2012

Constructing liberal and conservative supertrees and exact solutions for reduced consensus problems

Jianrong Dong
Iowa State University

Constructing liberal and conservative supertrees and exact solutions for reduced consensus problems

by

Jianrong Dong

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:

David Fernández-Baca, Major Professor

Oliver Eulenstein

Giora Slutzki

Soma Chaudhuri

Stephen Willson

Iowa State University

Ames, Iowa

2012

DEDICATION

I would like to dedicate this dissertation to my wife, Liming, and to our sons, Richard, Steven and Jason, without whose support I would not have been able to complete this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis.

First and foremost, Prof. David Fernández-Baca, my mentor, for his guidance, patience and support throughout all these years, not only in the research and the writing of this thesis, but also in the publication of various papers and my career design and development. His knowledge, insights, judgment, planning and word of encouragement have often inspired me to think differently, study more on these interesting problems and publish more.

I would also like to thank my committee members for their efforts and contributions to this work: Prof. Oliver Eulenstein, Prof. Giora Slutzki, Prof. Soma Chaudhuri and Prof. Stephen Willson.

I would additionally like to thank Prof. F. R. McMorris and Dr. Mark Wilkinson for their seminal works to include in their formulations such nice properties that enable us to tackle large problems.

ABSTRACT

This thesis studies two different approaches to extracting information from collections of phylogenetic trees: supertrees and reduced consensus. Supertree methods combine the phylogenetic information from multiple partially-overlapping trees into a larger phylogenetic tree called a supertree. Several supertree construction methods have been proposed to date, but most of these are not designed with any specific properties in mind. Recently, Cotton and Wilkinson proposed extensions of the majority-rule consensus tree method to the supertree setting that inherit many of the appealing properties of the former.

We study a variant of one of Cotton and Wilkinson's methods, called majority-rule (+) supertrees. After proving that a key underlying problem for constructing majority-rule (+) supertrees is NP-hard, we develop a polynomial-size exact integer linear programming formulation of the problem. We then present a data reduction heuristic that identifies smaller subproblems that can be solved independently. While this technique is not guaranteed to produce optimal solutions, it can achieve substantial problem-size reduction. Finally, we report on a computational study of our approach on various real data sets, including the 121-taxon, 7-tree Seabirds data set of Kennedy and Page. The results indicate that our exact method is computationally feasible for moderately large inputs. For larger inputs, our data reduction heuristic makes it feasible to tackle problems that are well beyond the range of the basic integer programming approach. Comparisons between the results obtained by our heuristic and exact solutions indicate that the heuristic produces good answers. Our results also suggest that the majority-rule (+) approach, in both its basic form and with data reduction, yields biologically meaningful phylogenies.

Generalizations of the strict and loose consensus methods to the supertree setting, recently introduced by McMorris and Wilkinson, are studied. The supertrees these methods produce are conservative in the sense that they only preserve information (in the form of splits) that is

supported by at least one the input trees and that is not contradicted by any of the input trees. Alternative, equivalent, formulations of these supertrees are developed. These are used to prove the NP-completeness of the underlying optimization problems and to give exact integer linear programming solutions. For larger data sets, a divide and conquer approach is adopted, based on the structural properties of these supertrees. Experiments show that it is feasible to solve problems with several hundred taxa and several hundred trees in a reasonable amount of time.

A rogue taxon in a collection of phylogenetic trees is one whose position varies drastically from tree to tree. The presence of such taxa can greatly reduce the resolution of the consensus tree (e.g., the majority-rule or strict consensus) for a collection. The reduced consensus approach aims to identify rogue taxa and to produce more informative consensus trees. Given a collection of phylogenetic trees over the same leaf set, the goal is to find a set of taxa whose removal maximizes the number of internal edges in the consensus tree of the collection. This problem is proven to be NP-hard for strict and majority-rule consensus. We describe exact integer linear programming formulations for computing reduced strict, majority and loose consensus trees. In experimental tests, our exact solutions show significant improvement over heuristic methods on several problem instances.

CHAPTER 1. INTRODUCTION

A supertree method begins with a profile of phylogenetic trees with possibly different leaf (taxon) sets, and assembles them into a larger phylogenetic tree, a *supertree*, whose taxon set is the union of the taxon sets of the input trees. Interest in supertrees was sparked by Gordon's paper [Gordon (1986)]. Since then, particularly during the past decade, there has been a flurry of activity with many supertree methods proposed and studied from the algorithmic, theoretical, and biological points of view. The appeal of supertree synthesis is that it can combine disparate data to provide a high-level perspective that is harder to attain from individual trees. A recent example of the use of this approach is the species-level phylogeny of nearly all extant Mammalia constructed by Bininda-Emonds et al. (2007) from over 2,500 partial estimates. Several of the known supertree methods are reviewed in the book edited by Bininda-Emonds (2004) — more recent publications with good bibliographies include [Wilkinson et al. (2007); Ranwez et al. (2007); Scornavacca (2009)]. There is still much debate about what specific properties should (can), or should not (cannot), be satisfied by supertree methods. Indeed, it is often a challenging problem to rigorously determine the properties of a supertree method that gives seemingly good results on data, but is heuristic.

We cannot discuss supertree construction without referring to matrix representation with parsimony (MRP) [Baum and Ragan (2004); Baum (1992); Ragan (1992)], by far the most commonly used method. MRP first encodes the input trees as incomplete binary characters, and then builds a maximum-parsimony tree for this data. The popularity of MRP is perhaps due to the widespread acceptance of the philosophy underlying parsimony approaches and the availability of excellent parsimony software. While MRP often performs well [Bininda-Emonds and Sanderson (2001)], it is essentially an *ad hoc* adaptation of parsimony to the supertree setting. This is reflected in the fact that MRP supertrees may display relationships that are

not supported by any of the input trees [Pisani and Wilkinson (2002); Goloboff (2005)].

Here we consider methods based on *splits*, where a split is the bipartition of the taxon set induced the removal of an edge from a tree. Following Wilkinson et al. (2004), we say that a supertree method is *liberal* if it allows the inclusion of splits that are contradicted by some subset of the input trees, provided a certain optimization criterion is met. *Conservative* methods produce supertrees that only display splits that are not contradicted by any input tree. Previous work [Cotton and Wilkinson (2007); Dong and Fernández-Baca (2009); Dong et al. (2010a,b)] has characterized the mathematical and computational properties of majority-rule supertrees, a liberal approach.

The well-studied consensus tree problem can be viewed as the special case of the supertree problem where the input trees have identical leaf sets. Consensus methods in systematics date back to Adams III (1972); since then, many consensus methods have been designed. Good surveys of these methods, their properties, and their interrelationships are given by Bryant (2003) and Scornavacca (2009), while the axiomatic approach and the motivation from the social sciences is found in Day and McMorris' book [Day and McMorris (2003)]. One of the most widely used methods is the majority-rule consensus tree [Barthélemy and McMorris (1986); Margush and McMorris (1981)], which is the tree that contains the splits displayed by the majority of the input trees. Not only does this tree always exist, but it is also unique, can be efficiently constructed [Amenta et al. (2003a)], and has the property of being a *median tree* relative to the symmetric-difference distance (also known as the Robinson-Foulds distance [Pattengale et al. (2007); Robinson and Foulds (1981)]). That is, the majority-rule consensus tree is a tree whose total Robinson-Foulds distance to the input trees is minimum.

The appealing qualities of the majority-rule consensus method have made it attractive to try to extend the method to the supertree setting, while retaining as many of its good characteristics as possible. Cotton and Wilkinson (2007) were able to define two such extensions (despite some doubts about whether such an extension was possible [Goloboff and Pol (2005)]) and at least two additional ones have been studied since Dong and Fernández-Baca (2009). Here we study one of the latter variants, called *graft-refine majority-rule (+) supertrees* in Dong and Fernández-Baca (2009), and here simply referred to as *majority-rule (+) supertrees*. These supertrees

satisfy certain desirable properties with respect to what information from the input trees, in the form of splits, is displayed by them (see the Preliminaries). The key idea in this method is to expand the input trees by grafting leaves onto them to produce trees over the same leaf set. The expansion is done so as to minimize the distance from the expanded trees to their median relative to the Robinson-Foulds distance. The supertree returned is the strict consensus of the median trees with minimum distance to the expanded input trees; these median trees are called *optimal candidate supertrees*.

After showing that computing an optimal candidate supertree is NP-hard, we develop a characterization of these supertrees that allows us to formulate the problem as a polynomial-size integer linear program (ILP). We then describe an implementation that enables us to solve moderately large problems exactly. We show that, in practice, the majority-rule (+) supertree can be constructed quickly once an optimal candidate supertree has been identified. Furthermore, we observe that the supertrees produced are similar to biologically reasonable trees, adding further justification to the majority-rule (+) approach.

In addition to the exact ILP formulation, we also introduce a data reduction heuristic that identifies "reducible" sets of taxa. Informally, these are taxa that are clustered in the same way by all the input trees. By restricting the original profile to the taxa in any such set, we get a "satellite profile" that can be much smaller than the original one. At the same time, the original profile can be reduced by representing all the taxa in the set by a single supertaxon. A supertree for the original profile is obtained by solving each of these supertree problems independently and combining the answers. This approach allows us to tackle supertree problems that are well beyond the limits of the basic ILP method. Thus, whereas the latter allowed us to solve instances at most 40 taxa, the former enabled us to handle the Seabirds data set of Kennedy and Page (2002), which as 121 taxa. While the data reduction technique is not guaranteed to produce the same answers as the exact method, we present empirical evidence that it produces good results. Moreover, reducible sets often correspond to meaningful biological classification units that likely should be respected by any supertree.

There has been previous work on ILP in phylogenetics, much of it dealing with parsimony or its relative, compatibility [Brown and Harrower (2006); Gusfield (2003, 2009); Gusfield et al.

(2007); Sridhar et al. (2008)]. Our work uses some of these ideas (especially those of Gusfield et al. (2007)), but the context and the objective function are quite different. In particular, the need to handle all possible expansions of the input trees necessitates the introduction of new techniques.

Recently, McMorris and Wilkinson (2011) introduced two conservative methods, strict and loose supertrees, which generalize strict and loose consensus trees.

In a loose supertree, each split is compatible with every input tree and is supported by at least one input tree. Each split in a strict supertree must meet the same requirements as in a loose supertree, plus one additional property: Every input tree either supports the split or it has no say on it, because its leaf set does not have sufficient overlap with both sides of the split (i.e., the split is trivial in the input tree. The formal definitions of strict and loose supertrees are more complex than this, since one must be careful in expressing the notions of support and conflict in the context of partial overlap among trees. One appealing property of strict and loose supertrees is that, in a well-defined sense, they express the least that we could expect from any split-based supertree method. Thus, they offer a baseline against which other methods can be compared. Furthermore, while the requirements imposed by their formulation might seem stringent, conservative supertrees can be quite well-resolved.

We provide alternative and equivalent definitions of loose and strict supertrees, based on Robinson-Foulds distance, along the lines of those proposed for majority-rule supertrees [Cotton and Wilkinson (2007); Dong et al. (2010a)]. They rely on the idea of filling in the input trees with the taxa missing in them in such a way as to maximize the amount of agreement. There may be multiple ways to achieve this; the supertree returned is the strict consensus of all optimal solutions. Based on our formulations, we show that obtaining strict and loose conservative supertrees is NP-complete. The new definition also enables us to express supertree construction as an optimization problem based on Robinson-Foulds distance, which we can solve exactly using integer linear programming (ILP). This method can handle problems of moderate size, with at most 40 taxa and 10 trees. For larger data sets, we turn to heuristics.

The idea behind our heuristics is simple: If we were somehow able to guess any of the splits in the strict or loose supertree, we could break down the original problem into two subproblems,

one for each side of the split. Since we do not know the splits in advance, we adopt a greedy approach and seek a split that at each step optimizes a function based on Robinson-Foulds distance. The computational experience reported here shows that the downsizing effect of divide and conquer can drastically reduce the running time, allowing us to solve problems with hundreds of taxa and trees.

While the above divide-and-conquer technique performs extremely well, it is still only a heuristic. To increase our confidence in the solution, we verify each of its edges, checking to see if there is a better tree that does not include it. We also attempt multiple runs with different decompositions. Even with all these safeguards, our supertrees may not be precisely *the* strict or loose supertrees originally defined by McMorris and Wilkinson. Nevertheless, the constraints we impose guarantee that our heuristically-built supertrees are conservative. That is, every loose supertree that we generate has the property that each of its splits is supported by some tree and contradicted by none. Also, every strict supertree we generate has the property that for each input tree, each of the splits of the supertree is either trivial in the tree or is supported by the tree.

Consensus trees can be greatly affected by *rogue taxa* (sometimes called *wandering taxa*); that is taxa whose positions can vary dramatically without having a strong effect on a tree's overall score. The presence of just a few such taxa can lead to poorly-resolved consensus trees, even when there is substantial agreement relative to the remaining taxa [Redelings (2009); Wilkinson (1994); Thomson and Shaffer (2010); Sullivan and Swofford (1997); Nadler et al. (2007)]. Here we consider the problem of finding a set of leaves (i.e., possible rogue taxa) to be removed, so as to maximize the number of internal edges in the consensus tree on the reduced leaf set. Our main results are the following.

- Proofs that the underlying decision problem is NP-complete for three trees in the strict consensus case and for four trees in the case of majority rule trees.

- A polynomial-time algorithm for reduced strict consensus when the maximum degree of the strict consensus tree of the original profile of trees is fixed.

- An integer linear programming (ILP) formulation for obtaining exact solutions for reduced

strict, majority and loose consensus.

- An experimental comparison with the heuristic proposed in Pattengale et al. (2011), showing that in several cases our reduced consensus formulations allow us to uncover increased common phylogenetic information without discarding many more leaves.

My contribution is mainly on the last two items. It is part of a paper accepted by ISBRA 2012 [Deepak et al. (2012)].

CHAPTER 2. PRELIMINARIES

Our terminology largely follows [Semple and Steel (2003)]. A *phylogenetic tree* is an unrooted leaf-labeled tree where every internal node has degree at least three. We will use "tree" and "phylogenetic tree" interchangeably. The leaf set of a tree T is denoted by $\mathcal{L}(T)$. Each leaf is called a *taxon* (plural, *taxa*).

