Trees and Terraces

Sarah Jayne Mark

A thesis submitted in partial fulfilment

of the requirements for the degree of

Masters of Science in Mathematics

University of Canterbury

New Zealand

June 2016

Contents

Abstract

The reconstruction of evolutionary trees from data sets on overlapping sets of species is a central problem in phylogenetics. Provided that the tree reconstructed for each subset of species is rooted and that these trees fit together consistently, the space of all parent trees that 'display' these trees was recently shown to satisfy the following property: there exists a path from any one parent tree to any other parent tree by a sequence of local rearrangements (nearest neighbour interchanges) so that each intermediate tree also lies in this same tree space. However, the proof of this result uses a non-constructive argument. In this thesis we describe a specific, polynomial-time procedure for navigating from any given parent tree to another while remaining in this tree space. We then investigate a related problem, the conditions under which there is only one parent tree. These results are of particular relevance to the recent study of 'phylogenetic terraces'.

Deputy Vice-Chancellor's Office
Postgraduate Office

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Co-Authorship Form

This form is to accompany the submission of any thesis that contains research reported in co-authored work that has been published, accepted for publication, or submitted for publication. A copy of this form should be included for each co-authored work that is included in the thesis. Completed forms should be included at the front (after the thesis abstract) of each copy of the thesis submitted for examination and library deposit.

Please indicate the chapter/section/pages of this thesis that are extracted from co-authored work and provide details of the publication or submission from the extract comes:

Chapter 2 of the thesis is extracted from Sections 2.3, 3, and 4 of the paper A navigation system for tree space *published in the Journal of Graph Algorithms and Applications, Vol 20, no 2, pages 247-268.*

Some parts of Chapter 1 of the thesis are extracted from Section 2 of the aforementioned paper.

Please detail the nature and extent (%) of contribution by the candidate:

The work in the paper was carried out by the candidate under the supervision and guidance of Jeanette McLeod and Mike Steel. (100%)

Certification by Co-authors:
If there is more than one co-author then a single co-author can sign on behalf of all
The undersigned certifys that:

- The above statement correctly reflects the nature and extent of the PhD candidate's contribution to this co-authored work
- In cases where the candidate was the lead author of the co-authored work he or she wrote the text

Name: *Sarah Mark* Signature: *Sarah Mark* Date: *14-06-16*

Acknowledgements

I would like to thank my supervisors Dr Jeanette McLeod and Professor Mike Steel for their guidance and support. I could not have imagined having a better supervisory team and mentors. They were always there to listen and give advice, and their patience and encouragement helped me persevere and overcome any obstacles I faced. I am also thankful for their insightful comments on countless revisions of this thesis.

I would also like to thank my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

This research was supported by scholarships from the Biomathematics Research Centre and the University of Canterbury.

Chapter 1

Introduction

1.1 Overview

A central goal in systematic biology is to reconstruct and analyze a (phylogenetic) tree to describe the evolutionary relationships among present-day species. This reconstruction is based on a comparison of the species' genetic data [4], an activity which has accelerated greatly in recent years due to the rapid advances in new genomic sequencing technology. While biologists in the 1970s might have reconstructed a tree for a dozen species using a single gene, today, phylogenetic trees are routinely constructed for hundreds or thousands of species, often based on hundreds or thousands of genes. These phylogenetic trees reveal how species today trace back to a common ancestor, by displaying the branching pattern and timing of separation events. For the group of species under study, they also provide insights into how particular evolutionary innovations arose (e.g. multicellularity, photosynthesis, wings, large brains, etc). Phylogenetic trees can also shed light on the amount of biodiversity captured by different subsets of species, and how much of this biodiversity may be at risk from extinction in the near future (a recent example is the analysis in [5] of the reconstructed tree for all $\sim 10,000$ species of birds).

Tree reconstruction methods often attempt to combine the evolutionary information of many different genes. One of the problems with such an approach is that each gene may be present in only a subset of the species; this is known as *patchy taxon coverage*. This may be because the gene simply does not exist in some species or because the gene, though present, is yet to be sequenced for those species. Moreover, the set of species that lack a given gene typically varies from gene to gene. Attempting to combine the information in these overlapping data sets often results in large collections of trees, each of which display all of the available information/evolutionary relationships.

Patchy taxon coverage has a direct combinatorial consequence for tree reconstruction methods, which often seek to optimize (e.g. minimize) some objective function based on how well the data 'fit' each tree. The result can be large collections of equally-optimal trees (i.e. a flat landscape of trees), that form a (phylogenetic) 'terrace' [9]. Consider a set of species X and the subset X_G of these species for which gene G is present. Suppose that T is a fully-resolved (i.e. binary) tree for set of species X. Consider a scoring function s that assigns a positive real value for each such pair (G, T). In biological applications, s will generally satisfy the following equation:

$$s(G, T) = s(G, T|X_G), \tag{1.1}$$

where $T|X_G$ is the phylogenetic tree with set of species X_G obtained from T by deleting all species in X for which gene G is not present. This condition essentially says that species for which the gene is not present should not affect how well the data for the gene 'fits' the tree under consideration.

Now suppose the data contains of a sequence of genes $\mathcal{G} = (G_1, G_2, \ldots, G_k)$ rather than just a single gene. Given the score $s(G_i, T)$ for each i, how might we combine them to obtain a score $s(\mathcal{G}, T)$, called the s-score of T, for how well this collection of genes

'fits' T? One option is simply to form a linear sum and let

$$s(\mathcal{G}, T) = \sum_{i=1}^{k} s(G_i, T). \tag{1.2}$$

We say that any such scoring scheme is *linear*. Given the sequence of genes $\mathcal{G}$, we seek to find a tree that minimizes $s(\mathcal{G}, T)$. While linearity may seem a strong condition to impose, it turns out that some standard phylogenetic methods select a tree that minimizes a linear scoring scheme, such as maximum likelihood or maximum parsimony (see [9, 8] for further details). Both of these methods also satisfy Eqn. (1.1), as do several others that fail linearity.

Now suppose that T^* is a fully-resolved tree that has some particular score (e.g. the optimal score) for $\mathcal{G}$ under a scoring function s that satisfies Eqn. (1.1) and is linear (Eqn. (1.2)). Suppose that T is any other tree for which $T|X_i = T^*|X_i$ for all i, i.e. when we restrict both T and T^* to the set of species X_{G_i}, we obtain the same tree. The tree T then has the same score as T^* for $\mathcal{G}$. To see this, simply observe that

$$s(\mathcal{G}, T) = \sum_{i=1}^{k} s(G_i, T) = \sum_{i=1}^{k} s(G_i, T|X_{G_i}) = \sum_{i=1}^{k} s(G_i, T^*|X_{G_i}) = \sum_{i=1}^{k} s(G_i, T^*) = s(\mathcal{G}, T^*).$$

The set of all phylogenetic trees T with set of species X for which $T|X_i = T^*|X_i$ for all i is referred to as the *terrace* containing T^*. Note that all trees on this terrace have the same s–score (when Eqns. (1.1) and (1.2) hold). In real applications, a terrace can be very large, for example, 61 million equally-optimal (maximum likelihood) trees for a data set consisting of 298 species of grasses on three genes [9]. The existence of large flat landscapes of trees can make the search for optimal trees by hill-climbing approaches more problematic, and ways in which search times on terraces can be improved are still being sought [3].

In this thesis, we explore a further combinatorial consequence of patchy taxon coverage: namely, for any terrace of trees (which thus have the same s-score), it is possible to move from any one tree on the terrace to any other tree on the terrace by making

a series of local elementary tree re-arrangments, while always remaining on that terrace (i.e. not altering the s-score). These local re-arrangements are called 'nearest neighbour interchange' (NNI) operations. This result follows from a theorem, first stated and proved in the PhD thesis of Magnus Bordewich [2] using an inductive argument. The motivation for this thesis is to provide an explicit algorithm for constructing a sequence of NNI operations to move from any tree on a terrace to any other tree on the same terrace. In our approach, the details of the scoring function s play no real role since a terrace is the set of binary trees displaying the set of trees $\{T|X_1, \ldots, T|X_k\}$, where T is some tree on that terrace. Thus we deal simply with sets of trees on overlapping sets of species as our input.

It is important to note that, although the trees comprising a terrace have the same s–score (when Eqns. (1.1) and (1.2) hold), there may be other trees with the same s–score that are not on this terrace. We can see this even in the simple case where $X_i = X$ for all i (i.e. we have complete taxon coverage). Then for any fully-resolved tree T, the terrace containing T is just T itself, yet for certain data there may be many resulting maximum parsimony trees. Moreover, in this setting of complete taxon coverage, we have known since the early 1990s that for particular data, there may be two or more optimal trees with optimal parsimony scores that cannot be connected by a sequence of NNI operations passing only through optimal trees [6]). The result in this thesis concerns connectivity under NNI within a single terrace and so its relevance is particular to the setting of partial taxon coverage. In general, the set of trees with a given s–score is be a union of one or more terraces.

The structure of this thesis is as follows. The remainder of this chapter is dedicated to defining necessary terms and operations on trees. In Chapter 2 we summarize a result from Bordewich's PhD thesis [2], then present our main result, namely that we can construct, in polynomial time, a sequence of trees from any tree on a terrace to any other

tree on the terrace, where all of the intermediate trees are also on the terrace. This proof yields a polynomial-time algorithm to obtain the aforementioned sequence of trees. In Chapter 3, we investigate the related problem of the conditions under which two trees on overlapping sets of species define a single tree. We present and independently prove two theorems previously proved in [1], then summarize our investigations into two much harder, more general versions of this problem and discuss the many cases and conditions which need to be considered.

1.2 Graphs

We begin by providing some necessary definitions and direct the reader to Semple and Steel [10] for further definitions and terminology. The focus of this thesis is trees, but before we can formally define a tree, we must first define a graph. A *graph* $G = (V(G), E(G))$ is a nonempty set $V(G)$ of *vertices*, together with a (possibly empty) set $E(G)$ of unordered pairs of vertices of G, called *edges*. Let $e = \{u, v\}$ be an edge of a graph G. We call u and v the *endpoints* of e. We say u and v are *adjacent* vertices, and edge e is *incident* to vertices u and v. The *degree* of a vertex is the number of vertices adjacent to it. A *subgraph* of a graph G is a graph $G' = (V(G'), E(G'))$ for which $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$.

A *walk* in a graph G is a non-empty, finite sequence of vertices $u_1, u_2, \ldots u_k$ for which $u_i \in V(G)$ for all i $(1 \leq i \leq k)$, and $\{u_j, u_{j+1}\} \in E$ for each j $(1 \leq j \leq k-1)$. The *length* of a walk is the number of edges it contains. A *path* in G is a walk in which all the vertices are distinct. A *cycle* is a walk of length at least 3 in which $u_1 = u_k$. A graph is *connected* if, given any two vertices u and v of G, there is a path from u to v. Otherwise, the graph is *disconnected*.

1.3 Phylogenetic trees

A *tree* $T = (V(T), E(T))$ is a graph that is connected and contains no cycles. Note that we are only dealing with finite trees (i.e. trees in which V(T) is finite) in this thesis. A vertex of T with degree one is called a *leaf*. All other vertices are called *interior vertices*. An *interior edge* of T is an edge in which both endpoints are interior vertices. Any subgraph S of a tree is itself a tree and we call S a *subtree* of T.

A *labeled tree* is a tree in which each vertex has been assigned a unique label by which it can be identified. A *semi-labeled tree* is a tree in which some of the vertices have been assigned unique labels.

A *phylogenetic tree* T is a semi-labeled tree in which only the leaves of T are labeled and each interior vertex has degree at least three, as illustrated in Figure 1.1. Note that we sometimes label the interior vertices of a tree (in addition to the leaves) for ease of reference. A *binary phylogenetic tree* is a phylogenetic tree in which each interior vertex has degree three.

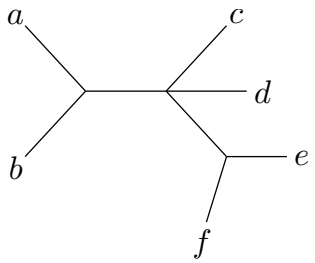

Figure 1.1: An example of a phylogenetic tree.

Let $P(X)$ be the set of all phylogenetic trees with leaf label set X. Consider $\mathcal{P} = \{T_1, \ldots, T_k\}$, where trees $T_1, \ldots T_k$ have leaf sets $X_1, \ldots X_k$ respectively. Then

$$\mathcal{L}(\mathcal{P}) = \bigcup_{i=1}^{k} X_i$$

denotes the leaf set of $\mathcal{P}$. For a single tree T_i, for ease of notation, we write $\mathcal{L}(\{T_i\}) = \mathcal{L}(T_i) = X_i$.

1.3.1 Equivalence

Two trees $T, T' \in P(X)$ are *equivalent* if there is a map $\phi : V(T) \to V(T')$ such that $\phi(l) = l$ for all $l \in X$ and a map $\psi : E(T) \to E(T')$ such that adjacency is preserved. If T and T' are equivalent, we write $T \cong T'$. If T has subtree t and T' has subtree t', where $t \cong t'$, then, to aid exposition, we say that T' has subtree t.

1.4 Rooted phylogenetic trees

A *rooted tree* is a tree that has exactly one distinguished vertex called the *root*; in Figure 1.2 this is the vertex r. For two vertices v and w of a rooted tree T, if the path from the root of T to w includes v, then we say that w is a *descendant* of v, and v is an *ancestor* of w. Note that a vertex of T is both a descendant and an ancestor of itself. For example, in Figure 1.2, v is an ancestor of v, w, a, b, and c, and these vertices are all descendants of v. If v is an ancestor of w and $\{v, w\}$ is an edge of T (as in Figure 1.2), we say that v is the *parent* of w and w is a *child* of v. In this case, we call the edge $\{v, w\}$ an *arc* from v to w and denote this edge (v, w). Note that a parent of a vertex v is always unique, but v may have many children. For example, in Figure 1.2, v is the parent of w and c, and so w and c are the children of v. The number of children of a vertex v is called the *outdegree* of v.

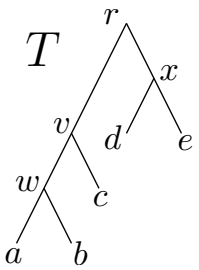

Figure 1.2: An example of a rooted phylogenetic tree.

A *rooted binary tree* is a rooted tree in which every interior vertex has exactly two

children; in other words, all interior vertices have degree 3, except the root, which has degree 2. For example, the tree T in Figure 1.2 is binary.

A *rooted phylogenetic tree* is a rooted tree that is also a phylogenetic tree, i.e. a rooted tree T in which the leaves of T are labeled. Let $RP(X)$ be the set of all such trees with leaf label set X. A *rooted binary phylogenetic tree* is a rooted phylogenetic tree in which all interior vertices have outdegree two. Let $RB(X)$ denote the set of all such trees with leaf label set X. Note that $RB(X) \subseteq RP(X)$. A tree $T \in RP(X)$ with only one interior vertex (which is the root) is called a *star*.

1.4.1 Pendant subtree

Consider a tree $T \in RP(X)$ with vertex v. A *pendant subtree* t_v of T is a subtree of T whose vertex set consists of vertex v and all of its descendants in T. The vertex v is the root of the subtree t_v. If v is not the root of T, then t_v is a *proper pendant subtree* of T. If v is a child of the root of T, then t_v is a *maximal proper pendant subtree* of T. Throughout the rest of this thesis, 'subtree' will refer to a pendant subtree unless otherwise specified.

The *most recent common ancestor* of two vertices v and w is the root of the minimal subtree containing both v and w. For example, in Figure 1.2, the most recent common ancestor of a and x is r, and the most recent common ancestor of v and w is v.

