) = \emptyset$,
- $\Gamma^2_{e_3}(N) = \emptyset$,
- $\Gamma^3_{e_3}(N) = \{B^3(1,6,4), B^3(2,6,4), B^3(3,6,4), B^3(7,6,4)\}$,
- $\Gamma^4_{e_3}(N) = \{B^4(6,4,5)\}$,
- $\Gamma^5_{e_3}(N) = \emptyset$.

Moreover,

- $\Gamma_0(N) = \{B^0(3,1,2), B^0(4,1,2), B^0(5,1,2), B^0(6,1,2), B^0(7,1,2), B^0(4,1,3), B^0(5,1,3), B^0(6,1,3), B^0(7,1,3), B^0(1,4,5), B^0(1,4,6), B^0(7,1,4), B^0(1,5,6), B^0(7,1,5), B^0(7,1,6), B^0(4,2,3), B^0(5,2,3), B^0(6,2,3), B^0(7,2,3), B^0(2,4,5), B^0(2,4,6), B^0(7,2,4), B^0(2,5,6), B^0(7,2,5), B^0(7,2,6), B^0(3,4,5), B^0(3,4,6), B^0(7,3,4), B^0(3,5,6), B^0(7,3,5), B^0(7,3,6), B^0(6,4,5), B^0(7,4,5), B^0(7,5,6)\}$.

Then, $\Gamma(N) = \Gamma_{e_1}(N) \cup \Gamma_{e_2}(N) \cup \Gamma_{e_3}(N) \cup \Gamma_0(N)$.

4.3 Redundant arcs and coverings

In this section we show that there may be secondary arcs that, in case of being removed from an LGT network, do not alter its tri-lgt decomposition. That is, the basic tri-lgt-nets induced by them are also induced by some other arc. Afterwards, we study how a set of secondary arcs should be so that, if they are removed from the network, the tri-lgt decomposition does not change.

Given N an LGT network and e a secondary arc, we say that e is a *redundant* arc in N when all basic tri-lgt-nets of $\Gamma_e(N)$ are displayed by $N \setminus \{e\}$. More precisely, e is a redundant arc if there exists a set of secondary arcs E' that does not contain e and such that:

$$\Gamma_e(N) \subseteq \bigcup_{e' \in E'} \Gamma_{e'}(N).$$

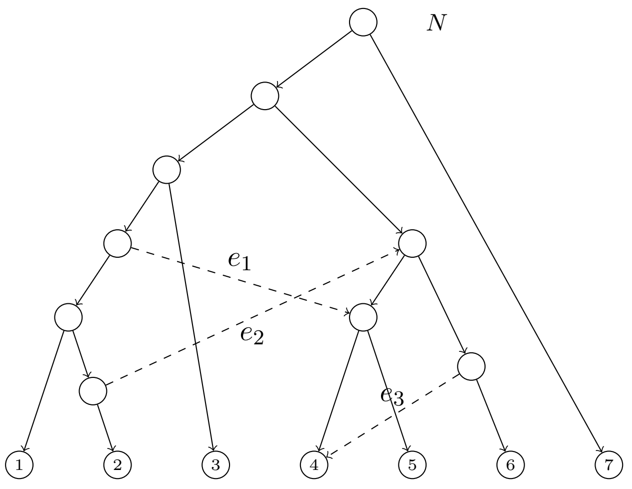

Figure 4.6: The LGT network N used in Example 4.4.

In that case, we say that the set of arcs E' is a *covering* of e. Given $N^e_{x,y,z} \in \Gamma_e(N)$, if $e' \in E'$ satisfies that $N^e_{x,y,z} \in \Gamma_{e'}(N)$, we say that e' *covers* $N^e_{x,y,z}$ or that $N^e_{x,y,z}$ is *covered* by e'.

Note that, if we add new secondary arcs to a covering of a fixed secondary arc, the resulting set is a (different) covering for the same arc. A covering of an arc is said to be *minimal* if none of its proper subsets is a covering of the arc. Notice that there may exist different minimal coverings for the same arc (see Example 4.5).

Example 4.5. *Let N be the LGT network depicted in the Figure 4.7 (left). The secondary arcs e_1 and e_2 are the only redundant arcs in N. Note that $\{e_2, e_3, e_4\}$, $\{e_3, e_4, e_5, e_6\}$, $\{e_2, e_3, e_4, e_5, e_6\}$ are coverings of e_1 but only the first two are minimal coverings of e_1. For example, the covering $\{e_2, e_3, e_4\}$ is minimal given that $B^4(1,4,6) \in \Gamma_{e_1}(N) \cap \Gamma_{e_2}(N)$, but $B^4(1,4,6) \notin \Gamma_{e_3}(N) \cup \Gamma_{e_4}(N)$; $B^5(1,4,5) \in \Gamma_{e_1}(N) \cap \Gamma_{e_3}(N)$, but $B^5(1,4,5) \notin \Gamma_{e_2}(N) \cup \Gamma_{e_4}(N)$ and $B^4(1,5,6) \in \Gamma_{e_1}(N) \cap \Gamma_{e_4}(N)$, but $B^4(1,5,6) \notin \Gamma_{e_2}(N) \cup \Gamma_{e_3}(N)$. Note also that, for instance, $\{e_3, e_5, e_6\}$ and $\{e_1, e_5, e_6\}$ are minimal coverings of e_2.*

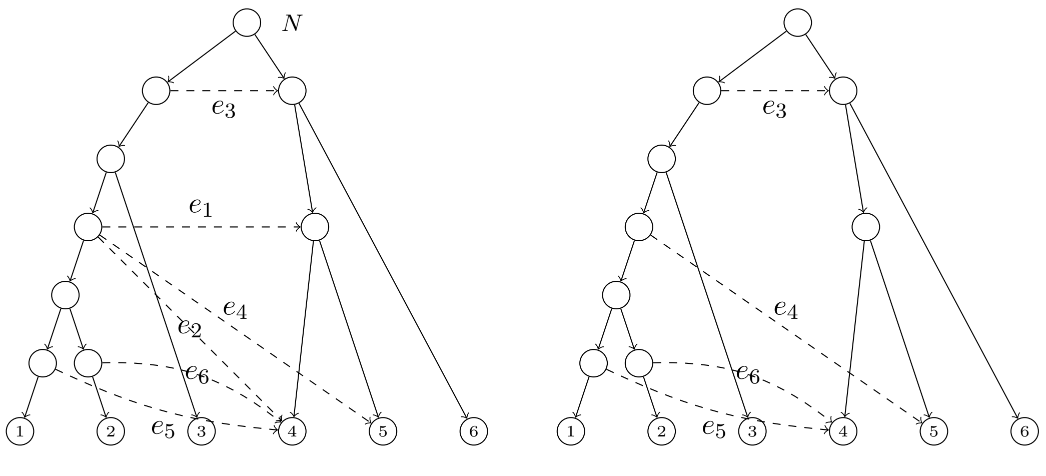

Figure 4.7: An LGT network N, left; and the LGT network $N \setminus \{e_1, e_2\}$ obtained by removing both secondary arcs e_1 and e_2 from N, right.

The tri-lgt decomposition of the LGT network obtained by removing a single redundant arc from an LGT network is always the same as the tri-lgt decomposition of the original network (cf. Proposition 4.4). Nevertheless, if we remove two redundant secondary arcs from the network, then the tri-lgt decomposition of the resulting LGT network can remain invariant (see Example 4.6), but it can also be different (see Example 4.7).

In Proposition 4.5 we prove that, if two redundant arcs are removed from an LGT network, the tri-lgt decomposition remains unchanged respect to the original one when each of these two arcs has a covering that does not contain the other arc. The generalization of this result to more than two secondary arcs is given in Proposition 4.6.

Proposition 4.4. *Given N an LGT network and e^* a secondary arc in N,*

(a) $\Gamma(N) = \Gamma(N \setminus \{e^*\})$ *if and only if e^* is a redundant arc in N.*

(b) $\Gamma_*(N \setminus \{e^*\}) = \Gamma_*(N) \setminus \Gamma_{e^*}(N)$ *if, and only if, none of the basic tri-lgt-nets of type i with $i \neq 0$ in $\Gamma_{e^*}(N)$ is covered by a secondary arc of N (apart from e^* itself).*

Proof. Let N be an LGT network and let e^* be a secondary arc. To prove (a), note that $\Gamma(N) = \Gamma(N \setminus \{e^*\})$ if and only if all tri-lgt-nets of $\Gamma_{e^*}(N)$ are in $\Gamma(N \setminus \{e^*\})$. This is, if and only if e^* is a redundant arc in N. To prove (b), write

$$\Gamma_*(N) = \left(\bigcup_{i=1}^{5} \Gamma^i_{e^*}(N)\right) \cup \left(\bigcup_{e \in E_s \setminus \{e^*\}} \bigcup_{i=1}^{5} \Gamma^i_e(N)\right) = \left(\bigcup_{i=1}^{5} \Gamma^i_{e^*}(N)\right) \cup \Gamma_*(N \setminus \{e^*\}).$$

Now, the equality of sets in the statement of the proposition can be written as

$$\Gamma_*(N \setminus \{e^*\}) = \left(\left(\bigcup_{i=1}^{5} \Gamma^i_{e^*}(N)\right) \cup \Gamma_*(N \setminus \{e^*\})\right) \setminus \Gamma_{e^*}(N)$$

and it holds if, and only if, $\bigcup_{i=1}^{5} \Gamma^i_{e^*}(N)$ and $\Gamma_*(N \setminus \{e^*\})$ are disjoint. This last condition is equivalent to saying that none of the basic tri-lgt-nets of type i with $i \neq 0$ in $\Gamma_{e^*}(N)$ is covered by any other arc of N. □

Proposition 4.5. *Let N be an LGT network and let e_1 and e_2 be two secondary arcs in N. Then, $\Gamma(N) = \Gamma(N \setminus \{e_1, e_2\})$ if and only if (1) both e_1 and e_2 are redundant arcs, and (2) each one of them has a covering that does not contain the other one.*

Proof. If either e_1 or e_2 is not redundant, then $\Gamma(N) \neq \Gamma(N \setminus \{e_1, e_2\})$ by definition of redundant arc. Hence we may assume from now on that both arcs are redundant and we need to prove that $\Gamma(N) = \Gamma(N \setminus \{e_1, e_2\})$ if and only if e_1 has a covering in N that does not contain e_2 and viceversa. Notice that thanks to Proposition 4.4 we have that

$$\Gamma(N) = \Gamma(N \setminus \{e_1\}) = \Gamma(N \setminus \{e_2\}).$$

Let us assume that $\Gamma(N) = \Gamma(N \setminus \{e_1, e_2\})$, which can be written as $\Gamma(N \setminus \{e_1\}) = \Gamma(N \setminus \{e_1, e_2\})$. Thanks to Proposition 4.4(a) this implies that e_2 is a redundant arc in $N \setminus \{e_1\}$, and hence it has a covering in N that does not contain e_1. Analogously we have that e_1 has a covering that does not contain e_2.

To prove the converse, notice that, since e_1 has a covering that does not include e_2, we have that e_1 is redundant in $N \setminus \{e_2\}$ and hence, thanks to Proposition 4.4(a), we get

$$\Gamma(N \setminus \{e_2\}) = \Gamma(N \setminus \{e_1, e_2\}).$$

Since $\Gamma(N) = \Gamma(N \setminus \{e_2\})$ the result follows. □

Example 4.6. *Let N be the LGT network depicted in Figure 4.7 (left) and let $N \setminus \{e_1, e_2\}$ be the one depicted in Figure 4.7 (right). As we have seen in Example 4.5, there exists a covering for e_1 that does not contain e_2 and a covering for e_2 that does not contain e_1. Then, by Proposition 4.5 we have that $\Gamma(N) = \Gamma(N \setminus \{e_1, e_2\})$.*

Example 4.7. *Let N be the LGT network depicted in Figure 4.8. The secondary arcs e_1 and e_2 are the only redundant arcs in N. Moreover, $\{e_2, e_3\}$ and $\{e_1, e_4, e_5\}$ are the only minimal coverings for e_1 and e_2, respectively. Then, e_2 is present in each covering of e_1 and, viceversa. Note that $B^2(4,2,1) \in \Gamma_{e_1}(N) \cap \Gamma_{e_2}(N)$ but $B^2(4,2,1) \notin \Gamma_{e_3}(N) \cup \Gamma_{e_4}(N) \cup \Gamma_{e_5}(N)$. Then, $\Gamma(N) \neq \Gamma(N \setminus \{e_1, e_2\})$. Notice that it still holds that $\Gamma(N) = \Gamma(N \setminus \{e_1\}) = \Gamma(N \setminus \{e_2\})$.*

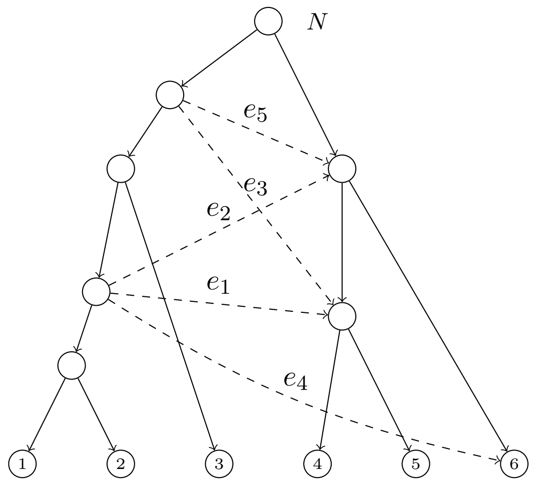

Figure 4.8: An LGT network N with two secondary redundant arcs e_1, e_2 satisfying that $\Gamma(N) \neq \Gamma(N \setminus \{e_1, e_2\})$.

Proposition 4.6. *Let N be an LGT network and let R be a set of secondary arcs in N. Then, $\Gamma(N) = \Gamma(N \setminus R)$ if and only if (1) all arcs in R are redundant arcs, and (2) each one of them has a covering in which no other arc of R is present.*

Proof. If any arc in R was not redundant, the proposition would follow in a straightforward way. Hence we assume from now on that all arcs in R are redundant.

Let us assume that $\Gamma(N) = \Gamma(N \setminus R)$ and let $e \in R$. Say $R' = R \setminus \{e\}$. Now, all inclusions in the chain $\Gamma(N) \supseteq \Gamma(N \setminus R') \supseteq \Gamma(N \setminus R)$ must be equalities, hence $\Gamma(N \setminus R') = \Gamma(N \setminus R) = \Gamma((N \setminus R') \setminus \{e\})$. By Proposition 4.4(a), this implies that e is a redundant arc in $N \setminus R'$, and hence it has a covering in $N \setminus R'$, which means that it has a covering in N that does not contain any arc of R'.

To prove the converse, let $R := \{e_1, \dots, e_k\}$ and consider the chain

$$\Gamma(N) \supseteq \Gamma(N \setminus \{e_1\}) \supseteq \Gamma(N \setminus \{e_1, e_2\}) \supseteq \cdots \supseteq \Gamma(N \setminus \{e_1, e_2, \dots, e_k\}).$$

In order to get a contradiction, assume that some inclusion in the chain is strict. Let us assume that the first strict inclusion is located at the i-th position:

$$\Gamma(N) = \Gamma(N \setminus \{e_1\}) = \cdots = \Gamma(N \setminus \{e_1, e_2, \dots, e_{i-1}\}) \supsetneq \Gamma(N \setminus \{e_1, e_2, \dots, e_i\}).$$

If $\Gamma(N \setminus \{e_1, e_2, \dots, e_{i-1}\}) \supsetneq \Gamma(N \setminus \{e_1, e_2, \dots, e_i\})$, this implies that e_i is not redundant in $N \setminus \{e_1, e_2, \dots, e_{i-1}\}$. Since e_i is not redundant in $N \setminus \{e_1, e_2, \dots, e_{i-1}\}$ and e_i is a redundant arc ($e_i \in R$) in N, then any covering of e_i must contain some of the arcs $e_1, e_2, \dots, e_{i-1}$, against our hypothesis. □

4.4 Characterization of partial coverings of a redundant arc

In this section we study the secondary arcs that must be present in an arbitrary set of secondary arcs to become a covering for a fixed secondary arc of an LGT network. We do not give a full characterization of coverings in all its generality, but only those required to study a particular case of LGT networks in the next section.

A covering of a secondary arc covers, particularly, all basic tri-lgt networks of every type i ($i = 1, \dots, 5$) that the arc induces. Then, a covering of a secondary arc is composed by a set of arcs covering each one of the types of basic tri-lgt networks: some arcs cover the tri-lgt networks of type 1, some (not necessarily disjoint from the previous ones) cover the tri-lgt networks of type 2, and so on. Then, given e a secondary redundant arc in N, we say that the set E'_i is a *partial* covering of type i of e ($i = 1, \dots, 5$), if the arcs in E'_i cover all basic tri-lgt networks of type i induced by e. More precisely, if

$$\Gamma^i_e(N) \subseteq \bigcup_{e' \in E'_i} \Gamma^i_{e'}(N).$$

Similarly to what we did in the previous section, when we can not remove any secondary arc present in a partial covering of type i of a redundant arc, we say that the partial covering is *minimal*. Note that, if $E'_1, \dots, E'_5$ are minimal partial coverings of respective types $1, \dots, 5$ of an arc e, then their union is a covering of e, but not necessarily minimal. See Examples 4.8, where the union is a minimal covering, and 4.9, where the union is not minimal.

Example 4.8. *Let N be the LGT network depicted in Figure 4.7. Note that, the sets $E'_1 = \{e_3\}$, $E'_2 = \{e_2, e_4\}$, $E'_3 = \emptyset$, $E'_4 = \{e_2, e_4\}$ and $E'_5 = \{e_3\}$ are minimal partial coverings of the redundant arc e_1 for each of the types. Moreover, $\{e_2, e_3, e_4\}$ is a minimal covering of e_1 as we have seen in Example 4.5.*

Example 4.9. *Let N be the LGT network depicted in Figure 4.9. The set $\{e_1, e_2, e_3, e_4\}$ is a covering of the secondary arc e but it is not a minimal covering. Note that, no type 3 basic tri-lgt-net is displayed by e. The singleton $\{e_1\}$ is a minimal partial covering of types 1 and 5 of e; the singleton $\{e_2\}$ is a minimal partial covering of types $1, 2$ and 5 of e; the set $\{e_3, e_4\}$ is a minimal partial covering of type $2, 4$ and 5. Then, $\{e_1, e_2, e_3, e_4\}$ is a covering of e, but it is not minimal. On the other hand, both sets $\{e_1, e_3, e_4\}$ and $\{e_2, e_3, e_4\}$ are minimal coverings of e.*

Let $e = (u, v)$ be a secondary arc, $l = LCA_{T_0(N)}(u, v)$ the least common ancestor in $T_0(N)$ of its extremes, and consider the principal paths $l \to w_u \rightsquigarrow u$, $l \to w_v \rightsquigarrow v$ in $T_0(N)$. Let $\mathcal{C}_e$ and $\mathcal{D}_e$ be the set of nodes of the paths $w_u \rightsquigarrow u$ and $w_v \rightsquigarrow v$, respectively. Let $\mathcal{A}_e$ be the set of internal nodes that are principal descendant nodes of u and let $\mathcal{B}_e$ be the set of principal descendants of v. See Figure 4.10.

Given E' a set of secondary arcs, we denote by $\pi_1(E')$ and $\pi_2(E')$ the sets of source and target nodes, respectively, of the arcs in E'. That is, $\pi_1(E') = \{u : \exists v \text{ such that } (u, v) \in E'\}$ and $\pi_2(E') = \{v : \exists u \text{ such that } (u, v) \in E'\}$.

Lemma 4.7. *Let $e = (u, v)$ be a redundant arc in an LGT network N defined on S, and let E' be a minimal covering of e. Then,*

(a) $\pi_1(E') \subset \mathcal{C}_e \cup \{u\} \cup \mathcal{A}_e$,

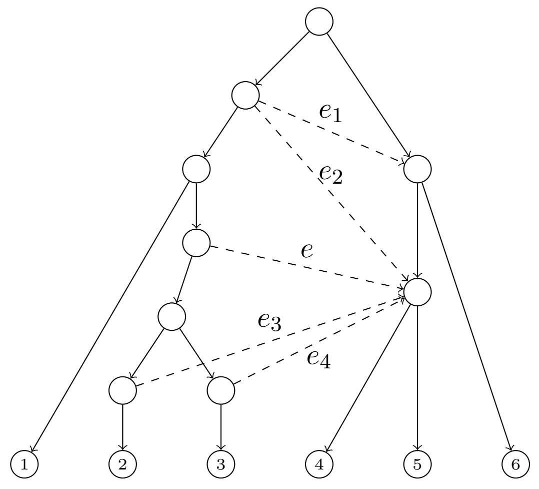

Figure 4.9: An LGT network with a redundant secondary arc e used in Example 4.9.

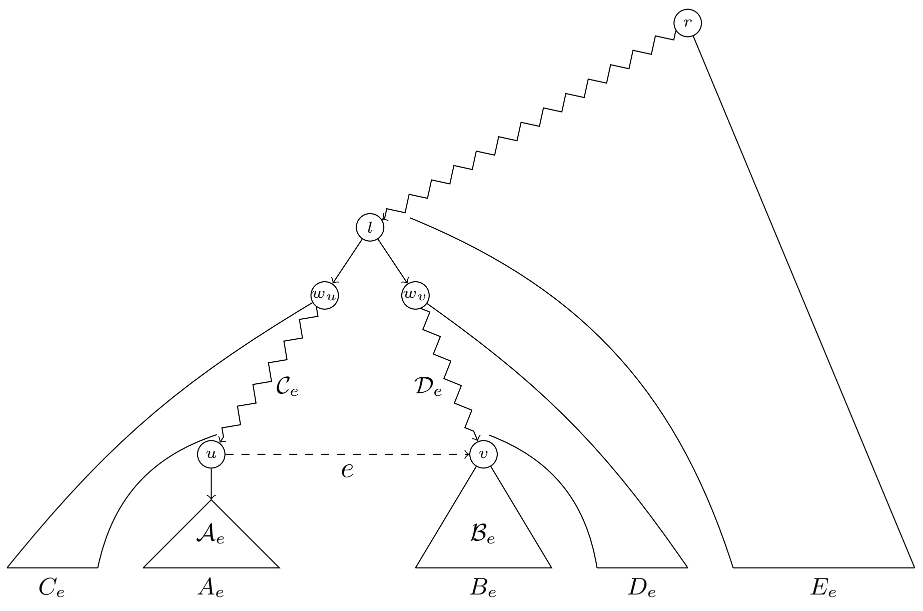

Figure 4.10: Representative scheme of the different set of nodes $\mathcal{A}_e, \mathcal{B}_e, \mathcal{C}_e, \mathcal{D}_e$ relative to the secondary arc $e = (u, v)$.

(b) $\pi_2(E') \subset \mathcal{D}_e \cup \{v\} \cup \mathcal{B}_e$.