A *profile* is a tuple of trees $P = (t_1, \ldots, t_k)$. Each t_i in P is called an *input tree*. Let $\mathcal{L}(P) = \bigcup_{i \in K} \mathcal{L}(t_i)$, where K denotes the set $\{1, \ldots, k\}$. An input tree t_i is *plenary* if $\mathcal{L}(t_i) = \mathcal{L}(P)$. Tree T is a *supertree* for profile P if $\mathcal{L}(T) = \mathcal{L}(P)$.

A *split* is a bipartition of a set. The split whose parts are A and B is denoted $A|B$. The order here does not matter, so $A|B$ is the same as $B|A$. Split $A|B$ is *nontrivial* if each of A and B has at least two elements; otherwise it is *trivial*. Split $A|B$ *extends* another split $C|D$ if $A \supseteq C$ and $B \supseteq D$, or $A \supseteq D$ and $B \supseteq C$.

Tree T *displays* split $A|B$ if there is an edge in T whose removal gives trees T_1 and T_2 such that $A \subseteq \mathcal{L}(T_1)$ and $B \subseteq \mathcal{L}(T_2)$. A split $A|B$ is *full* with respect to a tree T if $A \cup B = \mathcal{L}(T)$. $A|B$ is *partial* with respect to T if $A \cup B \subset \mathcal{L}(T)$. Split $A|B$ is *plenary* with respect to a profile P if $A \cup B = \mathcal{L}(P)$.

The set of all nontrivial full splits displayed by T is denoted $\mathit{Spl}(T)$. It is well known that the full splits of T uniquely identify T [Semple and Steel (2003)]. Thus, we will often view trees as sets of splits and write "$A|B \in T$" if tree T displays split $A|B$. A tree T with $\mathit{Spl}(T) = \emptyset$ is called a *fan*.

Let $S \subseteq \mathcal{L}(T)$. The *restriction of T to S*, denoted $T|S$, is the phylogenetic tree with leaf set S such that

$$\mathit{Spl}(T|S) = \{A \cap S | B \cap S : A|B \in \mathit{Spl}(T) \text{ and } |A \cap S|, |B \cap S| > 1\}.$$

Let T' be a phylogenetic tree such that $S = \mathcal{L}(T') \subseteq \mathcal{L}(T)$. Then, T *displays* T' if $Spl(T') \subseteq Spl(T|S)$. Trees T_1 and T_2 are *compatible* if there exists a tree T such that T displays T_1 and T_2.

A set of splits is *compatible* if there is a tree T that displays them all. Tree T is *compatible with* a set of splits $\mathcal{X}$ if there is a tree T' that displays T and $\mathcal{X}$.

Let t be an input tree in a profile P, T be a supertree for P, and $A|B \in T$. Let $A' = A \cap \mathcal{L}(t)$ and $B' = B \cap \mathcal{L}(t)$. We say that t *supports* $A|B$ if $A'|B' \in t$; t is *in conflict with* $A|B$ if $A'|B'$ is incompatible with t. If t neither supports nor is in conflict with $A|B$, t is *irrelevant* to $A|B$.

Let T_1 and T_2 be two phylogenetic trees over the same leaf set. The *symmetric-difference distance*, also known as *Robinson-Foulds distance* [Robinson and Foulds (1981)], between T_1 and T_2, denoted $d(T_1, T_2)$, is defined as

$$d(T_1, T_2) = |(Spl(T_1) \setminus Spl(T_2)) \cup (Spl(T_2) \setminus Spl(T_1))| . \tag{2.1}$$

The *asymmetric difference* from T_1 to T_2, denoted $AD(T_1, T_2)$, is defined as

$$AD(T_1, T_2) = |(Spl(T_1) \setminus Spl(T_2))| \tag{2.2}$$

The *majority splits* in a profile $P = (t_1, \ldots, t_k)$ are the splits displayed by more than $\frac{k}{2}$ of the input trees. A *majority plenary split* is a plenary split that is also a majority split. Similarly, a *majority partial split* is a partial split that is also a majority split.

Rooted phylogenetic trees can be viewed as a special case of unrooted trees. That is, we can view a profile of rooted trees as unrooted trees, all of which have a common taxon called the root. Thus, in a split in a rooted tree, one of the two parts must contain the root; the part that does not contain it is called a *cluster* (or *clade*, or *monophyletic group*). All of the above concepts (e,g., compatibility and distance), as well as those introduced in the rest of this paper, directly apply to rooted trees.

The *consensus problem* is the special case of the supertree problem where the profile $P = (T_1, \ldots, T_k)$ consists of trees that have the same leaf set. The *strict consensus* of P, denoted $Str(P)$ is the tree that displays exactly the full splits that are in every tree in the profile. The *loose consensus* of P, denoted $Loose(P)$, is the tree that displays exactly the full splits that are

in some tree of P and compatible with every tree in P. The *majority-rule consensus tree* of P, denoted $Maj(P)$, is the tree that displays every full split that is in the majority of the trees in P [Margush and McMorris (1981)].

For any phylogeny T with $\mathcal{L}(T) = \mathcal{L}(P)$, define the *distance* from T to P as $dist(T, P) = \sum_{i=1}^{k} d(T, T_i)$, where d denotes the symmetric-difference distance. Any T with leaf set $\mathcal{L}(P)$ that minimizes $dist(T, P)$ is called a *median tree* for P. It is known that $Maj(P)$ is a median tree for P; indeed, it follows from [Barthélemy and McMorris (1986)] that $Maj(P)$ is the strict consensus of the median trees for P. The (median) *score* of P is defined as $\text{score}(P) = \min_{T:\mathcal{L}(T)=\mathcal{L}(P)} dist(T, P)$. Thus, $s(P) = dist(Maj(P), P)$.

For the supertree problem, we introduce two different notions of the *span* of a tree. Let t be a tree in a profile P. The *graft-only span* of t is the set of trees

$$\langle t \rangle_g = \{T : T | \mathcal{L}(t) = t \text{ and } \mathcal{L}(T) = \mathcal{L}(P)\}.$$

The *graft/refine span* of t is the set of trees

$$\langle t \rangle_{gr} = \{T : T \text{ displays } t \text{ and } \mathcal{L}(T) = \mathcal{L}(P)\}.$$

Observe that $\langle t \rangle_g \subseteq \langle t \rangle_{gr}$, since, as their names suggest, the graft-refine span allows refinement, whereas the graft-only span does not. In particular, in the consensus setting (where $\mathcal{L}(t) = \mathcal{L}(P)$), $\langle t \rangle_g = \{t\}$, while $\langle t \rangle_{gr}$ includes all the refinements of t. The *graft-only span* of a profile P is $\langle P \rangle_g = (T_1, \ldots, T_k)$ where $T_i \in \langle t \rangle_g$ for every i. The *graft/refine span* of a profile P is $\langle P \rangle_{gr} = (T_1, \ldots, T_k)$ where $T_i \in \langle t \rangle_{gr}$ for every i.

A node is *internal* if it is not a leaf node. An edge is *internal* if it does not connect to a leaf. If a phylogeny T is rooted, we require that each internal node have at least two children; if T is unrooted, every internal node is required to have degree at least three. Here, we limit our discussion to collections of trees over the same set of taxa. Without loss of generality, we can view such trees as rooted, since, when the trees are unrooted, we can arbitrarily pick any taxon as an out-group, which is equivalent to fixing a common root [Semple and Steel (2003); Amenta et al. (2003b)].

Let C denote some consensus method. Given a collection of trees P, the *reduced C consensus problem* is to find a set $X \subseteq \mathcal{L}_P$ that maximizes the number of internal edges of $C(P|X)$).

CHAPTER 3. REVIEW OF THE LITERATURE

More than a quarter of century ago, Gordon (1986) introduced the concept of supertrees. Since then, a variety of supertree construction methods have been proposed; these have been surveyed by Eulenstein (2005), Bininda-Emonds (2004) and Scornavacca (2009).

When reviewing supertree methods, we discuss the following categories since they are either representative of widely used methods or related to the new methods that are developed in this dissertation.

1. Supertree methods related to BUILD algorithm
2. Matrix representation-based methods
3. Median supertrees
4. Supertrees with conservative properties

But before jumping into supertree methods, we need to mention some of the famous consensus methods, which are special cases of supertree methods when all the input trees have the same leaf sets.

3.1 Consensus Trees

When all the trees in a profile have identical leaves, the supertree problem becomes a consensus problem. Familiar examples of consensus methods are the Adams consensus [Adams III (1972)], strict consensus [Sokal and Rohlf (1981)], majority-rule consensus [Margush and McMorris (1981)], and loose consensus [Meacham (1982); Barthélemy et al. (1992); Bryant (2003)] (also known as combinable component [Bremer (1990)] or semi-strict consensus [Swofford

(1991)]). Supertrees extending from their respective consensus methods are attractive because we might be able to extend the nice properties of the methods in the supertree setting.

Consensus methods based on splits (majority-rule, strict, and loose consensus) can often be reformulated as problems of finding the strict consensus of all trees of a certain kind that minimize a score function based on the Robinson-Foulds distance to the input trees. For example, Barthélemy and McMorris (1986) showed that majority rule consensus trees are the strict consensus of all trees at minimum total distance from the input trees.

3.2 Supertree Methods Related to the BUILD Algorithm

These supertree methods encode triplets in the source trees in a graph known as the Aho graph. They are only for rooted trees and are efficient algorithms based on greedy handling of conflicts.

3.2.1 The BUILD Algorithm

The BUILD algorithm is proposed by Aho et al. (1981). It is an algorithm for constructing a tree to satisfy a set of lineage constraints on common ancestors. In a rooted trees, the lowest common ancestor of two taxa nodes x and y is denoted as (x, y). A set of constraints are of the form $(i, j) < (k, l)$ where $i \neq j$ and $k \neq l$, meaning that the lowest common ancestor of i and j is a proper descendant of the lowest common ancestor of k and l. The central idea is to determine for a potential tree T the sets of leaves that are descendants of each child of the root of T. Name these sets $S_1, \ldots, S_r$. There are two conditions that these sets must satisfy for each constraint $(i, j) < (k, l)$.

(1) i and j must be in the same set. Otherwise (i, j) is the root of T, and the root cannot be a proper descendant of (k, l).

(2) Either k and l are in different sets, or i, j, k and l are all together in one set. Otherwise (i, j) cannot be a proper descendent of (k, l).

These conditions are also sufficient. Thus, if we can partition the nodes into two or more sets satisfying them, and if we can recursively build trees from each set, then a tree exists; otherwise,

no trees exist. Given a set of constraints C, we define a partition π_c on the leaves $1, 2, \ldots, n$ using the following rules:

(1) If $(i, j) < (k, l)$ is a constraint, then i and j are placed in one block of π_c.

(2) If $(i, j) < (k, l)$ is a constraint, and k and l are in one block, then i,j,k and l are placed in the same block.

(3) No two leaves are in the same block of π_c unless it follows from (1) and (2).

When the BUILD algorithm runs, it first computes $\pi_c = S_1, \ldots, S_r$ based on C. If there is only one S_1 in π_c, BUILD is a null tree. If not, in a loop from 1 to r, it creates $C_m = \{(i, j) < (k, l)$ in C and i, j, k, l are in $S_m\}$, so that it recursively applies BUILD on S_m and C_m. It is this C_m that greatly reduces the number of constraints involved.

Other than its extremely simplicity, it is a fast algorithm. Its running time is highly dependent on the method used to partition the set of constraints. By imposing the restriction that all constraints be of the form $(i, j) < (i, k)$, the running time is $O(n^2)$. When all constraints are of the form $(i, j) < (i, k)$, rule (2) of the needs never be applied explicitly. Accordingly it suffices to implement rule (1) alone.

If we take each leaf appearing in one or more constraints to be a node, and each constraint $(i, j) < (i, k)$ to represent an edge between i and j, then we shall have a multigraph whose connected components represent the blocks of the partition that we are looking for. This is the famous Aho graph.

Since every triplet $ab|c$ is equivalent to $(a, b) < (a, c)$. Aho graph can be used to represent the relationships between a set of triplets.

BUILD provides an elegant base for all these methods originating from it. It is a yes-or-no algorithm that tells whether a collection of triplets R on a leaf set $\mathcal{L}$ is compatible or not. However, in face of incompatible triplets, it provides a null tree, which is unsatisfactory. Also it works strictly with triplets, while some input tree could have fans for three taxa in it.

3.2.2 The SUPERB, OneTree, AllTrees, and AllMinTrees Algorithms

The trees returned by BUILD are not necessarily binary. Constantinescu and Sankoff (1995) developed an efficient algorithm SUPERB based on BUILD to generate a set of rooted binary trees. Given k rooted binary trees $(t_1, \ldots, t_k)$, a unique system of lineage constraints on common ancestors is generated. Then SUPERB constructs the set of all rooted binary trees that are compatible with all of $(t_1, \ldots, t_k)$. The running time to obtain one such supertree is $O(k^2n^2)$, where n is the number of distinct leaves in all of the trees. Similar to BUILD, SUPERB could also provide a null tree.

The OneTree algorithm, proposed by Ng and Wormald (1996), is based on the BUILD algorithm, but it includes both triplets and fans. They combine all the triples and fans to build output trees that are compatible with them. Moreover, they produce all trees compatible with a given input with an AllTrees algorithm. The running times of their algorithms are polynomial in the size of the input for OneTree, and polynomial in the size of the output for AllTrees. The output size of AllTrees may be exponential in the size of the input.

The OneTree algorithm is simplified by Bryant (1997) when he removes the fans. For a profile P it runs in $O(mn)$ time where n is the number of taxa in P and m is the number of rooted triples in R, the triplet set of P. Some faster implementations were developed by Henzinger et al. (1996) and Berry and Semple (2006).

There could be more than one tree displaying a compatible triplet set R. Moreover, the number of qualifying trees may be exponential in the size of R. Hence it is important to find minimal trees with this property. A tree T being *minimal* means that no internal edge of T can be contracted so that the resulting tree displays R. Semple (2003) presents the method AllMinTrees that returns all trees that display R and are minimal.

The methods so far cannot handle the incompatible triplets. Thus how to handle incompatibility yields different methods. Among them, the most famous one is the MinCut supertrees.