1.4.2 Deletion, contraction, and suppression

Consider a tree $T = (V(T), E(T)) \in RP(X)$, and an arc $e = (v, u)$ of T. The tree $T \setminus e = (V(T), E(T) \setminus \{e\})$ is obtained from T by *deleting* e. Given two vertices u and x of a tree T, *identifying* these two vertices u and x with a single vertex produces a tree in which the vertices u and x are replaced with a new vertex w such that all arcs

incident to u or x are now incident to w. The tree obtained from T by deleting edge $e = (v, u)$ and identifying its endpoints u and v with a single vertex w is denoted T/e and is obtained from T by *contracting* e. Let x be an interior vertex of T. Delete all but one of its outgoing arcs to produce a tree T' in which x has outdegree one. Suppose x is not the root of T' and let $e_1 = (w, x)$ and $e_2 = (x, y)$ be arcs of T'. The tree T'' obtained from T' by deleting vertex x and arcs e_1 and e_2 from T' and inserting arc (w, y) is said to be obtained from T' by *suppressing* x. Note that a tree equivalent to T'' can also be obtained as T'/e_1 or T'/e_2. Now suppose that x is the root of T' and let (x, y) be its incident arc. Then x is suppressed by deleting x and the arc (x, y), producing a tree T'' with root y.

Let v be a vertex of T and let $e_1, \ldots, e_k$ be the arcs of T incident to v. The treee $T \setminus v = (V(T) \setminus \{v\}, E(T) \setminus \{e_1, \ldots, e_k\})$ is obtained from T by *deleting* v. Let $e = (v, u)$ be an arc of T and let t_u be the subtree of T with root u. We define $T \setminus t_u$ to be the tree obtained from T by deleting t_u and the arc e.

1.4.3 Distance

Consider a tree T. The *distance between two vertices*, say u and v, of T, $\text{dist}_T(u, v)$, is the length of the shortest path between u and v in T. The *distance between two edges* $e = \{u_1, u_2\}$ and $e' = \{v_1, v_2\}$ in T is $\text{dist}_T(e, e') = \min\{\text{dist}_T(u_i, v_j) : i, j \in \{1, 2\}\}$. In Figure 1.2, $\text{dist}_T(w, d) = 4$ and $\text{dist}_T((w, b), (x, e)) = 3$ (which is the distance between the vertices w and x).

1.4.4 Cluster

A *cluster* of a tree $T \in RP(X)$ is a subset of X consisting of all leaves that are descendants of a given vertex v in T, $\{x \in X : x \text{ is a descendant of } v\}$, i.e. the leaf set of the subtree

t_v of T. The collection of all clusters of T, denoted $\mathcal{C}(T)$, defines T. A *maximal proper cluster* of T is the leaf set of a maximal proper subtree of T. In Figure 1.2,

$$\mathcal{C}(T) = \{\{a, b, c, d, e\}, \{a, b, c\}, \{a, b\}, \{d, e\}, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}\}.$$

Except for the cluster $\{a, b, c, d, e\}$, these are all maximal proper clusters of T.

1.4.5 Restriction, refinement and display

Let tree $T \in RP(X)$ and let $X' \subseteq X$. Then $T|X' \in RP(X')$, called the *restriction of* T *to* X', is the tree for which

$$\mathcal{C}(T|X') = \{C \cap X' : C \in \mathcal{C}(T) \text{ and } C \cap X' \neq \emptyset\}.$$

We can obtain $T|X'$ from T by deleting all maximal subtrees whose leaf set is a subset of $X \setminus X'$ and then suppressing all vertices with outdegree one. In Figure 1.3, $T|\{a, b, d\} = T_1$.

Let tree $T' \in RP(X)$. We say that T *refines* T' (or *is a refinement of* T') if $\mathcal{C}(T') \subseteq \mathcal{C}(T)$. In Figure 1.3, T_1 is a refinement of T_2 as $\mathcal{C}(T_2) = \{\{a, b, d\}, \{a\}, \{b\}, \{d\}\}$ and $\mathcal{C}(T_1) = \mathcal{C}(T_2) \cup \{\{a, b\}\}$ so $\mathcal{C}(T_2) \subset \mathcal{C}(T_1)$.

Let $X'' \subseteq X$, and let $T'' \in RP(X'')$. We say that T *displays* T'' if $T|X''$ is a refinement of T''. In Figure 1.3, $X'' = \{a, b, d\}$ and $X = \{a, b, c, d\}$, and tree T_1 displays trees T_2 and T_3. Note that T_2 also displays T_3.

1.4.6 Compatibility

A set $\mathcal{P}$ of rooted phylogenetic trees (respectively, rooted binary phylogenetic trees) is *compatible* if there exists a tree $T \in RP(X)$ (respectively, $T \in RB(X)$) such that T displays each tree in the set $\mathcal{P}$. We then say that T *displays* $\mathcal{P}$. Note that T is

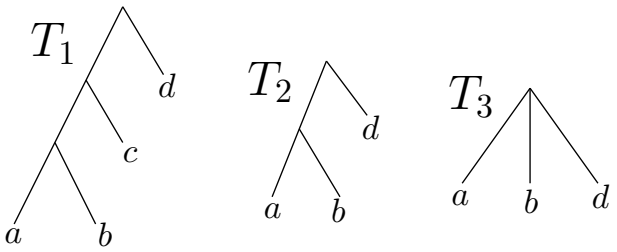

Figure 1.3: An example of a binary phylogenetic tree T_1 and two trees T_2 and T_3 that are displayed by T_1.

not necessarily in the set $\mathcal{P}$. Let $\langle\mathcal{P}\rangle$ (respectively $\langle\mathcal{P}\rangle_B$) denote the set of all rooted phylogenetic trees (respectively rooted binary phylogenetic trees) that display $\mathcal{P}$. Note that $\langle\mathcal{P}\rangle_B \subseteq \langle\mathcal{P}\rangle$.

1.4.7 Rooted triples

A *rooted triple* is a tree $T \in RB(X)$ where $|X| = 3$. A rooted triple with $X = \{a, b, c\}$ is denoted $ab|c$ if the path from a to b does not intersect the path from c to the root of T. Let $r(T)$ denote the set of all rooted triples displayed by T. Figure 1.4 shows an example of a set $\mathcal{R}$ of rooted triples and two trees $T, T' \in \langle\mathcal{R}\rangle_B$.

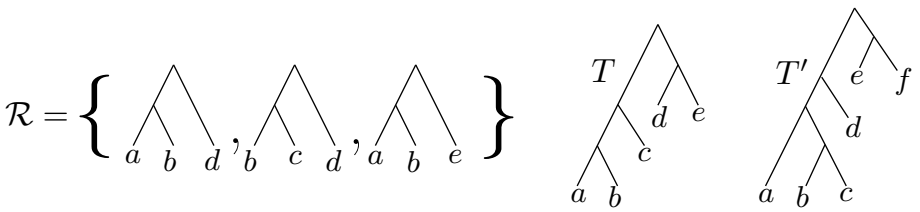

Figure 1.4: An example of a set $\mathcal{R}$ of rooted triples and two rooted binary phylogenetic trees T and T' that both display $\mathcal{R}$.

Note that we only need to consider a set of rooted triples $\mathcal{R}$ rather than a more general set of rooted phylogenetic trees $\mathcal{P}$, since the latter can be converted into the former so that the trees displaying $\mathcal{R}$ are exactly those which display $\mathcal{P}$. Let $\mathcal{P}'$ be a set of rooted phylogenetic trees such that each tree in $\mathcal{P}'$ has at least one internal arc, i.e. there are no

stars. Then $\langle \mathcal{P}' \rangle_B = \langle \mathcal{R}_{\mathcal{P}'} \rangle_B$, where $\mathcal{R}_{\mathcal{P}'}$ is the set of rooted triples such that, for each rooted triple $ab|c \in \mathcal{R}_{\mathcal{P}'}$, there is some tree in $\mathcal{P}'$ which displays $ab|c$. The case in which $\mathcal{P}$ contains at least one tree that is a star will be dealt with later.

1.4.8 Terraces

Consider a tree T with leaf set X and the subsets $X_1, \ldots, X_k$, where $X_i \subsetneq X$. Consider the set of trees $\mathcal{P} = \{T|X_1, \ldots, T|X_k\}$ (a set of trees on overlapping leaf sets). Let tree T' display $\mathcal{P}$. The set of all such trees T' is called a *terrace* of trees [9]. Note that this terrace contains the tree T. In the simplest case, where $X_i = X$ for all i, the terrace containing T is just T itself.

Relating this back to the biology, X is a set of species and X_G is the subset of X of species for which gene G is present. Let T be a tree that displays all of the evolutionary relationships given by this data. Then T is a fully-resolved (binary) tree with leaf set (set of species) X. Consider the set of induced subtrees $T|X_{G_1}, T|X_{G_2}, \ldots, T|X_{G_n}$. The set of trees displaying $T|X_{G_1}, T|X_{G_2}, \ldots, T|X_{G_n}$ is called a *terrace*. Each of the trees in this terrace, including T, display all of the evolutionary information given by the subsets X_G.

1.5 Operations on rooted phylogenetic trees

Consider tree $T \in RB(X)$ and let $e = (v, u)$ be an arc of T. The tree T' given by introducing a new vertex w into T, deleting arc e, and inserting arcs (v, w) and (w, u) into T is obtained from T by *subdividing e with w*. So the tree $T' = (V(T'), E(T'))$ where $V(T') = V(T) \cup \{w\}$ and $E(T') = (E(T) \setminus \{e\}) \cup \{(v, w), (w, u)\}$.

The following operation allows us to "prune" a subtree of a tree T and "regraft" it elsewhere in T.

rSPR (rooted subtree prune and regraft) operation: Let $T \in RB(X)$ and let $e = (v, u)$ be an arc of T. We say that $T' \in RB(X)$ is an *rSPR-neighbour* of T if T' can be obtained from T by the following procedure. Let t_u be the subtree of T rooted at u. Delete arc e, *pruning* the subtree t_u. Choose an arc $f = (w, z)$ of $T \setminus t_u$. Then, to *regraft* t_u:

If w is not the root of $T \setminus t_u$,

(i) subdivide f with a vertex x.

If w is the root of $T \setminus t_u$, either do (i) or

(ii) introduce a vertex x and insert the arc (x, w). Note that x is the root of the resulting tree.

Now insert the arc (x, u) into T, *regrafting* t_u, and, lastly, suppress v.

We have now obtained tree T', an rSPR-neighbour of T. We write $T \overset{\text{rSPR}}{\sim} T'$ and we say that this rSPR operation is performed *with respect to* t_u. Note that $\overset{\text{rSPR}}{\sim}$ is an equivalence relation. Figure 1.5 shows two examples of rSPR operations with respect to subtree t_u of T_1. The tree T_2 is obtained from T_1 by option (i) above, where $f = (v_3, v_4)$, and T_3 is obtained from T_1 by option (ii) above, where $f = (r, v_2)$.

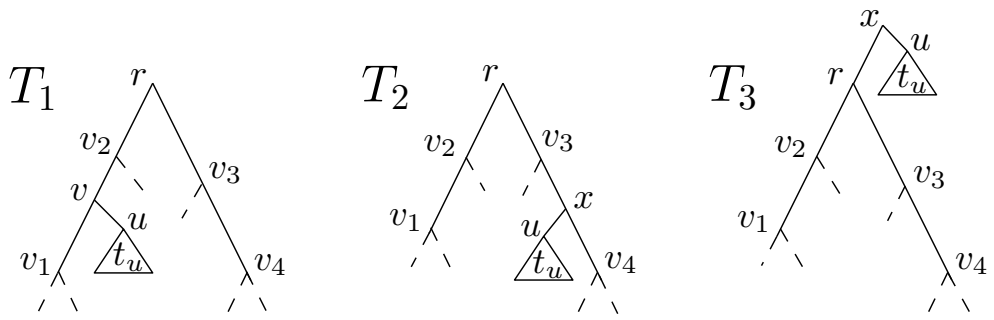

Figure 1.5: An example of a tree T_1 and two trees T_2 and T_3 each obtained from T_1 by performing an rSPR operation with respect to t_u.

An *NNI (nearest neighbour interchange) operation on a rooted tree* is a special case of an rSPR operation in which $f = \{w, z\}$ and w is adjacent to v ($w \neq u$). If a tree T' is

obtained from a tree T by an NNI operation, we say that T' is an NNI-neighbour of T and write $T \overset{\text{NNI}}{\sim} T'$. Note that $T \overset{\text{NNI}}{\sim} T$ and, if $T \overset{\text{NNI}}{\sim} T'$, then $T' \overset{\text{NNI}}{\sim} T$.

1.5.1 NNI- and rSPR-related trees

We define a *sequence of rSPR-related trees* to be a sequence of trees, say $(T_1, \ldots, T_n)$, for which $T_i \overset{\text{rSPR}}{\sim} T_{i+1}$ for all $1 \leq i < n$; that is, each tree in the sequence can be obtained from the previous tree by a single rSPR operation (not necessarily with respect to the subtree t_u). We refer to these rSPR operations as a *sequence of rSPR operations*. A sequence of rSPR operations from T_1 to T_n is called a *minimum sequence of rSPR operations* if it is the shortest sequence of rSPR operations that starts with the tree T_1 and produces the tree T_n. If each of the rSPR operations in a sequence is performed with respect to the subtree t_u, then we refer to this as a *sequence of rSPR operations with respect to* t_u. We define an analogous set of terms for NNI operations.

1.5.2 A note on vertex labels

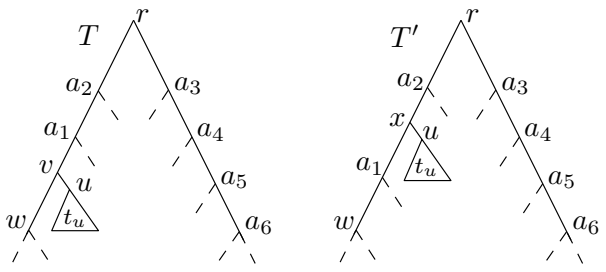

Figure 1.6: An example of the labeling of vertices before (T) and after (T') an NNI operation with respect to t_u.

When we perform an operation (such as a contraction or an NNI operation) on a tree T, existing vertices may be deleted or new vertices inserted to produce the tree T'. These new vertices will be labeled as required and all other vertices retain the same labels in

T' as they had in T. Figure 1.6 shows an example of this for an NNI operation. In this example, the subtree t_u rooted at u is pruned and regrafted. Vertex v of T is suppressed and new vertex x is inserted, so $V(T') = (V(T) \setminus \{v\}) \cup \{x\}$. All other vertex labels remain the same.

Chapter 2

A navigation system for tree space

2.1 Introduction

In this chapter, we explore a further combinatorial consequence of patchy taxon coverage: namely, for any terrace of trees, it is possible to move from any one tree on the terrace to any other tree on the terrace by performing a sequence of NNI operations, while always remaining on that terrace. This result follows from a theorem, first stated and proved in the PhD thesis of Magnus Bordewich [2], based on an inductive argument. The motivation for this chapter is to provide an explicit algorithm for constructing a sequence of NNI operations from any one tree T on the terrace to any other tree T' on the terrace. In our approach, the details of the scoring function s play no real role since a terrace is the set of binary trees displaying the set of rooted trees $\{T|X_1, \ldots, T|X_k\}$, where T is some tree on that terrace. Thus we deal simply with sets of rooted trees on overlapping leaf sets as our input.

The structure of this chapter is as follows. Later in this section, we summarize a result from [2] and state the main result of this chapter. In Section 2.2 we present and prove some preliminary results and then, in Section 2.3, we provide a proof of the main

result. This is followed, in Section 2.4, by an algorithm and an analysis of its complexity. We end with some brief concluding comments in Section 2.5.

For any two trees $T, T' \in RB(X)$, there is a sequence of trees $(T_1, T_2, \ldots, T_n)$ such that $T = T_1$, $T' = T_n$, and $T_i \overset{\text{NNI}}{\sim} T_{i+1}$ for all i $(1 \leq i < n)$ [7]. In this chapter, we consider two trees, T and T', that display a set of rooted triples $\mathcal{R}$, and find a sequence of trees satisfying the aforementioned conditions, along with the additional condition that each tree in the sequence displays $\mathcal{R}$.

The following result was stated and proved in the PhD Thesis of Bordewich [2].