Proof. Let $e' = (u', v') \in E'$ an arc in a minimal covering of e. Let x, y, z be any 3-tuple of taxa such that $B^i(x, y, z) \in \Gamma_e(N) \cap \Gamma_{e'}(N)$ with $i \neq 0$. Notice that this basic tri-lgt network must exist, since otherwise e' could be removed from E' and still get a covering of e. Let $a = LCA_{T_0(N)}(x, y) = LCA_{T_0(N)}(x, z)$ and $b = LCA_{T_0(N)}(y, z)$. Since e and e' induce the same basic tri-net on x, y, z, the respective extremes of both edges must be on the same of the four paths $a \rightsquigarrow x$, $a \rightsquigarrow b$, $b \rightsquigarrow y$ and $b \rightsquigarrow z$. That is, if u (resp. v) is in one of these four paths, then u' (resp. v') must be in the same path. Whichever is the path containing u, this full path is contained in $\mathcal{C}_e \cup \{u\} \cup \mathcal{A}_e$ and hence u' belongs to this set. Analogously, the path containing v is contained in $\mathcal{D}_e \cup \{v\} \cup \mathcal{B}_e$, and hence v' belongs to this set. □

Let N be an LGT network and a, b be two nodes, none of which is descendant of the other. We denote by $P^{a,b}$ (resp. $P^{b,a}$) the principal path in $T_0(N)$ from $LCA_{T_0(N)}(a,b)$ to a (resp. to b), excluding $LCA_{T_0(N)}(a,b)$.

Remark 4.8. *Let $e=(u,v)$ be a secondary arc in N. It is straightforward to check that:*

- $\cup_{i\in A_e} P^{i,v} = \mathcal{C}_e \cup \{u\} \cup \mathcal{A}_e$.
- $P^{u,v} = \mathcal{C}_e \cup \{u\}$.
- $P^{p_u,v} = \mathcal{C}_e$, *where p_u is the principal parent of u in N.*
- $\cup_{j\in B_e} P^{j,u} = \mathcal{D}_e \cup \{v\} \cup \mathcal{B}_e$.
- $P^{v,u} = \mathcal{D}_e \cup \{v\}$.
- $P^{p_v,u} = \mathcal{D}_e$, *where p_v is the principal parent of v in N.*

In the following five subsections we study each one of the partial coverings of a redundant arc, from type 1 to type 5.

Partial coverings of type 1

Now we analyze the set of secondary arcs that must be present in any partial covering of type 1 of a fixed secondary arc.

Proposition 4.9. *Let N be an LGT network and let $e=(u,v)$ be a secondary arc of N. Then, a set of secondary arcs $E' \subset E$ not containing e is a partial covering of type 1 of e if, and only if, for all $j \in B_e$, E' contains at least one secondary arc $(u',v') \in P^{u,v} \times P^{j,u}$.*

Proof. Let us assume that E' is a partial covering of type 1 of e and prove that it always contains arcs as in the statement of the proposition. Let $j \in B_e$ and take a, b descendant leaves of u such that its LCA is the single child of u; in particular $a, b \in A_e$. Now, thanks to Proposition 4.3 we have that $B^1(j,a,b) \in \Gamma^1_e(N)$ and since E' is a covering of type 1 of e, then $B^1(j,a,b) \in \Gamma^1_{e'}(N)$ for some $e'=(u',v') \in E'$. Using again Proposition 4.3, we get that $j \in B_{e'}$ (v' is an ascendant of j) and $a,b \in A_{e'}$ (a and b are descendants of u'). This last condition implies that u' is an ascendant of u (possibly equal to u). Now, neither u' nor v' can be ancestors of (or equal to) $LCA_{T_0}(u,j) = LCA_{T_0}(u,v)$, since otherwise the extremes of e' would be connected by a principal path, against our assumption on LGT networks. Then, $u' \in P^{u,v}$ and $v' \in P^{j,u}$.

In order to prove the converse, let E' be a set of secondary arcs satisfying the condition above. In order to prove that it is a partial covering of $e=(u,v)$ of type 1, let $B^1(x,y,z) \in \Gamma^1_e(N)$, which implies that $x \in B_e$ (v is an ancestor of x) and $y, z \in A_e$ (u is a common ancestor of y and z). Using the condition above we have that there exists $e'=(u',v')$ with $u' \in P^{u,v}$ and $v' \in P^{x,u}$. Since u' is an ancestor of u, then both y and z are descendants of u', and hence $y, z \in A_{e'}$. Since v' is an ancestor of x we have that $x \in B_{e'}$. Finally, $B^1(x,y,z) \in \Gamma^1_{e'}(N)$ and this basic tri-lgt of type 1 is covered by some arc in E'. □

Now, we prove a necessary condition to achieve a minimal partial covering of type 1 of a redundant arc.

Lemma 4.10. *Let E' be a minimal partial covering of type 1 of a redundant arc $e=(u,v)$. Then,* *(a) each arc (u', v') in E' satisfies that $(u', v') \in P^{u,v} \times P^{j,u}$ for some $j \in B_e$,*

(b) for each $j \in B_e$, there is at least one arc $(u', v') \in E'$ such that $(u', v') \in P^{u,v} \times P^{j,u}$.

Proof. The second part of the statement follows directly from the previous proposition.

In order to prove the first part, let $E'' \subseteq E'$ be the set of arcs in E' that do satisfy the condition. The arcs in $(u', v') \in E' \setminus E''$ satisfy that $(u', v') \notin P^{u,v} \times P^{j,u}$ (for all $j \in B_e$). Since E' satisfied the condition of Proposition 4.9, this same condition is also satisfied by E'' and hence E'' is a partial covering of type 1 of E. By the minimality assumption on E', we have that $E'' = E'$ and the result follows. □

Corollary 4.11. *If E' is a minimal partial covering of type 1 of an arc $e = (u, v)$, then $\pi_1(E') \subseteq \{u\} \cup \mathcal{C}_e$.*

Proof. The proof is a direct consequence of Lemma 4.10. □

Now we prove that if we modify the condition in Proposition 4.9 so that, instead of asking for the existence of *at least* one arc with certain properties, we ask for the existence of *exactly* one arc with those properties, we get a complete characterization of the minimal partial coverings of type 1 of secondary arcs.

Proposition 4.12. *Let N be an LGT network and $e = (u, v)$ a redundant arc in N. Then, a set E' of secondary arcs not containing e is a minimal partial covering of type 1 of e if, and only if, for each $j \in B_e$ there exists exactly one arc $e' = (u', v') \in E'$ with $(u', v') \in P^{u,v} \times P^{j,u}$.*

Proof. Let us assume that E' is a minimal partial covering of type 1 of e. Then Proposition 4.9 implies that for each $j \in B_e$ there exists at least one arc $e' = (u', v') \in E'$ with $(u', v') \in P^{u,v} \times P^{j,u}$. Let us assume that for a given j there exist at least two secondary arcs $e' = (u', v')$ and $e'' = (u'', v'')$ satisfying the condition. Since both v' and v'' belong to $P^{j,u}$, one of them is ascendant of the other; let us assume that v' is an ascendant of v'' (or $v' = v''$), and hence $B_{e''} \subseteq B_{e'}$. Also, since both u' and u'' belong to $P^{u,v}$ we get that $A_e \subseteq A_{e'}$ and $A_e \subseteq A_{e''}$. Using Proposition 4.3 we get that $\Gamma^1_{e''} \cap \Gamma^1_e \subseteq \Gamma^1_{e'} \cap \Gamma^1_e$. Now, $E' \setminus \{e''\}$ is still a partial covering of type 1 of e, against our assumption on the minimality of E', hence we have reached a contradiction and for each $j \in B_e$ there is exactly one arc $e' \in E'$ satisfying the condition above.

In order to prove the converse, thanks to Lemma 4.10, we only have to prove that if we remove an arc $(u', v') \in P^{u,v} \times P^{j,u}$ for some $j \in B_e$, then, the resulting set of arcs is not a partial covering of type 1 of e. This is easy to prove because otherwise, taking a, b descendant leaves of u such that their LCA is the single child of u, the basic tri-lgt network $B^1(j, a, b) \in \Gamma^1_e$ would not be covered by any arc. Consequently, the resulting set of arcs would not be a partial covering of type 1 of e. □

As per Proposition 4.12, if E' is a partial covering of type 1 of a secondary arc e (which means that E' satisfies the conditions in Proposition 4.9) we can obtain from it a minimal partial covering of type 1 by following these steps:

1. Filter the elements in E' so that we keep only those belonging to $P^{u,v} \times P^{j,u}$ for some $j \in B_e$.

2. For each $j \in B_e$, if there is more than one arc in $P^{u,v} \times P^{j,u}$, take away all of these arcs except the one whose target node is in a higher position on $P^{j,u}$ of the target node of all the other ones.

Figure 4.11 shows some examples of secondary arcs and minimal partial coverings of type 1 of them.

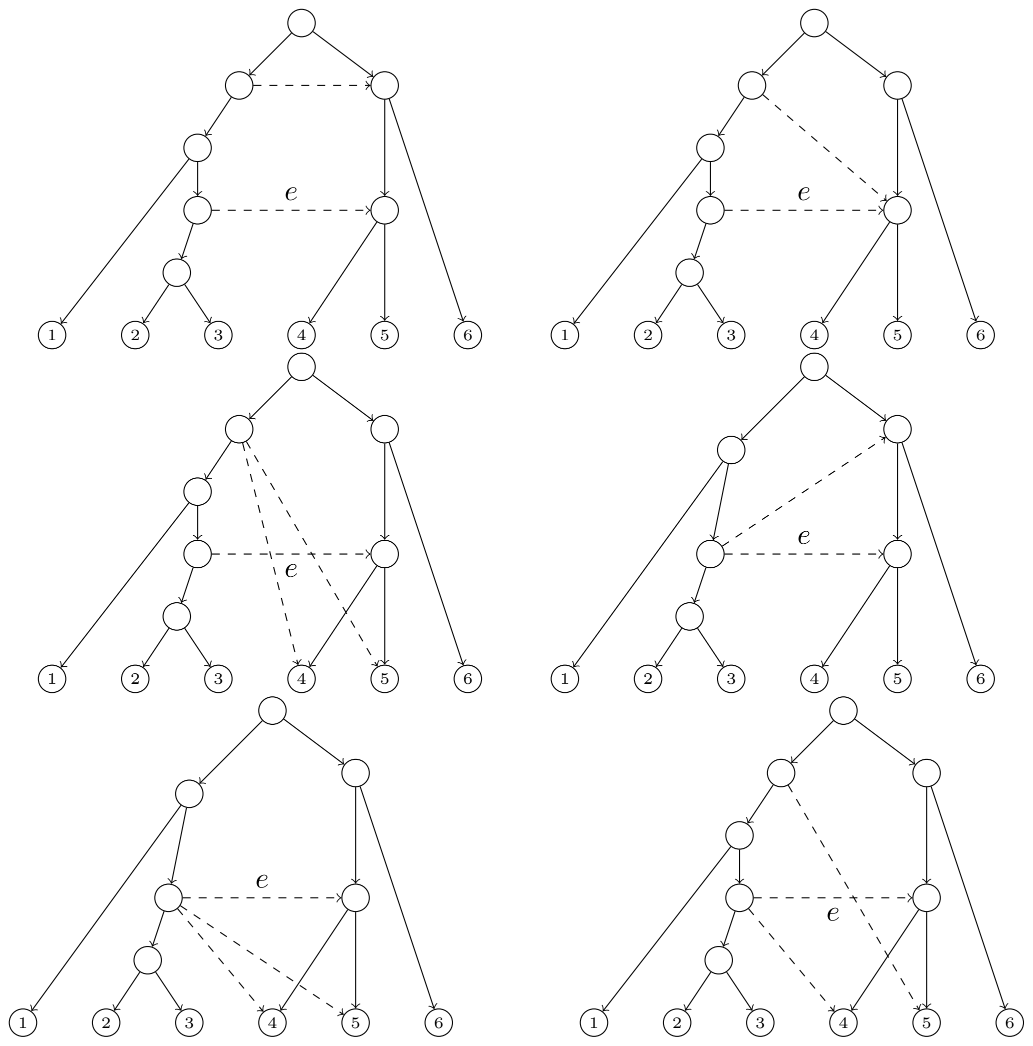

Figure 4.11: In each of these networks, the secondary arc labelled with e has a minimal partial covering of type 1 composed by the unlabelled ones.

Partial coverings of type 2

Now we analyze the set of secondary arcs that must be present in any partial covering of type 2 of a fixed secondary arc.

Proposition 4.13. *Let N be an LGT network and let $e = (u, v)$ be a secondary arc of N. Then, a set of secondary arcs $E' \subset E$ not containing e is a partial covering of type 2 of e if, and only if, for all $i \in A_e$ and for all $j \in B_e$, E' contains at least one secondary arc $(u', v') \in (P^{i,j} \setminus P^{p_u,v}) \times P^{j,u}$, where p_u denotes the principal parent of u.*

Proof. Let us assume that E' is a partial covering of e of type 2 and prove that it always contains arcs as in the statement of the proposition. Let $i \in A_e$ and $j \in B_e$ and take a a descendant leaf of p_u such that $LCA_{T_0}(a,i) = p_u$; in particular $a \in C_e$. Now, thanks to Proposition 4.3 we have that $B^2(j,i,a) \in \Gamma^2_e(N)$ and since E' is a covering of type 2 of e, then $B^2(j,i,a) \in \Gamma^2_{e'}(N)$ for some $e' = (u',v') \in E'$. Using again Proposition 4.3, we get that $j \in B_{e'}$ (v' is an ascendant of j), $i \in A_{e'}$ (u' is an ascendant of i) and $a \in C_{e'}$ (p_u is a principal ascendant of u'). Now, neither u' nor v' can be ancestors of (or equal to) $LCA_{T_0}(i,j) = LCA_{T_0}(u,v)$, since otherwise the extremes of e' would be connected by a principal path, against our assumption on LGT networks. Then, $u' \in (P^{i,j} \setminus P^{p_u,v})$ and $v' \in P^{j,u}$.

In order to prove the converse, let E' be a set of secondary arcs satisfying the condition above. In order to prove that it is a partial covering of $e = (u,v)$ of type 2, let $B^2(x,y,z) \in \Gamma^2_e(N)$, which implies that $x \in B_e$ (v is an ancestor of x), $y \in A_e$ (u is an ancestor of y) and $z \in C_e$ (u is a descendant of $LCA_{T_0}(y,z)$). Using the condition above we have that there exists $e' = (u',v')$ with $u' \in (P^{y,x} \setminus P^{p_u,v})$ and $v' \in P^{x,u}$. Since u' is an ancestor of y, $y \in A_{e'}$. Since v' is an ancestor of x we have that $x \in B_{e'}$. Since u' is an ancestor of y and it is not an ancestor of z ($\mathcal{C}_e \subseteq \mathcal{C}_{e'}$) we have that $z \in C_{e'}$. Finally, $B^2(x,y,z) \in \Gamma^2_{e'}(N)$ and this basic tri-lgt of type 2 is covered by some arc in E'. □

The minimal partial coverings of type 2 are more sophisticated than those of type 1. Hence, the characterization of minimal partial coverings of type 2 will be examined over a subclass of LGT networks (temporal consistent) which will be introduced in the next section. Now, we prove a necessary condition to achieve a minimal partial covering of type 2 of a redundant arc.

Lemma 4.14. *Let E' be a minimal partial covering of type 2 of a redundant arc $e = (u,v)$ and p_u be the principal parent of u. Then,*

(a) each arc (u',v') in E' satisfies that $(u',v') \in (P^{i,j} \setminus P^{p_u,v}) \times P^{j,u}$ for some $i \in A_e$ and $j \in B_e$,

(b) for each $i \in A_e$ and $j \in B_e$, there is at least one arc $(u',v') \in E'$ such that $(u',v') \in (P^{i,j} \setminus P^{p_u,v}) \times P^{j,u}$.

Proof. The second part of the statement follows directly from the previous proposition.

In order to prove the first part, let $E'' \subseteq E'$ be the set of arcs in E' that do satisfy the condition. The arcs in $(u',v') \in E' \setminus E''$ satisfy that $(u',v') \notin (P^{i,j} \setminus P^{p_u,v}) \times P^{j,u}$ (for all $i \in A_e$ and $j \in B_e$). Since E' satisfies the condition of Proposition 4.13, this same condition is also satisfied by E'' and hence E'' is a partial covering of type 2 of E. By the minimality assumption on E', we have that $E'' = E'$ and the result follows. □

Corollary 4.15. *If E' is a minimal partial covering of type 2 of an arc $e = (u,v)$, then $\pi_1(E') \subseteq \{u\} \cup \mathcal{A}_e$.*

Proof. The proof is a direct consequence of Lemma 4.14. □

Figure 4.12 shows some examples of secondary arcs and minimal partial coverings of type 2 of them.

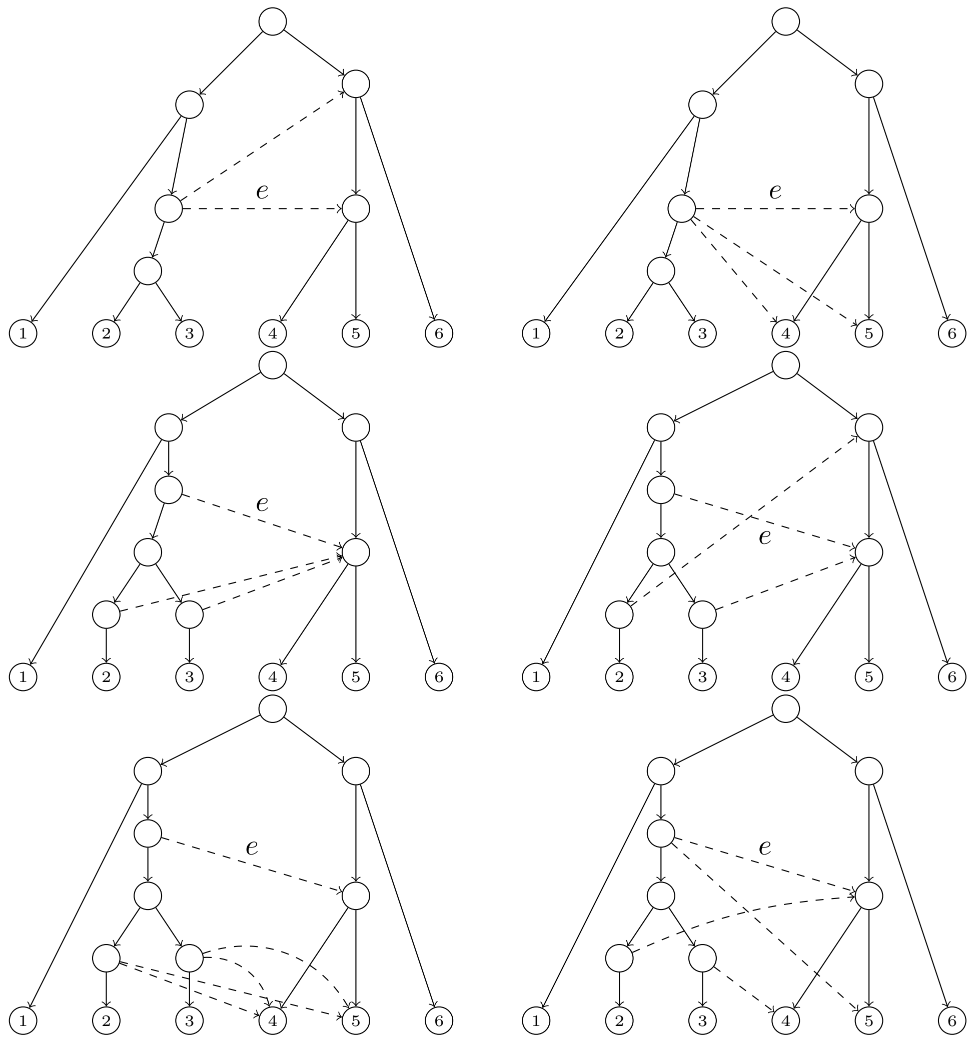

Figure 4.12: In each of these networks, the secondary arc labelled with e has a minimal partial covering of type 2 composed by the unlabelled ones.

Partial coverings of type 4

Now we analyze the set of secondary arcs that must be present in any partial covering of type 4 of a fixed secondary arc. Similarly to the partial coverings of type 2, we prove a necessary condition to achieve a minimal partial covering of type 4 of a redundant arc. However, we fully characterize these minimal coverings in the next section for time-consistent LGT networks.

In this subsection, for the sake of clarity, and in order not to lose the thread of the chapter, we omit the proofs of the results. This is done because the basic tri-lgt networks of type 4 are symmetric to those of type 2 and the proofs can be adapted following those for type 2.

Proposition 4.16. *Let N be an LGT network and let $e = (u, v)$ be a secondary arc of N. Then, a set of secondary arcs $E' \subset E$ not containing e is a partial covering of type 4 of e if, and only if, for all $i \in A_e$ and for all $j \in B_e$, E' contains at least one secondary arc $(u', v') \in P^{i,j} \times (P^{j,u} \setminus P^{p_v,u})$; where p_v denotes the principal parent of v.*

Lemma 4.17. *Let E' be a minimal partial covering of type 4 of a redundant arc $e = (u, v)$ and p_v be the principal parent of v. Then,* *(a) each arc (u', v') in E' satisfies that $(u', v') \in P^{i,j} \times (P^{j,u} \setminus P^{p_v,u})$ for some $i \in A_e$ and $j \in B_e$.*

(b) for each $i \in A_e$ and $j \in B_e$, there is at least one arc $(u', v') \in E'$ such that $(u', v') \in P^{i,j} \times (P^{j,u} \setminus P^{p_v,u})$.

Corollary 4.18. *If E' is a minimal partial covering of type 4 of an arc $e = (u, v)$, then $\pi_2(E') \subseteq \{v\} \cup \mathcal{B}_e$.*

Figure 4.13 shows some examples of secondary arcs and minimal partial coverings of type 4 of them.

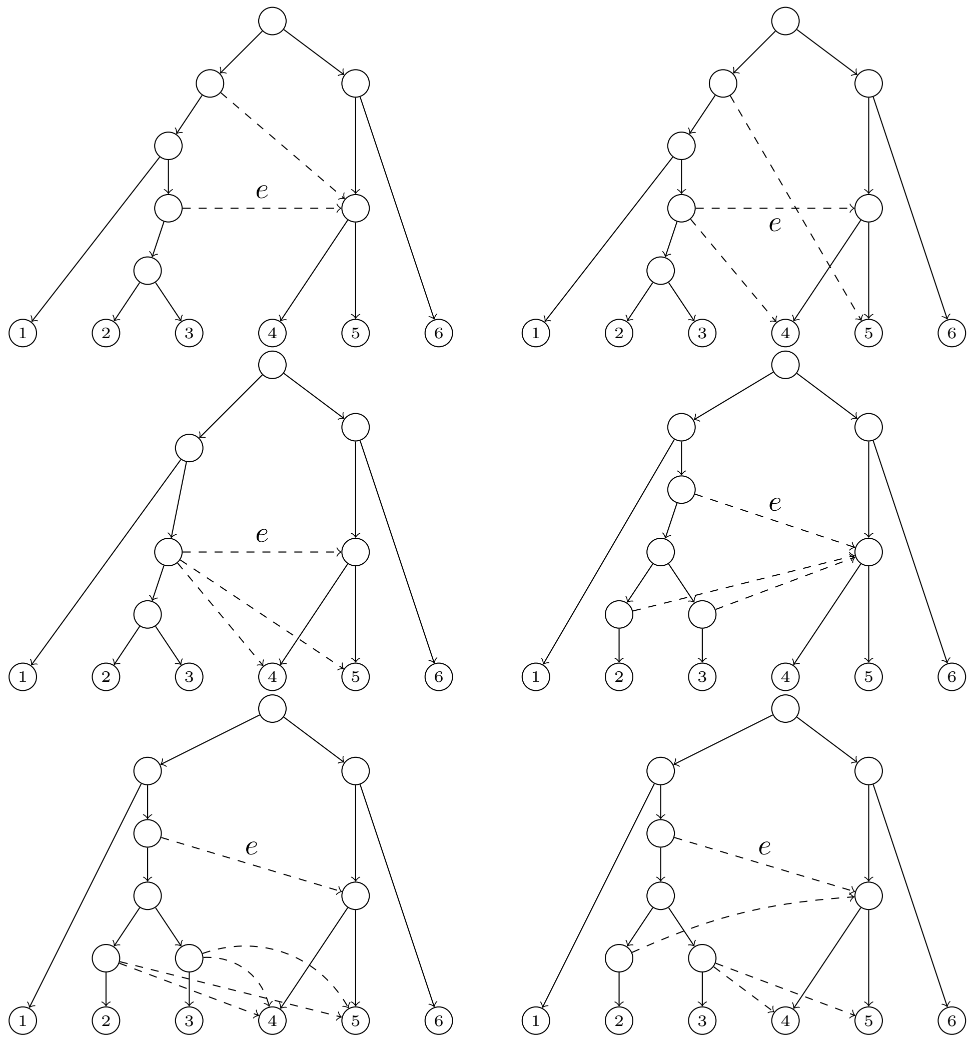

Figure 4.13: In each of these networks, the secondary arc labelled with e has a minimal partial covering of type 4 composed by the unlabelled ones.