3.2.3 MinCut (MC) Supertrees

The MinCut supertree algorithm [Semple and Steel (2000)] modifies OneTree so that it always returns a tree. Their method essentially is to cut connected components in a greedy way to enable the continuation of the BUILD algorithm. However, more arrangements are made in order to preserve some nice properties. The key is that when the Aho graph is a connected weighted graph, first, a second weighted "collapsed" Aho graph is created to preserve the edges whose triplets are in every trees, then, we find a minimum-weight cut set to disconnect the "collapsed" graph. At the subtree level, we recursively apply the cuts when necessary. The weight on each edge (a, b) is defined as the sum of weights of the trees such that the tree has at least one triplet $ab|c$. Obviously this is a greedy approach.

The MC supertree has some nice properties. First of all, if the profile P is compatible, it is the same as the OneTree and thus displays all the triplets in P. Moreover, it returns a tree displaying all nestings and triplets shared by all input trees in P. If all trees display a common induced subtree, then MC supertree displays it too. Also it is quick to compute.

However, it can be sensitive to the size of the input trees, favoring the resolutions contained in the biggest trees. Moreover, it can fail to include information that is not contradicted in the set of input trees. It is this last point that prompts Page to provide his own modified version.

3.2.4 Modified-MinCut (MMC) Supertrees

Page (2002) pointed out that the MC supertree neither maximizes the uncontradicted information nor minimizes the information that contradicts the source trees. To avoid those drawbacks, he proposed a modification, called the Modified-MinCut (MMC) supertree method. The MMC supertree method is a heuristic to avoid as much as possible contradicted information, having as consequence to permit to insert more uncontradicted information in the supertree than the MC method, while still returning a tree that has all the properties of the MC supertree.

Page distinguishes edges in his special graph into three different types: unanimous, uncontradicted and contradicted. The aim was to modify the "collapsed" graph used in MC such as to minimize the number of uncontradicted edges that are cut in the MC method. Thus,

Page extended Semple and Steel's approach of merging nodes linked by unanimous edges to include nodes linked by uncontradicted edges. Then he removes all contradicted edges. If this way we disconnect the "collapsed" graph, all uncontradicted edges are preserved at this step. Otherwise, at least one uncontradicted edge must be cut to disconnect the graph. Hence, he builds the set all edges in adjacent to a contradicted edge and remove them. If the graph is disconnected, he can proceed.

Consequently, MMC supertree contains much more uncontradicted triplets than MC tree one. However, both MC and MMC supertrees can contain clusters not present in any source tree.

Page's inclusion of uncontradicted information far exceeds its original application on MC. Indeed it evolves into a common practice nowadays that emphasizes the enrichment of the backbone trees with uncontradicted information so that the final tree agrees with the input trees and keeps as much information as possible. It is this "filler" part that makes trees usable to biologists.

3.2.5 BUILD-WITH-DISTANCES

Most supertree methods make use only of the tree topology. Yet many methods of building phylogenetic trees create trees in which each edge has a numerical branch length. Typically the branch length estimates the rate of DNA mutation along the edge. A rooted tree in which each edge has a branch length is an *additive tree*.

These distances carry a great deal of information about phylogeny. When only topology information exists, there are typically many different topologies compatible with the data. In contrast, when the branch lengths are used, the number of topologies is often more highly constrained.

Willson (2004) proposed a polynomial-time method BUILD-WITH-DISTANCES which uses input distance information to construct a supertree when an additive supertree exists. This algorithm generalizes the BUILD algorithm. When an additive supertree exists, BUILD-WITH-DISTANCES finds a supertree (although not necessarily an additive supertree), here called the "null threshold tree". Moreover, when the algorithm outputs a tree, the null threshold tree

will be a supertree. It is proved in this situation that the null threshold tree contains at least as much resolution as the supertree obtained by BUILD. When the algorithm fails to produce a supertree, a polynomial time modification is also proposed which outputs a tree called the "minimal threshold tree" with interesting properties.

The idea of using branch lengths to greatly limit the search space is applicable well beyond BUILD-WITH-DISTANCES. As in any physical science and engineering fields, the use of physical meaning and engineering intuition is very important not only in making educated guesses on the forms of solutions, but also in discarding meaningless answers that result from pure mathematical operations.

Given the current topology-only formulations, the phylogenetic problems are mostly hard. The search space for tree is simply too large. The fundamental limitations of the current exact solutions and heuristics cannot be resolved in the near future. Nevertheless, biologists want quick answer to their specific problems, and they can provide some domain knowledge. Maybe in order to break the stalemate, we need to add more domain knowledge into the phylogenetic problem formulation to limit the search space and to seek specific solutions that are realistic and fast to obtain. To that direction, BUILD-WITH-DISTANCES is a true trailblazer.

3.3 Matrix Representation-based Methods

These supertree methods convert input trees into matrices of binary sequences or distances, and the matrices are subsequently analyzed using a standard phylogenetic tree reconstruction method.

3.3.1 Matrix Representation with Parsimony (MRP)

We cannot discuss supertree construction without referring to matrix representation with parsimony (MRP) [Baum and Ragan (2004)], by far the most commonly used method, but also one of the most criticized. It is essentially an *ad hoc* adaptation of parsimony to the supertree setting. It has been independently developed by Baum (1992) and Ragan (1992).

Given a profile P, the MRP method first encodes it into a binary matrix. Row for species and column for splits of input trees. Then, a parsimony analysis is performed on the matrix.

The details are as follows. First, Baum (1992) suggested each tree of the input forest is rooted by using a taxon common to all input trees. If a tree is already rooted, re-root it. Ragan (1992) suggested to root them by an all-zero output. Then, all the splits of each tree, except for those trivial splits that have one taxon in one of its blocks, are put into the matrix. The block without the root is encoded with 1 while the block with the root is encoded with '0'. All taxa that do not belong to the tree are encoded by a ?. The matrix then is analyzed by the parsimony criterion. Finally the strict consensus of the most parsimonious trees is returned.

Foulds and Graham (1982) showed that finding the most parsimonious trees given a character matrix is an NP-complete problem, and heuristics have been proposed.

MRP is a widely used but also highly controversial method. Among the arguments there are

- It lacks of an underlying model.
- It can propose clusters not supported by any combination of input trees, when source trees conflict.
- Clusters contradicted by every input tree can be present. It even happens in a consensus setting.
- It can also fail to display some triplets common to every input tree, even in the consensus setting.
- It is biased and some tree topologies can unduly affect the MRP supertree. It may also favor source trees that are more unbalanced.
- The reason of its bias could be due to the different sizes of input trees, or due to different sizes of the trees in the region of conflict. It could also be that the information given by the matrix columns is not independent.

Given so many shortcomings, it is obvious that MRP is a method of convenience with many side effects. It should be used with caution. On one hand, maybe the widespread acceptance of the philosophy underlying parsimony and the availability of excellent parsimony software brings

its popularity. On the other hand, the lack of better competitors that are sound in principle and efficient in execution seems another reason. More importantly, new and better methods should be developed that incorporate the properties that we believe are nice and important, if theoretically possible.

3.3.2 Matrix Representation with Flipping (MRF)

The Matrix Representation with Flipping (MRF) method was developed by our group [Chen et al. (2003); Eulenstein et al. (2004)]. The idea is that input trees conflict because of errors. Some clusters have incorrect taxa, some need correct taxa. Such errors correspond to flips from 0 to 1 or 1 to 0. The flip problem is to find the minimum number of flips needed to resolve all incompatibilities, where each flip moves a taxon into or out of a cluster. The resulting clusters forms a flip supertree. If more than one such supertree exists, their loose consensus is the MRF supertree.

It is shown that the decision version of the minimum-flip is NP-complete. The MRF supertrees preserve strict, but not Adams or majority-rule consensus. In fact MRF consensus trees preserves loose consensus. However, preserving loose consensus for supertrees is not as important as preserving strict consensus. The authors showed simulation studies for which the MRF supertrees are at least as accurate as supertrees built with MRP. Unfortunately, as MRP, Goloboff (2005) showed that this method can propose new clusters contradicted by every input trees.

3.3.3 Matrix Representation with Compatibility (MRC)

The Matrix Representation using Compatibility (MRC) supertree is developed by Rodrigo (1996) and Ross and Rodrigo (2004). It is based on sound principle. However, since its incidence, MRC has received scant attention while MRP has become a popular technique. MRC identifies the largest set of mutually compatible splits (maximum clique) in the MR matrix. The supertree can be determined directly from this clique, without recourse to arguments involving parsimony. Identifying a maximum clique is NP-hard. Note that in the consensus case, pairwise compatibility is equivalent to compatibility. However, in a supertree setting, pairwise

compatibility does not guarantee compatibility. When computing the pairwise compatibility of two matrix columns, rows involving one or two missing entries are ignored. Experiments were conducted to compare the powers of MRC and MRP to construct a supertree reliably by simulating sets of consistent and inconsistent sample trees derived from an original model tree. Under stringent definitions of success, MRP and MRC were successful . Overall MRP is a little better in recovering the original tree.

Goloboff (2005) showed that MRC can also propose new clusters contradicted by each of the input trees.

3.4 Median Supertrees

The median supertree minimizes the sum of distances to the source trees. Using a median tree, like an average value, is a way to summarize the profile. Depending on how distances are defined, there are different formulations.

3.4.1 Majority-rule Supertrees

Cotton and Wilkinson (2007) extended the majority-rule consensus method to the supertree setting. They defined the majority-rule (-) supertree and the majority-rule (+) supertree. A median (-) supertree for a profile P of trees is the supertree T that minimizes $\sum_i dist(T|L_i, T_i)$ over all supertrees for P, where the distance *dist* is the Robinson-Foulds or symmetric difference distance. The Maj^- supertree is the strict consensus of all median (-) supertrees. The definition is very simple. However, finding a median (-) supertree is an NP-hard problem, proven by Bryant (1997).

Define the binary supertree span of an input tree t, denoted by $\langle t \rangle$, to be the set of binary trees on $\mathcal{L}(P)$ that display t. A representative selection for a profile $P = (t_1, \ldots, t_k)$ is a k-tuple $R = (T_1, \ldots, T_k)$, where $T_i \in \langle t_i \rangle$. The median score of R, denoted by $s(R)$, is defined as: $s(R) = \min_T \sum_{T_i \in R} dist(T_i, T)$, where T is a supertree with leaf set $\mathcal{L}(P)$. The candidate supertree of R, denoted by T_R, is the majority-rule consensus tree for R. Note that T_R minimizes $s(R)$ as the majority-rule consensus tree is a median tree. An optimal candidate supertree is the candidate supertree T_R of any representative selection $R = (T_1, \ldots, T_k)$ for P that has the

smallest possible median score $s(R)$. The Maj^+ supertree is the strict consensus of all optimal candidate trees. The main drawback is that it requires the enumeration of an exponential number of representative selections R, given a profile P.

A split is full with respect to a tree T if its leaf set is $\mathcal{L}(T)$, otherwise it is partial. A split is said to be plenary with respect to a profile P if its leaf set is $\mathcal{L}(P)$. A split is a majority split if it is displayed by a majority of the input trees. A split $A|B$ extends another split $C|D$ if $A \supseteq C$ and $B \supseteq D$, or $A \supseteq D$ and $B \supseteq C$. Cotton and Wilkinson (2007) conjectured that, for each profile P, majority-rule supertrees T had the following desirable properties:

CW1 All majority plenary splits in P are in T.

CW2 T is compatible with each majority partial split in P.

CW3 All splits in T are compatible with a majority of the trees in P.

CW4 Every plenary split in T extends at least one input tree full split.

3.4.2 Followups of Majority-rule Supertrees

Dong and Fernández-Baca (2009) showed that Maj^- satisfies CW1 and CW4 while Maj^+ supertrees satisfy CW1, CW2 and CW3. Moreover, Maj^-, in the consensus setting, is the majority-rule consensus, while Maj^+ is not. Two variants of the Maj^+ supertree method, i.e., the majority-rule $(+)_s$ tree (or $(+)_{gr}$ in later chapters) and the majority-rule $(+)_g$ tree, are proposed. Both of them satisfy all conjectured properties. The majority-rule $(+)_g$ supertree method is equivalent to majority-rule consensus in the consensus setting, while the majority-rule $(+)_{gr}$ supertree is not.

Several followup work of majority-rule supertrees are done. First, Dong et al. (2010b) studies the characteristics of $(+)_{gr}$ consensus trees. Among the results, we will completely determine the internal cluster properties of consensus trees constructed by this method and give an analog to the characterization of median trees in majority-rule consensus trees. The relationship between this method and other consensus methods are revealed. Second, axioms that characterize a new majority-rule $(+)$ consensus function are developed by Dong et al.

(2011). In addition, two other related consensus functions are characterized. Third, Dong et al. (2010a) studied a variant of one of Cotton and Wilkinsons methods, called majority-rule (+) supertrees. After proving that a key underlying problem for constructing majority-rule (+) supertrees is NP-hard, we develop a polynomial-size exact integer linear programming formulation of the problem. We then present a data reduction heuristic that identifies smaller subproblems that can be solved independently. While this technique is not guaranteed to produce optimal solutions, it can achieve substantial problem-size reduction. Finally, we report on a computational study of our approach on various real data sets, including the 121-taxon, 7-tree Seabirds data set of Kennedy and Page (2002). It indicates that our exact method is computationally feasible for moderately large inputs. For larger inputs, our data reduction heuristic makes it feasible to tackle problems that are well beyond the range of the basic integer programming approach. Comparisons between the results obtained by our heuristic and exact solutions indicate that the heuristic produces good answers. Our results also suggest that the majority-rule (+) approach, in both its basic form and with data reduction, yields biologically meaningful phylogenies.

3.4.3 Robinson-Foulds Supertrees

Bansal et al. (2010) introduced efficient, local search based, hill-climbing heuristics for the intrinsically hard binary median tree problem on rooted trees. The Robinson-Foulds distance is the same distance used to in Maj^{-} supertrees. The only difference is that RF supertrees must be binary median tree while the median trees in Maj^{-} can be non-binary. These heuristics use novel non-trivial algorithms for the SPR and TBR local search problems which improve on the time complexity of the best known solutions by a factor of $\Theta(n)$ and $\Theta(n^2)$ respectively, where n is the number of taxa of the profile. Experiment comparison with MRP and the triplet supertree method using four supertree data sets showed fast estimates and more retained information in all data sets.