Theorem 2.1.1. *Let $\mathcal{R}$ be a set of rooted triples. Suppose that $T, T' \in \langle \mathcal{R} \rangle_B$ and $\mathcal{L}(T) = \mathcal{L}(T')$. Then there is a sequence of trees $(T_1, T_2, \ldots, T_n)$ such that:*

1. *$T_1 = T$ and $T_n = T'$,*
2. *$T_i \overset{\text{NNI}}{\sim} T_{i+1}$ for $1 \leq i < n$, and*
3. *$T_i \in \langle \mathcal{R} \rangle_B$ for $1 \leq i \leq n$ (i.e. each T_i displays $\mathcal{R}$).*

We first note that Theorem 2.1.1 is equivalent to the following:

Theorem 2.1.2. *Let $T, T' \in \langle \mathcal{R} \rangle_B$, where $\mathcal{R} = r(T) \cap r(T')$ and $\mathcal{L}(T) = \mathcal{L}(T')$. Then there is a sequence of trees $(T_1, T_2, \ldots, T_n)$ such that Properties (1)–(3) of Theorem 2.1.1 hold.*

To see why these two theorems are equivalent, first assume that Theorem 2.1.1 holds. Let $\mathcal{R}'$ be the set of rooted triples in Theorem 2.1.1 (for clarity of notation). If we let $\mathcal{R}' = r(T) \cap r(T')$ in Theorem 2.1.1, we have Theorem 2.1.2. Now assume that Theorem 2.1.2 holds. Once again let $\mathcal{R}'$ be the set of rooted triples in Theorem 2.1.1 (for clarity of notation). Since $\mathcal{R}' \subseteq r(T) \cap r(T')$ and each T_i displays $r(T) \cap r(T')$, then T, T', and each T_i will also display $\mathcal{R}'$, so Theorem 2.1.1 holds.

Figure 2.1 shows an example to illustrate Theorem 2.1.2. In this example, $\mathcal{R} =$

$r(T) \cap r(T') = \{ac|d, ac|e, ac|f, de|a, de|c, ef|d\}$. It is easy to check that $T_1, \ldots, T_4$ all display $\mathcal{R}$, and each tree can be obtained from the previous tree by a single NNI operation.

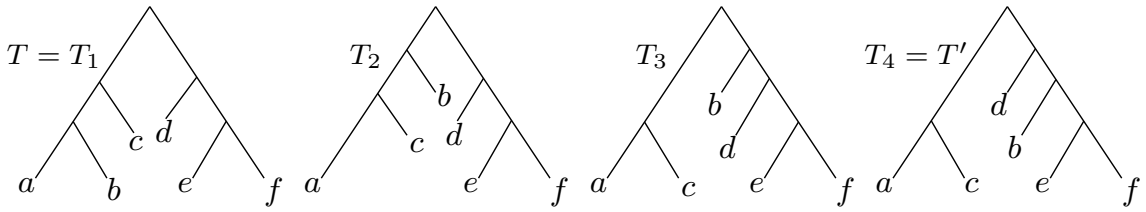

Figure 2.1: An example to illustrate Theorem 2.1.2 with $\mathcal{R} = \{ac|d, ac|e, ac|f, de|a, de|c, ef|d\}$.

We are only considering rooted trees in this chapter because there is no result directly analogous to Theorem 2.1.1 for unrooted trees and quartet trees (the unrooted analogue of rooted triples). To see this, consider the following counterexample. The two unrooted trees in Figure 2.2 both display the quartet trees 12|45, 34|16, and 56|23. However, it is straightforward to check that the two trees in Figure 2.2 are the only two trees which display these quartet trees and that they are not one (unrooted) NNI operation apart. See [10] for further information and definitions.

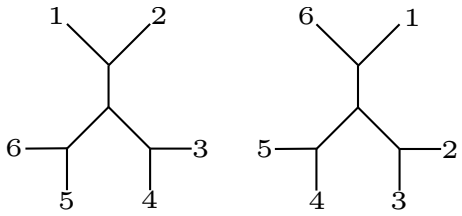

Figure 2.2: The two unrooted trees that display the quartets 12|45, 34|16, and 56|23.

The proof of Theorem 2.1.1 in [2] is based on an inductive argument. We present an alternative proof that provides an explicit procedure for obtaining the sequence of trees $(T_1, \ldots, T_n)$.

The main result of this chapter is the following theorem.

Theorem 2.1.3. *Given $T, T' \in \langle \mathcal{R} \rangle_B$, where $\mathcal{R} = r(T) \cap r(T')$ and $\mathcal{L}(T) = \mathcal{L}(T')$, one can construct (in polynomial time) a sequence of trees $(T_1, T_2, \ldots, T_n)$ such that:*

1. *$T_1 = T$ and $T_n = T'$,*
2. *$T_i \overset{\text{NNI}}{\sim} T_{i+1}$ for $1 \leq i < n$, and*
3. *$T_i \in \langle \mathcal{R} \rangle_B$ for $1 \leq i \leq n$ (i.e. each T_i displays $\mathcal{R}$).*

This takes into consideration the case in which $\mathcal{L}(\mathcal{R}) \subset \mathcal{L}(T) = \mathcal{L}(T')$, that is, there are leaves in T and T' that are not in any rooted triple in $\mathcal{R}$. We call these leaves (the leaves in $\mathcal{L}(T) \setminus \mathcal{L}(\mathcal{R})$) *free leaves*.

Let $\mathcal{P}''$ be a set of rooted phylogenetic trees such that at least one of these trees is a star. Let $\mathcal{P}_S$ be the subset of $\mathcal{P}$ consisting of all trees in $\mathcal{P}$ which are stars and let $\mathcal{P}' = \mathcal{P}'' \setminus \mathcal{P}_S$. All leaves which are in $\mathcal{L}(\mathcal{P}_S) \setminus \mathcal{L}(\mathcal{P}')$ (i.e. all leaves that are in a tree in $\mathcal{P}_S$ and are not in any tree in $\mathcal{P}'$) are free leaves. So $\langle P'' \rangle_B = \{T \in \langle \mathcal{R}_{P'} \rangle_B : \mathcal{L}(P'') \subseteq \mathcal{L}(T)\}$ where $\langle \mathcal{R}_{P'} \rangle_B$ is defined as in section 2.2.

Before we prove Theorem 2.1.3, we need some preliminary results.

2.2 Tree Manipulations

Lemma 2.2.1. *Let $\mathcal{R}$ be a set of rooted triples and let $L = \mathcal{L}(\mathcal{R})$. Suppose that $T, T' \in \langle \mathcal{R} \rangle_B$ and $\mathcal{L}(T) = \mathcal{L}(T')$. For $T^* \in \{T, T'\}$, it is possible to construct, in polynomial time, a sequence of NNI-related trees from T^* to a tree $\widetilde{T^*}$ with subtree $T^*|L$ such that each tree in the sequence displays $\mathcal{R}$ and $\widetilde{T} \setminus (T|L)$ is equivalent to $\widetilde{T'} \setminus (T'|L)$.*

Informally, this means that we can disregard the free leaves, transform $T|L$ into $T'|L$, and then reinstate the free leaves in T'. As these free leaves do not appear in any of the rooted triples in $\mathcal{R}$, this process will not affect whether a particular tree displays $\mathcal{R}$.

Proof of Lemma 2.2.1. First note that T and T' have the same leaf set, and let $\mathcal{L}(T) = \mathcal{L}(T') = \{x_1, \ldots, x_m\}$ such that $x_1, \ldots, x_n$ (for some $n \leq m$) are the free leaves. The following steps describe NNI operations which give a sequence of NNI-related trees from T to a tree $\widetilde{T}$ that has subtree $T|L$. An example of this can be seen in Figure 2.3.

(a) Consider $\mathcal{L}(T)\backslash L = \{x_1, \ldots, x_n\}$, the free leaves of T. Let $U_0 = T$ and let $S = (U_0)$.

(b) For $j = 1, \ldots, n$:

  (i) If $j \neq 1$, let v_{j-1} be the root of the subtree $T|(\mathcal{L}(T)\backslash\{x_1, x_2, \ldots, x_{j-1}\})$ of U_{j-1}. Otherwise (if $j = 1$), let v_{j-1} be the root of U_0.

  (ii) Consider U_{j-1}. If x_j is a child of v_{j-1}, let $U_j = U_{j-1}$. Otherwise,

    (1) Perform a minimum sequence of NNI operations with respect to x_j that results in a tree in which x_j is the grandchild of v_{j-1}. Append the sequence of trees (each of which is the result of one NNI operation in the sequence) to S.

    (2) If $j \neq 1$, perform the following NNI operation: prune x_j, subdivide the arc between v_{j-1} and its parent with a vertex w, insert arc (w, x_j), and suppress the vertex with outdegree one.

    Otherwise (if $j = 1$), perform the following NNI operation: prune x_1, introduce a vertex r', insert arc (r', v_0) and arc (r', x_1), and suppress the vertex with outdegree one.

    (3) Call the resulting tree U_j and append U_j to the sequence S.

(c) Let $\widetilde{T} = U_n$, the last tree of the sequence S. The tree $\widetilde{T}$ has subtree $T|L$, as required.

The trees in S will only differ on rooted triples that contain some x_j (for $1 \leq j \leq n$), but by our assumption x_j is not in any rooted triple. Therefore S is a sequence of NNI-related trees from T to $\widetilde{T}$ for which each tree in the sequence displays $\mathcal{R}$.

Recall that $\mathcal{L}(T) = \mathcal{L}(T') = \{x_1, \ldots, x_m\}$, where $x_1, \ldots, x_n$ (for some $n \leq m$) are

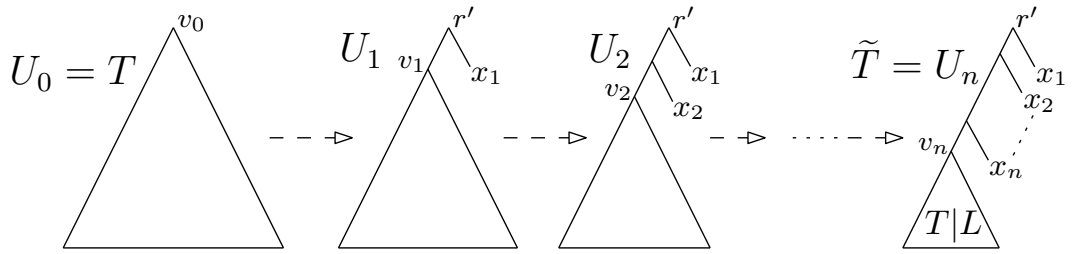

Figure 2.3: An example of the process in the proof of Lemma 2.2.1, where $x_1, \ldots, x_n$ are the free leaves.

the free leaves. Repeating steps (a) through (c) for T', we obtain a sequence of trees S', where the first tree in the sequence is T' and the last tree is $\widetilde{T'}$ which has subtree $T'|L$. Now S' is a sequence of NNI-related trees from T' to $\widetilde{T'}$ for which each tree in the sequence displays $\mathcal{R}$. Since step (b) is acting on the leaf sets in T and T' in the same order, $\widetilde{T} \setminus (T|L)$ is equivalent to $\widetilde{T'} \setminus (T'|L)$, as required. □

The above result simplifies our analysis so that we can always consider $T|L$ instead of T, where L is the set of leaves that are not free leaves. We have a sequence of trees from T to $\widetilde{T}$ and a sequence of trees from T' to $\widetilde{T'}$ (which can be reversed to give a sequence of trees from $\widetilde{T'}$ to T' as NNI operations are invertible). We now need a sequence of trees from $\widetilde{T}$ to $\widetilde{T'}$. We find a sequence of NNI operations which transforms $T|L$ into $T'|L$ and apply these NNI operations to $\widetilde{T}$, transforming the subtree $T|L$ into the subtree $T'|L$ and giving the tree $\widetilde{T'}$.

The following result further simplifies our analysis.

Lemma 2.2.2. *Suppose that $\mathcal{R}$ is a set of rooted triples and $T, T' \in \langle \mathcal{R} \rangle_B$, with $\mathcal{L}(T) = \mathcal{L}(T') = \mathcal{L}(\mathcal{R})$. Suppose that T' is obtained from T by one rooted subtree prune and regraft operation. Then there is a sequence of trees $(T_0, T_1, \ldots, T_n)$ such that:*

1. *$T_0 = T$ and $T_n = T'$,*
2. *$T_i \overset{\text{NNI}}{\sim} T_{i+1}$ for $0 \leq i < n$, and*
3. *$T_i \in \langle \mathcal{R} \rangle_B$ for $0 \leq i \leq n$ (i.e. each T_i displays $\mathcal{R}$).*

Proof. Consider the trees T and T'. By our assumption, $T \overset{\text{rSPR}}{\sim} T'$. If $T = T'$ or $T \overset{\text{NNI}}{\sim} T'$, we are done. So suppose that this is not the case. Let the rSPR operation be with respect to some subtree t_u of T. Then T and T' both have subtree t_u. Let v_0 be the parent of u in T (the vertex v_0 always exists because t_u is a proper subtree of T). Similarly, let v' be the parent of u in T'. When defining the neighbours of v' in T', there are two cases to consider. The first case is that v' has three neighbours, u, v_n, and v_{n+1}. Note that v_n and v_{n+1} are both in T, so there is a path $v_0, \ldots, v_n, v_{n+1}$ in T. The second case is that v' has two neighbours, u and v_n. In this case, v' is the root of T'. As v_n is in T, there is a path $v_0, \ldots, v_n$ in T.

We now describe the minimum sequence of NNI operations with respect to t_u that is performed to obtain T' from T.

a) In the first NNI operation, delete the arc (v_0, u) from T, subdivide the arc $\{v_1, v_2\}$ with a vertex w_1, insert arc (w_1, u), and then suppress v_0. Call the resulting tree T_1.

b) For $i = 2, \ldots, n-1$, perform the following (i^{th}) NNI operation: Delete the arc (w_{i-1}, u) from T_{i-1}, subdivide the arc $\{v_i, v_{i+1}\}$ with a vertex w_i, insert arc (w_i, u), and then suppress the vertex w_{i-1}. Call the resulting tree T_i.

c) The last (n^{th}) NNI operation is as follows. If v_n is the root of T_{n-1} and v' is the root of T', delete the arc (w_{n-1}, u) from T_{n-1}, introduce a vertex w_n, and insert arc (w_n, v_n). (Note that, in this case, w_n is the root of the resulting tree.) Otherwise, delete the arc (w_{n-1}, u) from T_{n-1}, and subdivide the arc $\{v_n, v_{n+1}\}$ with a vertex w_n.

d) Insert arc (w_n, u) and then suppress the vertex w_{n-1}. The resulting tree is T', where $w_n = v'$.

We now have a sequence of NNI-related trees from T to T'. Next, we prove that for each i ($0 \leq i \leq n$), $T_i \in \langle \mathcal{R} \rangle_B$. Let r_i denote the root of T_i for each i. We proceed using induction on i and note that $T_0 = T \in \langle \mathcal{R} \rangle_B$ by assumption.

Assume that $T_i \in \langle \mathcal{R} \rangle_B$ for some $i < n$. To see that $T_{i+1} \in \langle \mathcal{R} \rangle_B$, consider a rooted triple $ab|c \in \mathcal{R}$. Recall that a tree displays $ab|c$ if and only if the path from a to b does not intersect the path from c to the root of the tree. This is the case in T_i and we show that it is also the case in T_{i+1}. There are six possible cases to be considered with respect to the possible locations of a, b, and c in T_i. Let v_{ab} be the most recent common ancestor of a and b in T_i and let v_{abc} be the most recent common ancestor of a, b, and c in T_i. Recall that all of the NNI operations are with respect to t_u.

Case 1: $a, b, c \in t_u$. In this case, in T_i, the (a-b)-path (the path from a to b) does not intersect the (c-r_i)-path, so, in t_u, the (a-b)-path does not intersect the (c-u)-path. The tree T_{i+1} contains the subtree t_u, so, as before, the (a-b)-path does not intersect the (c-u)-path and hence, in T_{i+1}, the (a-b)-path does not intersect the (c-r_{i+1})-path. Therefore, T_{i+1} displays $ab|c$.