Partial coverings of type 5

Similarly to partial coverings of type 1, here we provide a characterization of the minimal partial coverings of type 5 of secondary arcs. As in the previous subsection, since the basic tri-lgt networks of type 5 are symmetric to those of type 1, we omit the proofs of the results.

Proposition 4.19. *Let N be an LGT network and let $e = (u, v)$ be a secondary arc of N. Then, a set of secondary arcs $E' \subset E$ not containing e is a partial covering of type 5 of e if, and only if, for all $i \in A_e$, E' contains at least one secondary arc $(u', v') \in P^{i,v} \times P^{v,u}$.*

Lemma 4.20. *Let E' be a minimal partial covering of type 5 of a redundant arc $e = (u, v)$. Then,*

(a) each arc (u', v') in E' satisfies that $(u', v') \in P^{i,v} \times P^{v,u}$ for some $i \in A_e$,

(b) for each $i \in A_e$, there is at least one arc $(u', v') \in E'$ such that $(u', v') \in P^{i,v} \times P^{v,u}$.

Corollary 4.21. *If E' is a minimal partial covering of type 5 of an arc $e = (u, v)$, then $\pi_2(E') \subseteq \{v\} \cup \mathcal{D}_e$.*

Proposition 4.22. *Let N be an LGT network and $e = (u, v)$ a redundant arc in N. Then, a set E' of secondary arcs not containing e is a minimal partial covering of type 5 of e if, and only if, for each $i \in A_e$ there exists exactly one arc $e' = (u', v') \in E'$ with $(u', v') \in P^{i,v} \times P^{v,u}$.*

Figure 4.14 shows some examples of secondary arcs and minimal partial coverings of type 5 of them.

Partial coverings of type 3

The reason why we have left the study of partial coverings of type 3 for the final subsection is that this coverings depend, to a great extent, on the partial coverings of type 1, 2, 4 and 5.

We first analyze the set of secondary arcs that must be present in any partial covering of type 3 of a fixed secondary arc.

Proposition 4.23. *Let N be an LGT network and let $e = (u, v)$ be a secondary arc of N. Then, a set of secondary arcs $E' \subset E$ not containing e is a partial covering of type 3 of e if, and only if, for all $i \in A_e$ and $j \in B_e$, E' contains at least one secondary arc $(u', v') \in P^{i,v} \times P^{j,u}$.*

Proof. Let us assume that E' is a partial covering of type 3 of e and prove that it always contains arcs as in the statement of the proposition. Let $i \in A_e$, $j \in B_e$ and take a a leaf which is not a descendant leaf of $LCA_{T_0}(i, j) = LCA_{T_0}(u, v)$. Now, thanks to Proposition 4.3 we have that $B^3(a, i, j) \in \Gamma^3_e(N)$ and since E' is a covering of type 3 of e, then $B^3(a, i, j) \in \Gamma^3_{e'}(N)$ for some $e' = (u', v') \in E'$. Using again Proposition 4.3, we get that $j \in B_{e'}$ (v' is an ascendant of j), $i \in A_{e'}$ (i is a descendant of u') and $a \in E_{e'}$ (a is not a descendant of $LCA_{T_0}(u', v')$). Then, $u' \in P^{i,v}$ and $v' \in P^{j,u}$.

In order to prove the converse, let E' be a set of secondary arcs satisfying the condition above. In order to prove that it is a partial covering of $e = (u, v)$ of type 3, let $B^3(x, y, z) \in \Gamma^3_e(N)$, which implies that $x \in E_e$ (x is not a descendant of $LCA_{T_0}(u, v)$), $y \in A_e$ (u is an ancestor of y) and $z \in B_e$ (v is an ancestor of z). Using the condition above we have that there exists $e' = (u', v')$ with $u' \in P^{y,v}$ and $v' \in P^{z,u}$. Since y is a descendant of u' we have that $y \in A_{e'}$. Since z is a descendant of v' we have that $z \in B_{e'}$. Since $E_e = E_{e'}$, we have that $x \in E_{e'}$. Finally, $B^3(x, y, z) \in \Gamma^3_{e'}(N)$ and this basic tri-lgt of type 3 is covered by some arc in E'. □

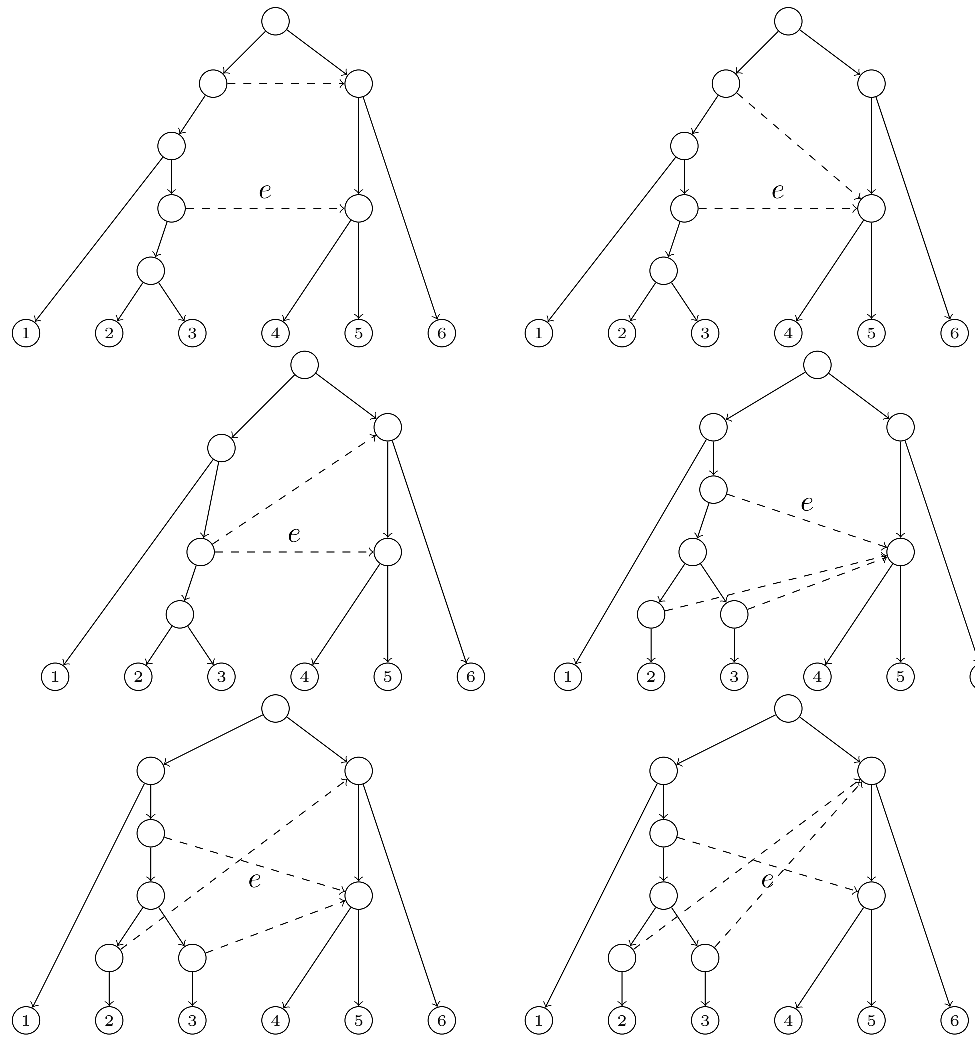

Figure 4.14: In each of these networks, the secondary arc labelled with e has a minimal partial covering of type 5 composed by the unlabelled ones.

Similarly to the partial coverings of type 2 and 4 we prove a necessary condition to achieve a minimal partial covering of type 3 of a redundant arc. However, we fully characterize these minimal coverings in the next section for time-consistent LGT networks.

Lemma 4.24. *Let E' be a minimal partial covering of type* 3 *of a redundant arc* $e = (u, v)$. *Then,*

(a) each arc (u', v') in E' satisfies that $(u', v') \in P^{i,v} \times P^{j,u}$ for some $i \in A_e$ and $j \in B_e$.

(b) for each $i \in A_e$ and $j \in B_e$, there is at least one arc $(u', v') \in P^{i,v} \times P^{j,u}$.

Proof. The proof of (a) and (b) is a direct consequence of Lemma 4.7 and Proposition 4.23, respectively. □

Note that, all minimal partial covering of type 1, 2, 4 and 5 are, particularly, a minimal partial covering of type 3. In general terms, the main difference between them lies on the relative position of the extreme nodes of the arcs of the covering respect to the nodes u and v. For instance, in a minimal partial covering of type 1, the source node of the arcs are either ascendant nodes of u or node u itself, see Corollary 4.11. Since the leaf occupying the first component in a basic tri-lgt network of type 3 is not a descendant of $LCA_{T_0}(u, v)$, the restriction on the relative position of the extremes disapears and, consequently, the arcs which form a minimal partial covering of type 3 are determined by Lemma 4.7.

Figure 4.15 shows an example of secondary arc and minimal partial covering of type 3 of it which is not a minimal partial covering of types 1, 2, 4 and 5.

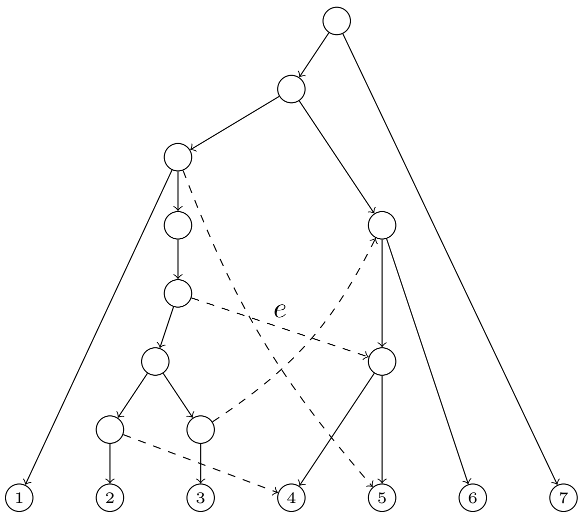

Figure 4.15: In the network, the secondary arc labelled with e has a minimal partial covering of type 3 composed by the unlabelled ones.

Proposition 4.25. *Given an LGT network, Lemmas 4.10, 4.14, 4.17, 4.20 and 4.24, allow us to detect the redundant arcs of the network.*

Proof. Lemmas 4.10, 4.14, 4.17, 4.20 and 4.24 allow us to detect if there is a partial covering of type 1, 2, 4, 5 and 3 for a fixed secondary arc, respectively. Consequently, they allow to detect if there is, or there is not, a covering for each secondary arc in the network. □

Since the decomposition $\Gamma(N)$ of an LGT network contains the triplets associated to its principal subtree $T_0(N)$, it follows that we can recover (with unicity) the reduced principal subtree $\widetilde{T_0}(N)$ from $\Gamma(N)$. Notice, however, that the non-reduced principal subtree cannot be recovered from $\Gamma(N)$, as the following example shows.

Example 4.10. *Both LGT networks depicted in Figure 4.16 have the same tri-lgt decomposition. The network on the right side is the same as the one on the left side, except that the unique redundant arc has been deleted. However, their respective (non-reduced) principal subtrees are not isomorphic.*

Remark 4.26. *Recall that if two different arc-node LGT networks have isomorphic reduced principal subtrees, then the nodes and arcs of both trees can be identified. Hence, when two LGT networks have the same tri-lgt decomposition, we can also assume that they also share the reduced principal subtree and hence the only differences between two such networks can be their respective sets of secondary arcs. That is, if we have N and N' two LGT networks such that $\Gamma(N) = \Gamma(N')$, say $N = T + F$ and $N' = T + F'$, where $T = \widetilde{T_0}(N) = \widetilde{T_0}(N')$, are due to differences between N and N'. More precisely, let $e = (u, v) \in E_s(N)$, let u_1 and u_2 be the (principal) parent and (principal) child of u respectively, and let $\hat{e} = (u_1, u_2) \in E(T)$. Note that $v \in V(T)$ and $(\hat{e}, v) \in F$. Then,*

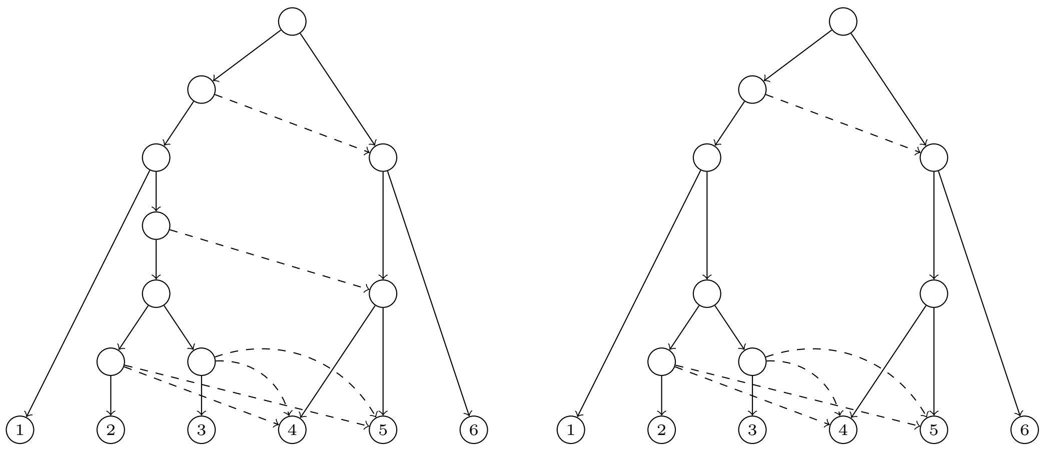

Figure 4.16: Two LGT networks that share the tri-lgt decomposition but whose principal subtrees are not isomorphic.

we say that N and N' share *the secondary arc e, if $(\hat{e}, v) \in F \cap F'$. Also, the network obtained for* adding *e to N' is $T + F \cup \{(\hat{e}, v)\}$. Also, we say that the secondary e in N is* redundant *in N' if either $(\hat{u}, v) \in F'$ or if e is redundant in the network $T + F \cup \{(\hat{e}, v)\}$.*

Proposition 4.27. *Two LGT networks have the same tri-lgt decomposition if and only if each secondary arc in one of the networks is redundant in the other one and viceversa.*

Proof. Let N_1 and N_2 be two LGT networks with the same tri-lgt decomposition. Particularly, both networks have the same reduced principal subtree. Suppose for the sake of contradiction that e is a secondary arc in N_1 that is not redundant in N_2. Then, there exists at least one basic tri-lgt network in $\Gamma_e(N_1) \subseteq \Gamma(N_1)$ that is not present in $\Gamma(N_2)$. Therefore $\Gamma(N_1) \neq \Gamma(N_2)$.

Conversely, if each arc in N_1 is redundant in N_2 and each arc in N_2 is redundant in N_1, then all basic tri-lgt networks displayed by N_1 are also displayed by N_2 and viceversa. Consequently, $\Gamma(N_1) = \Gamma(N_2)$. $\square$

Notice that two networks with the same reduced principal subtree but disjoint sets of secondary arcs can have the same tri-lgt decomposition, as the following example shows.

Example 4.11. *Let N_1 and N_2 be two LGT networks depicted in Figure 4.17. Both networks have the same reduced principal subtree but do not share any secondary arc. However, $\Gamma(N_1) = \Gamma(N_2)$. Each arc in N_1 has a covering in N_2 and viceversa. The coverings (in N_2) of the arcs (of N_1) $e_1^1, e_2^1, e_3^1, e_4^1$ and e_5^1 are $\{e_1^2, e_2^2, e_3^2, e_4^2\}$, $\{e_1^2, e_3^2\}$, $\{e_1^2, e_4^2\}$, $\{e_2^2, e_4^2\}$ and $\{e_2^2, e_3^2\}$, respectively. On the other hand, the coverings of the arcs e_1^2, e_2^2, e_3^2 and e_4^2 are $\{e_1^1, e_2^1, e_3^1\}$, $\{e_1^1, e_4^1, e_5^1\}$, $\{e_1^1, e_2^1, e_5^1\}$ and $\{e_1^1, e_3^1, e_4^1\}$, respectively.*

4.5 Temporal consistency and minimum LGT network

In previous sections we have investigated the decomposition of LGT networks in basic tri-lgt networks, and found out that this decomposition does not uniquely determine the network, one of the reasons being the presence of redundant arcs. We have fully studied the coverings of a particular redundant arc, and found out that there always exist minimal

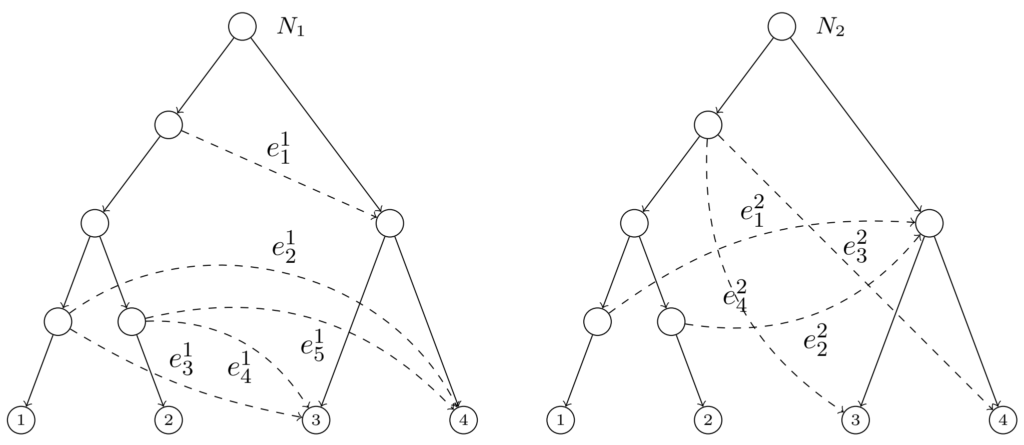

Figure 4.17: Two LGT networks N_1 and N_2 with the same tri-lgt decomposition.

coverings, but these coverings may not be unique. In this section we present a class of LGT networks where the possible coverings of a redundant arc admit a unique minimum covering and, more importantly, such that they are fully characterized by its decomposition in tri-lgt networks.

We say that an LGT network $N = (V, E)$, with decomposition in principal and secondary arcs as $E = E_p \sqcup E_s$, is *time-consistent* if there exists a time-assignment $\delta : V \to \mathbb{N}$ such that:

- if $(u, v) \in E_p$, then $\delta(u) < \delta(v)$, and
- if $(u, v) \in E_s$, then $\delta(u) = \delta(v)$.

This definition of time-consistency for LGT networks is slightly different from other definitions used in other kinds of phylogenetic networks [Baroni et al. (2006); Cardona et al. (2008c)], given that the time-assignment of a parent of a reticulation node must be the same time-assignment as that the node itself only if its parent is the principal one; and similar to the one used in Górecki (2004); see Section 1.7. We denote by (N, δ) a time-consistent LGT network N with time-assignment δ. See Example 4.12.

Example 4.12. *Figure 4.18 shows a depiction of a time-consistent LGT network, together with the time-assignment for each node.*

Now we prove that in an LGT network (N, δ) secondary arcs cannot "intersect".

Lemma 4.28. *Let (N, δ) be a time-consistent LGT network, and (u, v) and (u', v') two different secondary arcs of N. If u' is a principal descendant of u (possibly with $u' = u$), then v' is not a principal ascendant of v (excluding also the case $v' = v$). Symmetrically, if v' is a principal descendant of v (possibly with $v' = v$), then u' is not a principal ascendant of u (excluding also the case $u' = u$).*

Proof. Since (u, v) and (u', v') are secondary arcs, we have that $\delta(u) = \delta(v)$ and $\delta(u') = \delta(v')$. Assume that u' is a descendant of u, and hence we have that $\delta(u') \geq \delta(u)$. If v' was an ascendant of v, we would have that $\delta(v') \leq \delta(v)$ and hence $\delta(u) = \delta(v) = \delta(u') = \delta(v')$. Since the paths $u \rightsquigarrow u'$ and $v' \rightsquigarrow v$ are principal the only possibility is that $(u, v) = (u', v')$, against our hypothesis. The other part of the statement follows the same way. □

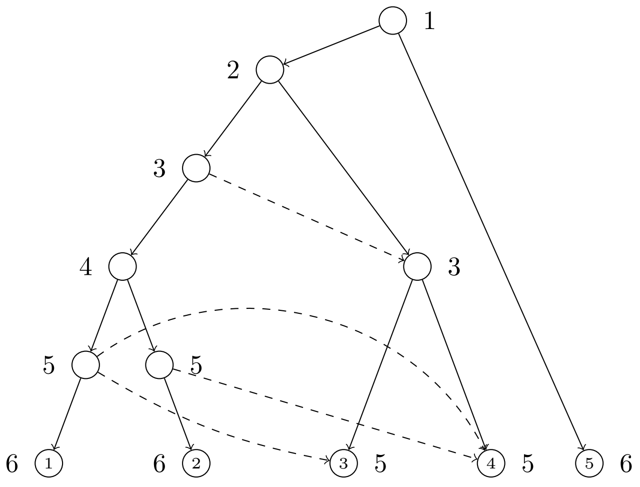

Figure 4.18: A time-consistent LGT network.

Let i and j be two leaves in an LGT network N. We denote by $S_N^{i\to j}$ the set of secondary arcs whose source and target are, respectively, in the principal paths $LCA_{T_0(N)}(i,j) \rightsquigarrow i$ and $LCA_{T_0(N)}(i,j) \rightsquigarrow j$. Let w be a node that is a principal descendant of $LCA_{T_0(N)}(i,j)$. In these conditions, we denote by $l_N^{i,j}(w)$ the number of non-principally elementary nodes in the principal path $LCA_{T_0(N)}(i,j) \rightsquigarrow w$, excluding the extremes.