Chaudhary et al. (2011) developed the unrooted version of the Robinson-Foulds supertree. Indeed, for technical reasons, both the Robinson-Foulds supertrees are required to be fully resolved, which guarantees many unsupported splits.

3.5 Supertrees with Conservative Properties

Up to now, the most closely related conservative supertree methods are the PhySIC method developed by Ranwez et al. (2007),PhySIC_IST by Scornavacca et al. (2008), and the conservative supertrees developed by McMorris and Wilkinson (2011).

3.5.1 PI and PC

Scornavacca (2009) explained in detail the underlying conservative criteria used in PhySIC and PhySIC_IST as well as the optimization objectives and the algorithms. Any rooted tree T can be equivalently described by the set of triplets, denoted by $R(T)$. Given a collection P of rooted phylogenetic trees, $R(P)$ denotes the set of triplets present in at least one tree of P. Given a triplet t, $\bar{t}$ denotes any of the two other triplets on the same set of leaves. Given a compatible set R of triplets, R *induces* a triplet t, denoted by $R \vdash t$, if and only if $R \cup \{\bar{t}\}$ is not compatible, or equivalently if any tree T that displays R contains t. In case of an incompatible set of triplets R, we say that a set R of triplets *induces* a triplet t when there is a compatible subset R' of R that induces t. Given a collection P of input trees and a candidate supertree T, $R(T, P)$ denotes the set of triplets of P for which T proposes a resolution. More formally, $R(T, P) = .\{ab|c \in R(P)$ such that $\{ab|c, ac|b, bc|a\} \cap R(T) \neq \emptyset\}$. The set $R(T, P)$ corresponds to all topological information present in P that is related to the information present in supertree T. We can express the induction property PI and the non-contradiction property PC as follows:

- T satisfies PI for P if and only if for all $t \in R(T)$, it holds that $R(T, P) \vdash t$. In other words, PI requires that each and every triplet of T is induced by $R(T, P)$.
- T satisfies PC for P if and only if for all $t \in R(T)$ and all $\bar{t}$, it holds that $R(T, P) \nvdash \bar{t}$, i.e., for each and every triplet of T, $R(T, P)$ induces no alternative resolution.

3.5.2 PhySIC

PhySIC stands for Phylogenetic Signal with Induction and non-Contradiction. It tries to infer supertrees that satisfy PI and PC and resolve as many triplets as possible. The optimization problem is as follows: Given a profile P of rooted trees, find output a supertree

T such that T satisfies PI and PC for P and $|R(T,P)|$ is maximum among the trees satisfying PI and PC. The problem is conjectured to be hard.

PhySIC takes two steps. Given a profile of rooted trees P, a supertree T_{PC} satisfying PC for P is first computed by $PhySIC_{PC}$. Then some branches of T_{PC} are collapsed by $PhySIC_{PI}$ until the modified T_{PC} satisfies PI. $PhySIC$ runs in $O(kn^3 + n^4)$ time where n is the number of taxa and k is the number of trees.

3.5.3 PhySIC_IST

To get a more refined supertree, one way is to reduce the number of (rogue) taxa studied in the profile. Thus, PhySIC_IST (PHYlogenetic Signal with Induction and non-Contradiction Inserting a Subset of Taxa) is proposed to infer non-plenary supertrees, i.e. supertrees without rogue taxa. PhySIC_IST aims at inferring supertrees that satisfy the same appealing theoretical properties as with PhySIC, while being as informative as possible under this constraint.

The information in a supertree is estimated using a variation of the Cladistic Information Content (CIC) criterion, that includes the effects of the number of multifurcations as well as the number of missing taxa. The input is a collection P of rooted trees. Its output a supertree T (plenary or not) such that T satisfies PI and PC for P and CIC is maximum among the trees satisfying PI and PC. So it is a change of the optimization objective as well as algorithm. This problem is conjectured to be hard, too.

PhySIC_IST is a polynomial-time heuristics. It is heuristics only on finding maximum CIC, since it always outputs a supertree satisfying PI and PC. The algorithm operates successive insertions of taxa on a backbone topology. Since it is a greedy algorithm, the order of the insertions is chosen carefully. Once a taxon is inserted, its presence in the supertree is questioned any more. It is therefore preferable to first insert the taxa with a strong and unambiguous signal. The first taxa inserted are thus those present in as many source trees as possible and involved in as few contradictions as possible. Indeed a priority order is established for insertion.

Additionally, a statistical preprocessing step called STC (Source Trees Correction) is proposed to correct the source trees prior to the supertree inference. STC is a liberal step that removes the parts of each source tree that significantly conflict with other source trees. Combin-

ing STC with a veto method allows an explicit trade-off between veto and liberal approaches, tuned by a single parameter.

Large-scale simulations show that the combination of STC and PhySIC_IST infers much more informative supertrees than PhySIC, while preserving low type I error compared to the well-known MRP method. Two biological case studies on animals confirm that the STC pre-process successfully detects anomalies in the source trees while the combination of STC and PhySIC_IST provides well-resolved supertrees agreeing with current knowledge in systematics.

3.5.4 Conservative Supertrees

Recently, McMorris and Wilkinson (2011) introduced two conservative methods, strict and loose supertrees, which generalize strict and loose consensus trees. The detailed formulation of conservative supertrees are left in later chapters. The exact relationship between PhySIC and conservative supertree is still an open question. My recent study shows that PC in triplets is likely to be equivalent to compatibility of splits.

3.6 Rogue Taxa

There is a sizable literature on identifying rogue taxa [Cranston and Rannala (2007); Pattengale et al. (2011); Thomson and Shaffer (2010); Wilkinson (1996, 1995, 1994)]. Agreement and frequent subtree approaches — where one seeks induced subtree common to all or some fraction of the input trees — have been suggested by some. For example, Cranston and Rannala use it to summarize the posterior distribution of a collection of trees resulting from Bayesian analysis [Cranston and Rannala (2007)]. The idea is, of course, that since the placement of rogue taxa varies widely from tree to tree, they will therefore be excluded from the agreement subtrees. There are several papers on computing agreement subtrees (e.g., [Finden and Gordon (1985); Amir and Keselman (1994); Farach et al. (1995); Lee et al. (2005); Swenson et al. (2011)]) and frequent subtrees (e.g., [Chi et al. (2004); Xiao and Yao (2003); Zaki (2005); Zhang and Wang (2008)]); however, agreement-based approaches can tend to be more conservative than consensus trees. Further, the underlying optimization problems are NP-hard [Amir and Keselman (1994); Pattengale et al. (2011)].

Wilkinson appears to have been the first to propose the use of reduced majority rule and strict consensus as a means to identify rogue taxa [Wilkinson (1994, 1995, 1996)]. While the underlying principle behind all proposed reduced consensus methods is to gain internal edges at the expense of dropping leaves [Redelings (2009)], the precise cost measure can vary. The criterion use here is to maximize the resolution of the consensus tree, without taking into account how many leaves are eliminated. In contrast, Pattengale et al. (2011) propose using an objective function that is a weighted sum of the total number of leaves retained and the number of clusters (actually, bipartitions) obtained. This problem is shown NP-hard for strict and majority-rule consensus, when the weight assigned to the leaves is zero. To our knowledge, the complexity of the problem in the general case has not been determined and is conjectured hard. Indeed, Pattengale et al. only offer a heuristic for it.

Compared with various supertree methods, the rogue taxa problem seems much simpler. The input trees are known and on the same leaf set. The difficult part is which taxa to remove to achieve various optimization goals. Since for n taxa there are 2^n brutal force choices, these problems are NP complete. Except for very small number of taxa, exhaustive search or exact solution have not been tried before our work based on integer linear programming. The solutions available are all based on heuristics.

The use of integer linear programming in phylogenetics appears to be relatively new [Gusfield et al. (2007); Sridhar et al. (2008)]. The formulations presented here are related to the author's previous work on constructing majority-rule supertrees and conservative supertrees [Dong et al. (2010a); Dong and Fernández-Baca (2011)]. Nevertheless, there are some important differences with that work. The main difference is that for the rogue taxa problem, the key variables are the selection variables for taxa, while for the supertree problems, especially for the (+) route, the key variables are the filling variables. While our ILP formulations can only be used for relatively small data sets, they offer a valuable benchmark against which to compare heuristics. They have also allowed us to identify certain interesting characteristics and limitations of the various consensus methods. In particular, our experiments suggest that assigning equal weight to the number clusters and the number of taxa in the consensus tree might be too conservative an approach.

CHAPTER 4. MAJORITY-RULE (+) SUPERTREES

In this chapter, we describe a variant (suggested by William H. E. Day) of the majority-rule (+) supertree.

4.1 Definition

The *span* of an input tree t, denoted by $\langle t \rangle_{gr}$, is the set of all trees on $\mathcal{L}(P)$ that display t. The *span* of a profile $P = (t_1, \ldots, t_k)$, denoted $\langle P \rangle_{gr}$, is the set of all k-tuples $R = (T_1, \ldots, T_k)$, where $T_i \in \langle t_i \rangle_{gr}$ for each $i \in K$ where K denotes the set $\{1, \ldots, k\}$. Each $R \in \langle P \rangle_{gr}$ is called a *representative selection* for P and $Maj(R)$ is called a *candidate supertree*. An *optimal representative selection* is a representative selection R with minimum score $s(R)$ over all $R \in \langle P \rangle_{gr}$. We refer to $Maj(R)$ as the *optimal candidate supertree* associated with R. The *majority-rule (+) supertree* of profile P, denoted by $Maj^{+}(P)$, is the strict consensus of all the optimal candidate supertrees.

Dong and Fernández-Baca (2009) have shown that $Maj^{+}(P)$ satisfies the following appealing properties (originally conjectured by Cotton and Wilkinson).

(CW1) $Maj^{+}(P)$ displays all of the majority plenary splits in P.

(CW2) $Maj^{+}(P)$ is compatible with each majority partial split in P.

(CW3) Each split in $Maj^{+}(P)$ is compatible with a majority of the trees in P.

(CW4) Every plenary split in $Maj^{+}(P)$ extends at least one input tree full split.

The majority-rule (+) supertrees, as defined above, do not generalize majority-rule consensus. That is, when used in the consensus setting, $Maj^{+}(P)$ is not, in general, the same as

Maj(P). Nevertheless, majority-rule (+) consensus trees have a simple characterization that yields an efficient algorithm for computing them (see Theorem 1).

The majority-rule (+) supertrees we study differ from other variants in the way the span of an input tree is defined. Cotton and Wilkinson (2007) originally defined the span of a tree t as the set of all plenary *binary* trees that display t. Dong and Fernández-Baca (2009) showed that this version does not generalize majority-rule consensus and does not satisfy (CW4). In a more recent version, suggested by Wilkinson (personal communication), the span of t is the set of all plenary trees T such that $T|\mathcal{L}(t) = t$. We name this span as $\langle t \rangle_g$. This definition of span prohibits refinement of any original polytomies in t. Dong and Fernández-Baca (2009) showed that the supertree method that results from using this definition generalizes majority-rule consensus, and that it satisfies properties (CW1)–(CW4). Nonetheless, we have preferred Day's variant for two reasons. First, we have found it computationally easier to deal with than the others. More importantly, it can be argued that a strict generalization of majority-rule consensus might not be the ideal approach for supertree construction: In practice, one often encounters profiles where different trees "specialize" in different groups of taxa, leaving other groups largely unresolved or unrepresented. In a combined analysis, each input tree should contribute its own specialized information so that, jointly, the trees lead to a well-resolved supertree. A strict generalization of majority rule would disallow this, since the method discards minority information. In contrast, the majority-rule (+) supertrees presented here preserve this fine-grained information, unless it were substantially contradicted by the remaining trees (the sense in which this is true can be gleaned from Theorem 1).

4.2 Constructing Optimal Candidate Supertrees

We first consider the consensus version of the problem. Let $P = (T_1, \ldots, T_k)$ be a profile of trees over the same leaf set. K denotes the set $\{1, \ldots, k\}$. Given a plenary split $X = A|B$, define

$$K_X(P) = \{i \in K : X \text{ is displayed by } T_i\}$$

and

$$K_{\overline{X}}(P) = \{i \in K : X \text{ is not compatible with } T_i\}.$$

The theorem below, proved elsewhere by Dong et al. (2010b), characterizes the majority-rule (+) consensus tree of a profile and implies that this tree can be computed in polynomial time.

Theorem 1. *For any profile P, $Maj^+(P)$ is precisely the tree that displays every split X such that $|K_X(P)| > |K_{\overline{X}}(P)|$. Furthermore, $Maj^+(P)$ is an optimal candidate tree for P, as well as the strict consensus of all optimal candidate trees for P.*

On the other hand, the next result suggests that finding the majority-rule (+) supertree for a profile of trees with partially overlapping leaf sets may be hard.

Theorem 2. *There is no polynomial-time algorithm to construct an optimal candidate supertree unless $P = NP$.*

Proof. We show that if there is a polynomial time algorithm to compute an optimal candidate supertree, then there exists a polynomial-time algorithm for the quartet compatibility problem, which is known to be NP-complete by Steel (1992). The quartet compatibility problem asks whether, given a collection Q of trees on four leaves, there exists a single tree that displays them all. If the answer is "yes", we say that Q is *compatible*.

Let Q be an instance of quartet compatibility. Construct a profile P that simply consists of the trees in Q in some arbitrary order. We claim that Q is compatible if and only if P has an optimal candidate supertree with a score of zero. Suppose first that Q is compatible and that T is any tree that displays each element of Q. Then, for each tree t in P, $T \in \langle t \rangle_{gr}$, because all the splits in T must be compatible with t, so any split in T that is not in t can be added to t. Hence, T is a candidate tree for P with a score of zero, and thus T is also an optimal candidate supertree. Conversely, if P has an optimal candidate supertree with zero score, it can be seen that T displays all the quartets in Q; i.e., Q is compatible. □

In the next sections, we show that despite the above result, moderately large majority-rule (+) supertree problems can be solved using integer linear programming. For this, we need to address a potential complication: Since the definition of $\langle t \rangle_{gr}$ allows refinement of

multifurcations in t, a tree $T \in \langle t \rangle_{gr}$ can contain many more nontrivial splits than t; indeed, we cannot predetermine the number of nontrivial splits T will contain. We circumvent this potential problem by defining a restricted version of the span.