Case 2: $a, b, c \notin t_u$. In this case, in T_i, no element of the (a-b)-path, or the (c-r_i)-path, is in t_u. Let q be the path between v_{ab} and v_{abc} in T_i and let p be a path in T_i with one endpoint in the (a-b)-path and the other endpoint in the (c-r_i)-path. Then p must contain q, so $|p| \geq |q|$. Therefore, the (a-b)- and (c-r_i)-paths intersect if and only if q has length zero, in which case $v_{ab} = v_{abc}$. Assume that T_{i+1} does not display $ab|c$, so the (a-b)- and (c-r_{i+1})-paths intersect and so, by the same argument, $v_{ab} = v_{abc}$. Then, in T_{i+1}, v_{abc} has three children and so T_{i+1} is not a binary tree. This is a contradiction because, by definition, an NNI operation on a binary tree produces another binary tree. Therefore T_{i+1} displays $ab|c$.

Case 3: $a, b \in t_u$ and $c \notin t_u$. In this case, in T_i, the (a-b)-path is contained entirely in t_u and the (c-r_i)-path contains no arcs or vertices in t_u, so these two paths do not intersect. Performing an NNI operation on T_i to obtain T_{i+1} will not affect this, so the same property will hold in T_{i+1}. In T_{i+1}, the (a-b)-path will be contained entirely in t_u and the (c-r_{i+1})-path will not contain any of these vertices or arcs, so these two paths do

not intersect. Therefore, T_{i+1} displays $ab|c$.

Case 4: $b, c \in t_u$ and $a \notin t_u$ (which is analogous to the case $a, c \in t_u$, $b \notin t_u$). In this case, in T_i, the (a-b)-path and the (c-r_i)-path both contain u, so these two paths intersect. Therefore, T_i does not display $ab|c$, which is a contradiction. It is thus not possible that $b, c \in t_u$ and $a \notin t_u$.

Case 5: $a \in t_u$ and $b, c \notin t_u$ (which is analogous to the case $b \in t_u$, $a, c \notin t_u$). Let v'_{ab} be the most recent common ancestor of a and b in T_{i+1} and let v'_{abc} be the most recent common ancestor of a, b, and c in T_{i+1}. First assume that, in T_{i+1}, v'_{ab} is a proper descendant of v'_{abc}. Then T_{i+1} displays $ab|c$. Now assume that this is not the case; that is, in T_{i+1}, v'_{ab} is not a proper descendant of v'_{abc}. Then the (a-b)-path and the (c-r_{i+1})-path intersect. Furthermore, in T_{i+1}, both the (a-b)-path and the (c-r_{i+1})-path contain v_{abc}. Now for each T_k, where $k > i$, both the (a-b)-path and the (c-r_k)-path contain v_{abc}. Hence, in $T_n = T'$, both the (a-b)-path and the (c-r_n)-path contain v_{abc}, so these two paths intersect, a contradiction of the assumption that $T' \in \langle \mathcal{R} \rangle_B$. Therefore, T_{i+1} displays $ab|c$.

Case 6: $c \in t_u$ and $a, b \notin t_u$. Let v'_{ab} and v'_{abc} be defined as in case 5. First assume that, in T_{i+1}, v'_{abc} is a proper ancestor of v'_{ab}. Then T_{i+1} displays $ab|c$. Now assume that this is not the case; that is, in T_{i+1}, v'_{abc} is not a proper ancestor of v'_{ab}. Then u (and therefore c) is a descendant of v_{ab}. So the (a-b)-path and the (c-r_{i+1})-path intersect. Furthermore, in T_{i+1}, both the (a-b)-path and the (c-r_{i+1})-path contain v_{ab}. Now for each T_k, where $k > i$, both the (a-b)-path and the (c-r_k)-path contain v_{ab}. Hence, in $T_n = T'$, both the (a-b)-path and the (c-r_n)-path contain v_{ab}, so these two paths intersect, a contradiction of the assumption that $T' \in \langle \mathcal{R} \rangle_B$. Therefore, T_{i+1} displays $ab|c$.

In each of the six cases, either the scenario is not possible (case 4) or T_{i+1} displays $ab|c$. Since $ab|c$ was an arbitrary rooted triple in $\mathcal{R}$, we can extend this result to all of the rooted triples in $\mathcal{R}$, so $T_{i+1} \in \langle \mathcal{R} \rangle_B$. Therefore, by induction, $T_j \in \langle \mathcal{R} \rangle_B$ for all $0 \leq j \leq n$. $\square$

Lemma 2.2.2 allows us to convert a sequence of rSPR operations into an equivalent sequence of NNI operations.

Lemma 2.2.3. *Let $\mathcal{R}$ be a set of rooted triples and suppose that $T \in \langle\mathcal{R}\rangle_B$ and $ab|c \in \mathcal{R}$. Then $a, b \in C$ for some maximal proper cluster C of T.*

Proof. Let r be the root of T and let C and $\bar{C}$ be the maximal proper clusters of T. For $ab|c \in \mathcal{R}$ suppose, without loss of generality, that $a \in C$ and $b \in \bar{C}$. The most recent common ancestor of a and b is then r, so the path from a to b will contain r and so this path will intersect the path from c to r. Therefore $ab|c \notin \mathcal{R}$, a contradiction. Hence if $ab|c \in \mathcal{R}$, then either $a, b \in C$ or $a, b \in \bar{C}$. □

Lemma 2.2.4. *Let $\mathcal{R}$ be a set of rooted triples and let $T \in \langle\mathcal{R}\rangle_B$, where $\mathcal{L}(T) = \mathcal{L}(\mathcal{R})$. Let C and $\bar{C}$ be non-empty subsets of $\mathcal{L}(T)$ for which $C \cup \bar{C} = \mathcal{L}(T)$, $C \cap \bar{C} = \emptyset$, and C and $\bar{C}$ are maximal proper clusters of at least one tree in $\langle\mathcal{R}\rangle_B$. Consider $T|C$ and $T|\bar{C}$, with roots r_C and $r_{\bar{C}}$ respectively. Let $\hat{T} \in \langle\mathcal{R}\rangle_B$ be the tree rooted at $\hat{r}$ composed of exactly the subtrees $T|C$ and $T|\bar{C}$ and the arcs $(\hat{r}, r_C)$ and $(\hat{r}, r_{\bar{C}})$. Then there is a sequence of trees $(T_1, T_2, \ldots, T_n)$ such that:*

1. *$T_1 = T$ and $T_n = \hat{T}$,*
2. *$T_i|C = T|C$ and $T_i|\bar{C} = T|\bar{C}$ for $1 \leq i \leq n$,*
3. *$T_i \overset{\text{rSPR}}{\sim} T_{i+1}$ for $1 \leq i < n$, and*
4. *$T_i \in \langle\mathcal{R}\rangle_B$ for $1 \leq i \leq n$.*

Proof. We first show how to obtain, from T, a tree T' with subtree $T'|C = T|C$. For any i $(1 \leq i \leq n)$, the tree T_i contains one or more maximal subtrees whose leaf sets are subsets of C. If T_i contains only one such subtree, then that subtree must be $T|C$ and so $T' = T_i$. Now consider the case in which T_i contains two or more such subtrees. Let t_v be a minimal subtree of T_i containing exactly two maximal subtrees, t_x and t_y, whose leaf sets are subsets of C. (Note that t_v may contain leaves in $\bar{C}$.) We assume $i \leq n - 2$

and apply the following two rSPR operations, starting at T_i, to produce a tree in which there is a subtree with leaf set $\mathcal{L}(t_x) \cup \mathcal{L}(t_y)$.

(a) Consider v, the most recent common ancestor of x and y. If v is not the root of T_i, let v' be the parent of v, and subdivide the arc (v', v) with a vertex u. Otherwise (if v is the root of T_i), introduce a vertex u and insert arc (u, v).

(b) Prune t_x, insert arc (u, x), regrafting t_x, and suppress the vertex with outdegree one. Call the resulting tree T_{i+1}. Note that $T_{i+1}|C = T|C$ (and similarly for $\bar{C}$).

(c) Subdivide the arc (u, x) with a vertex u'. Prune t_y, insert arc (u', y), regrafting t_y, and suppress the vertex with outdegree one. Call the resulting tree T_{i+2}. We now have a subtree $t_{u'}$ of T_{i+2} containing exactly the leaves in $\mathcal{L}(t_x) \cup \mathcal{L}(t_y)$. Note that $T_{i+2}|C = T|C$ (and similarly for $\bar{C}$).

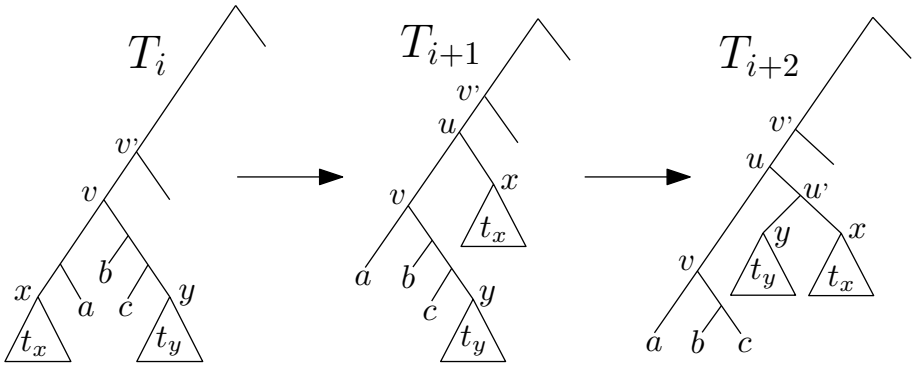

Figure 2.4: An example of steps (a) to (c) in the proof of Lemma 2.2.4

We now prove by induction on $|C|$ that we can repeatedly perform the above sequence of rSPR operations to obtain, from T, a tree T' with subtree $T'|C = T|C$. In the case $|C| = 1$, $T' = T$ and so the result holds. Consider the case $|C| = 2$. Let $C = \{x, y\}$ and $\bar{C} = \mathcal{L}(T) \setminus \{x, y\}$. Starting with T, apply the above steps (a) through (c), where t_x and t_y are each a single leaf (x and y respectively), to obtain a tree $T_3 = T'$ with a subtree containing exactly the leaves in $\mathcal{L}(t_x) \cup \mathcal{L}(t_y) = \{x\} \cup \{y\} = C$, as required.

Assume that, for tree T with $2 \leq |C| \leq k$ (for some $k \geq 2$), we can perform a sequence of rSPR operations to obtain a tree T' with subtree $T'|C = T|C$. Now consider a tree T

where $|C| = k + 1$, and let t_v be the minimal subtree of T such that $C \subseteq \mathcal{L}(t_v)$. Then t_v has two maximal proper subtrees, say t' and t'', each of which must contain at least one leaf in C, and so each contains no more than k leaves in C. By the induction assumption, we obtain a tree T^* with subtree t_v^* containing subtrees $t'|C$ and $t''|C$. Now t_v^* is the minimal subtree of T^* containing $t'|C$ and $t''|C$, so we apply steps (a) through (c) above, starting with the tree T^*, to obtain a tree T' with a subtree containing exactly the leaves in $\mathcal{L}(t'|C) \cup \mathcal{L}(t''|C) = C$, as required.

We now prove that neither of the rSPR operations in steps (a) through (c) violate any rooted triples in $\mathcal{R}$. Note that t_x or t_y (or both) may consist of only a single leaf. Consider the rSPR operation with respect to t_x (given in steps (a) and (b)) used to obtain T_{i+1} from T_i $(1 \leq i \leq n-2)$. Assume that T_i displays $\mathcal{R}$, but suppose, without loss of generality, that T_{i+1} does not display rooted triple $ab|c \in \mathcal{R}$. Consider T_i and $a, b, c \in \mathcal{L}(T_i)$, and the possible locations of the leaves a, b, and c in T_i with respect to the subtrees t_x, t_y, and t_v. The scenarios in which $a, b \in \mathcal{L}(t_x)$ or $a, b \notin \mathcal{L}(t_x)$ result in contradictions. So, without loss of generality, consider the scenarios in which $a \in \mathcal{L}(t_x)$ and $b \notin \mathcal{L}(t_x)$. The cases in which $c \notin \mathcal{L}(t_v)$ or $b \notin \mathcal{L}(t_v)$ also result in contradictions, so assume that $b, c \in \mathcal{L}(t_v)$. If $b \in \mathcal{L}(t_y)$, then $c \notin \mathcal{L}(t_v)$, which is a contradiction, hence $b \notin \mathcal{L}(t_y)$. However, since $b \notin \mathcal{L}(t_x)$ and $b \notin \mathcal{L}(t_y)$, $b \notin C$, which is a contradiction of Lemma 2.2.3 because C is a maximal proper cluster of some tree in $\langle \mathcal{R} \rangle_B$. This concludes the case analysis. All possibilities result in contradictions, hence T_{i+1} displays $ab|c$. Now consider the rSPR operation with respect to t_y (given in step (c)) used to obtain T_{i+2} from T_{i+1}. Assume that T_{i+1} displays $\mathcal{R}$, but suppose, without loss of generality, that T_{i+2} does not display rooted triple $ab|c \in \mathcal{R}$. Consider T_{i+1} and $a, b, c \in \mathcal{L}(T_{i+1})$, and the possible locations of the leaves a, b, and c in T_{i+1} with respect to the subtrees t_x, t_y, and t_u. Similar reasoning, replacing t_v with t_u (recalling u is the parent of v in T_{i+1}) and swapping t_x and t_y, again leads to contradictions. Hence, T_{i+2} displays $ab|c$.

We now show how to obtain $\hat{T}$ from T'. In T', let w be the root of subtree $T|C$, and let x be the parent of w (x always exists as w is not the root of T', otherwise $|\bar{C}| = 0$). If x is the root of T', then $T|C$ is a maximal proper subtree of T' and so $\hat{T} = T'$. Otherwise, the following rSPR operation is performed to obtain $\hat{T}$ from T'. Let r' be the root of T'. Prune subtree $T|C$, introduce a vertex $\hat{r}$, insert arc $(\hat{r}, r')$ and arc $(\hat{r}, w)$, regrafting $T|C$, and suppress the vertex with outdegree one. We have now obtained a tree $\hat{T}$ (with root $\hat{r}$) with maximal proper subtrees $T|C$ and $T|\bar{C}$, as required.

We now prove that this rSPR operation does not violate any rooted triples in $\mathcal{R}$. Assume that T' displays $\mathcal{R}$ but suppose, without loss of generality, that $\hat{T}$ does not display rooted triple $ab|c \in \mathcal{R}$. Consider T' and $a, b, c \in \mathcal{L}(T')$, and the possible locations of the leaves a, b, and c in T' with respect to the subtree $T|C$. The scenarios in which $a, b \in \mathcal{L}(T|C)$ or $a, b \notin \mathcal{L}(T|C)$ result in contradictions. So, without loss of generality, consider the scenario in which $a \in \mathcal{L}(T|C)$ and $b \notin \mathcal{L}(T|C)$. Then $a \in C$ and $b \notin C$, which is a contradiction of Lemma 2.2.3. Hence $\hat{T}$ displays $ab|c$.

We have established that each of $(T_2, T_3 \ldots, T_n = \hat{T})$ display $\mathcal{R}$. Furthermore, since $T = T_1$ displays $\mathcal{R}$ by definition, we have shown that T_i displays $\mathcal{R}$ for all $1 \leq i \leq n$. $\square$

The process described in Lemma 2.2.4 will be referred to as *disentangling* C from T.

We now have all the necessary preliminary results, so we return to the proof of Theorem 2.1.3.

2.3 Proof of Theorem 2.1.3

Proof of Theorem 2.1.3. We first prove the special case in which $\mathcal{L}(T) = \mathcal{L}(T') = \mathcal{L}(\mathcal{R})$ (i.e. there are no free leaves). We prove that we can obtain a sequence of NNI-related trees from T to T' for which each tree in the sequence displays $\mathcal{R}$. We use strong induction

on $m = |\mathcal{L}(T)|$.

Consider the case $m = 2$. Then the two children of the root of T' are leaves, and since $\mathcal{L}(T) = \mathcal{L}(T')$, then $T = T'$ and so the result holds.