We suppose that N is time-consistent. If $S_N^{i\to j}$ is not empty, let u^* be the source of an arc in $S_N^{i\to j}$ satisfying that:

$$l_N^{i,j}(u^*) \leq l_N^{i,j}(u), \forall u \in \pi_1(S_N^{i\to j})$$

and let v^* be the target of an arc in $S_N^{i\to j}$ satisfying that:

$$l_N^{i,j}(v^*) \leq l_N^{i,j}(v), \forall v \in \pi_2(S_N^{i\to j}).$$

Thanks to Lemma 4.28, u^* and v^* are well-defined, as well as the secondary arcs having them as source and target, respectively: there cannot be two different secondary arcs with the same source (resp. target) and whose targets (resp. sources) are connected by a principal path. Again by Lemma 4.28 the arc whose source is u^* and the arc whose target is v^* must be the same. We shall refer to this arc as $\min(S_N^{i\to j})$. Note that, if we do not impose the time-consistency constraint, this last statement can be false. See for example the network N_2 depicted in the right side of Figure 4.17 considering the leaves 1 and 3. Note that u^* is the source of the arc e_4^2 and the node v^* is the target of the arc e_1^2. However, the arc (u^*, v^*) is not present in N_2.

Similarly, taking nodes with maximum (instead of minimum) value of $l_N^{i,j}(u)$ we can define the arc $\max(S_N^{i\to j})$. Note that when $S_N^{i\to j}$ has a single arc, then $\min(S_N^{i\to j}) = \max(S_N^{i\to j})$.

These definitions are illustrated in Example 4.13.

Example 4.13. *Let N be the time-consistent LGT network depicted in Figure 4.19. Then, we have that: $x = LCA_{T_0(N)}(1,3)$, $y = LCA_{T_0(N)}(1,2)$, $r = LCA_{T_0(N)}(2,5)$, $S_N^{1\to 3} = \{e_1, e_4\}$, $S_N^{2\to 4} = \{e_1, e_3\}$, $l_N^{1,3}(a) = 0$, $l_N^{1,3}(b) = 0$, $l_N^{1,3}(c) = 1$, $l_N^{1,3}(3) = 1$, $\min(S_N^{1\to 3}) = e_1$, $\min(S_N^{1\to 4}) = e_2$, $\max(S_N^{1\to 3}) = e_4$.*

Proposition 4.29. *Let N_1 and N_2 be two time-consistent LGT networks defined on S such that $\Gamma(N_1) = \Gamma(N_2)$. Then, for each pair of leaves $i, j \in S$:*

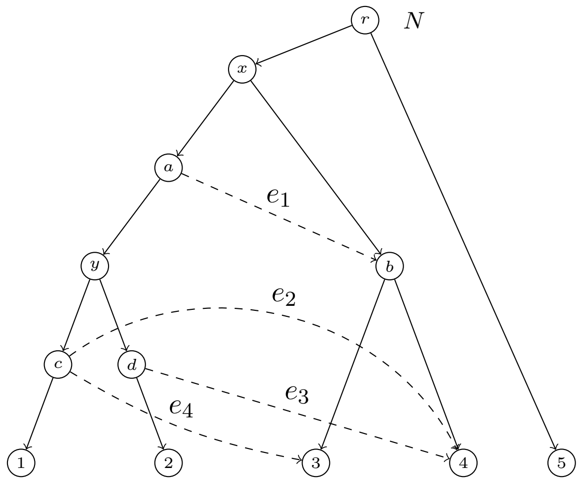

Figure 4.19: The LGT network used in Example 4.13.

- $\min(S^{i\to j}_{N_1}) = \min(S^{i\to j}_{N_2})$, *and*
- $\max(S^{i\to j}_{N_1}) = \max(S^{i\to j}_{N_2})$,

where the equality of arcs is taken according to Remark 4.26.

Proof. Let $\widetilde{T}_0$ be the reduced principal subtree common to N_1 and N_2. We suppose that $\min(S^{i\to j}_{N_1}) \neq \min(S^{i\to j}_{N_2})$ and we prove that $\Gamma(N_1) \neq \Gamma(N_2)$.

Let $e_1 = (u_1, v_1) = \min(S^{i\to j}_{N_1})$ and $e_2 = (u_2, v_2) = \min(S^{i\to j}_{N_2})$. Let $(n_1, m_1) = (l^{i,j}_{N_1}(u_1), l^{i,j}_{N_1}(v_1))$ and $(n_2, m_2) = (l^{i,j}_{N_2}(u_2), l^{i,j}_{N_2}(v_2))$. Note that the condition $\min(S^{i\to j}_{N_1}) \neq \min(S^{i\to j}_{N_2})$ (under the identification in Remark 4.26 from which both arcs $\min(S^{i\to j}_{N_1})$ and $\min(S^{i\to j}_{N_2})$ can be seen as pairs formed by an arc and a node in $\widetilde{T}_0$, that is as nodes in $\widetilde{T}_0$) is equivalent to $(n_1, m_1) \neq (n_2, m_2)$.

We analyse the different cases:

- If $n_1 \neq n_2$, assume without loss of generality that $n_2 < n_1$. Let u be the principal parent of u_1 in N_1. Note that $u \in \widetilde{T}_0$ and, since $n_2 < n_1$, u is a principal descendant of u_2. Let k be a principal descendant leaf of u such that $LCA_{T_0(N)}(i, k) = u$ (notice that k is simply a leaf descendant of the child of u different from u_1). Then, since $j \in B_{e_2}$ and $i, k \in A_{e_2}$, we have $B^1(j, i, k) \in \Gamma(N_2)$. We prove now that $B^1(j, i, k) \notin \Gamma(N_1)$. If $B^1(j, i, k) \in \Gamma(N_1)$, then by Proposition 4.3, there must exist $e \in S^{i\to j}_{N_1}$ such that $j \in B_e$ and $i, k \in A_e$. This implies that the source of e must be in $LCA_{T_0(N)}(i, j) \rightsquigarrow u$; then, $e = \min(S^{i\to j}_{N_1})$ which contradicts the assumption that $e_1 = \min(S^{i\to j}_{N_1})$.
- If $m_1 \neq m_2$, assume without loss of generality that $m_2 < m_1$. Let v be the principal parent of v_1 in N_1. Note that $v \in \widetilde{T}_0$ and, since $m_2 < m_1$, v is a principal descendant of v_2 (possibly with $v = v_2$). Let k be a principal descendant leaf of v such that $LCA_{T_0(N)}(j, k) = v$. Then, since $i \in A_{e_2}$ and $j, k \in B_{e_2}$, we have $B^5(i, j, k) \in \Gamma(N_2)$. We prove now that $B^5(i, j, k) \notin \Gamma(N_1)$. If $B^5(i, j, k) \in \Gamma(N_1)$, then by Proposition 4.3, there must exist $e \in S^{i\to j}_{N_1}$ such that $i \in A_e$ and $j, k \in B_e$. This implies that the target of e must be in $LCA_{T_0(N)}(i, j) \rightsquigarrow v$; then, $e = \min(S^{i\to j}_{N_1})$ which contradicts the assumption that $e_1 = \min(S^{i\to j}_{N_1})$.

The proof of the equality of maximum arcs follows analogously. □

Coverings in time-consistent LGT networks

In Section 4.4 we studied the different coverings of a redundant arc through the analysis of partial coverings. In particular, we studied minimal partial coverings but left some cases not fully characterized. Now, we fully characterize those minimal coverings for the case of time-consistent LGT networks.

Lemma 4.30. *Let N be a time-consistent LGT network and $e = (u, v)$ a redundant arc in N. Then, a set E' of secondary arcs not containing e is a minimal partial covering of type 1 and type 5 of e if, and only if, there exists exactly one arc $e' = (u', v') \in E'$ with $(u', v') \in P^{p_u,v} \times P^{p_v,u}$, where p_u and p_v are the principal parents of u and v in N, respectively.*

Proof. By Proposition 4.12, each minimal partial covering of type 1 is formed by a set E' of secondary arcs such that for each $j \in B_e$ there is one and only one $(u', v') \in E'$ with $(u', v') \in P^{u,v} \times P^{j,u}$. Since N is time-consistent, each node u' is different from u; then, each u' lies in $P^{p_u,v}$. Moreover, since $u' \in P^{p_u,v}$, by Lemma 4.28, each node v' must be a proper ascendant of v; then, $v' \in P^{p_v,u}$. Hence, the arc (u', v') does not depend on the leaf j chosen. Then, a single arc $(u', v') \in P^{p_u,v} \times P^{p_v,u}$ is enough to be a partial covering (which is obviously minimal) of type 1 in N since each $j \in B_e$ is a descendant of v'.

By Proposition 4.22, each minimal partial covering of type 5 is formed by a set E' of secondary arcs such that for each $i \in A_e$ there is one and only one $(u', v') \in E'$ with $(u', v') \in P^{i,v} \times P^{v,u}$. Since N is time-consistent, each node v' is different from v; then, each v' lies in $P^{p_v,u}$. Moreover, since $v' \in P^{p_v,u}$, by Lemma 4.28, each node u' must be a proper ascendant of u; this is, $u' \in P^{p_u,v}$. Hence the arc (u', v') is not dependant on the leaf i and a single arc $(u', v') \in P^{p_u,v} \times P^{p_v,u}$ is sufficient to be a partial covering of type 5 in N (which will be minimal) since each $i \in A_e$ is a descendant of u'. □

Lemma 4.31. *Let N be a time-consistent LGT network and $e = (u, v)$ a redundant arc in N. Then, a set E' of secondary arcs not containing e is a minimal partial covering of type 2 and type 4 of e if, and only if, for each $i \in A_e$ and $j \in B_e$ there exists exactly one arc $e' = (u', v') \in E'$ with $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$.*

Proof. We first prove the proposition for the covering of type 2.

Let us assume that E' is a minimal partial covering of type 2 of e. Then Proposition 4.13 implies that for each $i \in A_e$ and $j \in B_e$ there exists at least one arc $e' = (u', v') \in E'$ with $(u', v') \in (P^{i,j} \setminus P^{p_u,v}) \times P^{j,u}$. Since N is time-consistent, given that $u' \in P^{i,j} \setminus P^{p_u,v}$, by Lemma 4.28, u' must be a descendant of u; therefore v' must be a descendant of v. Then $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$. Let us assume that for a given i and j there exist at least two secondary arcs $e' = (u', v')$ and $e'' = (u'', v'')$ satisfying the condition. Given the time-consistency of N, (without loss of generality) we have that u' is an ascendant of u'' and v' is an ascendant of v''. Hence $A_{e''} \subseteq A_{e'}$ and $B_{e''} \subseteq B_{e'}$. Moreover, since u' and u'' are descendants of u, $C_e \subseteq C_{e'}$ and $C_e \subseteq C_{e''}$. Using Proposition 4.3 we get that $\Gamma^2_{e''} \cap \Gamma^2_e \subseteq \Gamma^2_{e'} \cap \Gamma^2_e$. Now, $E' \setminus \{e''\}$ is still a partial covering of type 2 of e, against our assumption on the minimality of E', hence we have reached a contradiction and for each $i \in A_e$ and $j \in B_e$ there is exactly one arc $e' \in E'$ satisfying the condition above.

In order to prove the converse, thanks to Lemma 4.14, we only have to prove that if we remove an arc $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$ for some $i \in A_e$ and $j \in B_e$, then, the resulting set of arcs is not a partial covering of type 2 of e. This is easy to prove because otherwise, taking a a descendant leaf of u such that $LCA_{T_0(N)}(a, i) = p_u$, the basic tri-lgt network $B^2(j, i, a) \in \Gamma_e^2$ would not be covered by any arc. Consequently, the resulting set of arcs would not be a partial covering of type 2 of e.

We now prove the proposition for the covering of type 4.

Let us assume that E' is a minimal partial covering of type 4 of e. Then Proposition 4.16 implies that for each $i \in A_e$ and $j \in B_e$ there exists at least one arc $e' = (u', v') \in P^{i,j} \times (P^{j,u} \setminus P^{p_v,u})$. Since N is time-consistent, given that $v' \in P^{j,u} \setminus P^{p_v,u}$, by Lemma 4.28, v' must be a descendant of v; therefore u' must be a descendant of u. Then $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$. Let us assume that for a given i and j there exist at least two secondary arcs $e' = (u', v')$ and $e'' = (u'', v'')$ satisfying the condition. Given the time-consistency of N, (without loss of generality) we have that u' is an ascendant of u'' and v' is an ascendant of v''. Hence $A_{e''} \subseteq A_{e'}$ and $B_{e''} \subseteq B_{e'}$. Moreover, since v' and v'' are descendants of v, $D_e \subseteq D_{e'}$ and $D_e \subseteq D_{e''}$. Using Proposition 4.3 we get that $\Gamma_{e''}^4 \cap \Gamma_e^4 \subseteq \Gamma_{e'}^4 \cap \Gamma_e^4$. Now, $E' \setminus \{e''\}$ is still a partial covering of type 4 of e, against our assumption on the minimality of E', hence we have reached a contradiction and for each $i \in A_e$ and $j \in B_e$ there is exactly one arc $e' \in E'$ satisfying the condition above.

In order to prove the converse, thanks to Lemma 4.17, we only have to prove that if we remove an arc $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$ for some $i \in A_e$ and $j \in B_e$, then, the resulting set of arcs is not a partial covering of type 4 of e. This is easy to prove because otherwise, taking a a descendant leaf of v such that $LCA_{T_0(N)}(a, j) = p_v$, the basic tri-lgt network $B^4(i, j, a) \in \Gamma_e^4$ would not be covered by any arc. Consequently, the resulting set of arcs would not be a partial covering of type 4 of e. □

Lemma 4.32. *Let N be a time-consistent LGT network and $e = (u, v)$ a redundant arc in N. Then, a set E' of secondary arcs not containing e is a minimal partial covering of type 3 of e if, and only if, either there exists exactly one arc $e' \in E'$ with $e' = (u', v') \in P^{p_u,v} \times P^{p_v,u}$ or for each $i \in A_e$ and $j \in B_e$ there exists exactly one arc $e' = (u', v') \in E'$ with $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$.*

Proof. By Lemma 4.24, each minimal partial covering of type 1, 2, 4 and 5 is a minimal partial covering of type 3. As per the Lemmas 4.30 and 4.31, we know that both the single arc $(u', v') \in P^{p_u,v} \times P^{p_v,u}$, and the set E' formed by one and only one arc $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$ for each $i \in A_e$ and $j \in B_e$, are minimal partial coverings of type 3. By the time consistency of N, there is not any other kind of minimal partial coverings of type 3 other than those. Each arc in the covering must have either its source and its target nodes that are ascendant nodes of u and v, respectively, or its source and its target nodes that are descendant nodes of u and v, respectively. Then, by Lemma 4.28, any other kinds of coverings of type 3 are not allowed. □

Lemma 4.33. *In a time-consistent LGT network, a minimal partial covering of type 1 and 5 of a redundant arc is never a minimal partial covering of type 2 or 4 for the same arc, and viceversa.*

Proof. Let $e = (u, v)$ be the considered redundant arc. Then, by Lemma 4.30, the source and target nodes of the arcs in a minimal partial covering of type 1 and 5 of e, are ascendant nodes of u and v, respectively. On the other hand, by Lemma 4.31, the source and target nodes of the arcs in a minimal partial covering of type 2 and 4 are descendant nodes of u and v, respectively. □

Proposition 4.34. *Let N be a time-consistent LGT network, $e = (u, v)$ a redundant arc in N and p_u, p_v the principal parents of u and v in N, respectively. Then, a set E' of secondary arcs not containing e is a minimal covering of e if, and only if, E' contains exactly:*

- *one arc $(u', v') \in P^{p_u,v} \times P^{p_v,u}$, and*
- *for each $i \in A_e$ and $j \in B_e$, exactly one arc $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$.*

Proof. The result is a direct consequence of Lemmas 4.30, 4.31, 4.32 and 4.33. □

Lemma 4.35. *Let N be a time-consistent LGT network and $e = (u, v)$ a redundant arc in N. Then, the set of arcs formed by $\min(S_N^{i\to j})$ and $\max(S_N^{i\to j})$ for all pairs $i \in A_e$, $j \in B_e$ is a covering of e.*

Proof. The proof is a direct consequence of Proposition 4.34, since fixed $i \in A_e$ and $j \in B_e$ we have that $\min(S_N^{i\to j}) \in P^{p_u,v} \times P^{p_v,u}$ and $\max(S_N^{i\to j}) \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$. □

We call the covering in Lemma 4.35 the *min-max* covering of $e = (u, v)$. Now we prove that the min-max covering may not be a minimal covering as Example 4.14 shows, but it is possible to obtain a such minimal covering removing some specific arcs from it.

First, note that, the arc $\min(S_N^{i\to j})$ is exactly the same for each pair i, j with $i \in A_e$ and $j \in B_e$. This results from the relative position of the source and the target nodes of $\min(S_N^{i\to j})$, which are ascendant nodes of u and v, respectively. The situation is different, in general, for arcs of the form $\max(S_N^{i\to j})$ where different pair of leaves i, j give rise to different $\max(S_N^{i\to j})$. However, for some relative positions of the extremes of the nodes of $\max(S_N^{i\to j})$, superfluous arcs in the covering can appear. This is, we consider two secondary arcs $e' = (u', v') = \max(S_N^{i_1\to j_1})$ and $e'' = (u'', v'') = \max(S_N^{i_2\to j_2})$ in the min-max covering when there are principal paths $u \rightsquigarrow u' \rightsquigarrow u''$ and $v \rightsquigarrow v' \rightsquigarrow v''$. In this conditions, $\Gamma_e^2(N) \cap \Gamma_{e''}^2(N) \subseteq \Gamma_e^2(N) \cap \Gamma_{e'}^2(N)$ and $\Gamma_e^4(N) \cap \Gamma_{e''}^4(N) \subseteq \Gamma_e^4(N) \cap \Gamma_{e'}^4(N)$. Then, the arc e'' can be removed from the min-max covering in order to obtain a minimal partial covering. The same procedure can be followed iteratively for each pair of secondary arcs under the same conditions. Finally, in the remaining set of arcs, none of these can be removed because the arcs of the form $\max(S_N^{i\to j})$ are the only arcs in the min-max which cover the basic tri-lgt-nets of the form $B^2(j, i, \cdot)$ and $B^4(i, j, \cdot)$.

Example 4.14. *Let N be the time-consistent LGT network depicted in Figure 4.20. The set $\{e_1, e_2, e_3, e_4, e_5, e_6\}$ is the min-max covering of arc e. This is formed by the arcs $\min(S_N^{i\to j})$ and $\max(S_N^{i\to j})$ for each pair $i \in A_e = \{2, 3, 4\}$ and $j \in B_e = \{5, 6, 7\}$. This set is not a minimal covering of e. Note that the source and the target of the arc $e_2 = \max(S_N^{2\to 5})$ are, respectively, ascendant nodes of the source and the target of arc $e_6 = \max(S_N^{3\to 5})$. Once removed, we obtain the set $\{e_1, e_2, e_3, e_4, e_5\}$, which is a minimal covering of e.*

Proposition 4.36. *Let N be a time-consistent LGT network. Then, a secondary arc e is non-redundant in N if and only if there exists a pair of leaves i and j such that either $e = \min(S_N^{i\to j})$ or $e = \max(S_N^{i\to j})$.*

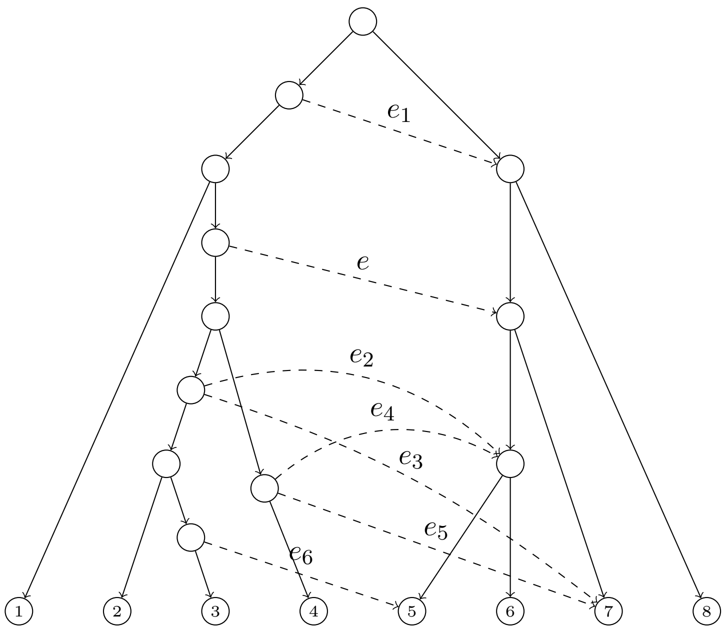

Figure 4.20: A time-consistent LGT network with a minimal covering of the secondary redundant arc e.

Proof. Let $e = (u, v)$ be a secondary arc in N. We first prove that if for each pair of leaves i, j, we have that $e \neq \min(S_N^{i\to j})$ and $e \neq \max(S_N^{i\to j})$, then e is a redundant arc in N. We can restrict our attention to the arcs in $S_N^{i\to j}$ with $i \in A_e$ and $j \in B_e$. Then, for each $i \in A_e$ and $j \in B_e$, in the set $S_N^{i\to j}$ there are, at least, three different arcs: $\min(S_N^{i\to j})$, e and $\max(S_N^{i\to j})$. By Lemma 4.35, the set of secondary arcs composed by $\min(S_N^{i\to j})$ and $\max(S_N^{i\to j})$ for each $i \in A_e$ and $j \in B_e$, is a covering of e. Consequently, e is a redundant arc.

To prove the converse, let i_f and j_f be a pair of fixed leaves such that either $e = \min(S_N^{i_f\to j_f})$ or $e = \max(S_N^{i_f\to j_f})$. We consider the case where $e = \max(S_N^{i_f\to j_f})$ (the other case can be proved similarly). By Proposition 4.34, in order to be e a redundant arc, there has to be in each of its coverings at least one arc $(u', v') \in (P^{i,j} \setminus P^{u,v}) \times (P^{j,u} \setminus P^{v,u})$ for each $i \in A_e$ and $j \in B_e$. Particularly, since $i_f \in A_e$ and $j_f \in B_e$, there must be in that covering an arc $(u'_f, v'_f) \in (P^{i_f,j_f} \setminus P^{u,v}) \times (P^{j_f,u} \setminus P^{v,u})$. Then u'_f is a descendant node of u and v'_f is a descendant of v; hence, the existence of the arc (u'_f, v'_f) contradicts our assumption on the maximality of e. □

Minimum LGT network

Given N an LGT network, we call its *minimum* network, denoted by $\mathrm{Min}(N)$, the subnetwork obtained from N by first removing all redundant secondary arcs and then suppressing the elementary nodes, if any. If there are no redundant arcs in a network, the minimum network coincides with the original one: $\mathrm{Min}(N) = N$. In Figures 4.7 and 4.16, the network depicted in the right side is the minimum network of the one depicted in the left.

Notice that, in the given procedure to obtain the minimum network, all the redundant arcs, without exception and all at once, are removed. Hence, the resulting network does not depend on the order of removal of the redundant arcs.