Given an input tree t in a profile P, the *restricted span* of t, denoted $\langle t \rangle_r$ is the set of all plenary trees T such that every nontrivial split in T extends a distinct nontrivial split in t. Thus, $|Spl(T)| = |Spl(t)|$. Note that T is obtained from t by filling in each of t's splits, by adding zero or more taxa to each part, to make them plenary splits in such a way that the resulting splits are compatible. Note also that $\langle t \rangle_r \subseteq \langle t \rangle_{gr}$. The *restricted span* of a profile $P = (t_1, \ldots, t_k)$, denoted $\langle P \rangle_r$ is the set of all $R = (T_1, \ldots, T_k)$ for P such that $T_i \in \langle t_i \rangle_r$ for each $i \in K$. Each $R \in \langle P \rangle_r$ is called a *restricted representative selection* for P. Since $\langle P \rangle_r \subseteq \langle P \rangle_{gr}$, the restricted span represents an intermediate level between the input profile and the original definition of span. The restricted span is more manageable than the original span because it does not allow any refinement of input trees. In the rest of this section, we will show how to obtain majority-rule (+) supertrees directly from the restricted span.

Before presenting the first of the two main results of this section, we need to introduce some new concepts. An optimal candidate supertree T for a profile P is *minimal* if contracting any edge in T yields a tree that is not an optimal candidate supertree. Let $R = (T_1, \ldots, T_k)$ and $R' = (T'_1, \ldots, T'_k)$ be two representative selections for a profile P. We say that R *displays* R' if T_i displays T'_i for every $i \in K$. Theorem 1 motivates the next definition. The *completion* of a representative selection $R = (T_1, \ldots, T_k)$ for a profile P is the representative selection $\widehat{R} = (\widehat{T}_1, \ldots, \widehat{T}_k)$ obtained as follows: For every $i \in K$, $\widehat{T}_i$ is the tree constructed by inserting into T_i each plenary split $X = A|B$ compatible with T_i such that $|K_X(R)| > |K_{\overline{X}}(R)|$.

Theorem 3. *Let T be a minimal optimal candidate supertree for a profile P and let $R \in \langle P \rangle_{gr}$ be such that $T = Maj(R)$. Consider any $G \in \langle P \rangle_r$ such that G is displayed by R. Then, R is the completion of G and $T = Maj^+(G)$.*

Proof. We begin by proving that T is an optimal candidate supertree for G. Assume the contrary. Then, there exists another candidate tree T' for G such that (i) $T' = Maj(R')$ for some $R' \in \langle G \rangle_{gr}$ and (ii) $s(R') < s(R)$. But then, since $\langle G \rangle_{gr} \subseteq \langle P \rangle_{gr}$, we have $R' \in \langle P \rangle_{gr}$,

and thus (ii) contradicts the optimality of T for P.

Next, we argue that T is a *minimal* optimal candidate supertree for profile G. Suppose this is not true. Then, T displays an optimal candidate supertree T' for G such that $T \neq T'$. Consider any $R' \in \langle G \rangle_{gr}$ such that $T' = \mathit{Maj}(R')$. Since T and T' are both optimal for G, $s(R) = s(R')$. Since R' displays P, we have $R' \in \langle P \rangle_{gr}$. Hence, T' is also an optimal candidate supertree for P. This, however, contradicts the assumption that T is a minimal optimal candidate tree for P.

By Theorem 1, $\mathit{Maj}^+(G)$ is an optimal candidate supertree for G, as well as the strict consensus of all optimal candidate supertrees for G. Therefore, $\mathit{Maj}^+(G)$ is the only minimal optimal candidate supertree for G. Hence $T = \mathit{Maj}^+(G)$.

Suppose $R = (T_1, \ldots, T_k)$ and let $\widehat{R} = (\widehat{T}_1, \ldots, \widehat{T}_k)$ be the completion of G. We claim that $\widehat{R} = R$. Assume, on the contrary, that there is some $i \in K$ such that $\widehat{T}_i \neq T_i$. That is, $\mathcal{X} \cup \mathcal{Y} \neq \emptyset$, where $\mathcal{X} = \mathit{Spl}(\widehat{T}_i) \setminus \mathit{Spl}(T_i)$ and $\mathcal{Y} = \mathit{Spl}(T_i) \setminus \mathit{Spl}(\widehat{T}_i)$. Set $\mathcal{X}$ consists of splits X such that $|K_X(G)| > |K_{\overline{X}}(G)|$ and $\mathcal{Y}$ consists of splits Y such that $|K_Y(G)| \leq |K_{\overline{Y}}(G)|$. By Theorem 1, $T = \mathit{Maj}^+(G)$ displays all splits X such that $|K_X(G)| > |K_{\overline{X}}(G)|$. Thus, $d(T, \widehat{T}_i) < d(T, T_i)$. As we are assuming that there is at least one such $i \in K$, we have $\sum_{i \in K} d(T, \widehat{T}_i) < \sum_{i \in K} d(T, T_i)$, contradicting the fact that T is a minimal optimal candidate supertree for G. □

Motivated by Theorem 3, we define the *adjusted score* of a representative selection R for a profile P, denoted $\widehat{s}(R)$, to be the score of the completion $\widehat{R}$ of R; i.e., $\widehat{s}(R) = s(\widehat{R})$. Recall that $s(\widehat{R}) = \mathit{dist}(\mathit{Maj}(\widehat{R}), \widehat{R})$.

Theorem 4. *Let P be a profile. Define $\mathcal{G} = \{G \in \langle P \rangle_r : \widehat{s}(G)$ is minimum$\}$ and $\mathcal{S} = \{T = \mathit{Maj}^+(G) : G \in \mathcal{G}\}$. Then, $\mathit{Maj}^+(P)$ is the strict consensus of $\mathcal{S}$.*

Proof. Let $\mathcal{O}$ be the set of all optimal candidate supertrees for P and let $\mathcal{M}$ be the set of all minimal optimal candidate supertrees of P. In what follows, we show that $\mathcal{M} \subseteq \mathcal{S} \subseteq \mathcal{O}$. This immediately implies the theorem, because not only is (by definition) $\mathit{Maj}^+(P)$ the strict consensus of $\mathcal{O}$, but it must also be the strict consensus of $\mathcal{M}$.

Suppose $T \in \mathcal{M}$. We claim that $T \in \mathcal{S}$ and, therefore, that $\mathcal{M} \subseteq \mathcal{S}$. Let R be a representative selection for P such that $T = \mathit{Maj}(R)$. Let G be any restricted representative selection for

P displayed by R. By Theorem 3, $T = Maj^+(G)$ and R is the completion of G. We claim that $G \in \mathcal{G}$; i.e., $\widehat{s}(G)$ is minimum. Assume, by way of contradiction, that there is another $G' \in \langle P \rangle_r$ such that $\widehat{s}(G') < \widehat{s}(G)$. Let R' be the completion of G'. Then, $s(R') = \widehat{s}(G') < \widehat{s}(G) = s(R)$, which contradicts the assumption that T is optimal. Therefore, $\widehat{s}(G)$ is minimum and $T \in \mathcal{S}$.

Suppose $T \in \mathcal{S}$. We claim that $T \in \mathcal{O}$ and, therefore, that $\mathcal{S} \subseteq \mathcal{O}$. Let $G \in \langle P \rangle_r$ be such that $T = Maj^+(G)$ and the adjusted score $\widehat{s}(G)$ is minimum. Let R be the completion of G. Assume, by way of contradiction, that $T \notin \mathcal{O}$. Then there is a $T' \in \mathcal{M}$ such that, if R' is a representative selection for P where $T' = Maj(R')$, then $s(R') < s(R)$. By Theorem 3, there is a $G' \in \langle P \rangle_r$ such that $T' = Maj^+(G')$ and $\widehat{s}(G') = s(R')$. Then $\widehat{s}(G') = s(R') < s(R) = \widehat{s}(G)$. This contradicts the assumption that $\widehat{s}(G)$ is minimum. □

4.3 ILP Formulation

In this section we first describe an ILP formulation of the optimal candidate supertree problem based on Theorem 4. The optimum solution to this ILP is a $G \in \langle P \rangle_r$ with minimum adjusted score. For ease of exposition, we divide the variables of our ILP into three categories: *fill-in variables*, which represent the way taxa are added to the input trees to create G; *objective function variables*, which are used to express $\widehat{s}(G)$; and *auxiliary variables*, which are used to establish a connection between the fill-in and objective function variables. All variables are binary. After presenting our ILP model, we discuss how to use it to generate $Maj^+(P)$.

4.3.1 Fill-in Variables

At the core of our ILP formulation is a matrix representation of the input trees similar to that used in MRP [Baum (1992); Ragan (1992)]. Let $P = (t_1, \ldots, t_k)$ be a profile where $|\mathcal{L}(P)| = n$. Assume input tree t_j has m_j nontrivial splits, which are assumed to be ordered in some fixed but arbitrary way. A *matrix representation* of t_j is a $n \times m_j$ matrix $M(t_j)$ whose columns are in one to one correspondence with the nontrivial splits of t_j. Suppose column i of $M(t_j)$ corresponds to split $A|B$ in t_j and let x be a taxon in $\mathcal{L}(P)$. Then, $M_{x,i}(t_j) = 1$ if $x \in A$, $M_{x,i}(t_j) = 0$ if $x \in B$, and $M_{x,i}(t_j) =?$ otherwise. We note that for unrooted trees the

assignment of 1 to the A side of the split and of 0 to the B side is arbitrary. For rooted trees, all taxa in the side of a split that contains the root are assigned a 1.

Let $m = \sum_{j \in K} m_j$. A *matrix representation* of P, denoted $M(P)$, is a $n \times m$ matrix $M(P)$ obtained by concatenating matrices $M(t_1), M(t_2), \ldots, M(t_k)$.

A *fill-in* of matrix $M(P)$ is a matrix representation for a restricted representative selection G for P. Note that $M(G)$ has no question marks and that, for every taxon x and split i such that $M_{xi}(P) \in \{0, 1\}$, we have $M_{xi}(G) = M_{xi}(P)$. To represent fill-ins of $M(P)$, the ILP associates a *fill-in variable* F_{xi} with each x and i. If $M_{xi}(P) \in \{0, 1\}$, then $F_{xi} = M_{xi}(P)$; i.e., F_{xi} is fixed. If $M_{xi}(P) =?$, F_{xi} will be assigned a value of 0 or 1, representing an assignment of taxon x to one of the two sides of split i. Our ILP has constraints (described below) to ensure that each value assignment to the F-variables corresponds to a restricted representative selection for P. That is, there must exist a $G \in \langle P \rangle_r$ such that $M_{xi}(G) = F_{xi}$ for every x and i.

4.3.2 Objective Function Variables

The objective is to minimize $\widehat{s}(G)$ over all $G \in \langle P \rangle_r$, where each G is represented by a fill-in of $M(P)$. By definition, $\widehat{s}(G) = dist(Maj^+(G), R)$, where $R = (\widehat{T_1}, \ldots, \widehat{T_k})$ is the completion of $G = (T_1, \ldots, T_k)$. We do not, however, construct $Maj^+(G)$ and R explicitly. Instead, we proceed indirectly, using the fact that, by Theorems 1 and 3, all splits in $Maj^+(G)$ and R are already in G. Indeed, those theorems imply that

$$dist(Maj^+(G), R) = \sum_{j \in K} |Spl(\widehat{T_j}) \setminus Spl(Maj^+(G))| + \sum_{j \in K} |Spl(Maj^+(G)) \setminus Spl(\widehat{T_j})|. \qquad (4.1)$$

The next result, which follows Theorems 1 and 3, allows us to count directly from G the contribution of each split $X \in Spl(Maj^+(G)) \cup Spl(\widehat{T_j})$ to $d(Maj^+(G), \widehat{T_j})$.

Lemma 1. *Let P be a profile and suppose $G \in \langle P \rangle_r$. Then, for each $j \in K$,*

(i) $X \in Spl(\widehat{T_j}) \setminus Spl(Maj^+(G))$ if and only if $|K_X(G)| \leq |K_{\overline{X}}(G)|$ and $j \in K_X(G)$.

(ii) $X \in Spl(Maj^+(G)) \setminus Spl(\widehat{T_j})$ if and only if $|K_X(G)| > |K_{\overline{X}}(G)|$ and $j \in K_{\overline{X}}(G)$.

Suppose we have a fill-in for $M(P)$ that corresponds to some $G = (T_1, \ldots, T_k) \in \langle P \rangle_r$. Our ILP has two kinds of objective function variables. The first group of variables are denoted $w_1, \ldots, w_m$, where w_i corresponds to the ith column of $M(G)$. Suppose this column corresponds to split X in tree T_j; thus, $j \in K_X(G)$. Our ILP has constraints such that $w_i = 1$ if and only if $|K_X(G)| > |K_{\overline{X}}(G)|$, that is, the i^{th} column is in $Maj^+(G)$. Thus, $w_i = 0$ means that $|K_X(G)| \leq |K_{\overline{X}}(G)|$, that is, the i^{th} column is not in $Maj^+(G)$. It, together with Lemma 1 (i), implies that $\sum_{i=1}^{m}(1 - w_i) = \sum_{j \in K} |Spl(\widehat{T}_i) \setminus Spl(Maj^+(G))|$.

The second group of variables are denoted z_{ij}, $1 \leq i \leq m$, $1 \leq j \leq k$. Suppose column i of $M(P)$ corresponds to split X. Our ILP has constraints such that $z_{ij} = 1$ if and only if $w_i = 1$ (i.e., $|K_X(G)| > |K_{\overline{X}}(G)|$), $j \in K_{\overline{X}}(G)$, and $i = \min\{\ell : \ell \in K_X(G)\}$. Thus, by Lemma 1 (ii), $\sum_{i=1}^{m}\sum_{j=1}^{k} z_{ij} = \sum_{j \in K} |Spl(Maj^+(G)) \setminus Spl(\widehat{T}_j)|$.

The objective function can now be expressed as

$$\text{minimize} \quad \sum_{i=1}^{m}\sum_{j=1}^{k} z_{ij} + \sum_{i=1}^{m}(1 - w_i).$$

4.3.3 Auxiliary Variables and Constraints

As mentioned earlier, all variables, including the auxiliary ones, are Boolean. We take advantage of this, expressing the constraints relating the variables as Boolean expressions in terms of the "and', "or," "exclusive or," and "if and only if" operators (denoted by the usual symbols, $\wedge$, $\vee$, $\oplus$, and $\Leftrightarrow$, respectively). We then convert these expressions into equivalent linear inequalities on zero-one variables.