Assume that the result holds for trees T and T' with at most $k-1$ leaves. Now consider two trees $T, T' \in \langle\mathcal{R}\rangle_B$ with $|\mathcal{L}(T)| = |\mathcal{L}(T')| = |\mathcal{L}(\mathcal{R})| = k$. Let C and $\bar{C}$ be the maximal proper clusters of T'. Note that $\mathcal{L}(T) = C \cup \bar{C}$. Consider T and apply Corollary 2.2.4 to disentangle C from T, giving a sequence of rSPR-related trees from T to a tree T_i $(1 < i \leq n)$ such that each tree displays $\mathcal{R}$, and the tree T_i has maximal proper subtrees $T|C$ and $T|\bar{C}$. Applying Lemma 2.2.2 to this sequence of trees, we obtain a sequence S of NNI-related trees from T to T_i for which each tree in the sequence displays $\mathcal{R}$.

The tree T' has maximal proper subtrees $T'|C$ and $T'|\bar{C}$. Let $\mathcal{R}_C = \{ab|c \in \mathcal{R} : a, b, c \in C\}$ (i.e. the set of rooted triples for which the leaves are all in C). Define $\mathcal{R}_{\bar{C}}$ similarly. Now note that $|\mathcal{L}(T|C)| = |\mathcal{L}(T'|C)| < k$ so, by the induction assumption, there is a sequence of NNI-related trees $(T|C, \ldots, T'|C)$ such that each tree in the sequence displays $\mathcal{R}_C$. Since $|\mathcal{L}(T|\bar{C})| = |\mathcal{L}(T'|\bar{C})| < k$, the same applies to $\bar{C}$.

Consider the sequence of NNI operations above that create the sequence $(T|C, \ldots, T'|C)$. Starting with tree T_i with subtree $T|C$, perform this sequence of NNI operations to obtain a tree T_j $(1 < i \leq j \leq n)$ with subtree $T'|C$, where the rest of the tree remains unchanged (that is, $T_i \setminus T|C$ is equivalent to $T_j \setminus T'|C$). Since each tree in the sequence $(T_i, \ldots, T_j)$ displays $\mathcal{R}_C$, and $T_i \setminus T|C$ is equivalent to $T_j \setminus T'|C$, each tree this sequence displays $\mathcal{R}$. Repeating this process for $\bar{C}$ (with set of rooted triples $\mathcal{R}_{\bar{C}}$), starting with the tree T_j, gives the tree T' with maximal proper subtrees $T'|C$ and $T'|\bar{C}$, as required. We now have a sequence of NNI-related trees from T_i to T' such that each tree in the sequence displays $\mathcal{R}$.

Combining this sequence of trees with the earlier sequence S, we obtain a sequence of

NNI-related trees $(T, \ldots, T')$ such that each tree in the sequence displays $\mathcal{R}$, as required.

We now turn to the general case in which $\mathcal{L}(\mathcal{R}) \subseteq \mathcal{L}(T) = \mathcal{L}(T')$. By Lemma 2.2.1, there is a sequence of NNI-related trees from T to a tree $\widetilde{T}$ with subtree $T|\mathcal{L}(\mathcal{R})$, and there is a sequence of trees from T' to a tree $\widetilde{T}'$ with subtree $T'|\mathcal{L}(\mathcal{R})$, such that each tree in these sequences displays $\mathcal{R}$ and $\widetilde{T} \setminus (T|\mathcal{L}(\mathcal{R}))$ is equivalent to $\widetilde{T}' \setminus (T'|\mathcal{L}(\mathcal{R}))$.

Next, we need a sequence of NNI-related trees from $\widetilde{T}$ to $\widetilde{T}'$ such that each tree in the sequence displays $\mathcal{R}$. By the special case (proved above), there is a sequence of NNI-related trees from $T|\mathcal{L}(\mathcal{R})$ to $T'|\mathcal{L}(\mathcal{R})$ such that each tree in the sequence displays $\mathcal{R}$. Performing the corresponding sequence of NNI operations, starting with the tree $\widetilde{T}$, transforms the subtree $T|\mathcal{L}(\mathcal{R})$ into $T'|\mathcal{L}(\mathcal{R})$, giving the tree $\widetilde{T}'$. We now have a sequence of NNI-related trees from $\widetilde{T}$ to $\widetilde{T}'$ such that each tree displays $\mathcal{R}$, as required. □

2.4 Algorithm and Complexity

2.4.1 Algorithm

In this section, we take the steps from the proofs of Lemmas 3.3, 3.4, and 3.6 and create an algorithm which takes two trees $T, T' \in \langle\mathcal{R}\rangle_B$ as input and produces a sequence of NNI-related trees from T to T', such that each tree in the sequence displays $\mathcal{R}$. The algorithm consists of the following five procedures.

The first procedure, FreeLeaves, takes a tree T and a set of rooted triples $\mathcal{R}$ as input and uses steps (a) through (c) in the proof of Lemma 2.2.1 to create a sequence of NNI-related trees ending with a tree $\widetilde{T}$which has a subtree containing exactly the leaves in $\mathcal{R}$ (as illustrated in Figure 2.3). It returns the sequence of NNI-related trees.

Procedure FreeLeaves:

Input: A set $\mathcal{R}$ of rooted triples; a tree $T \in \langle\mathcal{R}\rangle_B$.

Output: A sequence F of NNI-related trees from T to a tree with subtree $T|\mathcal{L}(\mathcal{R})$.

1. Label the free leaves $\mathcal{L}(T) \setminus \mathcal{L}(\mathcal{R}) = \{x_1, x_2, \ldots x_n\}$.
2. Apply steps (a) through (c) in the proof of Lemma 2.2.1 where T is the starting tree and the free leaves are labeled as in step 1 to produce F.
3. Return F.

The second procedure, ToNNI, uses steps (a) through (e) in the proof of Lemma 2.2.2 to produce a sequence of NNI-related trees from a sequence of rSPR-related trees.

Procedure ToNNI:

Input: A sequence $S = (S_1, \ldots, S_k)$ of rSPR-related trees; a set $\mathcal{R}$ of rooted triples displayed by each tree in S.

Output: A sequence $\widetilde{S}$ of NNI-related trees.

1. Let $\widetilde{S} = (\ )$.
2. For $i = 1, \ldots, k-1$:
   (i) Apply steps (a) through (e) from the proof of Lemma 2.2.2 (given in the appendix) to the trees S_i and S_{i+1} to obtain a sequence U_i of NNI-related trees, where S_i is the first tree in the sequence U_i and S_{i+1} is the last tree in the sequence U_i. Each tree in the sequence U_i displays $\mathcal{R}$ by Lemma 2.2.2.
   (ii) If $i \neq k$, remove the last tree (S_{i+1}) from U_i (so that it will not be repeated) and append U_i to $\widetilde{S}$.
3. Return $\widetilde{S}$.

The third procedure, Disentangle, takes a tree $T_{current}$ as input and uses steps (a) through (c) in the proof of Lemma 2.2.4 to disentangle a given leaf set from a specified subtree of $T_{current}$, and returns the resulting sequence of rSPR-related trees.

Procedure Disentangle:

Input: A set $\mathcal{R}$ of rooted triples; a tree $T_{current} \in \langle \mathcal{R} \rangle_B$; the root w of the subtree of $T_{current}$ to disentangle; the leaf set C to disentangle.

Output: A sequence S of rSPR-related trees.

1. Let t_w be the subtree of $T_{current}$ rooted at w. Let $S = (\ )$ and let $T_{working} = T_{current}$.
2. While $T_{working}$ does not have subtree $T_{current}|C = T_{working}|C$:

   Let t_w be the subtree of $T_{working}$ rooted at w. There is a minimal subtree of t_w containing exactly two maximal subtrees t_x and t_y, whose leaf sets are subsets of C (note that t_x or t_y may contain only a single leaf). Perform steps (a) through (c) in the proof of Lemma 2.2.4, starting with the tree $T_{working}$, to obtain two trees T^* and T^{**}, where T^{**} has a subtree with leaf set $\mathcal{L}(t_x) \cup \mathcal{L}(t_y)$. Append T^* and T^{**} to S. Let $T_{working} = T^{**}$.
3. The tree $T_{working}$ now has subtree $T_{current}|C$. Consider the root r_C of the subtree $T_{current}|C$ of $T_{working}$ and the root r_L of the subtree $T_{working}|\mathcal{L}(\mathcal{R})$ of $T_{working}$. If r_C is not a child of r_L, perform one more rSPR operation to prune the subtree $T_{current}|C$ and regraft it to a vertex subdividing the arc between r_L and its parent, giving tree $\hat{T}$. Append $\hat{T}$ to S.
4. Return S.

The fourth procedure, TraverseTree, is a recursive procedure that traverses a tree depth-first, calls the procedure Disentangle for each subtree, and combines the resulting sequences of trees. It then returns the entire sequence of rSPR-related trees produced by all of the recursive calls.

Procedure TraverseTree:

Input: a set $\mathcal{R}$ of rooted triples; a tree $T_{current} \in \langle \mathcal{R} \rangle_B$ to traverse; the root w of a subtree of $T_{current}$; a tree $T' \in \langle \mathcal{R} \rangle_B$.

Output: A sequence S of rSPR-related trees from $T_{current}$ to a tree with subtree T'.

1. Let $S = (\ )$ and let t_w be the subtree of $T_{current}$ rooted at w. If $|\mathcal{L}(t_w)| \in \{1, 2\}$, go to step 5 (i.e. if t_w consists of only a single leaf or a cherry, return an empty sequence.)

2. Let C be a maximal proper cluster of T'.
3. Do $S = S$+Disentangle$(\mathcal{R}, T_{current}, w, C)$. If $|S| \neq 0$, let $T_{current}$ be the last tree in the sequence S.
4. For $A \in \{C, \bar{C}\}$ (where $\bar{C}$ is the complement of C with respect to $\mathcal{L}(t_w)$):
   (i) Do $S = S$+TraverseTree$(\mathcal{R}_A, T_{current}, r_A, T'|A)$, where $\mathcal{R}_A = \{ab|c \in \mathcal{R} : a, b, c \in A\}$ and r_A is the root of the subtree $T_{current}|A$ of $T_{current}$ (the subtree containing exactly the leaves in A).
   (ii) If $|S| \neq 0$, let $T_{current}$ be the last tree in the sequence S.
5. Return S.

The last procedure, TreeSequence, takes two trees, T and T', as input and uses all of the above procedures to produce a sequence of NNI-related trees from T to T' such that each tree in the sequence displays $\mathcal{R}$. TreeSequence first calls FreeLeaves with input T (respectively T') to produce a sequence of NNI-related trees from T to a tree $\widetilde{T}$ (respectively from T' to a tree $\widetilde{T}'$). TraverseTree is then applied to produce a sequence of rSPR-related trees from $\widetilde{T}$ to $\widetilde{T}'$, which ToNNI converts into a sequence of NNI-related trees. Lastly, these three sequences are combined to produce the required sequence of NNI-related trees from T to T'.

Procedure TreeSequence:

Input: Two rooted binary phylogenetic trees, T and T'.

Output: A sequence of NNI-related trees from T to T' such that each tree in the sequence displays $\mathcal{R}$.

1. Let $\mathcal{R} = r(T) \cap r(T')$, so $T, T' \in \langle \mathcal{R} \rangle_B$. Let $L = \mathcal{L}(\mathcal{R})$. Do $F =$ FreeLeaves$(\mathcal{R}, T)$ and $F' =$ FreeLeaves$(\mathcal{R}, T')$. Let $\widetilde{T}$ and $\widetilde{T}'$ be the last trees in the sequences F and F' respectively. Note that $\widetilde{T} \setminus (T|L)$ is equivalent to $\widetilde{T}' \setminus (T'|L))$.
2. Reverse F' and call this sequence of trees $\overleftarrow{F'}$.
3. Do $S =$ TraverseTree$(\mathcal{R}, \widetilde{T}, r_{T|L}, \widetilde{T}')$, where $r_{T|L}$ is the root of the subtree $T|L$ of $\widetilde{T}$.

Now S is a sequence of rSPR-related trees from $\widetilde{T}$ to $\widetilde{T}'$ for which each tree in the sequence displays $\mathcal{R}$.

4. Do $\widetilde{S} = \text{ToNNI}(S)$.
5. Return $F + \widetilde{S} + \overleftarrow{F'}$, which is a sequence of NNI-related trees from T to T' satisfying the required properties.

2.4.2 Complexity

In this section we calculate the complexity of each procedure. We start by noting that one rSPR operation is $O(1)$, as is one NNI operation. Recall that $T, T' \in \langle \mathcal{R} \rangle_B$ and $\mathcal{L}(\mathcal{R}) \subseteq \mathcal{L}(T) = \mathcal{L}(T')$. Let $n = |\mathcal{L}(T)|$. Let $n_R = |\mathcal{L}(\mathcal{R})|$, the number of leaves in $\mathcal{R}$, and let $n_F = |\mathcal{L}(T)| - n_R$, the number of free leaves in T, so that $n = n_R + n_F$.

First consider the procedure FreeLeaves. This procedure uses NNI operations to produce, from T, a tree with subtree $T|\mathcal{L}(\mathcal{R})$, as described in the procedure. Let $D = d(T)$ be the depth of T. For tree T, each leaf requires $O(D)$ NNI operations. There are n_F leaves for which this must be repeated, so this procedure is $O(n_F D)$.

Next, we consider the procedure ToNNI applied to a sequence $S = (S_1, \ldots, S_k)$ of rSPR-related trees. This procedure produces from S a sequence of NNI-related trees. Let $D_S = \max\{d(S_i) \text{ for } 1 \leq i \leq k\}$. Each rSPR operation corresponds to at most $2D_S$ NNI operations since, in the worst case, arcs e and f (given in the definition of an rSPR operation) are distance $2D_S - 2$ apart. Therefore, each consecutive pair of trees in S produces a sequence of up to $2D_S$ NNI-related trees. There are $k - 1$ consecutive pairs of trees in S, so this procedure is $O(kD_S) = O(|S|D_S)$.

Consider the procedure Disentangle. Let t_w be the subtree of $T_{current}$ rooted at w. Disentangling the leaf set C from t_w requires up to $2|C| - 1$ rSPR operations. Therefore, the total number of rSPR operations required is at most $2|C| - 1$. Since $2|C| - 1 < 2|C| <$

$2n_{\mathcal{R}}$, the procedure Disentangle is $O(n_{\mathcal{R}})$.

Now consider the procedure TraverseTree. The maximum recursion depth is $n_{\mathcal{R}}$. The call to the procedure Disentangle in step 3 is $O(n_{\mathcal{R}})$, as described above. Step 4 is $O(|C|)\times O(n_{\mathcal{R}})+O(|\bar{C}|)\times O(n_{\mathcal{R}}) = (O(|C|)+O(|\bar{C}|))\times O(n_{\mathcal{R}}) = O(n_{\mathcal{R}})\times O(n_{\mathcal{R}}) = O(n_{\mathcal{R}}^2)$. So the overall complexity of the procedure TraverseTree is $O(n_{\mathcal{R}}) + O(n_{\mathcal{R}}^2) = O(n_{\mathcal{R}}^2)$.

Lastly, consider the procedure TreeSequence. Let $D_T = \max\{d(T), d(T')\}$. Step 1 is $O(n_F D_T)$ (two calls to the procedure FreeLeaves). This gives two sequences of trees, F and F', where the length of F' is $An_F D_T$ for some constant A. Step 2 is therefore $O(n_F D_T)$, reversing the sequence F'. Step 3 is $O(n_{\mathcal{R}}^2)$ (call to the procedure TraverseTree). This gives a sequence of trees S' of length $Bn_{\mathcal{R}}^2$ for some constant B. Step 4 is $O(|S'|D_{S'}) = O(Bn_{\mathcal{R}}^2 D_{S'}) = O(n_{\mathcal{R}}^2 D_{S'})$ (call to the procedure ToNNI). Step 5 concatenates three sequences, the complexity of which can be $O(1)$ (depending on the implementation). Therefore, the procedure TreeSequence is $O(n_{\mathcal{R}}^2 D_{S'} + n_F D_T)$. Letting $D_{max} = \max\{D_{S'}, D_T\}$ and noting that $n_{\mathcal{R}} \leq n$ and $n_F \leq n$, the complexity is $O(n^2 D_{max})$.