Notice also that for a generic LGT network N, the tri-lgt decomposition of N does not necessarily coincide with that of $\mathrm{Min}(N)$, see Example 4.7. We shall now prove that if we restrict to time-consistent LGT networks, its tri-lgt decomposition coincides with the tri-lgt decomposition of its minimum network.

Proposition 4.37. *Let N be a time-consistent LGT network and R a set of redundant arcs in N. Then $\Gamma(N) = \Gamma(N \setminus R)$.*

Proof. Since R is a set of redundant arcs in N, by Proposition 4.6, we know that $\Gamma(N) = \Gamma(N \setminus R)$ if and only if for each arc in R, there exists a covering of this arc that does not contain any arc of R. For the sake of contradiction, let us assume that there exists $e \in R$ such that all its coverings contain some other arc of R. That is, if E' is an arbitrary covering of e, then there is some arc $e^* \neq e$ with $e^* \in E' \cap R$. Using Lemma 4.35, take as covering E' of e its min-max covering. Then, there is some $e^* \neq e$ such that $e^* \in E' \cap R$. This means that there is a pair of leaves $i \in A_e$ and $j \in B_e$ with either $e^* = \min(S_N^{i \to j})$ or $e^* = \max(S_N^{i \to j})$. Hence, by Proposition 4.36, the arc e^* is not redundant arc, against our hypothesis. $\square$

As a direct consequence of this Proposition we get the following result.

Corollary 4.38. *Let N be a time-consistent LGT network. Then, $\Gamma(\mathrm{Min}(N)) = \Gamma(N)$.*

Basic tri-lgt networks determine time-consistent LGT networks

In this subsection we prove the main result of this chapter, which allows us to fully characterize when time-consistent LGT networks have the same tri-lgt decomposition and give a criterion to decide isomorphism of networks based on their tri-lgt decomposition.

Theorem 4.39. *Let N_1 and N_2 be two time-consistent LGT networks. Then, $\Gamma(N_1) = \Gamma(N_2)$ if and only if $\mathrm{Min}(N_1) = \mathrm{Min}(N_2)$.*

Proof. Let N_1 and N_2 be two time-consistent LGT networks such that $\Gamma(N_1) = \Gamma(N_2)$. We prove that $\mathrm{Min}(N_1) = \mathrm{Min}(N_2)$. Recall that $\mathrm{Min}(N_1)$ (resp. $\mathrm{Min}(N_2)$) is the LGT network obtained by removing all redundant arcs from N_1 (resp. N_2). It suffices to prove that each non redundant secondary arc in N_1 is a non redundant secondary arc in N_2 and viceversa. If e is a non redundant secondary arc in N_1, by Proposition 4.36, there are a pair of leaves i, j in N_1 which either $e = min(S_{N_1}^{i \to j})$ or $e = \max(S_{N_1}^{i \to j})$. In both cases, since $\Gamma(N_1) = \Gamma(N_2)$, by Proposition 4.29, we have that either $e = \min(S_{N_2}^{i \to j})$ or $e = \max(S_{N_2}^{i \to j})$. Hence, e is a non redundant secondary arc in N_2. Similarly, we can prove that each non redundant arc in N_2 is a non redundant arc in N_1. Consequently, $\mathrm{Min}(N_1) = \mathrm{Min}(N_2)$.

To prove the other implication, we suppose that $\mathrm{Min}(N_1) = \mathrm{Min}(N_2)$. In such case, $\Gamma(\mathrm{Min}(N_1)) = \Gamma(\mathrm{Min}(N_2))$. By Lemma 4.37, we have that $\Gamma(N_1) = \Gamma(\mathrm{Min}(N_1))$ and $\Gamma(N_2) = \Gamma(\mathrm{Min}(N_2))$. Then, we can conclude that $\Gamma(N_1) = \Gamma(N_2)$. $\square$

Note that, both implications in the Theorem are false if we do not require the temporal consistency of the LGT networks N_1 and N_2. We show examples of both situations.

As a counterexample of the first implication, consider the two networks in Figure 4.17, and notice that only one of them is time-consistent. These two different networks have the same tri-lgt decomposition, but since they have no redundant arcs, their minimum networks are also different.

As for the other implication, consider the networks depicted in the figure used in Example 4.7. This is a case where the tri-lgt decomposition of a network does not coincide with the tri-lgt decomposition of its minimum network. In this case, since $\mathrm{Min}(\mathrm{Min}(N)) = \mathrm{Min}(N)$, the original network and the minimum network have the same minimum network but they do not have the same tri-lgt decomposition.

Finally, we can give this classification result, which is a direct consequence of the theorem above.

Corollary 4.40. *Let N_1 and N_2 be two LGT networks without secondary redundant arcs. Then, $N_1 = N_2$ if, and only if, $\Gamma(N_1) = \Gamma(N_2)$.*

Consequently, we conclude that, tri-lgt networks represented in an LGT network completely determine it if it belongs to the subclass of time-consistent arc-node LGT networks, and up to redundant arcs. In case that the network contains redundant arcs, only the minimum network can be reconstructed.

Chapter 5

A reconciliation problem between gene trees and LGT networks

Contents

5.1 Introduction

Reconciliation methods aim at explaining the discrepancies between the evolutionary histories of genes and species. These differences are caused by evolutionary events like speciations, duplications, losses and lateral transfer of genes, among others. The most used models for reconciliation scenarios between phylogenetic trees take into account either duplications and losses, $\mathbb{DL}$ reconciliations [Goodman et al. (1979); Page (1994); Guigo et al. (1996); Doyon et al. (2009); Górecki et al. (2011); Rasmussen and Kellis (2012); To and Scornavacca (2015)], or also transfers, $\mathbb{DTL}$ reconciliations [Gorbunov and Lyubetsky (2009); Doyon et al. (2010); Tofigh et al. (2011); Bansal et al. (2012); Scornavacca et al. (2013)].

In the parsimony framework, a cost is associated to each evolutionary event and the cost of an evolutionary scenario is the sum of the costs of all events that it implies; the aim is to obtain such scenario with a minimum cost over the set of all possible evolutionary scenarios. This is called the most parsimonious reconciliation (or MPR, for short). For the $\mathbb{DL}$ model, the MPR can be found in linear time [Zhang (1997)]; however, for the $\mathbb{DTL}$ model, finding the MPR becomes NP-hard mainly because we must have a time-consistent scenario [Hallett et al. (2004); Tofigh et al. (2011)]. To overcome this issue and to ensure that transfer events happen only between co-existing species, different approaches have been followed, using either dated species trees [Conow et al. (2010); Merkle et al. (2010); Gorbunov and Lyubetsky (2009); Doyon et al. (2010)] or fixing a priori the branches which allow the transfers events [Hallett et al. (2004); Górecki (2004)], obtaining polynomially solvable problems for the $\mathbb{DTL}$ model.

When moving from trees to phylogenetic networks, the situation is much less understood, and little work has been done on the problem of finding parsimonious reconciliations [Górecki (2004); Libeskind-Hadas and Charleston (2009); To and Scornavacca (2015)]. In this chapter we study the reconciliation problem where the evolution of species is modelled by an LGT network (see Chapter 2). We consider a scenario with duplications, losses and transfers, but restrict transfers to happen only through the secondary arcs of the network. Then, extending previous results in To and Scornavacca (2015) and in Doyon et al. (2010) we provide fast algorithms that give, on the one hand, the MPR between a gene tree and a tree displayed by the LGT network, and, on the other hand, the MPR between a gene tree and the LGT network. These results improve upon previous results in Górecki (2004).

The results in this chapter have led to a manuscript (joint with Dr. Céline Scornavacca and Dr. Gabriel Cardona) that has been submitted to *Journal of Theoretical Biology.*

5.2 Parsimonious reconciliations between gene trees and species trees

Even though trees can be reconciled using probabilistic models, following a criterion of maximum likelihood, this approach, according to Arvestad et al. (2004), leads to algorithms that are very time-consuming, and hence they are only suitable for small sets of taxa or small collections of genes. As an alternative, the combinatorial principle of parsimony overcomes these limitations.

Parsimony methods have been vastly used to infer evolution history, as well as in reconciliation problems. These methods are explicit discrete models of gene evolution that, given the elementary costs of each evolutionary event, seek for the solution of the reconciliation problem with minimum cost. A variety of models exist in order to reconcile a gene tree with a species tree, being Goodman et al. (1979) and Page (1994) the pioneers papers in this field.

In this manuscript we will focus on a model, where a reconciliation α between a gene tree G and a species tree S is defined as a function that maps each node u of G into an ordered sequence of nodes of S, denoted by $\alpha(u) = (\alpha_1(u), \alpha_2(u), \ldots, \alpha_\ell(u))$, and hence allows to track the evolution of genes into the species tree [Doyon et al. (2010); Scornavacca et al. (2013)]. These mappings are restricted by conditions aimed at having an evolutionary history coherent with the chosen evolutionary model.

We present here the definition for the $\mathbb{DTL}$ reconciliation between a gene tree and a species tree, which considers duplications ($\mathbb{D}$), transfers ($\mathbb{T}$) and losses ($\mathbb{L}$), and which is the foundation for more complex reconciliations using species networks. Such definition is a simplified version of the definition of a reconciliation given in Doyon et al. (2010), as our definition does not require the species tree to be dated (for more information on this model, see Ranwez et al. (2015) or Jacox et al. (2016), among others).

We recall some notations before introducing the definition. We focus on binary rooted phylogenetic trees, *trees* for short. Given a tree T, we denote its root by $r(T)$ and the set of its nodes, internal nodes, arcs, leaves and labels respectively by $V(T)$, $I(T)$, $E(T)$, $L(T)$ and $\mathcal{L}(T)$. An internal node u of T has two interchangeable children (u_l, u_r). Then a *species tree* S is a tree such that each element of $\mathcal{L}(S)$ represents an extant species that labels exactly one leaf of S. This is, each leaf u in S is associated to a species, denoted $s(u)$. A *gene tree* G is a tree whose leaves are labeled by contemporary genes. From now on we consider a species tree S and a gene tree G such that $\{s(u)|u \in L(G)\} \subseteq \mathcal{L}(S)$. The same notations and considerations will be used to deal with species networks in the following sections.

Definition 5.1. *Given a species tree S and a gene tree G, α is said to be a $\mathbb{DTL}$ reconciliation between G and S if and only if exactly one of the following events occurs for each pair of nodes u of G and $\alpha_i(u)$ of S (for simplicity, let $x := \alpha_i(u)$ below):*

a) if x is the last node of $\alpha(u)$, one of the cases below is true:

1. *$u \in L(G)$, $x \in L(S)$ and $s(x) = s(u)$;* *(extant leaf)*
2. *$\{\alpha_1(u_l), \alpha_1(u_r)\} = \{x_l, x_r\}$;* ($\mathbb{S}$)
3. *$\alpha_1(u_l) = x$ and $\alpha_1(u_r) = x$;* ($\mathbb{D}$)
4. *$\alpha_1(u_l) = x$, and $\alpha_1(u_r)$ is any species node that is a not a descendant or ancestor of x (or symmetrically interchanging the roles of u_l and u_r);* ($\mathbb{T}$)

b) otherwise, one of the cases below is true:

5. *$\alpha_{i+1}(u) \in \{x_l, x_r\}$;* ($\mathbb{SL}$)
6. *$\alpha_{i+1}(u)$ is any node that is a not a descendant or ancestor of x;* ($\mathbb{TL}$)

From this definition, a $\mathbb{DL}$ reconciliation is a $\mathbb{DTL}$ one where $\mathbb{T}$ and $\mathbb{TL}$ events are not allowed (omit [4.] and [6.]). This is not the common definition of a $\mathbb{DL}$ reconciliation, but it is equivalent [Jacox et al. (2016)] to the more widespread one, which is used for example in To and Scornavacca (2015).

The evolutionary events modelled by this definition are:

- Speciation ($\mathbb{S}$), duplication ($\mathbb{D}$) and transfer ($\mathbb{T}$) events are self-explanatory (see Section 1.2).
- A speciation-loss ($\mathbb{SL}$) is a speciation where the original gene is absent in one of the two species resulting from the speciation.
- A transfer-loss ($\mathbb{TL}$) is a transfer of one of the two descendants of a gene combined with the loss of its sibling lineage.

Note that each loss is coupled with either a speciation or a transfer. Indeed, duplication-loss events leave no trace in the data and therefore are not considered by the model since they will never yield to a parsimonious solution. The reconciliation α is said to be time-consistent if all $\mathbb{T}$ events can be guaranteed to happen between co-existing species.

Given costs δ, τ and λ for respectively $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events, the *cost* of a reconciliation α is defined as the sum of the costs of all events it implies. The most parsimonious reconciliation between G and S is one with minimum cost over all possible reconciliations. We denote this minimum cost as *cost*(G, S).

In order to identify the "most parsimonious reconciliations" (MPR for the $\mathbb{DL}$ model), as per Górecki and Tiuryn (2006) the simple polynomial time algorithm, called *LCA mapping*, can be applied. This mapping between G and S is constructed by finding for each node u of G, the smallest cluster in S that contains the set of genes represented by u. When only $\mathbb{D}$ and $\mathbb{L}$ events are considered, finding the most parsimonious reconciliation between G and

S is feasible by using a polynomial algorithm [Zhang (1997); Vernot et al. (2008)]; however, when transfers are also considered in the evolutionary model, the problem becomes NP-hard [Hallett et al. (2004); Tofigh et al. (2011)]. Such phenomena is due to the timing constraints among nodes of S that are induced by $\mathbb{T}$ events, which are difficult to handle.

To palliate this problem, two strategies are commonly used to reduce the NP-complete problem to a polynomially-solvable one. The first strategy consists in altering the definition of MPR to accept a dated tree S (imposing time scales) as input [Conow et al. (2010); Merkle et al. (2010); Gorbunov and Lyubetsky (2009); Doyon et al. (2010)]. The other strategy uses a priori information to fix which pairs of branches are allowed to be involved in a transfer event: a representation called lateral transfer scheme in Hallett et al. (2004) and species graph in Górecki (2004). The latter is the most similar strategy to the one we follow.

5.3 Reconciliation between gene trees and species networks

The reconciliation problem for phylogenetic networks, generalizing the reconciliation problem from phylogenetic trees, has its origins in a similar problem called the *cophylogeny problem*. Cophylogeny is the reconstruction of ancient relationships among ecologically linked groups of organisms, like hosts and their parasites, from their phylogenetic information. Then, similarly as in the "general" reconciliation between gene trees and species trees, in the host-parasite reconciliation, the *cophylogeny mapping* considers associations between the tips of two evolutionary histories representing the parasite phylogeny and the host phylogeny, where one can consider the evolutionary events of cospeciation, duplication, loss, and host switch (coespeciation and host switching correspond respectively to speciation and transfer events in the $\mathbb{DTL}$ jargon). Libeskind-Hadas and Charleston (2009) present the first contribution toward solving the cophylogeny problem on networks. They attempt to map one network into the other in order to construct the set of coevolutionary events that best explains the current observations. However, they also prove that the cophylogeny reconstruction problem is NP-complete when the host phylogeny is reticulate, even if the parasite phylogeny is a tree. They prove that the reconstruction problem can be solved in polynomial time for a parasite tree and a host network if the time function defined on the host phylogeny, that is the function that assigns to each node (event) in the host network the set of relative times when the event may have occurred, has only one possible assignment for each node. Moreover, they prove that the problem becomes NP-complete when the fixed-time constraint is slightly relaxed.

In order to improve the time-complexity of the algorithm presented in Libeskind-Hadas and Charleston (2009), To and Scornavacca (2015) solve the reconciliation problem between gene trees and species networks restricting the set of events to duplications and losses (i.e. transfer events are not taken into account).

In this chapter we shall focus on two different problems related to the reconciliation between a gene tree and a species network, using LGT networks (see Section 2.2) and extending the models presented in To and Scornavacca (2015) to allow transfer events via secondary arcs. Principal arcs are used to represent descent with modification while secondary arcs represent possible transfers. Efficient polynomial algorithms to solve the MPR reconciliation will be presented.

We will model species networks with LGT networks subject to certain restrictions. For the rest of this chapter we will assume that LGT networks are binary, that is, tree nodes are indegree-1 and outdegree-2 nodes (except for the root that has indegree 0 and the leaves, that have outdegree 0), while reticulation nodes have indegree-2 and outdegree-1.

To ease notations, given an internal node u of a species network, if u has a single child, we shall denote this child as u_l; otherwise, it has two interchangeable children, that we shall denote by u_l and u_r.

Moreover, in order to avoid the inconsistency of the $\mathbb{T}$ events and overcome the NP-completeness of the problem, in the following, we will only consider, without further mention, time-consistent LGT networks (see Section 4.5). Remember that we say that an LGT network N is *time-consistent* if there is a function $t : V(N) \to \mathbb{N}$ such that:

- $t(u) = t(v)$, if $(u, v) \in E_s$, and
- $t(u) < t(v)$, if $(u, v) \in E_p$.

Figure 5.1 shows a time-consistent LGT network modelling a species network.

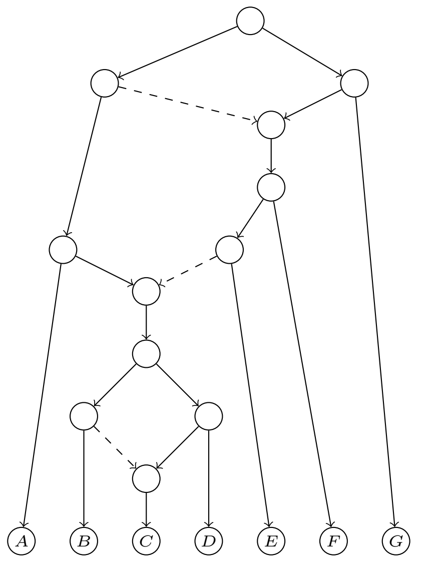

Figure 5.1: A species network which is a time-consistent LGT network.

Given a gene tree and a species network modeled by an LGT network, two approaches to the reconciliation problem are possible. The first one is based on choosing a switching of the network (hence a tree) and finding the best reconciliation of the gene tree with this switching. Remember that, given a network N, a *switching* S of N is obtained from N by choosing, for each reticulation, an incoming arc to *switch on* and the others to *switch off*. Once this is done, we also recursively switch off all switched-on arcs whose target node has only switched-off outgoing arcs (see Section 1.3). The second possible approach is to define directly a reconciliation between the gene tree and the network itself and try to find the best one.

Then, fixed a gene tree and an LGT network, in the first problem, we ask how to find the switching of the network – and consequently, the tree displayed by the network – having a most parsimonious reconciliation with the given gene tree according to the $\mathbb{DTL}$ model [Doyon et al. (2010)], extending the result obtained in To and Scornavacca (2015), which is designed for the $\mathbb{DL}$ model [Doyon et al. (2009)]. In the second problem, we extend the $\mathbb{DTL}$ model for trees (see Section 5.2) to species networks where transfers are allowed only on secondary arcs – similarly to what was proposed in Górecki (2004) – and ask how to find a most parsimonious reconciliation (a reconciliation with minimum cost over all possible reconciliations, denoted as $cost(G, N)$ when G is the gene tree and N is the species network) between the given gene tree and species network. The first model is more adapted for ancient duplication, for which often only one copy of the gene has been retained, while the latter is more suitable for recent duplications.

Then, the two main goals in this chapter are to solve the following problems:

PROBLEM 5.1 *Best Switching*
INPUT: A gene tree G, an LGT network N, the costs δ, τ and λ for respectively $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events.
OUTPUT: A switching S of N such that the $cost(G, S)$ is minimum over all switchings of N.

PROBLEM 5.2 *Best Reconciliation*
INPUT: A gene tree G, an LGT network N, the costs δ, τ and λ for respectively $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events.
OUTPUT: A $\mathbb{DTL}$ reconciliation between G and N with minimum cost $cost(G, N)$.

5.4 The best tree displayed by an LGT network

In this section, we present an algorithm to find the tree displayed by an LGT network N having the most parsimonious $\mathbb{DTL}$ reconciliation with a given gene tree G. We extend the $\mathbb{DTL}$ reconciliation definition given in Section 5.2 for a species tree and a gene tree to the case of a switching S of an LGT network and a gene tree G allowing here transfers only via secondary arcs.

Given a network N and a *switching* S of N, a *path* of S is a path of N that uses only switched-on arcs. We denote by $V_{on}(S)$ the set of nodes of a switching S that are not an extreme node of any switched-off arc. A tree T is said to *be displayed* by a network if T is isomorphic to the tree obtained by using only the switched-on arcs of a switching S of N and by contracting all nodes not in $V_{on}(S)$. See Section 1.3

Definition 5.2. *Given an LGT network N, a switching S of N and a gene tree G, α is said to be a $\mathbb{DTL}$ reconciliation between G and S if and only if exactly one of the following events occurs for each pair of nodes u of G and $\alpha_i(u)$ of S (for simplicity, let $x := \alpha_i(u)$ below):*

a) if x is the last node of $\alpha(u)$, one of the cases below is true:

1. *$u \in L(G)$, $x \in L(S)$ and $s(x) = s(u)$;* *(extant leaf)*
2. *$\{\alpha_1(u_l), \alpha_1(u_r)\} = \{x_l, x_r\}$ and both outgoing arcs of x are in E_p and switched-on;* ($\mathbb{S}$)
3. *$\alpha_1(u_l) = x$ and $\alpha_1(u_r) = x$;* ($\mathbb{D}$)
4. *$\alpha_1(u_l) = x$, $\alpha_1(u_r) = y$ and (x, y) is in E_s and switched-on (or symmetrically interchanging the roles of u_l and u_r);*

or $\alpha_1(u_l) = x_l$, $\alpha_1(u_r) = x_r$ and (x, x_r) is in E_s and both outgoing arcs of x are switched-on (or symmetrically interchanging the roles of u_l and u_r, and x_l and x_r, so 4 possibilities); ($\mathbb{T}$)

b) otherwise, one of the cases below is true:

5. *$\alpha_{i+1}(u) = y$, (x, y) is in E_p and x has two outgoing arcs that are switched-on;* ($\mathbb{SL}$)
6. *$\alpha_{i+1}(u) = y$, (x, y) is in E_s and x has two outgoing arcs that are switched-on;* ($\mathbb{TL}$)
7. *$\alpha_{i+1}(u) = y$, (x, y) is in E_p and is the only switched-on arc of x;* ($\emptyset$)
8. *$\alpha_{i+1}(u) = y$, (x, y) is in E_s and is the only switched-on arc of x;* ($\mathbb{T}$)

In the following, we use $\alpha_l(\cdot)$ to denote the last node of $\alpha(\cdot)$. Moreover, we focus on reconciliations such that $|\alpha(r(G))| = 1$, since this is the case for all most parsimonious reconciliations. Indeed, since Definition 5.2 allows to map the root of G to any node of the switching S, any reconciliation α where $|\alpha(r(G))| > 1$ can be changed into another one with equal or lower cost simply by mapping $r(G)$ to the last node of $\alpha(r(G))$.

Note that, since both nodes of a secondary arc have the same time-stamps, all lateral gene transfers are *time-consistent*, i.e. are guaranteed to happen between co-existing species.