We first describe the variables and constraints that are used to ensure that the settings of the fill-in variables (the F variables) correspond to a restricted representative selection. That is, the assignments to the F variables must be such that, for each input tree t_j, the resulting plenary splits associated with the tree are pairwise compatible, so that they yield a plenary tree $T_j \in \langle t_j \rangle_r$. For this purpose, we define variables C_{pq}, $1 \leq p, q \leq m$ and add constraints linking these variables and the F variables such that $C_{pq} = 1$ if and only if columns p and q are compatible under the fill-in represented by the F variables. To guarantee that the assignment to the F variables corresponds to a restricted representative selection, we require that $C_{pq} = 1$

for every two column indices p, q that correspond to splits in the same input tree. We note that the constraints relating the fill-in variables F and the C-variables closely resemble the ones used by Gusfield et al. (2007). One difference is that for our problem we need "if and only if" relationships, whereas Gusfield et al. require only one direction of the implication.

The constraints on the C-variables use the fact that splits p and q are incompatible if and only if 00, 01, 10, and 11 all appear in some rows of columns p and q (the "four gametes condition"). The presence or absence of these patterns for columns p and q is indicated by the settings of variables $B_{pq}^{(ab)}$, $a, b \in \{0, 1\}$, where $B_{pq}^{(ab)} = 1$ if and only if there is a taxon r such that $F_{rp} = a$ and $F_{rq} = b$. The $B_{pq}^{(ab)}$s are determined from the settings of variables $\Gamma_{rpq}^{(ab)}$, where r ranges over the taxa (i.e., the rows of $M(P)$). The Γ variables satisfy $\Gamma_{rpq}^{(ab)} \Leftrightarrow ((F_{rp} = a) \wedge (F_{rq} = b))$. This condition is expressed by the following constraints.

$$\begin{aligned} (-1)^a F_{rp} + (-1)^b F_{rq} + \Gamma_{rpq}^{(ab)} &\geq 1 - a - b, \\ (-1)^a F_{rp} + (-1)^b F_{rq} + 2\Gamma_{rpq}^{(ab)} &\leq 2 - a - b. \end{aligned} \tag{4.2}$$

We have that $B_{pq}^{(ab)} \Leftrightarrow \bigvee_r \Gamma_{rpq}^{(ab)}$, which is expressed by the inequalities below.

$$\begin{aligned} -\sum_r \Gamma_{rpq}^{(ab)} + B_{pq}^{(ab)} &\leq 0, \\ \sum_r \Gamma_{rpq}^{(ab)} - nB_{pq}^{(ab)} &\leq 0 \end{aligned} \tag{4.3}$$

Observe that $\neg C_{pq} \Leftrightarrow B_{pq}^{(00)} \wedge B_{pq}^{(01)} \wedge B_{pq}^{(10)} \wedge B_{pq}^{(11)}$. Equivalently we have the constraints below.

$$\begin{aligned} B_{pq}^{(00)} + B_{pq}^{(01)} + B_{pq}^{(10)} + B_{pq}^{(11)} + 4C_{pq} &\geq 4, \\ B_{pq}^{(00)} + B_{pq}^{(01)} + B_{pq}^{(10)} + B_{pq}^{(11)} + C_{pq} &\leq 4. \end{aligned} \tag{4.4}$$

We now consider the variables and constraints that enable us to express the objective function variables. There are three main sets of variables:

- For $1 \leq p \leq m$, D_p equals 1 if and only if column p represents the same split as some column with smaller index.
- For $1 \leq i \leq m$, $1 \leq j \leq k$, $S_{ij}^{(1)}$, equals 1 if and only if split i is in tree j.

- For $1 \leq i \leq m$, $1 \leq j \leq k$, $S_{ij}^{(2)}$ equals 1 if and only if split i is compatible with tree j.

As we shall see, the values of the w and the z variables in the objective function are determined, respectively from the $S^{(1)}$ and $S^{(2)}$ variables, and from the w, $S^{(2)}$, and D variables.

The D and $S^{(1)}$ variables depend on variables E_{pq}, $1 \leq p, q \leq m$, where $E_{pq} = 1$ if and only if columns p and q of the filled-in matrix represent the same split. Here we have to make a distinction between rooted and unrooted trees. In the rooted case, there exists a root taxon r such that $M_{ri}(P) = 1$ for every column i. The same is not true for unrooted trees.

The value of E_{pq} depends on the patterns that appear in columns p and q, which can be deduced from the values of $B_{pq}^{(ab)}$ for different choices of a and b as follows.

- For rooted trees, $E_{pq} \Leftrightarrow \neg B_{pq}^{(01)} \wedge \neg B_{pq}^{(10)}$. This is expressed as follows.

$$\begin{aligned} B_{pq}^{(01)} + B_{pq}^{(10)} + 2E_{pq} &\leq 2, \\ B_{pq}^{(01)} + B_{pq}^{(10)} + E_{pq} &\geq 1. \end{aligned} \tag{4.5}$$

- For unrooted trees, we introduce two auxiliary variables $\delta_{pq}^{(1)}$ and $\delta_{pq}^{(2)}$ such that

$$\delta_{pq}^{(1)} \Leftrightarrow \neg B_{pq}^{(01)} \wedge \neg B_{pq}^{(10)} \quad \text{and} \quad \delta_{pq}^{(2)} \Leftrightarrow \neg B_{pq}^{(00)} \wedge \neg B_{pq}^{(11)}.$$

  Then,

$$E_{pq} \Leftrightarrow \delta_{pq}^{(1)} \oplus \delta_{pq}^{(2)}.$$

  These logical constraints are expressed by the following inequalities.

$$\begin{aligned} B_{pq}^{(01)} + B_{pq}^{(10)} + 2\delta_{pq}^{(1)} &\leq 2, \\ B_{pq}^{(01)} + B_{pq}^{(10)} + \delta_{pq}^{(1)} &\geq 1, \\ B_{pq}^{(00)} + B_{pq}^{(11)} + 2\delta_{pq}^{(2)} &\leq 2, \\ B_{pq}^{(00)} + B_{pq}^{(11)} + \delta_{pq}^{(2)} &\geq 1, \\ E_{pq} - \delta_{pq}^{(1)} - \delta_{pq}^{(2)} &= 0. \end{aligned} \tag{4.6}$$

We are now ready to give the constraints for the D, $S^{(1)}$ and $S^{(2)}$ variables. Observe that

$D_1 = 0$ and that, for $1 < p \leq m$, $D_p \Leftrightarrow \bigvee_{i=1}^{p-1} E_{ip}$. Equivalently we have

$$\begin{aligned} D_p - \sum_{i=1}^{p-1} E_{ip} &\leq 0, \\ \sum_{i=1}^{p-1} E_{ip} - p \cdot D_p &\leq 0. \end{aligned} \tag{4.7}$$

In describing the constraints for the $S^{(1)}$ and $S^{(2)}$ variables, we adopt the convention that the splits of the jth tree correspond to columns $j_1, \ldots, j_d$ of $M(P)$. Then, $S_{ij}^{(1)} \Leftrightarrow E_{ij_1} \oplus \cdots \oplus E_{ij_d}$. This translates into the equality constraint

$$S_{ij}^{(1)} - \sum_{r=1}^{d} E_{ij_r} = 0. \tag{4.8}$$

On the other hand, $S_{ij}^{(2)} \Leftrightarrow C_{ij_1} \wedge \cdots \wedge C_{ij_d}$. This is equivalent to the two constraints below.

$$\begin{aligned} d \cdot S_{ij}^{(2)} - \sum_{r=1}^{d} C_{ij_r} &\leq 0, \\ 1 - d - S_{ij}^{(2)} + \sum_{r=1}^{d} C_{ij_r} &\leq 0. \end{aligned} \tag{4.9}$$

Finally, we describe how the objective function variables relate to the auxiliary variables. For each i, $w_i = 1$ if and only if $\sum_{j=1}^{k} S_{ij}^{(1)} > k - \sum_{j=1}^{k} S_{ij}^{(2)}$. This is expressed by the following two constraints.

$$\begin{aligned} k \cdot w_i + 1 - \sum_{j=1}^{k} S_{ij}^{(1)} - \sum_{j=1}^{k} S_{ij}^{(2)} &\leq 0, \\ \sum_{j=1}^{k} S_{ij}^{(1)} + \sum_{j=1}^{k} S_{ij}^{(2)} - k - k \cdot w_i &\leq 0. \end{aligned} \tag{4.10}$$

It follows from the definition of the z variables that, for every i, j, $z_{ij} \Leftrightarrow w_i \wedge \neg S_{ij}^{(2)} \wedge \neg D_i$. Equivalently we have the following.

$$\begin{aligned} -2 - w_i + S_{ij}^{(2)} + D_i + 3 \cdot z_{ij} &\leq 0, \\ w_i - S_{ij}^{(2)} - D_i - z_{ij} &\leq 0. \end{aligned} \tag{4.11}$$

There are a total of $O(nm^2)$ variables and constraints of the ILP formulation. It should be noted that $O(nm^2)$ assumes that all the variables listed are indeed variables. In reality, the values of many of the F variables are fixed because they correspond to non-question-mark

entries in $M(P)$. This in turn fixes the values for several Γ variables, as well as those of other variables. As a consequence, the number of true variables in the ILP formulation is typically much smaller than the worst case estimates. In general, the larger the number of question marks in matrix $M(P)$, the closer the problem size will be to the worst case estimates.

4.3.4 Building Majority-rule (+) Supertree

The ILP model just outlined allows us to find a $G \in \langle P \rangle_r$ corresponding to some optimal candidate supertree T^*. To build $Maj^+(P)$ we need, in principle, the set of all such G. While there are ways to enumerate this set [Danna et al. (2007)], we have found that an alternative approach works much better in practice. The key observation is that, since $Maj^+(P)$ is the strict consensus of all optimal candidate supertrees, each split in $Maj^+(P)$ must also be in T^*. Thus, once we have T^*, we simply need to verify which splits in T^* are in $Maj^+(P)$ and which are not. To do this, for each split $A|B$ in T^*, we put additional constraints on the original ILP requiring that the optimal tree achieve an objective value equal or smaller than that of T^* and *not* display split $A|B$. The resulting ILP has only $O(mn)$ more variables and constraints than the original one. If the new ILP is feasible, then $A|B \notin Spl(Maj^+(P))$; otherwise, $A|B \in Spl(Maj^+(P))$. We have found that detecting infeasibility is generally much faster than finding an optimal solution.

4.4 A Data Reduction Heuristic

The ILP formulation described in the previous section allows us to solve supertree problems of moderate size. Here we describe a data reduction heuristic that allows us to extend the range of our method significantly in practice, by exploiting the structure that is present in certain supertree problems. Our data reduction heuristic applies when the input profile $P = (t_1, \ldots, t_k)$ contains a subset of taxa S that can be treated as a single super-taxon. Roughly stated, we are looking for a set S such that every tree in P respects the split implied by S. We now define this concept more precisely.

Let $Spl_0(T)$ denote the set of *all* full splits displayed by T. That is, $Spl_0(T)$ includes the non-trivial and the trivial splits displayed by T; in particular, $\mathcal{L}(T)|\emptyset \in Spl_0(T)$. We say

that $S \subseteq \mathcal{L}(P)$ with $1 < |S| < |\mathcal{L}(P)| - 1$ is a *reducible set* if, for each $j \in K$, there is a split $A|B \in Spl_0(t_j)$ such that $A \cap S = A$ and $B \cap S = \emptyset$. Ideally, a reducible set should correspond to a widely-acknowledged biological classification unit. For example, a subset of the trees in a collection of phylogenies may contain subtrees corresponding to different (possibly empty) subsets of the primates. While these subsets may not be identical and may even disagree somewhat in their topologies, the trees are likely to separate primates from other organisms. In settings like this, it makes intuitive sense to restrict our attention to supertrees where reducible sets appear as clusters.

Given a reducible set S for P, we can define two smaller subproblems.

- The *reduced profile associated with a reducible set* S is the profile $P^{Red} = (t_1^{Red}, \ldots, t_k^{Red})$ where, for each $j \in K$, t_j^{Red} is the tree obtained from t_j by contracting the minimal subtree of t_j containing $S \cap \mathcal{L}(t_j)$ to a single leaf node β_S. If $S \cap \mathcal{L}(t_j) = \emptyset$, then $t_j^{Red} = t_j$. We refer to β_S as the *supertaxon associated with* S.

- The *satellite profile associated with* S is the profile $P^{Sat} = (t_1^{Sat}, \ldots, t_k^{Sat})$ where t_j^{Sat} is obtained from t_j by contracting the minimal subtree of t_j containing $(\mathcal{L}(P) \setminus S) \cap \mathcal{L}(t_j)$ to a single leaf node ρ_S. Note that some of the trees in the satellite profile associated with S may contain only ρ_S. The *compressed satellite profile associated with* S is the satellite profile associated with S with all of the latter trees removed.

An *S-restricted* representative selection for P is a selection $R = (T_1, \ldots, T_k) \in \langle P \rangle$ such that $S|(\mathcal{L}(P) \setminus S) \in Spl(T_i)$ for all $i \in K$. An *optimal S-restricted candidate representative selection* is an S-restricted representative selection R with minimum score, and $Maj(R)$ an *optimal S-restricted candidate supertree*. The *S-restricted majority-rule (+) supertree* is the strict consensus of all the optimal S-restricted candidate supertrees.

It should be noted that, given an arbitrary reducible set S, it is not true in general that an optimal S-restricted candidate supertree will be an optimal candidate supertree, nor that an S-restricted majority-rule (+) supertree will also be a majority-rule (+) supertree.

On the other hand, a reducible set may represent useful biological knowledge that should be incorporated into a supertree analysis. There are also computational benefits. With the right

choice of S (one where $|S|$ is far from the extreme values of 2 and $|\mathcal{L}(P)|-2$), the reduced and satellite profiles can be considerably smaller than the original profile, and the corresponding integer programs will have fewer unknown variables. As the following theorem indicates, an optimal S-restricted candidate supertree can be found by solving the associated subproblems separately and combining their answers.