Hence, producing a sequence of NNI-related trees from T to T' has a complexity of $O(D_{max} n^2)$.

2.5 Concluding comments

In this chapter, we have provided an explicit polynomial-time procedure for moving between any two trees on a phylogenetic 'terrace' using elementary (NNI) operations, so that each tree in the sequence also belongs to the terrace. Thus if two trees have an optimal score under some linear scoring function satisfying Eqn. (1.1), each tree in the sequence is also optimal. Of course, there are likely to be many other such sequences

between the two trees that also lie on the terrace, so having some way of quantifying this number would be interesting. A further question, that is particularly relevant to many applications, asks for the development of a polynomial-time approximation procedure for sampling the trees on a terrace uniformly at random (or, equivalently, the trees that display a set of rooted triples). An approach based on random NNI or rSPR walks (sequences of NNI or rSPR operations) that move between trees on the terrace may provide a way to approach this problem; this was, in part, the motivation for this research. The development of an efficient randomized sampling scheme for trees on a terrace seems a worthy topic for further study.

Chapter 3

A terrace with only one tree

3.1 Introduction

Patchy taxon coverage may provide anywhere from very little to a lot of information on the evolutionary relationships within a set of species. Thus the process of reconstructing these relationships is capable of producing anywhere from a large terrace of trees [9] to only a single tree. In this chapter, we will examine the conditions under which a single tree is produced. We look at the simplest case of this problem - the conditions under which two trees with overlapping leaf sets define a single tree. Three cases to be considered: (1) the case in which two binary trees define a binary tree, (2) the case where two non-binary trees define a binary tree, and finally (3) where two non-binary trees define a non-binary tree. The conditions were established for the first two cases by Sebastian Böcker in [1]. We restate and independently prove those results here, and then investigate the third case, including a more general version of this case which is an open problem. When a set of trees does not define a single tree, we can ask: what is the maximum leaf set we can restrict these trees to such that they do define a single tree? This is the Maximum Defining Leaf Set Problem [9]. We ask this question for cases (1) and (2) and provide

the solution.

In this chapter, we begin by defining some necessary terms in Section 3.1.1. In Section 3.2, we consider the conditions under which two trees define a single tree. There are three cases, as described previously, investigated in Sections 3.2.1), 3.2.2, and Section 3.2.3 respectively. In cases (1) and (2), we also show how to construct the tree T and find the maximum defining leaf set when the two trees do not define a single tree. We initially hoped to solve the more general problem of the conditions under which three binary trees define a single binary tree. At the end of Section 3.2 we briefly investigate this more complicated scenario.

3.1.1 Definitions

In the previous chapter we considered rooted phylogenetic trees. Now we consider (unrooted) phylogenetic trees. Recall from Section 1.3 that a phylogenetic tree is a tree in which all the leaves are labeled and all interior vertices have degree at least three. Recall that $P(X)$ denotes the set of all phylogenetic trees with leaf set X. A binary phylogenetic tree is a phylogenetic tree in which all interior vertices have degree three. Let $BP(X)$ denote the set of all binary phylogenetic trees with leaf label set X. Note that $BP(X) \subseteq P(X)$. Figure 3.1 shows an example of a binary phylogenetic tree.

The definitions of deletion and contraction given in Section 1.5.1 also apply to phylogenetic trees that are not necessarily rooted (using edges in place of arcs).

An X-*split* is a partition of a set X into two non-empty sets, say A and B. This split is denoted $A|B$. For ease of notation we write, for example, $\{a, b, c\}|\{d, e\} = abc|de$ and $(A \cup B)|C = AB|C$.

Consider a tree $T \in P(X)$ and an edge e of T. Then $T \setminus e$ consists of two components, T' and T''. Now $\mathcal{L}(T')|\mathcal{L}(T'')$ is an X-split and we say that this X-split *cor-*

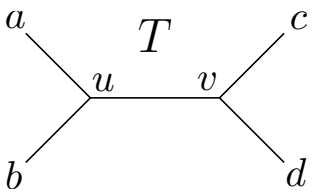

Figure 3.1: An example of an (unrooted) tree with leaf set $X = \{a, b, c, d\}$.

responds to edge e in T and no other edge of T. For each edge of T there is a corresponding X-split (and vice-versa); we denote the set of all these splits by $\Sigma(T)$. We refer to $\Sigma(T)$ as the *splits of T*. If follows that a tree $S \in BP(X)$ with $|X| \geq 2$ has $|\Sigma(S)| = 2|X| - 3$, since S has $2|X| - 3$ edges (see [10]). In Figure 3.1, the set of splits of T, $\Sigma(T) = \{ab|cd, a|bcd, b|acd, c|abd, d|abc\}$. These splits correspond to the edges $\{u, v\}, \{a, u\}, \{b, u\}, \{c, v\}$, and $\{d, v\}$ respectively.

The following terms were defined previously for rooted trees; here we define them for unrooted trees. Let $T \in P(X)$ and let X' be a non-empty subset of X. Then $T|X' \in P(X')$, called the *restriction of T to X'*, is the tree for which

$$\Sigma(T|X') = \{(A \cap X')|(B \cap X') : A|B \in \Sigma(T) \text{ and } A \cap X', B \cap X' \neq \emptyset\}$$

We can obtain $T|X'$ from T by deleting all maximal subtrees containing only leaves that are not in X' and then suppressing all vertices with degree two. Let $T' \in P(X)$. We say that T *refines* T' (or *is a refinement of* T') if $\Sigma(T') \subseteq \Sigma(T)$. Let $X'' \subseteq X$, and let $T'' \in P(X'')$. We say that T *displays* T'' if $T|X''$ is a refinement of T''.

A set $\mathcal{P}$ of phylogenetic trees (respectively, binary phylogenetic trees) is *compatible* if there exists a tree $T \in P(X)$ (respectively, $T \in BP(X)$) such that T displays each tree in $\mathcal{P}$. We then say that T *displays* $\mathcal{P}$. Note that T is not necessarily in $\mathcal{P}$. Let $\langle\mathcal{P}\rangle_P$ (respectively $\langle\mathcal{P}\rangle_{BP}$) denote the set of all phylogenetic trees (respectively binary phylogenetic trees) that display $\mathcal{P}$. Note that $\langle\mathcal{P}\rangle_{BP} \subseteq \langle\mathcal{P}\rangle_P$. If $\langle\mathcal{P}\rangle_P = \{T\}$ for some tree T, we say that $\mathcal{P}$ *defines* T. If $\langle\mathcal{P}\rangle_P = \{T_1, \ldots, T_n\}$ and there is some T_i $(1 \leq i \leq n)$

such that T_j is a refinement of T_i for all j $(1 \leq j \leq n, j \neq i)$, we say that $\mathcal{P}$ *identifies* T.

Consider a tree $T \in BP(X)$. Let the leaf set $Y \subset X$ and let tree $T' = T|Y$. Consider a split σ of T and a split σ' of T'. Let $\sigma = A|B \in \Sigma(T)$ and $\sigma' = A'|B' \in \Sigma(T')$. If $A \cap Y = A'$ and $B \cap Y = B'$ then we say that σ *extends* σ' and write $\sigma' \leq \sigma$.

Consider a tree $T_1 \in BP(X)$, a leaf set $X' \subset X$, and the tree $T_S = T|X' \in BP(X')$. We call tree T_S an *underlying tree* of T_1. Consider an edge $e = \{w, x\}$ of T'_S. This edge corresponds to some path $P_e(T_1, T_S)$ of T_1 from w to x. Note that $P_e(T_1, T_S)$ may consist of only a single edge. Consider an edge $\{u, v\}$ of T_1 where u is an interior vertex of the path $P_e(T_1, T_S)$ (not one of the endpoints) and v is not in $P_e(T_1, T_S)$. Consider the pendant subtree t_v of T rooted at the vertex v and containing nothing in the path $P_e(T_1, T_S)$. This pendant subtree t_v is called a *pendant subtree of* $P_e(T_1, T_S)$, and we say that this pendant subtree is *attached* to $P_e(T_1, T_S)$ at the vertex u (by the edge $\{v, u\}$). We call u the *attachment vertex* of t_v. See Figure 3.2 for an example of these definitions.

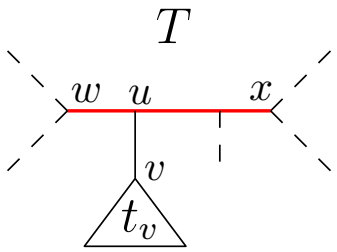

Figure 3.2: An example of an unrooted tree T with path $P_e(T)$ from w to x (shown by a thicker red line) and pendant subtree t_v attached to $P_e(T)$ at u.

Assume that the path $P_e(T_1, T_S)$ has length at least two. Then $P_e(T_1, T_S)$ has at least one pendant subtree attached. We can attach these pendant subtrees to edge e in T_S, giving a new tree T_2. Note that attaching these subtrees subdivides the edge e and the order of the attachment vertices remains the same. For example, in Figure 3.3, $e = \{w, x\}$ and $P_e(T_1, T_S) = wv_1v_2v_3x$. We attach the pendant subtrees t_1, t_2, and t_3 of $P_e(T_1, T_S)$ to edge e of T_S. This gives the tree T_2 with path $wv_1v_2v_3x$ and with the pendant subtrees

t_1, t_2, and t_3 attached to v_1, v_2, and v_3 respectively. These definitions can be formalized using splits, though this is not given here.

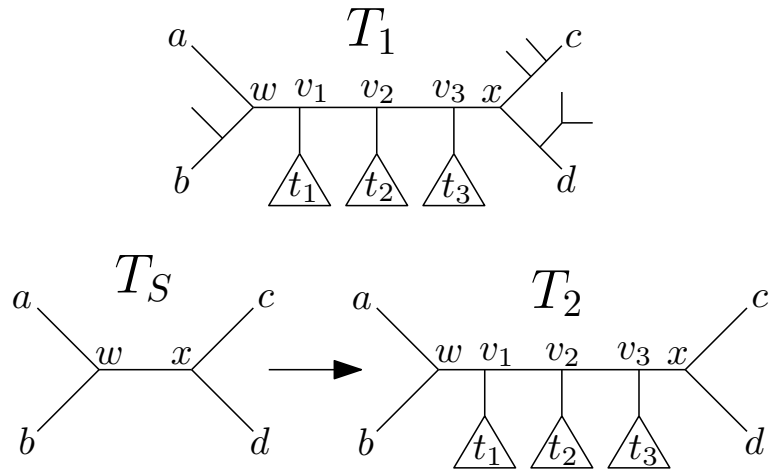

Figure 3.3: An example of a tree T_1 with underlying tree T_S and a tree T_2 obtained from T_S by attaching the pendant subtrees t_1, t_2, and t_3.

3.2 Two trees define a single tree

In this section we investigate the conditions under which two trees define a single tree. There are three cases to consider, which are dealt with in the following sections.

3.2.1 Binary to binary

We first state and prove the conditions under which two binary trees define a single tree.

Theorem 3.2.1. *Let $T_1 \in BP(X_1)$ and $T_2 \in BP(X_2)$ be compatible. Let $X = X_1 \cup X_2$ and $Y = X_1 \cap X_2$, and let $T_S \cong T_1|Y \cong T_2|Y$. Then T_1 and T_2 define (up to isomorphism) a tree $T \in BP(X)$ if and only if for each split $\sigma' \in \Sigma(T_S)$, either*

(a) $|\{\sigma \in \Sigma(T_1) : \sigma' \leq \sigma\}| = 1$ *or*

(b) $|\{\sigma \in \Sigma(T_2) : \sigma' \leq \sigma\}| = 1$.

For some edge $e = \{u, v\}$ of T_S with corresponding split σ' of T_S, if condition (a) holds then there is only one split, say σ_f, of T_1 that extends σ'. This split σ_f corresponds to some edge f of T_1 and so the path $P_e(T_1, T_S)$ consists of only the edge $f = \{u, v\}$. As $P_e(T_1, T_S)$ has length one, it contains no attachment vertices and therefore has no pendant subtrees. Hence, for each edge e of T_S, conditions (a) and (b) are equivalent to specifying that only $P_e(T_1, T_S)$ or $P_e(T_2, T_S)$, but not both, has pendant subtrees. We will use this notion in the proof of Theorem 3.2.1 below. Note that no pendant subtrees may be attached to the endpoints of any path $P_e(T_1, T_S)$ because then T_1 would be a non-binary tree. Similarly for T_2. This theorem was proved by Böcker in [1]. Here we explicitly state and independently prove this theorem.

Proof. $(\leftarrow)$ We assume that either (a) or (b) holds and prove that the tree T is unique. We use proof by induction on $k = |X_1| + |X_2|$. Note that the result holds in the case $X_1 = X_2 = Y$ because then $T_S \cong T_1 \cong T_2 \cong T$, so we assume that $X_1 \neq X_2$. Note also that, without loss of generality, if $Y = X_1$ then $T_S \cong T_1$ so T_2 displays T_1 and hence $T \cong T_2$ so the result holds. So we assume that $X_1 \neq Y$ and $X_2 \neq Y$.

Since X_1 and X_2 are non-empty, we first consider the case in which $k = 2$. Then, as $X_1 \neq X_2$, the leaf set X consists of exactly two distinct leaves, and so T is simply the tree consisting of these two leaves connected by an edge. As there are no other unrooted trees on two vertices, $\langle \{T_1, T_2\} \rangle_{BP} = \{T\}$ and so T_1 and T_2 define T, as required.

Assume that the result holds when $2 \leq k \leq n - 1$. Recall that $X_1 \neq Y$ and let $x \in X_1 \setminus Y$. Consider trees T_2 and $T_1|(X_1 \setminus \{x\})$ and underlying tree T_S. For each edge e of T_S, if $P_e(T_2, T_S)$ has pendant subtrees, attach these pendant subtrees to e, or if $P_e(T_1|(X_1 \setminus \{x\}), T_S)$ has pendant subtrees, attach these pendant subtrees to e. Call the resulting tree T^*. Note that since either condition (a) or (b) holds by assumption, only $P_e(T_2, T_S)$ or $P_e(T_1|(X_1 \setminus \{x\}), T_S)$, but not both, will have pendant subtrees for a given edge e of T_S. So tree T^* displays trees $T_1|(X_1 \setminus \{x\})$ and T_2 and so, by the induction

assumption, $T^* \in BP(X \setminus \{x\})$ is the unique tree displaying $T_1|(X_1 \setminus \{x\})$ and T_2.

Now consider the case in which $k = n$. Assume that there are two distinct trees, $T, T' \in BP(X)$, that each display both T_1 and T_2. Now T and T' can be obtained from T^* by attaching the leaf x. Figure 3.4 gives a graphical representation of the following labelings. Let e_1 (respectively e_2) be the edge of T^* that x is attached to in order to obtain the tree T (respectively T'). Note that e_1 and e_2 are distinct edges of T^* as $T \ncong T'$. In T^*, let the endpoints of edge e_1 be labeled u_1 and v_1 and let the endpoints of edge e_2 be labeled u_2 and v_2, such that there is a path P from u_1 to u_2 containing both v_1 and v_2. Let $\{v_1, w\}$ be the edge with endpoint v_1 that is not part of the path P. Let A be the pendant subtree of T^* containing vertex u_1 and with attachment vertex v_1. Similarly, let C be the pendant subtree of T^* containing vertex u_2 and with attachment vertex v_2, and let B be the pendant subtree of T^* containing vertex w and with attachment vertex v_1 (B must exist as e_1 and e_2 are distinct edges of T^*). Note that it may be the case that $v_1 = v_2$. If $v_1 \neq v_2$ then there will be other pendant subtrees attached to interior vertices of the path P. We call these pendant subtrees $t_1, \ldots, t_i$, where $i = dist_{T^*}(v_1, v_2) - 1$.