To and Scornavacca (2015) pointed out that, when all arcs are principal, the most parsimonious mapping between a switching S and G is completely determined and the most parsimonious reconciliation is always the LCA reconciliation which we define, adapting the definition to LGT networks as follows:

1. for each node $u \in L(G)$, $\alpha_l(u)$ is defined as the only node $x \in L(S)$ such that $s(u) = s(x)$;
2. for each node $u \in I(G)$ with child nodes $\{u_1, u_2\}$, $\alpha_l(u) := LCA_S(\alpha_l(u_1), \alpha_l(u_2))$;
3. once $\alpha_l(u)$ is fixed for all nodes u of G, $\alpha(u)$ is completed as follows. First, we insert in $\alpha(u)$ – before $\alpha_l(u)$ – the ordered list of nodes composing the unique path in S –extremities excluded– between $\alpha_l(u_p)$ and $\alpha_l(u)$, where u_p is the parent node of u. Then, if
   - $\alpha_l(u_p) \neq \alpha_l(u)$,
   - $\alpha_l(u_p) = \alpha_l(u')$ – where u' is the other child of u_p which is not u,
   - $(\alpha_l(u_p), \alpha_1(u))$ is not a secondary arc,

   we insert $\alpha_l(u_p)$ before $\alpha_1(u)$ in $\alpha(u)$.

As we previously stated, when N is a tree, there is a unique reconciliation (the LCA reconciliation) between G and N with minimum cost. Note that the algorithms used on trees to find the LCA reconciliation can also be used on switchings even when some arcs are secondary. Then, the most parsimonious mapping is still completely determined. Hereafter, when we refer to the reconciliation between a tree and a switching, we refer to the LCA reconciliation between them.

This observation will allow us to use several of the results in To and Scornavacca (2015) to solve the Best Reconciliation Problem. Given that *cost*(G, S) is the minimum cost over all possible $\mathbb{DTL}$ reconciliations between G and S, it corresponds to the cost of the LCA reconciliation.

Example 5.1. *Given the gene tree G and the switching S depicted in Figure 5.2 of the LGT network depicted in Figure 5.1, we give full details on how to compute the LCA reconciliation α between G and S, which coincides with the most parsimonious $\mathbb{DTL}$ reconciliation between G and S.*

The LCA reconciliation maps each node in G as we have summarized in Table 5.1. We have also included a third column giving the corresponding specific evolutionary events which are obtained following Definition 5.2. Moreover, in Figure 5.3 there is a depiction of the embedding modelling the reconciliation. Notice that this $\mathbb{DTL}$ reconciliation has 2 duplications, 3 transfers and 7 losses. Then, since α is the MPR, if we assume unitary cost for each of the three types of events, we have that $cost(G, S) = 2 + 3 + 7 = 12$.

Table 5.1: LCA reconciliation between G and S

Genes	α mapping	corresponding events
r	(z)	$(\mathbb{D})$
s	(z, h)	$(\mathbb{SL}, \mathbb{T})$
t	(k)	$(\mathbb{D})$
u	(k)	$(\mathbb{S})$
v	(k, n, o, q)	$(\mathbb{SL}, \varnothing, \mathbb{SL}, \mathbb{T})$
w	(z)	$(\mathbb{S})$
a	(A)	$(\mathbb{C})$
d	(n, o, p, D)	$(\varnothing, \mathbb{SL}, \varnothing, \mathbb{C})$
b	(B)	$(\mathbb{C})$
c	(x, C)	$(\varnothing, \mathbb{C})$
e	(j, l, m, E)	$(\varnothing, \mathbb{SL}, \varnothing, \mathbb{C})$
f	(h, j, l, F)	$(\mathbb{TL}, \varnothing, \mathbb{SL}, \mathbb{C})$
g	(i, G)	$(\varnothing, \mathbb{C})$

We now explain how we have been computed α for two especific nodes in G, namely $\alpha(f)$ and $\alpha(v)$, following the definition of the LCA reconciliation.

For the node f we have that its parent node is w and its sibling node is g. Note that, since $f \in L(G)$, then $\alpha_l(f) = F$. Moreover, $\alpha_l(w) = z$ and $\alpha_l(g) = G$. Then, for the first part in item 3 of the definition, we have, for the moment, $\alpha(f) = (h, j, l, F)$. Now, notice that $z \neq G$, then $\alpha_l(w) \neq \alpha_l(g)$ (it is not satisfied the second subitem in 3.). Consequently, it is not necessary to insert a new node in $\alpha(f)$.

On the other hand, for the node v, we have that its parent node is t and its sibling node is u. Moreover, $\alpha_l(v) = q$, $\alpha_l(t) = k$ and $\alpha_l(u) = k$. Then, for the moment, $\alpha(v) = (n, o, q)$. But since

- $\alpha_l(t) \neq \alpha_l(v)$,
- $\alpha_l(t) = \alpha_l(u)$, *and*
- $(\alpha_l(t), \alpha_1(v)) = (k, n)$ *is not a secondary arc,*

we insert k before $\alpha_1(v)$ in $\alpha(v)$. Then, we can conclude that $\alpha(v) = (k, n, o, q)$.

Note that Definition 5.2 is equivalent to Definition 5.1, apart from the fact that transfers are allowed only on secondary arcs (conditions [4.], [6.] and [8.]), and that losses are not counted on nodes not in $V_{on}(S)$ (conditions [7.] and [8.]). This last fact implies that, if

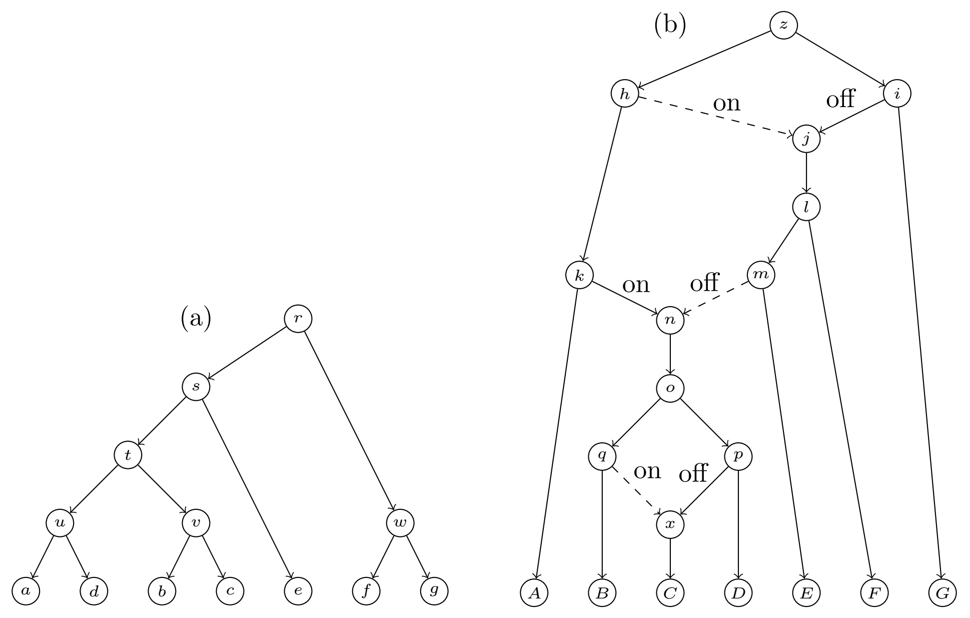

Figure 5.2: (a) A gene tree G, and (b) a switching S for the LGT network depicted in Figure 5.1 used for Example 5.1. The labels "on" and "off" on the arcs in (b) represent the switched-on and switched-off arcs, respectively.

transfers are allowed only on secondary arcs, for a tree S' displayed by the network and its associated switching S, we have $cost(G, S) = cost(G, S')$. Thus, to find the optimal tree displayed by the network, we can focus on switchings, as done in To and Scornavacca (2015), and we can solve the Best Switching Problem.

To and Scornavacca (2015) solve this problem when all arcs of the network are considered as principal. This implies that all the results and algorithms in the *Best switching* section of To and Scornavacca (2015) are valid to find a valid reconciliation according to Definition 5.2, except for optimising the cost of transfers. Indeed, since both incoming arcs of a reticulation are considered as principal, in their model of hybridizations there is no difference in taking one rather than the other incoming arc of a reticulation.

The time complexity of the reference algorithm does not depend on the number of reticulations in the species network but only on the level of the network (see Section 1.7). In order to solve the Best Switching Problem, we need to recall basic notation and some definitions relative to biconnected components in a network as introduced in To and Scornavacca (2015):

1. For a node u of N, N_u denotes the subgraph induced by the nodes accessible from u.

2. Given a biconnected component B (that is not a leaf) of N, the network $N(B)$ consists of B and all cut arcs coming out from B.

3. The mapping $B(\cdot)$ associates every $u \in G$ to the lowest (we say that a biconnected component B_i is lower than another one B_j if there is a path from the root of B_j to the root of B_i) biconnected component B of N such that $\mathcal{L}(N_{r(B)}) \supseteq \mathcal{L}(G_u)$.

4. The tree G_N is obtained from G by applying the following procedure for each child u' of each internal node u of G. If there are k biconnected components $B_1, \ldots, B_k$

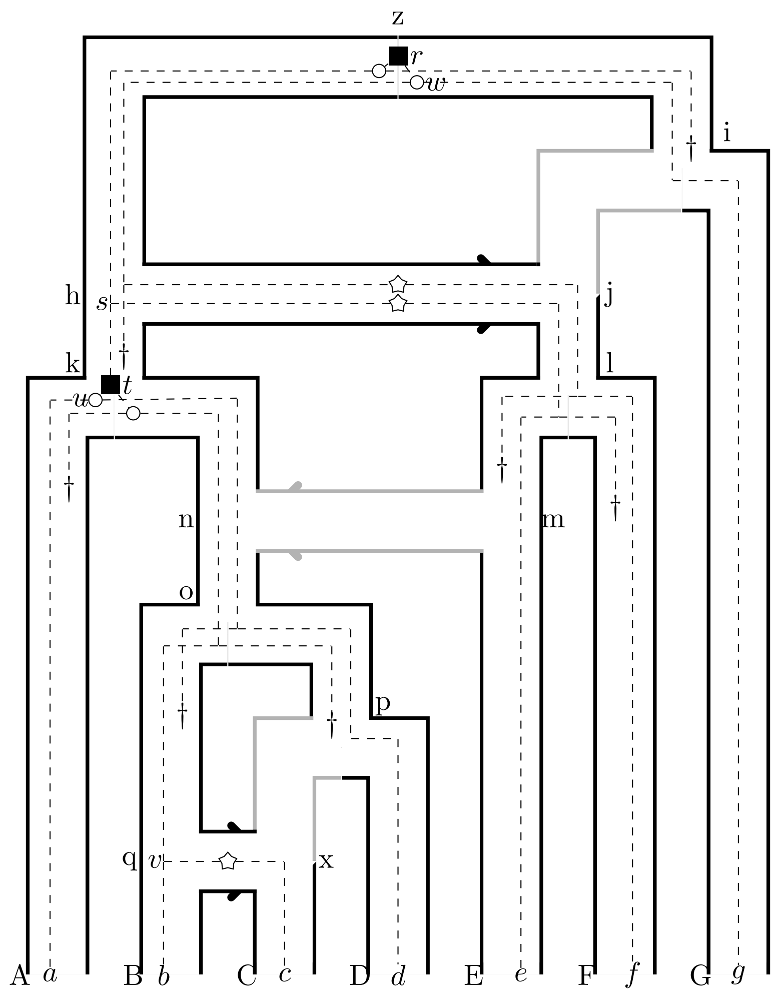

Figure 5.3: A $\mathbb{DTL}$ reconciliation α between the gene tree and the switching depicted in Figure 5.2. Solid lines represent network arcs, where secondary arcs are depicted horizontally and with "winglets" and grey ones represent switched-off arcs. The dashed lines represent the gene evolution. The $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ event are represented as filled square, star and cross dagger, respectively. Each gene label is represented in the figure and mapped to the element $\alpha_l(\cdot)$.

properly below $B(u)$ and properly above $B(u')$ in N, we add k indegree-1 outdegree-1 nodes on the arc (u, u'), respectively mapped to $B_1, \dots, B_k$.

5. Given a biconnected component B different from a leaf, we denote by G_B the set of all maximal connected subgraphs H of G_N satisfying that $B(u) = B$ for every internal node u of H. Note that these subgraphs are necessarily binary trees or arcs (see Lemma 2 of To and Scornavacca (2015)); we accordingly decompose G_B as $G_B^t \sqcup G_B^e$.

See Figures 5.4 and 5.5 for an illustration of these concepts.

We shall see that, in order to adapt the tools proposed in To and Scornavacca (2015) to solve the Best Switching Problem, it suffices to redefine the cost functions introduced in To and Scornavacca (2015) to properly count transfers, i.e. secondary arcs used in the reconciliations.

Let α be a reconciliation between G and S, for two nodes x and y in S, $lgt(x, y)$ and

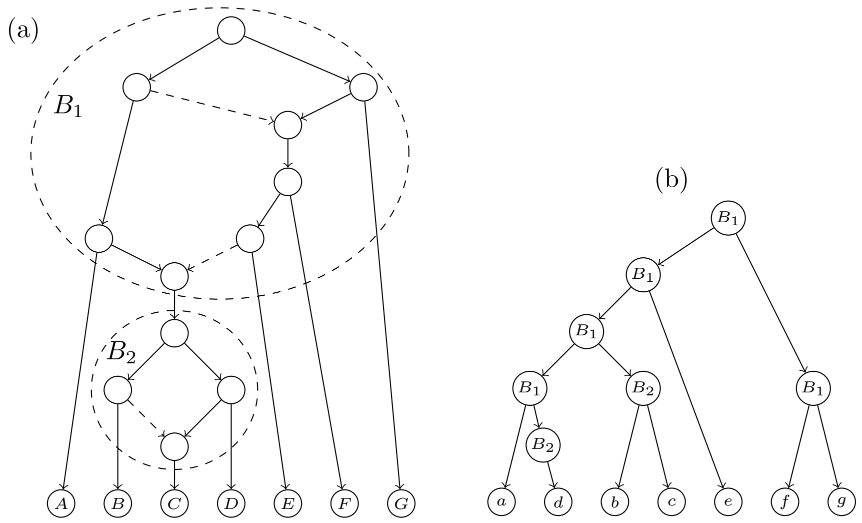

Figure 5.4: (a) The LGT network depicted in Figure 5.1 with highlighted the two non trivial biconnected components B_1 and B_2. (b) The tree G_N along with the labelling $B(\cdot)$ where the gene tree G is the one depicted in Figure 5.2(a). Notice that an artificial node labelled by B_2 is added on the arc (u, d) because $B(u) = B_1$, $B(d) = B_2$ and $B_1 > B_2$.

$dist(x, y)$ are defined as follows: If there exists a path $x \rightsquigarrow y$ in N, $lgt(x, y)$ and $dist(x, y)$ are defined respectively as the number of switched-on secondary arcs, and as the number of nodes in $V_{on}(S)$, in the path from x to y. Otherwise, $dist(x, y) = lgt(x, y) = \infty$. We denote $t_\alpha(u)$ the number of transfer events in α associated to an internal node u in G; then we have $t_\alpha(u) = lgt(\alpha_l(u), \alpha_l(u_l)) + lgt(\alpha_l(u), \alpha_l(u_r))$. Consequently, the number of transfers of the reconciliation α, denoted by $t(\alpha)$, is the sum of $t_\alpha(\cdot)$ for all internal nodes of G. We denote by $d(\alpha)$ and $l(\alpha)$ respectively the number of duplications and of losses of α.

Note that, as done in To and Scornavacca (2015), we suppose that no internal node of G has all its descendant leaves associated to the same species $\bar{s}$. Indeed, if such a node exists in G, say u, it is easy to be convinced that the most parsimonious way to reconcile G_u is via $(|L(G_u)| - 1)$ duplications. Thus, we can replace G_u by a leaf l such that $s(l) = \bar{s}$, reconcile the resulting tree and, a posteriori, add back G_u, along with all the duplications it implies, to the reconciliation.

Given a biconnected component B_i of N different from a leaf, S_i a switching of $N(B_i)$ and H a tree in $G^t_{B_i}$, we denote by $\beta^H_{S_i}$ the LCA reconciliation between H and S_i when, for each leaf u of H, $s(u) := r(B(u))$. Now, for $H \in G_{B_i}$, we define $cost(H, S_i)$ as follows:

- $\forall H \in G^t_{B_i}$, if $B_i = B(r(G))$,

$$cost(H, S_i) = cost(\beta^H_{S_i}),$$

- $\forall H \in G^t_{B_i}$, if $B_i \neq B(r(G))$,

$$cost(H, S_i) = cost(\beta^H_{S_i}) + \tau \cdot lgt(r(S_i), \beta^H_{S_i}(r(H))) + \lambda \cdot dist(r(S_i), \beta^H_{S_i}(r(H))),$$

- $\forall H \in G^e_{B_i}$ with u the only leaf of H,

$$cost(H, S_i) = \tau \cdot lgt(r(S_i), r(B(u))) + \lambda \cdot dist(r(S_i), r(B(u))).$$

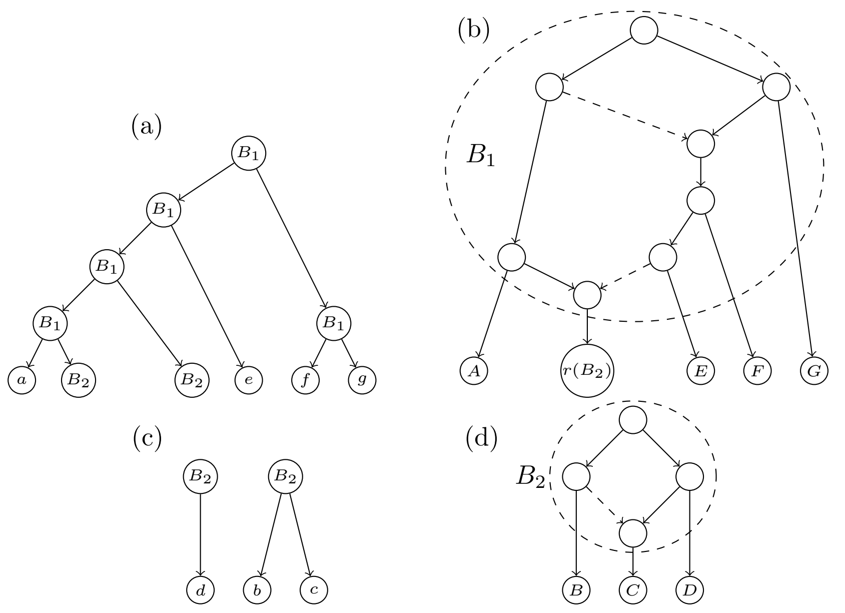

Figure 5.5: Given the LGT network of Figure 5.1 and the gene tree of Figure 5.2(a) we find: (a) G_{B_1}; (b) $N(B_1)$; (c) G_{B_2}; (d) $N(B_2)$.

Then, the following proposition permits to analyze independently each biconnected component of N and, hence, their cost.

Proposition 5.1. *Let $B_1, \dots, B_p$ be the biconnected components of N that are not leaf nodes, and let S be a switching of N. Moreover, for each network $N(B_i)$, let S_i its switching induced by S. Then, cost$(G, S) = \sum_{i=1}^{p} \sum_{H \in G_{B_i}}$ cost(H, S_i).*

Proof. Let α be a reconciliation between G and S with minimum cost. We denote by $d_\alpha(S_i)$ the number of duplications in S_i and by $l_\alpha(u, S_i)$ the number of losses in S_i associated with $u \in I(G)$. Moreover, given two nodes in S such that y is descendant of x, we define $lgt_{S_i}(x, y)$ as the number of switched-on secondary arcs in S_i on the path from x to y. Note that, given $u \in L(S_i)$, the single arc whose target is u is never a secondary arc. Then, using lgt_{S_i} instead of lgt, we can define the number of transfers associated with u in S_i, denoted by $t_\alpha(u, S_i)$, in the same way as $t_\alpha(u)$. Then, $t_\alpha(u) = \sum_{i=1}^{p} t_\alpha(u, S_i)$.

Now, given u in $I(G)$, by Lemma 3 of To and Scornavacca (2015), there exists $H \in G^t_{B(u)}$ such that $u \in I(H)$ and $\alpha_l(u) = \beta^H_{S(B(u))}(u)$. This also implies that $\alpha_l(u)$ and $\alpha_l(u')$ are respectively contained in $B(u)$ and $B(u')$, where u' is a child of u.

Note that $t_\alpha(u, S_i) > 0$ only if the path from $\alpha_l(u)$ to $\alpha_l(u')$ (being u' a child of u) in S contains at least one switched-on secondary arc of B_i. Then, $t_\alpha(u, S_i) \geq 0$ can hold only for the three sets of nodes V_i^1, V_i^2, V_i^3 of $I(G)$ defined below:

- $V_i^1 := \{u \in I(G) : \alpha_l(u) \in B_i\}$.
- $V_i^2 := \{u \in I(G) : \alpha_l(u)$ is above $r(B_i)$ and either $\alpha_l(u_l)$ or $\alpha_l(u_r)$ are in $B_i\}$.
- $V_i^3 := \{u \in I(G) : \alpha_l(u)$ is above $r(B_i)$ and either $\alpha_l(u_l)$ or $\alpha_l(u_r)$ are below $B_i\}$.

Note that V_i^2 and V_i^3 are empty if $B_i = B(r(G))$. By construction V_1 is disjoint from V_2 and V_3; moreover, V_2 and V_3 are disjoint because if the two children of u have their α_l one in B_i and one below B_i, then $\alpha_l(u)$ must be in B_i and thus cannot be above $r(B_i)$.

Thus,

$$t(\alpha) = \sum_{i=1}^{p} \sum_{u \in V_i^1 \cup V_i^2 \cup V_i^3} t_\alpha(u, S_i).$$

Recall that the contribution to $t_\alpha(u, S_i)$ of any child u' of u consists of the secondary arcs in the path between $\alpha_l(u)$ and $\alpha_l(u')$ contained in S_i.