Theorem 5. *Let P be a profile and S be a reducible set in P. Let T^{Red} and T^{Sat} be optimal candidate trees for the reduced profile associated with S and the compressed satellite profile associated with S. Let T be the tree obtained by identifying the node β_S in T^{Red} and node ρ_S in T^{Sat} and then suppressing the resulting degree-two vertex. Then, T is an optimal S-restricted candidate supertree for P. Further, if R, is the optimal S-restricted representative selection corresponding to T and R^{Red} and R^{Sat} are the optimal representative selections corresponding to T^{Red} and T^{Sat}, respectively, then $s(R) = s(R^{Red}) + s(R^{Sat})$.*

The straightforward proof of this result is omitted. A direct consequence is that the S-restricted majority-rule (+) supertree can be obtained by piecing together the majority-rule (+) supertrees for the reduced and satellite profiles. Observe that if multiple pairwise disjoint reducible sets are known, then each of the corresponding compressed satellite profiles can be solved independently, and the original profile can be reduced by replacing each reducible set to a distinct super taxon. In fact, the idea can be used recursively, so that a satellite profile can itself be decomposed to a reduced profile and (sub) satellites. As we shall see later, this can result in dramatic problem size reductions.

CHAPTER 5. EXPERIMENTAL STUDY FOR MAJORITY-RULE (+) SUPERTREES

We report on computational tests with the exact ILP method and the data reduction heuristic. All our experiments were conducted on real data sets, rather than simulated data. We did this because we were interested in seeing if the groupings of taxa generated by majority-rule (+) supertrees would coincide with those commonly accepted by biologists. Another goal of our experiments was to compare the performance of the ILP formulation without data reduction, which we refer to as the *basic* method, against that of ILP plus data reduction. All trees considered in our tests were rooted.

To conduct our tests of the basic method, we wrote a program to generate the ILPs from the input profiles. For our tests of the data reduction heuristic, we used different methods to find reducible sets in a profile; these are outlined later. Given the reducible sets, the corresponding reduced and satellite profiles, as well as the associated ILPs, were generated automatically. All ILPs were then solved using CPLEX (CPLEX is a trademark of IBM) on an Intel Core 2 64 bit quad-core processor (2.83GHz) with 8 GB of main memory and a 12 MB L2 cache per processor.

5.1 Experiments with the Basic ILP Formulation

We tested the basic ILP formulation on five published data sets. The *Drosophila A* data set is the example studied in Cotton and Wilkinson (2007), which was extracted from a larger Drosophila data set considered by Cotton and Page (2004). *Primates* is the smaller of the data sets from Ranwez et al. (2007). *Drosophila B* is a larger subset of the data studied in Cotton and Page (2004) than that considered in Cotton and Wilkinson (2007). *Chordata A and B* are

two extracts from a data set used in a widely-cited study by Delsuc et al. (2006). Chordata A consists of the first 6 trees with at least 35 taxa (out of 38). Chordata B consists of the first 12 trees with at least 37 taxa (out of 38).

The results are summarized in Table 5.1. Here n, m, and k are the number of taxa, total number of splits, and number of trees, respectively. N is the size of the CPLEX-generated reduced ILP. It shows the time to solve the ILP and produce an optimal candidate supertree T^* and the time to verify all the splits of T^* to produce $Maj^+(P)$.

Table 5.1 Summary of Experimental Results with the Basic ILP Method

Data set	n	m	k	N	Sol. (sec)	Verif. (sec)
Drosphila A	9	17	5	9.8 e5	0.83	1.6
Primates	33	48	3	7.8 e7	15.83	2.86
Drosophila B	40	55	4	1.25 e9	362	19
Chordata A	38	290	6	1.40 e8	120	258
Chordata B	38	411	12	1.05 e8	986	1784

Our results using the basic ILP formulation compare well with the published ones. For Drosophila A we obtained exactly the same tree reported in Cotton and Wilkinson (2007). For Primates, the output is exactly the same as Ranwez et al. (2007), which was produced by PhySIC method. The coincidence with PhySIC is noteworthy, since this supertree is less controversial than the MRP, Mincut, and $\text{PhySIC}_{\text{PC}}$ supertrees reported in Ranwez et al. (2007). The reason for the coincidence may lie in the fact that, while heuristic, PhySIC requires that all topological information contained in the supertree be present in an input tree or collectively implied by the input trees, which bears some similarity with properties (CW1)–(CW4) of majority (+) supertrees.

For Drosphila B, Cotton and Page (2004) show four supertrees: strict consensus of gene tree parsimony (GTP), Adams consensus of GTP, strict consensus of MRP, Adams consensus of MRP. Among the 10 clusters found by our ILP, two are in all four of these supertrees, three are found in the Adams consensus of GTP and Adams consensus of MRP, one is in the strict and Adams consensus of GTP, and one is found in the strict and Adams consensus of MRP. Thus, with only four input trees we were able to generate a tree that is quite similar to the

published results. For Chordata A, the 12 splits found matched published results [Delsuc et al. (2006)] exactly. For Chordata B, the 14 splits found matched Delsuc et al. (2006).

We have not mapped out the precise boundary within which it is feasible to use the basic ILP method. However, it appears that it may not extend much beyond the dimensions of the problems listed in Table 5.1. For example, Drosophila B contains four out of 6 of the trees studied in Cotton and Page (2004). Adding a fifth tree to the data set yields a problem that could not be solved by the basic ILP method. A major factor here is that the size of our ILP grows as the square of the total number of splits in all trees, and the solution time is exponential in the worst case. Incorporating a new tree to Drosophila B could easily add enough splits to the problem to put it well beyond the reach of our technique. We should add that model size does not appear to be the sole factor that makes instances hard — sparsity also seems to play a role.

5.2 Experiments with the Data Reduction Heuristic

As a preliminary test, we compared the results obtained via the reduction heuristic with the exact solutions, obtained using the basic ILP method, for two of the data sets listed in Table 5.1. For simplicity, only clusters from the input trees were used as reducible sets. (Note that unions of input clusters could have also been used as reducible sets.) We wrote a program that chooses clusters greedily. At every step, it selects the largest non-trivial cluster present in some input tree that does not overlap with any of the previously chosen clusters.

For the Primates data set, the optimal objective value (i.e., the score of an optimal candidate supertree) for the original profile is 9. We found six pairwise disjoint reducible sets, and built the corresponding reduced and satellite profiles. The optimal objective values of the reduced profile, first, second and third satellite profiles are 0, 4, 3, and 2, respectively. The other satellite profiles have an optimal objective value of 0. Thus, the total score of the reduced and satellite profiles matches the optimal score for the original profile, and the supertree obtained using the heuristic is also optimal. The reduction method also gives a correct optimal candidate supertree for Drosophila B. Here the original profile has an objective value of 8. We found nine pairwise disjoint reducible sets, and built the corresponding reduced and satellite profiles. The

reduced profile has an optimal objective value of 8 and all satellite profiles have an optimal objective value of 0.

It should be pointed out that the reducible sets used for Primates and Drosophila B do not necessarily correspond to clusters in the majority-rule (+) supertree, although they are displayed by some optimal candidate trees. Thus, one will not obtain a majority-rule (+) supertree by simply composing the solutions to the reduced problems and the satellites. This indicates the importance of choosing relatively few large and well-supported reducible sets. Biological knowledge can serve as a good guide. For example using the clade *Haplorrhini* as a reducible set for Primates data set, solving the corresponding reduced and satellite profiles and combining the respective majority-rule (+) supertrees one gets exactly the same supertree as through the basic (and exact) method. Similarly, using the subgenus *Sophophora* as a reducible set for Drosophila B, we, obtained precisely the majority-rule (+) supertree for the data set.

Next, we considered some data sets that are well beyond the reach of our basic ILP method. The *Drosphila C* data set is the full 6-tree Drosophila data set of Cotton and Page (2004) from which the Drosophila A and B data sets were extracted. The *Seabirds* data set consists of the 7 trees in the seabirds study by Kennedy and Page (2002); which encompasses 122 taxa (note that one of these taxa is an outgroup, so we do not count it in our study). We also examined the full Chordata set of Delsuc et al. (2006), which has 38 taxa and 146 trees.

We looked for reducible sets in the full Chordata data set by considering increasingly larger subprofiles, starting with one input tree and then including one more input tree at every step. For each subprofile, we conducted an exhaustive search for reducible sets. The number of reducible sets increased at first, then fluctuated, and finally declined. After the 20th tree, there were no reducible sets. Thus, the data reduction heuristic proved to be ineffective for this data set.

We identified seven reducible sets for Drosophila C. Six of these were found by the greedy approach; the seventh corresponded to the subgenus *Sophophora* (the latter was selected manually, after some of the subproblems identified by our program proved impossible to solve). Four of the associated satellites were trivially solvable, since each contained only two taxa. We then solved ILPs for the reduced and the nontrivial satellites. The running time statistics are

summarized in Table 5.2, which shows the same kind of data shown in Table 5.1, except that this time it reports these statistics for the original, reduced and satellite problems. Notably, even though the original ILP was too large to be solved, the reduced profile was solved in less than 10 minutes and the satellite profiles were solved almost instantly. Not listed in the table are four other trivially solvable two-taxon satellite profiles.

Table 5.2 Results of Drosophila C Analysis Using Data Reduction

Data set	n	m	N	Sol. (sec)	Verif. (sec)
Original	46	70	9.4e9	N/A	N/A
Reduced	17	33	1.7e7	543.16	50.4
Satellite 1	17	17	2.3e6	0.23	0.28
Satellite 2	6	4	0	0.00	0.00
Satellite 3	5	3	0	0.00	0.02

The majority-rule (+) supertree for Drosphila C constructed by our method (available upon request) has 15 nontrivial clusters, while the MRP strict consensus tree of Cotton and Page (2004) has 11. Of these only three appear in both trees. This rather surprising result motivated us to try to assess how well the input trees are represented by the supertree. To this end, we relied on the notions of support and conflict, along the lines proposed by Wilkinson et al. (2005).

Let t be an input tree for a profile P, T be a supertree for P, and S be a non-trivial cluster in T (i.e., S does not contain the root of T and $S|(\mathcal{L}(P) \setminus S) \in \mathit{Spl}(T)$). Let $S' = S \cap \mathcal{L}(t)$. We say that tree t *supports* S if S' is a non-trivial cluster in t. Tree t is *in conflict with* S if S' is incompatible with t; i.e., there is no tree t' with $\mathcal{L}(t') = \mathcal{L}(t)$ such that $\mathit{Spl}(t) \cup \{S' | (\mathcal{L}(t) \setminus S')\} \subseteq \mathit{Spl}_0(t')$. If t neither supports nor is in conflict with S, we say that t is *irrelevant* to S.

It hints that each cluster S in the majority-rule (+) supertree should have more input trees supporting it than contradicting it, even when most trees are irrelevant to S. This indeed holds for the Drosophila C majority-rule (+) supertree: Every one of its non-trivial clusters is supported by at least one input tree and does not conflict with any input tree. In contrast, of the five clusters in the MRP strict consensus supertree for which support outweighs conflict, only three have no conflict with any input tree. Of the remaining clusters, three have the same

amount of conflict as support, and for three others the amount of support is outweighed by the amount of conflict. In fact, among the latter, there is a cluster that is in conflict with five out of six of the input trees; the remaining tree is irrelevant to that cluster. We refrain from claiming the superiority of one supertree over the other, since the biological relevance of both trees needs to be studied in more detail.

To handle the Seabirds data set, we identified three reducible sets, which yielded a reduced profile and three satellite profiles, numbered 1, 2, and 3. Satellite profile 3 was too big to be solved by the basic ILP method, so it was further reduced by identifying three reducible sets within it, which resulted in three (sub-) satellite profiles, numbered 3.1, 3.2, and 3.3. The various reducible sets correspond to biologically meaningful classification units, as we explain next. In what follows, we refer to the 7 input trees of Kennedy and Page's seabirds data set by the same letters A–G that those authors used in Kennedy and Page (2002).

Satellite 1 comprises the family *Spheniscidae* (Penguins, 10 taxa), which agrees with widely-accepted classifications for seabirds [Brooke (2002)]. Members of this family appear in input trees E, F, and G of Kennedy and Page (2002), and clearly form clusters of their own. Satellites 2 and 3 correspond to *Diomedeinae* (Albatrosses, 22 taxa), and *Procellariinae* (gadfly petrels, shearwaters, fulmars and diving petrels, 73 taxa). This agrees with the Sibley-Ahlquist classification [Sibley and Ahlquist (1990)] (represented by tree G). The resulting reduced profile has 19 taxa (16 original taxa and three supertaxa).

Satellite 3 (*Procellariinae*) has three subsatellites. Satellite profile 3.1 comprises the genus *Pterodroma* (30 taxa). Satellite 3.2 is for genus *Pelecanoides* (four taxa). Satellite 3.2 is a combination of *Puffinus* and *Calonectris* (10 taxa), which is supported by Nunn and Stanley (1998) (tree E). With these three sub-satellites, the reduced *Procellariinae* profile has 23 taxa (20 original taxa and three supertaxa).

Table 5.3 summarizes the results on the Seabirds data set. The majority-rule (+) supertree is shown in Figure 5.1, along with the MRP strict consensus tree of Kennedy and Page (2002). While the original problem was too big for CPLEX to solve on our machine, the reduced model was solved in 6.5 seconds. Most subproblems were solved and verified in a negligible amount of time. A notable exception was the reduced version of satellite 3, which required almost a

minute to solve and nearly one hour and 45 minutes to verify.

Table 5.3 Results of Seabirds Analysis Using Data Reduction

Data set	n	m	N	Sol. (sec)	Verif. (sec)
Original	121	188	2.63e12	N/A	N/A
Reduced	19	24	7.1e6	6.51	156.2
Sat. 1	10	8	1.1e5	0.05	0.07
Sat. 2	22	29	1.2e6	0.09	1.06
Satellite 3 (reduced)	23	39	5.1e7	52.3	6110
Subsatellite 3.1	30	42	6.8e5	0.06	0.04
Subsatellite 3.2	4	2	0	0.00	0.00
Subsatellite 3.3	19	20	9.5e5	0.17	0.06

Figure 5.1 compares the majority-rule (+) supertree for the seabirds data set, constructed using the data reduction heuristic, with the MRP strict consensus supertree that Kennedy and Page presented for the same data set [Kennedy and Page (2002)]. The latter is the strict consensus of 10,000 equally parsimonious trees obtained using MRP. There are 66 nontrivial clusters in the majority-rule (+) supertree, compared with 75 nontrivial clusters in the MRP strict consensus tree (ignoring the outgroup). Among these clusters, 63 are present in both trees (95% of 66 and 84% of 75). The reducible sets used to construct the majority-rule (+) supertree are indicated by heavy lines. Note that these sets are also clusters in the MRP supertree.