We now prove that the subtrees A, B, and C each contain at least one leaf in X_1. Consider the tree T^* and underlying tree T_S. Assume that, $\mathcal{L}(B) \subseteq X_2$. Then no leaf of B is in Y, so, in T^*, B is attached to some path $P_e(T^*, T_S)$. We consider the possible locations of $P_e(T^*, T_S)$ in T^* and show that in each case, to obtain either T or T', the leaf x is also be attached to $P_e(T^*, T_S)$, a contradiction. There are three cases to consider. In the first case, $P_e(T^*, T_S)$ contains edge e_1 and the adjacent edge in path P, and B attaches directly to $P_e(T^*, T_S)$. In this case, to obtain T, both B and x attach to $P_e(T^*, T_S)$, a contradiction. In the other two cases, $P_e(T^*, T_S)$ is contained in the subtree A or is elsewhere in T^*. In these two cases, to obtain T, there exists a pendant subtree containing both x and B that attaches to $P_e(T^*, T_S)$, a contradiction. Therefore $\mathcal{L}(B) \not\subseteq X_2$, so B contains at least one leaf $b \in X_1$.

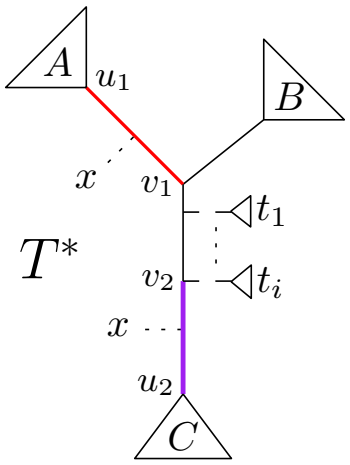

Figure 3.4: An illustration of the tree T^* showing the labels of the vertices and subtrees as in the proof of Theorem 3.2.1, and also showing the attachment points of vertex x to obtain trees T and T'. The edge $e_1 = \{u_1, v_1\}$ is shown in red, and $e_2 = \{u_2, v_2\}$ is shown in purple. Note that v_1 and v_2 may be the same vertex and so the path between them (and the subtrees $t_1, \ldots, t_i$) may not exist.

Now assume that $\mathcal{L}(A) \subseteq X_2$. Then, as before, subtree A attaches to some path $P_e(T^*, T_S)$. This path $P_e(T^*, T_S)$ must be contained in $T^* \setminus A$. Then, to obtain T, there exists a pendant subtree containing both A and the leaf x that attaches to $P_e(T^*, T_S)$, a contradiction. So A contains at least one leaf $a \in X_1$. Similarly, C contains at least one leaf $c \in X_1$.

Now A, B, and C each contain at least one leaf of X_1. So T displays the quartet $ay|bc$, and T' displays the quartet $ab|yc$, where $a, b, c, x \in X_1$. So T_1 must also display both of these quartets, a contradiction. Hence, by induction, T is unique and so T_1 and T_2 define T, as required.

($\rightarrow$) We assume that T is unique and prove that either (a) or (b) holds. Consider the trees T_1 and T_2 and underlying tree T_S. Assume that, for some edge e of T_S, the path $P_e(T_1, T_S)$ has at least one pendant subtree, say A, and the path $P_e(T_2, T_S)$ has at least

one pendant subtree, say B. To obtain tree T, which displays both T_1 and T_2 and also has underlying tree T_S, both A and B must be attached to edge e of T_S. However, these two subtrees can be attached to e in more than one arrangement while still satisfying the condition that the resulting tree displays both T_1 and T_2, as illustrated in Figure 3.5. Note that there are more possible arrangements than those shown in Figure 3.5. These different arrangements each give a distinct tree that displays both T_1 and T_2 and so T is not unique, a contradiction. Therefore only $P_e(T_1, T_S)$ or $P_e(T_2, T_S)$, but not both, may have pendant subtrees, and so either (a) or (b) must hold, as required. □

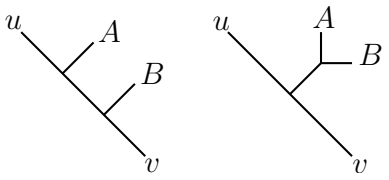

Figure 3.5: An example of two possible ways of attaching the pendant subtrees A (from tree T_1) and B (from tree T_2) to the edge $e = \{u, v\}$ of T_S.

To construct the tree T, start with the tree T_S. Consider an edge e of T_S. If the path $P_e(T_1, T_S)$ has pendant subtrees, then attach these subtrees to e in T_S. Similarly for $P_e(T_2, T_S)$. By Theorem 3.2.1, only one of these options is possible for a particular edge e of T_S. Consider another edge of T_S and repeat this process. Note that in this step we are actually working with the tree produced by the previous step, rather than the original tree T_S, but the tree T_S is the underlying tree of the current tree and it is these edges that we are attaching pendant subtrees to, as illustrated in Figure 3.6.

If trees $T_1 \in BP(X_1)$ and $T_2 \in BP(X_2)$ do not define a tree $T \in BP(X)$ (recall $X = X_1 \cup X_2$), we find the largest set $X' \subseteq X$ such that the trees $T_1' = T_1|X'$ and $T_2' = T_2|X'$ define a tree $T' \in RB(X')$. This is a simple case of the Maximum Defining Leaf Set Problem and was solved in [9]. We independently give the solution here.

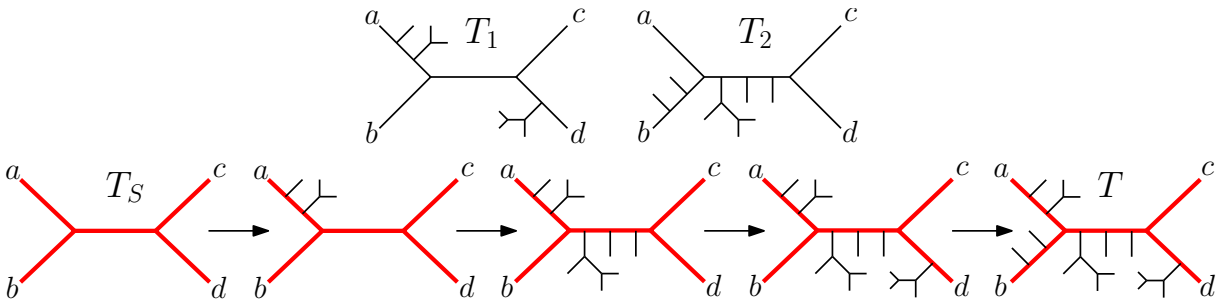

Figure 3.6: An example of the process used to obtain T from T_S. The underlying tree T_S is illustrated with thicker red lines.

We start with the leaf set X and trees T_1 and T_2, and remove as few leaves as possible from leaf set X and trees T_1 and T_2 so that the resulting trees define a single tree. The removal of these leaves is carried out as follows. Assume that trees T_1 and T_2 do not define a tree T. Hence, by Theorem 3.2.1, there is at least one edge e of underlying tree T_S such that both the path $P_e(T_1, T_S)$ and the path $P_e(T_2, T_S)$ have pendant subtrees. Let $E \subseteq E(T_S)$ be the set of all such edges. We create the trees T_1' and T_2' from trees T_1 and T_2 respectively, such that Theorem 3.2.1 is satisfied. So for each edge $e \in E$, at most one of the paths $P_e(T_1, T_S)$ or $P_e(T_2, T_S)$ has pendant subtrees, and so we must remove the pendant subtrees of either $P_e(T_1, T_S)$ or $P_e(T_2, T_S)$. Let $X_e(T_1, T_S)$ be the leaf set of the pendant subtrees of $P_e(T_1, T_S)$ and similarly for T_2. The trees T_1' and T_2' have the largest possible leaf set, so if $|X_e(T_1, T_S)| \geq |X_e(T_2, T_S)|$, delete the pendant subtrees of $P_e(T_2, T_S)$, otherwise, delete the pendant subtrees of $P_e(T_1, T_S)$. Note that if $|X_e(T_1, T_S)| = |X_e(T_2, T_S)|$, we could have deleted the pendant subtrees of $P_e(T_1, T_S)$ instead of those of $P_e(T_2, T_S)$. Repeating this for each edge $e \in E$ gives the trees T_1' and T_2' and the required leaf set $X' = \mathcal{L}(T_1') \cup \mathcal{L}(T_2')$. Note that for edges of T_S that are not in E, there is no contradiction of Theorem 3.2.1 and so we do not delete any pendant subtrees in those cases. The leaf set X' can be obtained as

$$X' = Y \cup \bigcup_{e \in E(T_S)} X_e'$$

where

$$X'_e = \begin{cases} X_e(T_1, T_S) & \text{if } |X_e(T_1, T_S)| \geq |X_e(T_2, T_S)| \\ X_e(T_2, T_S) & \text{otherwise} \end{cases} \tag{3.1}$$

3.2.2 Non-binary to binary

We now consider the case in which the trees T_1 and T_2 may be non-binary and investigate the conditions under which .

Theorem 3.2.2. *Let $T_1 \in P(X_1)$ and $T_2 \in P(X_2)$ be compatible. Let $X = X_1 \cup X_2$ and let $Y = X_1 \cap X_2$. Let $T_S \in BP(Y)$ such that $T_S \geq T_1|Y$ and $T_S \geq T_2|Y$. Then T_1 and T_2 define (up to isomorphism) a tree $T \in BP(X)$ if*

(i) for each split $\sigma' \in \Sigma(T_S)$, either

(a) $|\{\sigma \in \Sigma(T_1) : \sigma' \leq \sigma\}| \leq 1$ or

(b) $|\{\sigma \in \Sigma(T_2) : \sigma' \leq \sigma\}| \leq 1$.

and

(ii) $|\Sigma(T_i)| - |\Sigma(T_i|Y)| = 2|X_1| - 2|Y|$ for $i = 1, 2$.

Condition (ii) says that T_1 and T_2 must be binary except for the subtrees $T_1|Y$ and $T_2|Y$, which ensures that the resulting tree T will be binary.

This result was also proved by Böcker in [1]. We prove it independently here.

Proof. $(\leftarrow)$ Assume that conditions (i) and (ii) hold and assume that T_1 and T_2 do not define a binary tree $T \in BP(X)$. So either there is no binary tree T displaying T_1 and T_2, or T is not unique.

We first prove that there exists a binary tree that displays trees T_1 and T_2. By assumption, the trees T_1 and T_2 are compatible, so there must be some (not necessarily binary) tree T' that displays both T_1 and T_2. We can construct such a tree T' that

displays both T_1 and T_2 as follows. Consider trees T_1 and T_2 and underlying tree T_S. For each edge e of T_S, if the path $P_e(T_1, T_S)$ of T_1 has pendant subtrees then attach those pendant subtrees to e, otherwise attach the pendant subtrees of the path $P_e(T_2, T_S)$ of T_2 to edge e. Call the resulting tree T'. At least one of the paths $P_e(T_1, T_S)$ and $P_e(T_2, T_S)$ must exist, otherwise e would not be an edge of T_S. Also, by condition (i), either path $P_e(T_1, T_S)$ or path $P_e(T_2, T_S)$, but not both, has pendant subtrees. So tree T' displays both T_1 and T_2. By condition (ii), trees T_1 and T_2 are both binary, apart from the subtrees $T_1|Y$ and $T_2|Y$, so all pendant subtrees of the paths $P_e(T_1, T_S)$ and $P_e(T_2, T_S)$ are binary. Also, by assumption, T_S is binary. Therefore the tree T' is also binary. We now have a binary tree T' that displays both T_1 and T_2.

We now prove that T' is unique. Assume that T' is not unique. Following the proof of Theorem 3.2.1 in Section 3.2.1, we obtain a contradiction. (The only change in the proof of Section 3.2.1 is in the case $X_1 = X_2 = Y$. Now $T_S \ncong T'_1 \ncong T'_2$, but it is still true that $T_S \cong T$, so the result still holds in this case.) Hence T' is unique. Now T' is the unique binary tree displaying both T_1 and T_2, as required.

$(\rightarrow)$ The proof follows that of Theorem 3.2.1 in Section 3.2.1. $\square$

The construction of the tree T is the same as in the previous section.

If trees T_1 and T_2 do not define a tree T, we find the largest set $X' \subseteq X$ such that trees $T_1|X'$ and $T_2|X'$ define a tree $T' \in RB(X')$. This is another simple case of the Maximum Defining Leaf Set Problem, and the solution is given below.

Consider tree T_1 and underlying tree T_S. Consider an edge e of T_S and the path $P_e(T_1, T_S)$. Let t_1 be the subtree of T_1 consisting of the path $P_e(T_1, T_S)$ and its pendant subtrees. Define t_2 similarly. Note that either the path $P_e(T_1, T_S)$ or the path $P_e(T_2, T_S)$ may not exist (but at least one must exist, otherwise e is not an edge of T_S). So either the subtree t_1 of T_1 or the subtree t_2 of T_2 may not exist. As trees T_1 and T_2 do not

define a tree T, condition (ii) may no longer hold, and so the subtrees t_1 and t_2 may not be binary. In order for trees $T_1|X'$ and $T_2|X'$ to define a binary tree T, we require that condition (ii) holds for trees $T_1|X'$ and $T_2|X'$. So, for each edge e of T_S, the subtree t_1 of T_1 and the subtree t_2 of T_2 must be made binary while maintaining the largest possible leaf set.

Consider the path $P_e(T_1, T_S)$ of T_1. Consider a pendant subtree t_v of $P_e(T_1, T_S)$ attached by edge $\{v, u\}$ where vertex u is the attachment vertex. We can consider the subtree t_v as a rooted tree with root v. Visiting the vertices of subtree t_v in a depth-first order, for each interior vertex x of t_v, if the outdegree of vertex x is greater than two, prune the subtree of t_x with the smallest number of leaves, repeating until vertex x has outdegree two. Call the resulting binary subtree t'_v (which will still have root v) and replace the pendant subtree t_v of $P_e(T_1, T_S)$ with the subtree t'_v. Repeat this process for each pendant subtree of $P_e(T_1, T_S)$. Lastly, for any binary pendant subtrees that share an attachment vertex, keep the pendant subtree with the largest leaf set and prune the others, so that each pendant subtree has a unique attachment vertex. Note that none of the attachment vertices are deleted.

Repeating the above process for each edge e of T_S and corresponding path $P_e(T_1, T_S)$ gives a tree T_{1B} that satisfies condition (ii) while retaining the largest possible leaf set. The tree T_{2B} is obtained similarly.

We now use the same process as in Section 3.2.1, using the trees T_{1B} and T_{2B} instead of T_1 and T_2, to obtain the leaf set X', as required.

3.2.3 Non-binary to non-binary

Two compatible non-binary trees define a single tree T if and only if Theorem 3.2.2 holds. So, in this section, we investigate the related and more general problem of the conditions

under which two compatible non-binary trees T_1 and T_2 identify a non-binary tree T, i.e. the conditions under which there is some tree $T \in \langle\{T_1, T_2\}\rangle_P$ such that every tree in $\langle\{T_1, T_2\}\rangle_P$ is a refinement of T.

This investigation revealed that the result does not follow directly from the previous two sections. Firstly, as T may be non-binary, unlike the previous two cases we must consider that we may attach a pendant subtree to either an edge or a vertex of T_S. In this new scenario, where a pendant subtree is attached to a vertex v of T_S, the existing vertex v is the attachment vertex of that pendant subtree. Note that this situation applies only to this case.

However, this is not as straightforward as in the previous cases, and we cannot simply specify that we may not attach pendant subtrees of both tree T_1 and tree T_2 to the same edge of tree T_S, and similarly for attaching pendant subtrees to a particular vertex of T_S. If, in T_1, there are one or more pendant subtrees with the same attachment vertex and for which each of these pendant subtrees consists of only a single leaf, we call this collection of pendant subtrees a *tussock* of T_1. In the case where a tussock of T_1 and a tussock of T_2 both attach to the same edge or vertex w of T_S (to obtain T_1 and T_2 respectively), to obtain T we may attach both of these tussocks to w in T_S, as illustrated in Figure 3.7 for the case where w is an edge of T_S. This is because, in T_1, the tussock allows for all possible resolutions, as it does in T_2, and the combined larger tussock of T also allows for all possible resolutions. Hence any subtree displaying these tussocks of T_1 and T_2 will be a refinement of the larger tussock of T and vice-versa.