Given $u \in V_i^1$, by Lemma 3 in To and Scornavacca (2015), there exists $H \in G_{B_i}^t$ such that $\alpha_l(u) = \beta_{S_i}^H(u)$ while, for each child u' of u, $\alpha_l(u')$ is somewhere in $B(u')$. If $B(u') = B(u)$, then by definition of G_N, u' is a child of u in H. If $B(u') \neq B(u)$, let B_j be the highest biconnected component in N such that $r(B_i) > r(B_j) \geq r(B(u'))$. Then, by definition of G_N and of $\beta_{S_i}^H$, $r(B_j)$ will label a leaf l of H that is a child of u. Thus $t_\alpha(u, S_i) = t_{\beta_{S_i}^H}(u)$. Then:

$$\sum_{u \in V_i^1} \tau \cdot t_\alpha(u, S_i) = \sum_{H \in G_{B_i}^t} \tau \cdot \sum_{u \in I(H)} t_{\beta_{S_i}^H}(u) = \sum_{H \in G_{B_i}^t} \tau \cdot t(\beta_{S_i}^H).$$

The first equivalence holds because 1) any internal node u of a tree $H \in G_{B_i}^t$ is a node of G that is mapped to B_i and, by definition of G_N, is an internal node, thus $u \in V_i^1$; 2) any such $u \in V_i^1$ is an internal node for some $H \in G_{B_i}^t$. In a similar way, from To and Scornavacca (2015), we have:

$$\sum_{u \in V_i^1} \lambda \cdot l_\alpha(u, S_i) = \sum_{H \in G_{B_i}^t} \lambda \cdot l(\beta_{S_i}^H),$$

and $d_\alpha(S_i) = \sum_{H \in G_{B_i}^t} d(\beta_{S_i}^H)$. Then, the following holds:

$$\delta \cdot \sum_{H \in G_{B_i}^t} d(\beta_{S_i}^H) + \sum_{u \in V_i^1} \big(\tau \cdot t_\alpha(u, S_i) + \lambda \cdot l_\alpha(u, S_i)\big) = \sum_{H \in G_{B_i}^t} cost(\beta_{S_i}^H).$$

Given $u \in V_i^2$ and u' being any of the children of u with $B(u') = B_i$, by Lemma 3 of To and Scornavacca (2015), there exists $H \in G_{B_i}^t$ such that $\alpha_l(u') = \beta_{S_i}^H(u')$. Moreover, by definition of G_N, u' is the root of H, thus the contribution of u' to $t_\alpha(u, S_i)$ is $lgt(r(S_i), \beta_{S_i}^H(r(H)))$. From To and Scornavacca (2015), we know that $l_\alpha(u, S_i) = dist(r(S_i), \beta_{S_i}^H(r(H)))$, thus, joining the cases for $u \in V_i^1$ and $u \in V_i^2$, we have:

$$\delta \cdot \sum_{H \in G_{B_i}^t} d(\beta_i^H) + \sum_{u \in V_i^1 \cup V_i^2} \big(\tau \cdot t_\alpha(u, S_i) + \lambda \cdot l_\alpha(u, S_i)\big) \overset{(*1)}{=} \sum_{H \in G_{B_i}^t} cost(H, S_i).$$

Again, the last equivalence holds because any internal node $\bar{u}$ of a tree $H \in G_{B_i}^t$ is a node in V_i^2, with $B_i \neq B(r(G))$, and vice versa.

Given $u \in V_i^3$ and u' being any of the children of u below B_i, let B_j be the first biconnected component of N between B_i and the component containing $\alpha_l(u')$. Then the number of secondary arcs in the path between $\alpha_l(u)$ and $\alpha_l(u')$ contained in S_i is, by construction, equal to $lgt(r(S_i), r(B_j))$. Since, as noted above, $\alpha_l(u)$ and $\alpha_l(u')$ are respectively contained in $B(u)$ and $B(u')$, then, by definition of G_N, each arc (u_a, u_b) in $G_{B_i}^e$ corresponds to exactly one of the biconnected component, namely $B(u_b)$, properly between $\alpha_l(u)$ and $\alpha_l(u')$ for a given u in V_i^3. Then, the following holds:

$$\sum_{u \in V_i^3} \tau \cdot t_\alpha(u, S_i) = \sum_{H := (u_a, u_b) \in G_{B_i}^e} \tau \cdot lgt(r(S_i), r(B(u_b))).$$

Similarly, from To and Scornavacca (2015), we have:

$$\sum_{u \in V_i^3} \lambda \cdot l_\alpha(u, S_i) = \sum_{H:=(u_a,u_b) \in G^e_{B_i}} \lambda \cdot dist(r(S_i), r(B(u_b))).$$

Then, from the definition of $cost(H, S_i)$, the following holds:

$$\sum_{u \in V_i^3} \big(\tau \cdot t_\alpha(u, S_i) + \lambda \cdot l_\alpha(u, S_i)\big) \overset{(*2)}{=} \sum_{H \in G^e_{B_i}} cost(H, S_i).$$

Now, by Lemma 4 in To and Scornavacca (2015),

$$d(\alpha) = \sum_{i=1}^{p} d_\alpha(S_i) = \sum_{i=1}^{p} \sum_{H \in G^t_{B_i}} d(\beta^H_{S_i})$$

and

$$l(\alpha) = \sum_{i=1}^{p} l_\alpha(S_i) = \sum_{i=1}^{p} \sum_{u \in V_i^1 \cup V_i^2 \cup V_i^3} l_\alpha(u, S_i).$$

Moreover, above we proved that $t(\alpha) = \sum_{i=1}^{p} \sum_{u \in V_i^1 \cup V_i^2 \cup V_i^3} t_\alpha(u, S_i)$.

Then, combining this with (*1) and (*2), we can conclude the proof:

$$\begin{aligned}
cost(G, S) &= \delta \cdot d(\alpha) + \tau \cdot t(\alpha) + \lambda \cdot l(\alpha) \\
&= \sum_{i=1}^{p} \big(\delta \cdot d_\alpha(S_i) + \tau \cdot \sum_{u \in I(G)} t_\alpha(u, S_i) + \lambda \cdot \sum_{u \in I(G)} l_\alpha(u, S_i)\big) \\
&= \sum_{i=1}^{p} \Big(\delta \cdot \sum_{H \in G^t_{B_i}} d(\beta^H_i) + \sum_{u \in V_i^1 \cup V_i^2 \cup V_i^3} \big(\tau \cdot t_\alpha(u, S_i) + \lambda \cdot l_\alpha(u, S_i)\big)\Big) \\
&= \sum_{i=1}^{p} \Big([\delta \cdot \sum_{H \in G^t_{B_i}} d(\beta^H_i) + \sum_{u \in V_i^1 \cup V_i^2} \big(\tau \cdot t_\alpha(u, S_i) + \lambda \cdot l_\alpha(u, S_i)\big)] + \\
&+ [\sum_{u \in V_i^3} \big(\tau \cdot t_\alpha(u, S_i) + \lambda \cdot l_\alpha(u, S_i)\big)]\Big) \\
&= \sum_{i=1}^{p} \big(\sum_{H \in G^t_{B_i}} cost(H, S_i) + \sum_{H \in G^e_{B_i}} cost(H, S_i)\big) \\
&= \sum_{i=1}^{p} \sum_{H \in G_{B_i}} cost(H, S_i)
\end{aligned}$$

□

Given a gene tree and a level-k LGT network, the Best Switching Problem can be solved applying Algorithm 3 (which is Algorithm 1 in To and Scornavacca (2015)) using the new definition of $cost(H, S_i)$.

Algorithm 3 Compute a switching S of N such that $cost(G, S)$ is minimum given positive costs δ, τ, and λ, respectively for $\mathbb{D}$, $\mathbb{T}$, $\mathbb{L}$ events

```
1: Compute the tree $G_N$ and its labeling function $B(\cdot)$;
2: Compute $G_{B_i}$ for each biconnected component $B_i$ of $N$ that is not a leaf;
3: for each biconnected component $B_i$ of $N$ do
4:     for each switching $S_i^j$ of $N(B_i)$ do
5:         $cost_j = \sum_{H \in G_{B_i}} cost(H, S_i^j)$;
6:         $S_i^m \leftarrow$ the switching of $N(B_i)$ with the lowest value of $cost_j$ over all $j$;
7:     end for
8: end for
9: return the switching $S$ of $N$ in which each network $N(B_i)$ has $S_i^m$ as switching.
```

Theorem 5.2. *Given a gene tree G, a level-k LGT network N with p biconnected components and the costs δ, τ and λ for respectively $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events, Algorithm 3 solves the Best Switching Problem in $O(n + 2^k \cdot p \cdot m)$ time, where $n = |V(N)|$ and $m = |V(G)|$.*

Proof. See Theorem 2 of To and Scornavacca (2015) to check its correctness and running time analysis. □

5.5 The best reconciliation with the LGT network

In the previous section, we showed how to find the tree in the network that has a most parsimonious $\mathbb{DTL}$ reconciliation with a given gene tree G. Now, if G contains several copies of a gene tree, each of it following a different evolutionary scenario, this model may not be the more adapted one. Another way to approach the problem is to drop the requirement of reconciling with a switching, and directly reconcile the gene tree with the LGT network instead. What we want to do is to extend Definition 5.1 to networks, allowing transfer events only via secondary arcs. To obtain this, we can simply modify Definition 5.2 as follows. First, we consider all arcs of N as switched-on. Second, condition [7.] is modified so that $\alpha_{i+1}(u)$ is the only child of x through a principal arc. Note that, since in an LGT network all internal nodes have at least a principal outgoing arc, and here we consider all arcs of N as switched-on, condition [8.] will never be fulfilled. Hence:

Definition 5.3. *Given an LGT network N and a gene tree G, α is said to be a $\mathbb{DTL}$ reconciliation between G and N if and only if exactly one of the following events occurs for each pair of nodes u of G and $\alpha_i(u)$ of S (for simplicity, let $x := \alpha_i(u)$ below):*

a) if x is the last node of $\alpha(u)$, one of the cases below is true:

1. *$u \in L(G)$, $x \in L(S)$ and $s(x) = s(u)$;* *(extant leaf)*
2. *$\{\alpha_1(u_l), \alpha_1(u_r)\} = \{x_l, x_r\}$;* ($\mathbb{S}$)
3. *$\alpha_1(u_l) = x$ and $\alpha_1(u_r) = x$;* ($\mathbb{D}$)
4. *$\alpha_1(u_l) = x$, $\alpha_1(u_r) = y$ and (x, y) is in E_s (or symmetrically interchanging the roles of u_l and u_r);*

   or $\alpha_1(u_l) = x_l$, $\alpha_1(u_r) = x_r$ and (x, x_r) is in E_s (or symmetrically interchanging the roles of u_l and u_r, and x_l and x_r, so 4 possibilities); ($\mathbb{T}$)

b) otherwise, one of the cases below is true:

5. *$\alpha_{i+1}(u) = y$, (x, y) is in E_p;* ($\mathbb{SL}$)

6. $\alpha_{i+1}(u) = y$, (x, y) *is in* E_s; $(\mathbb{TL})$

7. $\alpha_{i+1}(u) = y$ *and* (x, y) *is the only outgoing arc of* x *in* E_p; $(\emptyset)$

In Algorithm 4 we present an algorithm to solve the Best Reconciliation Problem, which is an adaptation of the algorithm presented in Doyon et al. (2010) to consider only transfers on secondary arcs. Note that in this algorithm we use the fact that N is a time-consistent network and also that a bottom-up traversal of N exists.

Algorithm 4 Compute $cost(G, N)$ given positive costs δ, τ, and λ, respectively for $\mathbb{D}$, $\mathbb{T}$ $\mathbb{L}$ events

```
1: for each $u \in V(G)$ and $x \in V(N)$, do $c(u,x) \leftarrow \infty$ end for    ▷ Initialize the matrix
2: for node $u \in V(G)$ according to a bottom-up traversal do
3:     for node $x \in V(N)$ according to a bottom-up traversal do
4:         if $u \in L(G)$, $x \in L(S)$, and $s(u) = s(x)$ then
5:             $c(u,x) \leftarrow 0$. Go to the next iteration of the loop at line 3    ▷ Extant leaf
6:         end if
7:         for all $e \in \{\mathbb{S}, \mathbb{D}, \mathbb{T}, \varnothing, \mathbb{SL}, \mathbb{TL}\}$ do $c_e \leftarrow \infty$ end for
8:         if $u$ has two children then
9:             $c_{\mathbb{D}} \leftarrow c(u_l, x) + c(u_r, x) + \delta$    ▷ $\mathbb{D}$ event
10:            if $x$ has two outgoing principal arcs then
11:                $c_{\mathbb{S}} \leftarrow \min\{c(u_l, x_l) + c(u_r, x_r), c(u_l, x_r) + c(u_r, x_l)\}$    ▷ $\mathbb{S}$ event
12:            else if $x$ has two outgoing arcs and w.l.o.g. $(x, x_r)$ is secondary then
13:                $c_{\mathbb{T}} \leftarrow \min\{c(u_l, x) + c(u_r, x_r), c(u_l, x_r) + c(u_r, x)\} + \tau$    ▷ $\mathbb{T}$ event
14:                $c_{\mathbb{T}} \leftarrow \min\{c_{\mathbb{T}}, c(u_l, x_l) + c(u_r, x_r) + \tau, c(u_l, x_r) + c(u_r, x_l) + \tau\}$    ▷ $\mathbb{T}$ event
15:            end if
16:        else
17:            if $x$ has two outgoing principal arcs then
18:                $c_{\mathbb{SL}} \leftarrow \min\{c(u, x_l), c(u, x_r)\} + \lambda$    ▷ $\mathbb{SL}$ event
19:            else if $x$ has two outgoing arcs and w.l.o.g. $(x, x_r)$ is secondary then
20:                $c_{\mathbb{TL}} \leftarrow c(u, x_r) + \lambda + \tau$    ▷ $\mathbb{TL}$ event
21:                $c_{\varnothing} \leftarrow c(u, x_l)$    ▷ $\varnothing$ event
22:            else
23:                $c_{\varnothing} \leftarrow c(u, x_l)$    ▷ $\varnothing$ event
24:            end if
25:        end if
26:        $c(u,x) \leftarrow \min\{c_e : e \in \{\mathbb{S}, \mathbb{D}, \mathbb{T}, \varnothing, \mathbb{SL}, \mathbb{TL}\}\}$    ▷ Final cost for $c(u,x)$
27:    end for
28: end for
29: return $\min\{c(root(G), x) : x \in V(N)\}$.
```

Theorem 5.3. *Given a gene tree G, an LGT network N, the costs δ, τ and λ for respectively $\mathbb{D}$, $\mathbb{T}$ and $\mathbb{L}$ events, Algorithm 4 solves the Best Reconciliation Problem in $O(n \cdot m)$ space and time, where $n = |V(N)|$ and $m = |V(G)|$.*

Proof. We first prove the correctness of the algorithm.

The algorithm fills a matrix $c : V(G) \times V(N) \to \mathbb{N}$ through two nested loops, each one visiting all nodes through a bottom-up traversal of G and N, respectively.

Consider the node u at an iteration of the loop in line 2. This loop (lines 3–28) computes $c(u, x)$ for each node $x \in V(N)$ by considering all six possible events separately. The consistency of the computation of the cost for each of these events is ensured because for any child u' of u ($u' \in \{u_l, u_r\}$), any child x' of x ($x' \in \{x_l, x_r\}$) and any node $y \in V(N)$, the costs $c(u', y)$ and $c(u, x')$ have been previously computed thanks to the bottom-up traversal of G and N. Then, the final cost for $c(u, x)$ is computed by considering the minimum over all possible events. Since we can assume that a reconciliation with minimum cost maps the root of G to a single node of N, we can find this minimum cost by taking the minimum of $c(r(G), x)$ where $x \in V(N)$.

We now prove the running time and space cost of the algorithm. The loop over the nodes of G (line 2) runs for $O(m)$ iterations. The loop over the nodes of N (line 3) runs for $O(n)$ iterations. Thus, lines 4 to 27 run $O(m \cdot n)$ times. The computations of the costs of all possible events can be computed in constant time. As a result, the overall time complexity of the algorithm is in $O(n \cdot m)$. The space complexity is completely determined by the size of the matrix $c(G, N)$, which is $m \times n$, and hence the space cost is also $O(n \cdot m)$. $\square$

Note that, when all arcs are principal, the Best Reconciliation Problem coincides with Problem 2 in To and Scornavacca (2015). This implies that Algorithm 4 solves the latter problem for time-consistent networks in $O(n \cdot m)$ instead of $O(h^2 \cdot n \cdot m)$, as proposed in To and Scornavacca (2015), where h is the number of reticulations of N.

In conclusion, we have studied two variants of the reconciliation problem between a gene tree and a species network when we consider a scenario where the transfer events are restricted to happen only through the secondary arcs of LGT networks. We provide fast algorithms for the problem of finding the "best" (the most parsimonious) switching of the network, and also the MPR between a gene tree and an LGT network, improving previous results on this field.

Conclusions and future work

In this dissertation we have introduced *LGT networks*, which have proved to be a solid model for representing the evolution of species with presence of lateral genetic transfer events. Until now, the models used to describe the evolution of species under this kind of events were either phylogenetic networks, that did not take into account that there is a principal line of evolution, or trees with "extra information" that were not mathematically well founded. We have developed methods for the reconstruction of LGT networks over a set of species when only partial information is available. The general philosophy is that one can either have ambiguous information on the evolutionary relationship for all species (and get different large trees on the same set of taxa) or have very precise information on this evolution but only for 3-tuples of species (and get networks on different small sets of taxa). We have put these two generic problems into the context of LGT networks and described two different classes of LGT networks where they can be solved effectively. Finally we have shown how our model for networks is also well suited to study the reconciliation problem between the evolutionary history of genes and species, reducing the complexity of the problem from NP-hard to polynomial.

During the course of the research that has led to this dissertation, we have attacked problems that we have only been able to solve partially, and for some others we have completely failed to give a solution. We have also devised unexplored areas, but the lack of time has prevented us to visit them. All these lacks and failures are nothing but opportunities to keep working on the subject. We review now some possibilities for future work.

In Chapter 2 we generalized the Robinson-Foulds distance to generic LGT networks. While having a sound distance metric to compare LGT networks is important in order to assess the validity of reconstructions methods, this concrete distance may not be the more appropriate in some cases. For instance, the distance we introduced is very sensitive to small changes in the principal tree, since they change many secondary arcs (or at least how we compare them). We plan to develop new distances that can address this problem. For instance, modifying the definition of the clusters so that they reflect directly the existence of secondary paths to the leaves and hence those can be compared directly.

In Chapter 3 we showed how a certain kind of LGT networks can be singled out by means of its principal and secondary subtrees. For this kind of networks, the "first degree" approximation given by these subtrees are enough to determine the network. In some other cases, maybe, knowing not only the subtrees induced by a single secondary arc, but two (or more) of them could allow us to recover wider classes of networks. As a future work we plan to investigate whether a "higher degree" approximation of LGT networks can determine them completely.

One of the modern ways to tackle the reconstruction of phylogenetic trees is through the use of algebraic invariants associated to a given statistical model of evolution on them. In one of the experiments in Chapter 3 we introduced a simple statistical model of evolution on LGT networks. We plan to study further this model of evolution and, although our attempt to apply the techniques of algebraic invariants to LGT networks led us to a dead end, we will consider it again at some time in future.

In Chapter 4 we faced the problem of finding a full description of the sets of secondary arcs that are coverings of another secondary arc. For partial coverings of type 1 or 5 we could find such a description in a quite simple way for generic LGT networks. However, such a description for the remaining types was only found for time-consistent LGT networks. We think that having simple descriptions for generic networks for any kind of coverings could improve the clarity of our results, and hence we plan to investigate this coverings, which could lead us to characterize more generic LGT networks by means of the triplets that they display.

In Chapter 5 we studied the reconciliation problem, more precisely we obtained the most parsimonious reconciliation between a unique given gene tree and a given species network. Two different generalizations of the problem could arise: one would be the reconciliation between a gene tree forest into one common species network; and the second would be based on the assumption that the species network is not known and on trying to build the LGT network together with the most parsimonious reconciliation, simultaneously.

Bibliography

Abby, S. S., Tannier, E., Gouy, M., and Daubin, V. (2010). Detecting lateral gene transfers by statistical reconciliation of phylogenetic forests. *BMC bioinformatics*, 11(1):324.

Addario-Berry, L., Hallett, M. T., and Lagergren, J. (2003). Towards identifying lateral gene transfer events. In *Pacific Symposium on Biocomputing*, volume 8, pages 279–290. Citeseer.

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

Albrecht, B. (2015). Computing all hybridization networks for multiple binary phylogenetic input trees. *BMC bioinformatics*, 16(1):236.

Allen, B. L. and Steel, M. (2001). Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of combinatorics*, 5(1):1–15.

Arvestad, L., Berglund, A.-C., Lagergren, J., and Sennblad, B. (2004). Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 326–335. ACM.

Bansal, M. S., Alm, E. J., and Kellis, M. (2012). Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. *Bioinformatics*, 28(12):i283–i291.

Bansal, M. S., Banay, G., Harlow, T. J., Gogarten, J. P., and Shamir, R. (2013). Systematic inference of highways of horizontal gene transfer in prokaryotes. *Bioinformatics*, 29(5):571–579.

Baroni, M., Semple, C., and Steel, M. (2005). A framework for representing reticulate evolution. *Annals of Combinatorics*, 8(4):391–408.

Baroni, M., Semple, C., and Steel, M. (2006). Hybrids in real time. *Systematic Biology*, 55(1):46–56.

Baroni, M. and Steel, M. (2006). Accumulation phylogenies. *Annals of Combinatorics*, 10(1):19–30.

Beiko, R. G. et al. (2011). Telling the whole story in a 10,000-genome world. *Biol Direct*, 6:34.

Beiko, R. G. and Hamilton, N. (2006). Phylogenetic identification of lateral genetic transfer events. *BMC evolutionary biology*, 6(1):15.

Benveniste, R. E. and Todaro, G. J. (1974). Evolution of c-type viral genes: inheritance of exogenously acquired viral genes. *Nature*, 252:456–459.

Bordewich, M., Linz, S., St John, K., and Semple, C. (2007). A reduction algorithm for computing the hybridization number of two trees. *Evolutionary Bioinformatics*, 3.

Bordewich, M. and Semple, C. (2005). On the computational complexity of the rooted subtree prune and regraft distance. *Annals of combinatorics*, 8(4):409–423.

Bordewich, M. and Semple, C. (2007). Computing the minimum number of hybridization events for a consistent evolutionary history. *Discrete Applied Mathematics*, 155(8):914–928.

Bordewich, M. and Semple, C. (2015a). Determining phylogenetic networks from inter-taxa distances. *Journal of Mathematical Biology*, pages 1–21.

Bordewich, M. and Semple, C. (2015b). Reticulation-visible networks. *arXiv preprint arXiv:1508.05424*.

Boto, L. (2010). Horizontal gene transfer in evolution: facts and challenges. *Proceedings of the Royal Society of London B: Biological Sciences*, 277(1683):819–827.

Brady, A. and Salzberg, S. (2011). Phymmbl expanded: confidence scores, custom databases, parallelization and more. *Nature methods*, 8(5):367–367.

Brodal, G. S., Fagerberg, R., Mailund, T., Pedersen, C. N., and Sand, A. (2013). Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1814–1832. SIAM.

Brooks, D. R. and Ferrao, A. L. (2005). The historical biogeography of co-evolution: emerging infectious diseases are evolutionary accidents waiting to happen. *Journal of Biogeography*, 32(8):1291–1299.

Bryant, D. and Moulton, V. (2004). Neighbor-net: an agglomerative method for the construction of phylogenetic networks. *Molecular biology and evolution*, 21(2):255–265.

Campbell, P. J., Yachida, S., Mudie, L. J., Stephens, P. J., Pleasance, E. D., Stebbings, L. A., Morsberger, L. A., Latimer, C., McLaren, S., Lin, M.-L., et al. (2010). The patterns and dynamics of genomic instability in metastatic pancreatic cancer. *Nature*, 467(7319):1109–1113.

Cardona, G., Llabrés, M., and Rosselló, F. (2010a). Two results on distances for phylogenetic networks. In *Advances in Bioinformatics*, pages 93–100. Springer.

Cardona, G., Llabrés, M., Rosselló, F., and Valiente, G. (2008a). A distance metric for a class of tree-sibling phylogenetic networks. *Bioinformatics*, 24(13):1481–1488.

Cardona, G., Llabrés, M., Rosselló, F., and Valiente, G. (2009a). The comparison of tree-sibling time consistent phylogenetic networks is graph isomorphism-complete. *arXiv preprint arXiv:0902.4640*.