It is also the case that for a vertex v and incident edge e of tree T_S, we cannot attach pendant subtrees of tree T_1 to vertex v and pendant subtrees of tree T_2 to edge e, or vice-versa, as illustrated in Figure 3.8. Figure 3.8 shows trees T_1 and T_2, where underlying tree T_S is the quartet $ab|cd$. The trees T_3 and T_4 both display T_1 and T_2 but tree T_3 displays the quartet $ax|by$ while tree T_4 displays the quartet $ab|xy$. So there is no tree T

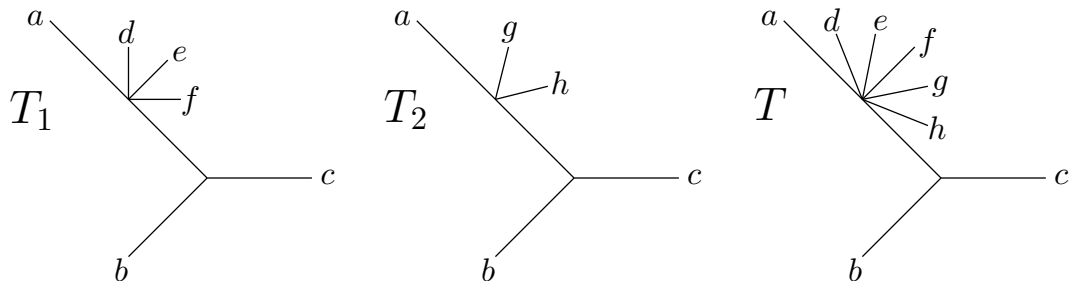

Figure 3.7: An example of attaching a tussock of T_1 and a tussock of T_2 to an edge of T_S.

such that T_3 and T_4 are both refinements of T and where T displays both T_1 and T_2. In any such tree T, the pendant edges incident to a, b, x, and y share an endpoint, but then T does not display T_2. This situation occurs any time pendant subtrees of T_1 and T_2 are attached to a vertex v and an incident edge e of T_S, respectively (or vice-versa).

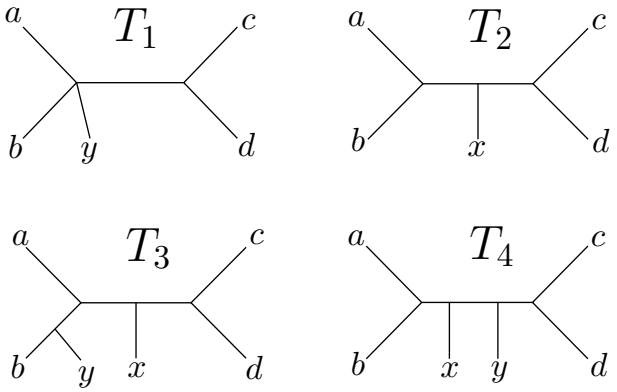

Figure 3.8: An example of pendant subtrees of T_1 and T_2 attaching to an incident vertex and edge, respectively, of T_S, and two trees, T_3 and T_4, that both display T_1 and T_2.

Another issue is illustrated in Figure 3.9. The trees T_3 and T_4 each display both tree T_1 and tree T_2 but there is no tree T such that T_3 and T_4 are refinements of T and where T displays both T_1 and T_2. In tree T_3, the pendant edges incident to a, b, and y share an endpoint, and, in tree T_4, the pendant edges incident to y and c share an endpoint. So, in any such tree T, the pendant edges incident to a, b, c, and y share an endpoint, but then T does not display T_2.

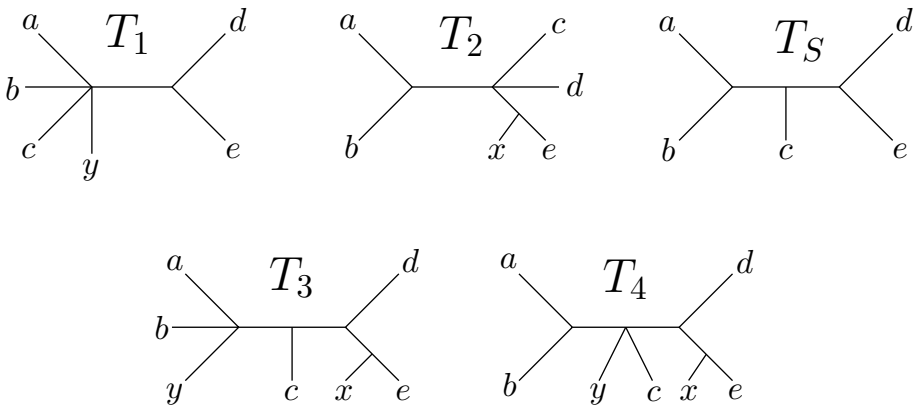

Figure 3.9: An example of two trees T_1 and T_2, the tree T_S, and two trees T_3 and T_4 that both display T_1 and T_2.

These examples illustrate that this case is far more complicated than the previous two cases and requires further investigation to determine exactly the set of conditions under which two non-binary trees identify a single non-binary tree.

3.2.4 A simple case in which three trees define a single tree

In this section we state and prove a simple case in which three trees define a single tree.

Theorem 3.2.3. *Let $T_i \in BP(X_i)$ for $i = 1, 2, 3$ be compatible. Let $X = X_1 \cup X_2 \cup X_3$ and let $Y = X_1 \cap X_2 \cap X_3$. Let $T_S = T_1|Y \cong T_2|Y \cong T_3|Y$. Then T_1, T_2, and T_3 define (up to isomorphism) a tree $T \in BP(X)$ if, for each split $\sigma' \in \Sigma(T_S)$, at least two of the following hold*

(a) $|\{\sigma \in \Sigma(T_1) : \sigma' \leq \sigma\}| = 1$,

(b) $|\{\sigma \in \Sigma(T_2) : \sigma' \leq \sigma\}| = 1$,

(c) $|\{\sigma \in \Sigma(T_3) : \sigma' \leq \sigma\}| = 1$.

This condition states that at most one of $P_e(T_1, T_S)$, $P_e(T_2, T_S)$, and $P_e(T_3, T_S)$ has pendant subtrees. If this is true, then T can be obtained by the same process as in Section 3.2.1. The proof, given below, follows that of Theorem 3.2.1 with a few minor

alterations.

Proof. We assume that at least two of $(a), (b)$, and (c) hold, and prove that the tree T is unique. We use proof by induction on $k = |X_1| + |X_2| + |X_3|$. Note that the result holds in the case $X_1 = X_2 = X_3 = Y$ because then $T_S \cong T_1 \cong T_2 \cong T_3 \cong T$, so we assume, without loss of generality, that $X_1 \neq X_2$.

Consider the case in which $X_1 = X_3$, without loss of generality. Then $T_1 \cong T_3$, so by Theorem 3.2.1 the result holds. So we also assume that $X_1 \neq X_3$ and $X_2 \neq X_3$.

Since X_1, X_2, and X_3 are non-empty, we first consider the smallest case, in which $k = 3$. Then, X consists of exactly three distinct leaves, and so T is simply the star with these three leaves. As there are no other unrooted trees on three vertices, $\langle\{T_1, T_2, T_3\}\rangle_{BP} = \{T\}$ and so T_1, T_2, and T_3 define T, as required.

Assume that the result holds when $3 \leq k \leq n-1$. Assume without loss of generality that $X_1 \neq Y$ and let $x \in X_1 \setminus Y$. Consider trees T_2, T_3, and $T_1|(X_1 \setminus \{x\})$ and underlying tree T_S. For each edge e of T_S, if $P_e(T_2, T_S)$ has pendant subtrees, attach these pendant subtrees to e, or similarly for T_3 and $T_1|(X_1 \setminus \{x\})$. Call the resulting tree T^*. Note that since at most one of conditions $(a), (b)$, and (c) does not hold, at most one of the three paths will have pendant subtrees for a given edge e of T_S. So tree T^* displays trees $T_1|(X_1 \setminus \{x\})$, T_2, and T_3 and so, by the induction assumption, $T^* \in BP(X \setminus \{x\})$ is the unique tree displaying trees $T_1|(X_1 \setminus \{x\})$, T_2, and T_3.

Now consider the case in which $k = n$. Assume that there are two distinct trees, T and T', that each display trees T_1, T_2, and T_3. Now T and T' can be obtained from T^* by attaching the leaf x. Figure 3.10 gives a graphical representation of the following labelings. Let e_1 (respectively e_2) be the edge of T^* that x is attached to in order to obtain the tree T (respectively T'). Note that e_1 and e_2 are distinct edges of T^* as $T \ncong T'$. In T^*, let the endpoints of edge e_1 be labeled u_1 and v_1 and let the endpoints of edge e_2

be labeled u_2 and v_2, such that there is a path P from u_1 to u_2 containing both v_1 and v_2. Let $\{v_1, w\}$ be the edge with endpoint v_1 that is not part of the path P. Let A be the pendant subtree of T^* containing vertex u_1 and with attachment vertex v_1. Similarly, let C be the pendant subtree of T^* containing vertex u_2 and with attachment vertex v_2, and let B be the pendant subtree of T^* containing vertex w and with attachment vertex v_1 (B must exist as e_1 and e_2 are distinct edges of T^*). Note that it may be the case that $v_1 = v_2$. If $v_1 \neq v_2$ then there will be other pendant subtrees attached to interior vertices of the path P. We call these pendant subtrees $t_1, \ldots, t_i$, where $i = dist_{T^*}(v_1, v_2) - 1$.

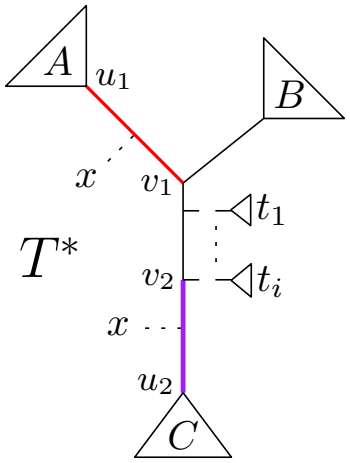

Figure 3.10: An illustration of the tree T^* showing the labels of the vertices and subtrees as in the proof of Theorem 3.2.3, and also showing the attachment points of vertex x to obtain trees T and T'. The edge $e_1 = \{u_1, v_1\}$ is shown in red, and $e_2 = \{u_2, v_2\}$ is shown in purple. Note that v_1 and v_2 may be the same vertex and so the path between them (and the subtrees $t_1, \ldots, t_i$) may not exist.

We now prove that the subtrees A, B, and C each contain at least one leaf in X_1. Consider the tree T^* and underlying tree T_S. Assume that, $\mathcal{L}(B) \subseteq X_2 \cup X_3$, i.e. B contains no leaves in X_1. If B contains a leaf in X_2 and a leaf in X_3 then B is not a pendant subtree of some path $P_e(T^*, T_S)$ (or contained within such a pendant subtree). Any other subtree of T^* contains at least one leaf in Y, so B contains a leaf in Y and

hence a leaf in X_1, a contradiction. So assume without loss of generality that $\mathcal{L}(B) \subseteq X_2$. Then no leaf of B is in Y, so, in T^*, B is attached to some path $P_e(T^*, T_S)$. We consider the possible locations of $P_e(T^*, T_S)$ in T^* and show that in each case, to obtain either T or T', the leaf x is also attached to $P_e(T^*, T_S)$, a contradiction. There are three cases to consider. In the first case, $P_e(T^*, T_S)$ contains edge e_1 and the adjacent edge in path P, and B attaches directly to $P_e(T^*, T_S)$. In this case, to obtain T, both B and x attach to $P_e(T^*, T_S)$, a contradiction. In the other two cases, $P_e(T^*, T_S)$ is contained in the subtree A or is elsewhere in T^*. In these two cases, to obtain T, there exists a pendant subtree containing both x and B that attaches to $P_e(T^*, T_S)$, a contradiction. Now all cases lead to contradictions, so $\mathcal{L}(B) \not\subseteq X_2 \cup X_3$ and therefore B contains at least one leaf $b \in X_1$, as required.

Now consider the subtree A. Assume that $\mathcal{L}(A) \subseteq X_2 \cup X_3$. If A contains a leaf in X_2 and a leaf in X_3 then, as in the case of the subtree B, this gives a contradiction. So assume without loss of generality that $\mathcal{L}(A) \subseteq X_2$. Then, as before, subtree A attaches to some path $P_e(T^*, T_S)$. This path $P_e(T^*, T_S)$ must be contained in $T^* \setminus A$. Then, to obtain T, there exists a pendant subtree containing both A and the leaf x that attaches to $P_e(T^*, T_S)$ a contradiction. Again, all cases lead to contradictions, so $\mathcal{L}(A) \not\subseteq X_2 \cup X_3$ and therefore A contains at least one leaf $a \in X_1$, as required. Similarly, C contains at least one leaf $c \in X_1$.

Now A, B, and C each contain at least one leaf of X_1 (a, b, and c respectively). So T displays the quartet $ax|bc$ and T' displays the quartet $ab|xc$, where $a, b, c, x \in X_1$. So T_1 also displays both of these quartets, a contradiction. Hence, by induction, T is unique and so T_1, T_2, and T_3 define T, as required.

□

This is only the simplest case in which T_1, T_2, and T_3 define a single tree T. Further

investigation revealed that there are far more cases to be considered, many of which are more complicated than the case described above.

3.3 Concluding comments

In this chapter we explored the simplest cases in which a set of trees defines (or identifies, in the case of non-binary trees) a single tree. But investigation revealed that some of these seemingly simple cases are actually quite hard and lead to many conditions and sub-cases. We established the conditions under which two binary trees define a single binary tree, which also led to the solution to the Maximum Defining Leaf Set problem for this case, and similarly for two non-binary trees defining a single binary tree. However, the more general problem of the conditions under which two non-binary trees identify a single, non-binary tree has been left open. Some observations were made that may help whoever next picks up this problem, if only just to serve as a reminder that it is not as simple as it may seem at first glance.

This chapter also briefly looked at the conditions under which three binary trees define a single binary tree and provided a proof of the simplest sub-case of this. There is plenty of further work to be done here, continuing the investigation into the conditions under which three binary trees define a single binary tree with a view to generalizing these findings, if possible, to obtain the conditions under which a set of binary trees (of any given size) defines a single binary tree. There is also the generalization of this problem to non-binary trees, and the generalization of a solution to the Maximum Defining Leaf Set Problem to be considered in each of these cases.

Bibliography

[1] Böcker, S. (1999). *From Supertrees to Subtrees*. PhD thesis, University of Bielefeld.

[2] Bordewich, M. (2003). *The Complexity of Counting and Randomised Approximation*. PhD thesis, University of Oxford.

[3] Chernomor, O., Minh, B. Q., and von Haeseler A, A. (2015). Consequences of common topological rearrangements for partition trees in phylogenomic inference. *J. Comput. Biol.*, 22:1–14.

[4] Felsenstein, J. (2004). *Inferring phylogenies*. Sinauer Press.

[5] Jetz, W., Thomas, G. H., Joy, J. B., Redding, D. W., Hartmann, K., and Mooers, A. O. (2014). Global distribution and conservation of evolutionary distinctness in birds. *Curr. Biol.*, 24:1–12.

[6] Maddison, D. R. (1991). The discovery and importance of multiple islands of most-parsimonious trees. *Syst. Zool.*, 40:315–328.

[7] Robinson, D. F. (1971). Comparison of labeled trees with valency three. *J. Comb. Theory B*, 11:105–119.

[8] Sanderson, M. J., McMahon, M. M., Stamatakis, A., Zwickl, D., and Steel, M. (2015). Impacts of terraces on phylogenetic inference. *Syst. Biol.*, page syv024.

[9] Sanderson, M. J., McMahon, M. M., and Steel, M. (2011). Terraces in phylogenetic tree space. *Science*, 333:448–450.

[10] Semple, C. and Steel, M. (2003). *Phylogenetics*. Oxford University Press.