Cardona, G., Llabrés, M., Rosselló, F., and Valiente, G. (2009b). Metrics for phylogenetic networks i: Generalizations of the robinson-foulds metric. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 6(1):46–61.

Cardona, G., Llabres, M., Rossello, F., and Valiente, G. (2009c). Metrics for phylogenetic networks ii: Nodal and triplets metrics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 6(3):454–469.

Cardona, G., Llabres, M., Rosselló, F., and Valiente, G. (2009d). On nakhleh's metric for reduced phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 6(4):629–638.

Cardona, G., Llabrés, M., Rosselló, F., and Valiente, G. (2010b). Nodal distances for rooted phylogenetic trees. *Journal of mathematical biology*, 61(2):253–276.

Cardona, G., Llabrés, M., Rosselló, F., and Valiente, G. (2010c). Path lengths in tree-child time consistent hybridization networks. *Information Sciences*, 180(3):366–383.

Cardona, G., Pons, J. C., and Rosselló, F. (2015). A reconstruction problem for a class of phylogenetic networks with lateral gene transfers. *Algorithms for Molecular Biology*, 10(1):1.

Cardona, G., Rosselló, F., and Valiente, G. (2008b). Extended newick: it is time for a standard representation of phylogenetic networks. *BMC bioinformatics*, 9(1):532.

Cardona, G., Rosselló, F., and Valiente, G. (2008c). Tripartitions do not always discriminate phylogenetic networks. *Mathematical biosciences*, 211(2):356–370.

Cardona, G., Rossello, F., and Valiente, G. (2009e). Comparison of tree-child phylogenetic networks. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 6(4):552–569.

Charleston, M. A. and Perkins, S. L. (2006). Traversing the tangle: algorithms and applications for cophylogenetic studies. *Journal of biomedical informatics*, 39(1):62–71.

Chen, Z.-Z. and Wang, L. (2010). Hybridnet: a tool for constructing hybridization networks. *Bioinformatics*, 26(22):2912–2913.

Chia, N. and Goldenfeld, N. (2011). Statistical mechanics of horizontal gene transfer in evolutionary ecology. *Journal of Statistical Physics*, 142(6):1287–1301.

Choy, C., Jansson, J., Sadakane, K., and Sung, W.-K. (2005). Computing the maximum agreement of phylogenetic networks. *Theoretical Computer Science*, 335(1):93–107.

Collins, J., Linz, S., and Semple, C. (2011). Quantifying hybridization in realistic time. *Journal of Computational Biology*, 18(10):1305–1318.

Conow, C., Fielder, D., Ovadia, Y., and Libeskind-Hadas, R. (2010). Jane: a new tool for the cophylogeny reconstruction problem. *Algorithms for Molecular Biology*, 5(1):1.

Crow, J. F. et al. (1986). *Basic concepts in population, quantitative, and evolutionary genetics.* WH Freeman and Company.

Dagan, T. and Martin, W. (2006). The tree of one percent. *Genome Biol*, 7(10):118.

Daubin, V. and Szöllősi, G. J. (2016). Horizontal gene transfer and the history of life. *Cold Spring Harbor Perspectives in Biology*, page a018036.

De Bruyn, A., Martin, D. P., and Lefeuvre, P. (2014). Phylogenetic reconstruction methods: an overview. *Molecular Plant Taxonomy: Methods and Protocols*, pages 257–277.

Doolittle, W. F. (1999). Phylogenetic classification and the universal tree. *Science*, 284(5423):2124–2128.

Doolittle, W. F. and Bapteste, E. (2007). Pattern pluralism and the tree of life hypothesis. *Proceedings of the National Academy of Sciences*, 104(7):2043–2049.

Doyon, J.-P., Chauve, C., and Hamel, S. (2009). Space of gene/species trees reconciliations and parsimonious models. *Journal of Computational Biology*, 16(10):1399–1418.

Doyon, J.-P., Ranwez, V., Daubin, V., and Berry, V. (2011). Models, algorithms and programs for phylogeny reconciliation. *Briefings in bioinformatics*, 12(5):392–400.

Doyon, J.-P., Scornavacca, C., Gorbunov, K. Y., Szöllősi, G. J., Ranwez, V., and Berry, V. (2010). An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. In *Comparative genomics*, pages 93–108. Springer.

Dress, A., Huber, K. T., and Koolen, J. (2012). *Basic phylogenetic combinatorics*. Cambridge University Press.

Edwards, S. V. (2009). Is a new and general theory of molecular systematics emerging? *Evolution*, 63(1):1–19.

Felsenstein, J. (2004). *Inferring phylogenies*, volume 2. Sinauer Associates Sunderland.

Francis, A. R. and Steel, M. (2015a). Tree-like reticulation networks—when do tree-like distances also support reticulate evolution? *Mathematical biosciences*, 259:12–19.

Francis, A. R. and Steel, M. (2015b). Which phylogenetic networks are merely trees with additional arcs? *Systematic biology*, 64(5):768–777.

Frank, A. C., Alsmark, C. M., Thollesson, M., and Andersson, S. G. (2005). Functional divergence and horizontal transfer of type iv secretion systems. *Molecular biology and evolution*, 22(5):1325–1336.

Gambette, P. (2010). Who is who in phylogenetic networks: Articles, authors and programs. *Published electronically at http://www. atgc-montpellier. fr/phylnet.*

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

Gambette, P., Berry, V., and Paul, C. (2012). Quartets and unrooted phylogenetic networks. *Journal of bioinformatics and computational biology*, 10(04):1250004.

Gambette, P., Gunawan, A. D., Labarre, A., Vialette, S., and Zhang, L. (2015a). Locating a tree in a phylogenetic network in quadratic time. In *Research in Computational Molecular Biology*, pages 96–107. Springer.

Gambette, P., Gunawan, A. D., Labarre, A., Vialette, S., and Zhang, L. (2015b). Solving the tree containment problem for genetically stable networks in quadratic time. In *IWOCA 2015*, pages to–appear. Springer.

Gambette, P., Huber, K., and Kelk, S. (2015c). On the challenge of reconstructing level-1 phylogenetic networks from triplets and clusters. *arXiv preprint arXiv:1511.08056*.

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

Gilbert, C., Schaack, S., Pace II, J. K., Brindley, P. J., and Feschotte, C. (2010). A role for host-parasite interactions in the horizontal transfer of transposons across phyla. *Nature*, 464(7293):1347–1350.

Goodman, M., Czelusniak, J., Moore, G. W., Romero-Herrera, A., and Matsuda, G. (1979). Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Biology*, 28(2):132–163.

Gorbunov, K. Y. and Lyubetsky, V. (2009). Reconstructing the evolution of genes along the species tree. *Molecular Biology*, 43(5):881–893.

Górecki, P. (2004). Reconciliation problems for duplication, loss and horizontal gene transfer. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 316–325. ACM.

Górecki, P. (2010). H-trees: a model of evolutionary scenarios with horizontal gene transfer. *Fundamenta Informaticae*, 103(1-4):105–128.

Górecki, P., Burleigh, G. J., and Eulenstein, O. (2011). Maximum likelihood models and algorithms for gene tree evolution with duplications and losses. *BMC bioinformatics*, 12(1):1.

Górecki, P. and Tiuryn, J. (2006). Dls-trees: a model of evolutionary scenarios. *Theoretical computer science*, 359(1):378–399.

Górecki, P. and Tiuryn, J. (2012). Inferring evolutionary scenarios in the duplication, loss and horizontal gene transfer model. In *Logic and Program Semantics*, pages 83–105. Springer.

Gray, R. D., Drummond, A. J., and Greenhill, S. J. (2009). Language phylogenies reveal expansion pulses and pauses in pacific settlement. *science*, 323(5913):479–483.

Grenfell, B. T., Pybus, O. G., Gog, J. R., Wood, J. L., Daly, J. M., Mumford, J. A., and Holmes, E. C. (2004). Unifying the epidemiological and evolutionary dynamics of pathogens. *science*, 303(5656):327–332.

Guigo, R., Muchnik, I., and Smith, T. F. (1996). Reconstruction of ancient molecular phylogeny. *Molecular phylogenetics and evolution*, 6(2):189–213.

Gunawan, A. D. and Zhang, L. (2015). Bounding the size of a network defined by visibility property. *arXiv preprint arXiv:1510.00115*.

Gusfield, D., Eddhu, S., and Langley, C. (2003). Efficient reconstruction of phylogenetic networks with constrained recombination. In *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 363–374. IEEE.

Habib, M. and To, T.-H. (2012). Constructing a minimum phylogenetic network from a dense triplet set. *Journal of bioinformatics and computational biology*, 10(05):1250013.

Hallett, M., Lagergren, J., and Tofigh, A. (2004). Simultaneous identification of duplications and lateral transfers. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 347–356. ACM.

Hallett, M. T. and Lagergren, J. (2001). Efficient algorithms for lateral gene transfer problems. In *Proceedings of the fifth annual international conference on Computational biology*, pages 149–156. ACM.

Hao, W. and Golding, G. (2004). Patterns of bacterial gene movement. *Molecular biology and evolution*, 21(7):1294–1307.

Hein, J. (1990). Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical biosciences*, 98(2):185–200.

Hein, J., Jiang, T., Wang, L., and Zhang, K. (1996). On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71(1):153–169.

Holmes, E. C., Worobey, M., and Rambaut, A. (1999). Phylogenetic evidence for recombination in dengue virus. *Molecular biology and evolution*, 16(3):405–409.

Hotopp, J. C. D. (2011). Horizontal gene transfer between bacteria and animals. *Trends in Genetics*, 27(4):157–163.

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

Huber, K. T., Moulton, V., Steel, M., and Wu, T. (2015a). Folding and unfolding phylogenetic trees and networks. *arXiv preprint arXiv:1506.04438*.

Huber, K. T., Van Iersel, L., Kelk, S., and Suchecki, R. (2011). A practical algorithm for reconstructing level-1 phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(3):635–649.

Huber, K. T., van Iersel, L., Moulton, V., Scornavacca, C., and Wu, T. (2015b). Reconstructing phylogenetic level-1 networks from nondense binet and trinet sets. *Algorithmica*, pages 1–28.

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

Huson, D. H. and Klöpper, T. H. (2007). Beyond galled trees-decomposition and computation of galled networks. In *Research in Computational Molecular Biology*, pages 211–225. Springer.

Huson, D. H. and Rupp, R. (2008). Summarizing multiple gene trees using cluster networks. In *Algorithms in bioinformatics*, pages 296–305. Springer.

Huson, D. H., Rupp, R., Berry, V., Gambette, P., and Paul, C. (2009). Computing galled networks from real data. *Bioinformatics*, 25(12):i85–i93.

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

Huson, D. H. and Scornavacca, C. (2011). A survey of combinatorial methods for phylogenetic networks. *Genome biology and evolution*, 3:23–35.

Jacox, E., Chauve, C., Szöllősi, G. J., Ponty, Y., and Scornavacca, C. (2016). eccetera: Comprehensive gene tree-species tree reconciliation using parsimony. *Bioinformatics*, page btw105.

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

Jansson, J. and Sung, W.-K. (2006). Inferring a level-1 phylogenetic network from a dense set of rooted triplets. *Theoretical Computer Science*, 363(1):60–68.

Jin, G., Nakhleh, L., Snir, S., and Tuller, T. (2006). Maximum likelihood of phylogenetic networks. *Bioinformatics*, 22(21):2604–2611.

Jukes, T. H. and Cantor, C. R. (1969). Evolution of protein molecules. *Mammalian protein metabolism*, 3(21):132.

Kanj, I. A., Nakhleh, L., Than, C., and Xia, G. (2008). Seeing the trees and their branches in the network is hard. *Theoretical Computer Science*, 401(1):153–164.

Keeling, P. J. and Palmer, J. D. (2008). Horizontal gene transfer in eukaryotic evolution. *Nature Reviews Genetics*, 9(8):605–618.

Keen, E. C. (2012). Paradigms of pathogenesis: targeting the mobile genetic elements of disease. *Frontiers in cellular and infection microbiology*, 2:161.

Kelk, S. and Scornavacca, C. (2014). Constructing minimal phylogenetic networks from softwired clusters is fixed parameter tractable. *Algorithmica*, 68(4):886–915.

Kellis, M., Patterson, N., Endrizzi, M., Birren, B., and Lander, E. S. (2003). Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423(6937):241–254.

Libeskind-Hadas, R. and Charleston, M. A. (2009). On the computational complexity of the reticulate cophylogeny reconstruction problem. *Journal of Computational Biology*, 16(1):105–117.

Libeskind-Hadas, R., Wu, Y.-C., Bansal, M. S., and Kellis, M. (2014). Pareto-optimal phylogenetic tree reconciliation. *Bioinformatics*, 30(12):i87–i95.

Linder, C. R., Moret, B. M., Nakhleh, L., Padolina, A., Sun, J., Tholse, A., Timme, R., and Warnow, T. (2003). An error metric for phylogenetic networks. *University of New Mexico, Tech. Rep. TR03-26*.

Maddison, W. P. (1997). Gene trees in species trees. *Systematic biology*, 46(3):523–536.

Makarenkov, V. (2001). T-rex: reconstructing and visualizing phylogenetic trees and reticulation networks. *Bioinformatics*, 17(7):664–668.

Mallet, J. (2007). Hybrid speciation. *Nature*, 446(7133):279–283.

Martin, D. P., Murrell, B., Golden, M., Khoosal, A., and Muhire, B. (2015). Rdp4: Detection and analysis of recombination patterns in virus genomes. *Virus Evolution*, 1(1):vev003.

Martin, W. F. (2011). Early evolution without a tree of life. *Biol Direct*, 6:36.

Mäser, P., Thomine, S., Schroeder, J. I., Ward, J. M., Hirschi, K., Sze, H., Talke, I. N., Amtmann, A., Maathuis, F. J., Sanders, D., et al. (2001). Phylogenetic relationships within cation transporter families of arabidopsis. *Plant Physiology*, 126(4):1646–1667.

McDaniel, L. D., Young, E., Delaney, J., Ruhnau, F., Ritchie, K. B., and Paul, J. H. (2010). High frequency of horizontal gene transfer in the oceans. *Science*, 330(6000):50–50.

Merkle, D. and Middendorf, M. (2005). Reconstruction of the cophylogenetic history of related phylogenetic trees with divergence timing information. *Theory in Biosciences*, 123(4):277–299.

Merkle, D., Middendorf, M., and Wieseke, N. (2010). A parameter-adaptive dynamic programming approach for inferring cophylogenies. *BMC bioinformatics*, 11(1):1.

Moret, B. M., Nakhleh, L., and Warnow, T. (2002). An error metric for phylogenetic networks. *University of New Mexico, Tech. Rep. TR02-09*.

Moret, B. M., Nakhleh, L., Warnow, T., Linder, C. R., Tholse, A., Padolina, A., Sun, J., and Timme, R. (2004). Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):13–23.

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

Morrison, D. A. (2013). The genealogical world of phylogenetic networks: Different topological restrictions of rooted phylogenetic networks. which make biological sense? URL http://phylonetworks.blogspot.com.es/2013/03/different-topological-restrictions-of.html.

Morrison, D. A. (2014). The genealogical world of phylogenetic networks: The first hgt network. URL http://phylonetworks.blogspot.com.es/2014/04/the-first-hgt-network.html.

Nakhleh, L. (2004). Phylogenetic networks.

Nakhleh, L. (2010a). Evolutionary phylogenetic networks: models and issues. In *Problem solving handbook in computational biology and bioinformatics*, pages 125–158. Springer.

Nakhleh, L. (2010b). A metric on the space of reduced phylogenetic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 7(2):218–222.

Nakhleh, L. (2013). Computational approaches to species phylogeny inference and gene tree reconciliation. *Trends in ecology &amp; evolution*, 28(12):719–728.

Nakhleh, L., Ruths, D., and Innan, H. (2009). Gene trees, species trees, and species networks.

Nakhleh, L., Ruths, D., and Wang, L.-S. (2005). Riata-hgt: a fast and accurate heuristic for reconstructing horizontal gene transfer. In *Computing and Combinatorics*, pages 84–93. Springer.

Nei, M. (1987). *Molecular evolutionary genetics*. Columbia university press.

Nieberding, C., Jousselin, E., Desdevises, Y., et al. (2010). The use of co-phylogeographic patterns to predict the nature of host–parasite interactions, and vice versa. *The biogeography of host-parasite interactions*, pages 631–641.

Nikolaidis, N., Doran, N., and Cosgrove, D. J. (2014). Plant expansins in bacteria and fungi: evolution by horizontal gene transfer and independent domain fusion. *Molecular biology and evolution*, 31(2):376–386.

Ochman, H., Lawrence, J. G., and Groisman, E. A. (2000). Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405(6784):299–304.

Oldman, J., Wu, T., van Iersel, L., and Moulton, V. (2016). Trilonet: Piecing together small networks to reconstruct reticulate evolutionary histories. *Molecular biology and evolution*, page msw068.

Otto, S. P. and Day, T. (2007). *A biologist's guide to mathematical modeling in ecology and evolution*, volume 13. Princeton University Press.

Page, R. D. (1994). Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Systematic Biology*, 43(1):58–77.

Page, R. D. and Charleston, M. A. (1998). Trees within trees: phylogeny and historical associations. *Trends in Ecology &amp; Evolution*, 13(9):356–359.

Paten, B., Herrero, J., Fitzgerald, S., Beal, K., Flicek, P., Holmes, I., and Birney, E. (2008). Genome-wide nucleotide-level mammalian ancestor reconstruction. *Genome research*, 18(11):1829–1843.

Plotree, D. and Plotgram, D. (1989). Phylip-phylogeny inference package (version 3.2). *cladistics*, 5:163–166.

Poggio, S., Abreu-Goodger, C., Fabela, S., Osorio, A., Dreyfus, G., Vinuesa, P., and Camarena, L. (2007). A complete set of flagellar genes acquired by horizontal transfer coexists with the endogenous flagellar system in rhodobacter sphaeroides. *Journal of bacteriology*, 189(8):3208–3216.

Polz, M. F., Alm, E. J., and Hanage, W. P. (2013). Horizontal gene transfer and the evolution of bacterial and archaeal population structure. *Trends in Genetics*, 29(3):170–175.

Purves, W. K., Orians, G. H., Sadava, D., and Heller, H. C. (2003). *Life: The Science of Biology: Volume III: Plants and Animals*, volume 3. Macmillan.

Ranwez, V., Scornavacca, C., Doyon, J.-P., and Berry, V. (2015). Inferring gene duplications, transfers and losses can be done in a discrete framework. *Journal of mathematical biology*, pages 1–34.

Rasmussen, M. D. and Kellis, M. (2012). Unified modeling of gene duplication, loss, and coalescence using a locus tree. *Genome Research*, 22(4):755–765.

Robinson, D. F. (1971). Comparison of labeled trees with valency three. *Journal of Combinatorial Theory, Series B*, 11(2):105–119.

Ronquist, F. (1995). Reconstructing the history of host-parasite associations using generalised parsimony. *Cladistics*, 11(1):73–89.

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

Scornavacca, C., Paprotny, W., Berry, V., and Ranwez, V. (2013). Representing a set of reconciliations in a compact way. *Journal of bioinformatics and computational biology*, 11(02):1250025.

Seehausen, O., Terai, Y., Magalhaes, I. S., Carleton, K. L., Mrosso, H. D., Miyagi, R., van der Sluijs, I., Schneider, M. V., Maan, M. E., Tachida, H., et al. (2008). Speciation through sensory drive in cichlid fish. *Nature*, 455(7213):620–626.

Semple, C. (2006). Hybridization networks.

Semple, C. and Steel, M. A. (2003). *Phylogenetics*, volume 24. Oxford University Press on Demand.

Szöllősi, G. J., Tannier, E., Lartillot, N., and Daubin, V. (2013). Lateral gene transfer from the dead. *Systematic biology*, 62(3):386–397.

To, T.-H. and Habib, M. (2009). Level-k phylogenetic network can be constructed from a dense triplet set in polynomial time. *arXiv preprint arXiv:0901.1657*.

To, T.-H. and Scornavacca, C. (2015). Efficient algorithms for reconciling gene trees and species networks via duplication and loss events. *BMC genomics*, 16(Suppl 10):S6.

Todd, J. D., Curson, A. R., Sullivan, M. J., Kirkwood, M., and Johnston, A. W. (2012). The ruegeria pomeroyi acui gene has a role in dmsp catabolism and resembles yhdh of e. coli and other bacteria in conferring resistance to acrylate. *PLoS One*, 7(4):e35947.

Tofigh, A., Hallett, M., and Lagergren, J. (2011). Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(2):517–535.

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

van Iersel, L. and Kelk, S. (2011). Constructing the simplest possible phylogenetic network from triplets. *Algorithmica*, 60(2):207–235.

Van Iersel, L. and Kelk, S. (2011). When two trees go to war. *Journal of theoretical biology*, 269(1):245–255.

van Iersel, L., Kelk, S., Lekić, N., and Scornavacca, C. (2014). A practical approximation algorithm for solving massive instances of hybridization number for binary and nonbinary trees. *BMC bioinformatics*, 15(1):1.

Van Iersel, L., Kelk, S., and Mnich, M. (2009b). Uniqueness, intractability and exact algorithms: reflections on level-k phylogenetic networks. *Journal of Bioinformatics and Computational Biology*, 7(04):597–623.

Van Iersel, L., Kelk, S., Rupp, R., and Huson, D. (2010a). Phylogenetic networks do not need to be complex: using fewer reticulations to represent conflicting clusters. *Bioinformatics*, 26(12):i124–i131.

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

Van Iersel, L., Semple, C., and Steel, M. (2010b). Locating a tree in a phylogenetic network. *Information Processing Letters*, 110(23):1037–1043.

Vernot, B., Stolzer, M., Goldman, A., and Durand, D. (2008). Reconciliation with nonbinary species trees. *Journal of Computational Biology*, 15(8):981–1006.

Wang, J., Guo, M., Xing, L., Che, K., Liu, X., and Wang, C. (2013). Bimlr: A method for constructing rooted phylogenetic networks from rooted phylogenetic trees. *Gene*, 527(1):344–351.

Warnow, L. N. J. S. T., Linder, C. R., and Tholse, B. M. M. A. (2003). Towards the development of computational tools for evaluating phylogenetic network reconstruction methods. In *Proc. Eighth Pacific Symp. Biocomputing (PSB'03)*, pages 315–326.

Willson, S. (2011). Regular networks can be uniquely constructed from their trees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(3):785–796.

Willson, S. J. (2010). Properties of normal phylogenetic networks. *Bulletin of mathematical biology*, 72(2):340–358.

Willson, S. J. (2013). Reconstruction of certain phylogenetic networks from their tree-average distances. *Bulletin of mathematical biology*, 75(10):1840–1878.

Wu, Y. (2013). An algorithm for constructing parsimonious hybridization networks with multiple phylogenetic trees. *Journal of Computational Biology*, 20(10):792–804.

Yu, Y. and Nakhleh, L. (2015). A maximum pseudo-likelihood approach for phylogenetic networks. *BMC genomics*, 16(10):1.

Yue, J., Hu, X., Sun, H., Yang, Y., and Huang, J. (2012). Widespread impact of horizontal gene transfer on plant colonization of land. *Nature communications*, 3:1152.

Zhang, L. (1997). On a mirkin-muchnik-smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology*, 4(2):177–187.