Three clusters, numbered 1–3 in Figure 5.1, are in the majority-rule (+) supertree but not in the MRP tree; 12 clusters, numbered 4–15 in Figure 5.1, appear in the MRP tree but not in the majority-rule (+) tree. For each of the seven input trees (labeled A–G in Kennedy and Page (2002)) and each of these 15 clusters, Table 5.4 indicates whether the tree supports (s), is in conflict with (c), or is irrelevant to (i) the cluster. The numbering of the clusters follows Figure 5.1. As we expected, each of clusters 1–3 (from the majority-rule (+) tree) has more input trees supporting it than in conflict with it. Of the 12 clusters (4–15) that are present only in the MRP strict consensus tree, seven have as many trees in support as in conflict. The others have more support than conflict.

In general, it appears that MRP may have a bias toward preserving clusters that are present

Table 5.4 Support and Conflict for the Seabirds Data Set

Cluster	A	B	C	D	E	F	G
1	i	s	i	i	i	i	i
2	s	s	s	i	s	s	s
3	s	s	s	i	c	s	s
4	s	s	s	i	s	c	c
5	i	s	s	i	s	i	c
6	i	s	s	i	s	c	c
7	i	i	i	s	c	i	i
8	i	i	i	s	c	i	i
9	i	i	i	s	c	i	i
10	i	i	i	s	c	i	i
11	i	i	i	c	s	i	i
12	i	i	i	s	c	i	i
13	i	i	i	i	s	i	c
14	i	i	s	i	s	i	c
15	i	i	s	i	s	i	c

in trees that contain many members of the families represented in those clusters. This is noticeable for *Pterodroma*, where the disagreement between trees D and E is resolved in favor of the former five times to one, in clusters 7, 8, 9, 10, and 12 versus cluster 11. This may be related to the "size bias" that previous researchers have observed in MRP [Purvis (1995)]: Here, even though E is the larger tree (90 taxa versus 30), D has more taxa in the *Pterodroma* genus (30 versus 16). Majority-rule (+) trees seem not to have such a bias, because the expansion process used to construct representative selections tends to put all input trees, regardless of their size, on equal footing. These are, of course, only preliminary observations; this issue clearly deserves further analysis.

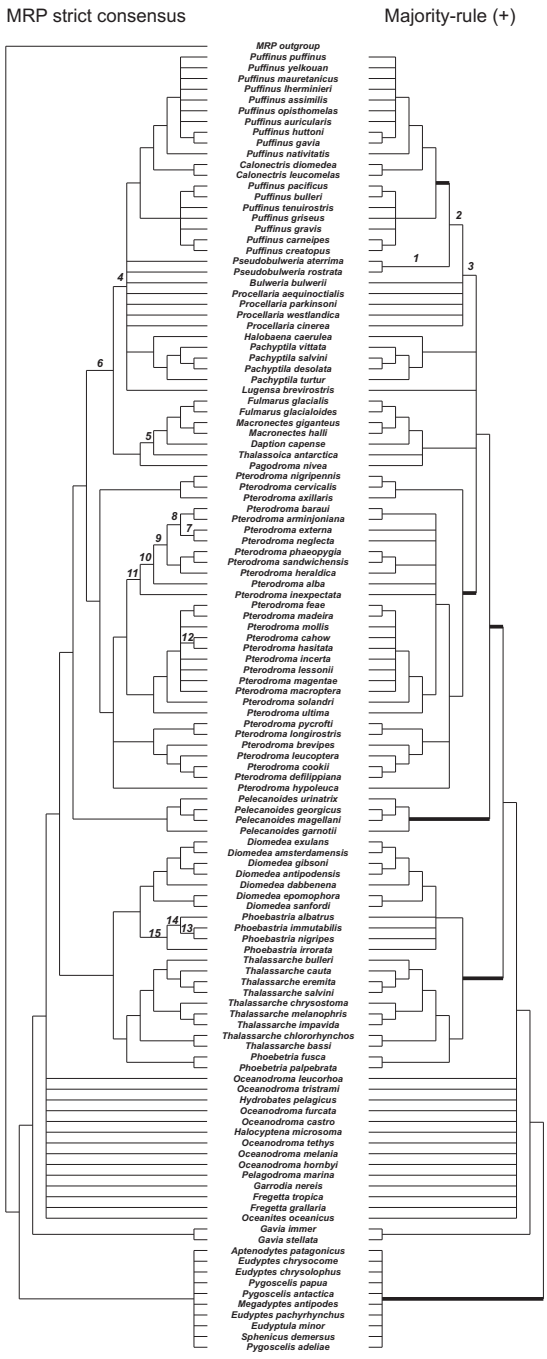

Figure 5.1 Comparing the Seabirds MRP Strict Consensus with the Majority-rule (+) Supertree

CHAPTER 6. CONSERVATIVE SUPERTREES

6.1 Original Formulations

McMorris and Wilkinson (2011) defined strict and loose supertrees using asymmetric distance as follows. Let $P = (t_1, \ldots, t_k)$ be a profile and S be a supertree for P. Define $AD^-(S, t_i) = AD(S|\mathcal{L}_i, t_i)$ and $AD^-(t_i, S) = AD(t_i, S|\mathcal{L}_i)$. Then $AD^-(S, P) = \sum_{i=1}^{k} AD^-(S, t_i)$ and $AD^-(P, S) = \sum_{i=1}^{k} AD^-(t_i, S)$. Define $AD_g^+(S, t_i) = \min\{AD(S, T_i) : T_i \in \langle t_i \rangle_g\}$ and $AD_g^+(t_i, S) = \min\{AD(T_i, S) : T \in \langle t_i \rangle_g\}$. Then $AD_g^+(S, P) = \sum_{i=1}^{k} AD_g^+(S, t_i)$ and $AD_g^+(P, S) = \sum_{i=1}^{k} AD_g^+(t_i, S)$. It is pointed out that $AD_g^+(t_i, S) = AD^-(t_i, S)$ and $AD_g^+(S, t_i) = 0$ if and only if $AD^-(S, t_i) = 0$. Therefore, the *strict supertree* of P, denoted $Str_s(P)$, is the strict consensus tree of candidate supertrees in the set $\sigma^+ = \{S : AD_g^+(S, P) = 0\}$ for which the sum $AD_g^+(P, S)$ is minimum over σ^+.

Define the following two sets of candidate supertrees. $\sigma^- = \{S : AD^-(S, P) = 0$ and $\mathcal{L}(S) = \mathcal{L}(P)\}$, $\sigma^+ = \{S : AD_g^+(S, P) = 0\}$. Obviously $\sigma^+ = \sigma^-$. $\lambda = \{S : S \otimes P$ and $\mathcal{L}(S) = \mathcal{L}(P)\}$. The symbol $\otimes$ means "to be compatible with". The *strict supertree* for P, denoted $Str_s(P)$, is the strict consensus of the supertrees in the set $\sigma^* = \{S \in \sigma^-(= \sigma^+) : AD^-(P, S)(= AP_g^+(P, S)$ is minimum$\}$. The *loose supertree* of P, denoted $Loose_s(P)$, is the strict consensus of all trees in the set $\lambda^* = \{S \in \lambda : AD^-(P, S)(= AP_g^+(P, S))$ is minimum$\}$.

6.2 Alternative Formulation of Strict Supertrees

We propose an alternative definition. Let $P = (t_1, \ldots, t_k)$ be a profile. Let $R = (T_1, \ldots, T_k)$ be a representative selection of P where $T_i \in \langle t_i \rangle_g$, $S = Str(R)$ (the strict consensus of R), and $dist(S, R)$ be the symmetric difference distance between S and R. If $dist(S, R)$ is minimum, S is an *optimal candidate tree*. $Str_s(P)$ is the strict consensus of all optimal candidate trees.

Define $ADIFF_g(P) = \min_{S' \in \sigma^+} \min_{R' \in \langle P \rangle_g} \{AD(R', S')\}$. Note that an optimal candidate tree S to generate Str_s requires that $S \in \sigma^+$ and $AD_g^+(P, S) \leq AD_g^+(P, S')$ where $S' \in \sigma^+$. Obviously such S satisfies that $AD_g^+(P, S) = ADIFF_g(P)$.

The following lemma establishes that for any $S \in \sigma^+$, there is a "bridge" representative selection R such that $AD(S, R) = AD_g^+(S, P) = 0$ and $AD(R, S) = AD_g^+(P, S)$. Moreover, if $S^* \in \sigma^+$, R^* is its bridge representative selection, and $AD(R^*, S^*) = AD_g^+(P, S^*) \leq AD_g^+(P, S)$ for any $S \in \sigma^+$, then $S^* = Str(R^*)$ and $ADIFF(P) = AD_g^+(P, S^*)$.

Lemma 2. *Given a profile $P = (t_1, \ldots, t_k)$, and any $S \in \sigma^+$, there is a representative selection $R = (T_1, \ldots, T_k)$ where $T_i \in \langle t_i \rangle_g$ such that*

(i) $T_i \setminus S$ has a bijection with the splits in t_i that cannot be reduced from S.

(ii) $AD_g^+(P, S) = AD(R, S)$.

(iii) $S \subseteq Str(R)$.

For $S^* \in \sigma^+$ and $R^* \in \langle P \rangle_g$ satisfying (i)-(iii) and $AD(R^*, S^*) = AD_g^+(P, S^*) \leq AD_g^+(P, S)$ for any $S \in \sigma^+$, $S^* = Str(R^*)$ and $ADIFF_g(P) = AD_g^+(P, S^*)$.

Proof. (i) Since $S \in \sigma^+$, $AD_g^+(S, P) = 0$. There is a representative selection $R = (T_1, \ldots, T_k)$ such that $T_i \in \langle t_i \rangle_g$, $AD(S, R) = 0$ (that is, $S \subseteq T_i \in R$) and $T_i \setminus S$ has a bijection to edges in t_i that cannot be reduced from S. If two or more edges in $T_i \setminus S$ correspond to one edge in t_i that cannot be reduced from S, then leave one edge and contract the rest.

(ii) We claim that $AD(R, S) = AD_g^+(P, S)$. For any $A|B \in t_i$ that S cannot reduce to, $T_i \setminus S$ displays $A|B$. $|T_i \setminus S|$ equals the number of such splits. This is the minimum number any $|AD(T, S)|$ can achieve where $T \in \langle t_i \rangle_g$. Thus, $AD(R, S) = AD_g^+(P, S)$.

(iii) Obviously $S \subseteq T_i$ and $S \subseteq Str(R)$.

We claim that $S^* = Str(R^*)$. Assume to the contrary, if $S^* \subset S' = Str(R^*)$, then $S' \in \sigma^+$, $AD(R^*, S') < AD(R^*, S^*) = AD_g^+(P, S^*) \leq AD_g^+(P, S')$. A contradiction. Hence, $S^* = Str(R^*)$.

$ADIFF_g(P) = \min_{S' \in \sigma^+} \min_{R' \in \langle P \rangle_g} \{AD(R', S')\} = \min_{S' \in \sigma^+} AD_g^+(P, S') = AD_g^+(P, S^*)$.

□

Therefore, instead of having two R_1 and R_2 such that $AD(S, R_1) = AD_g^+(S, P) = 0$ and $AD(R_2, S) = AD_g^+(P, S)$, respectively, we have $AD(S, R) = AD_g^+(S, P) = 0$ and $AD(R, S) = AD_g^+(P, S)$. R is a "bridge" that links both $AD_g^+(S, P)$ and $AD_g^+(P, S)$. Moreover, $dist(S, R) = AD(S, R) + AD(R, S) = AD(R, S)$. R joins the symmetric difference distance with the asymmetric difference. R^* relates to strict consensus since $dist(S^*, R^*) = dist(Str(R^*), R^*)$.

This leads to the next theorem to establish the equivalence of two definitions of strict supertrees.

Theorem 6. *Given any profile $P = (t_1, \ldots, t_k)$, Str_s defined via symmetric difference distance and asymmetric difference are identical.*

Proof. We show that an optimal candidate tree S defined via symmetric difference distance satisfies the definition via asymmetric difference, and vice versa.

Suppose that S is an optimal candidate tree defined via symmetric difference distance. There is a $R \in \langle P \rangle_g$ such that $S = Str(R)$, and $dist(R, S) \leq dist(R', Str(R'))$ where $R' \in \langle P \rangle_g$. Since $S \subseteq T_i \in R$ for every i, $AD(S, T_i) = 0$. Therefore, $AD_g^+(S, t_i) = 0$, $AD_g^+(S, P) = 0$, and $S \in \sigma^+$.

We show that S is an optimal candidate tree via asymmetric difference by showing $AD(R, S) = ADIFF_g(P)$. Assume to the contrary, there are $S'' \in \sigma^+$ and its "bridge" $R'' \in \langle P \rangle_g$ as defined in Lemma 2 and $ADIFF_g(P) = AD(R'', S'') < AD(R, S)$.

Since $S'' = Str(R'')$, $ADIFF_g(P) = AD(R'', S'') = dist(R'', S'')$ and we have $dist(R'', S'') < AD(R, S) = dist(R, S)$, violating the fact that $dist(R, S)$ is minimum. Therefore, S must be also an optimal candidate tree via asymmetric difference.

Conversely, suppose that S is an optimal candidate tree defined via asymmetric difference. Then $S \in \sigma^+$ and $AD_g^+(P, S) = ADIFF_g(P)$. By Lemma 2 there is a "bridge" R such that $AD(R, S) = AD_g^+(P, S)$. Moreover, $S = Str(R)$ and $AD(R, S) = dist(R, S)$.

We show that $dist(R, Str(R)) \leq dist(R', Str(R'))$ for any $R' \in \langle P \rangle_g$. Assume to the contrary, there is a $R'' \in \langle P \rangle_g$ such that $dist(R'', Str(R'')) < dist(R, Str(R))$. Note that $dist(R'', Str(R'')) = AD(R'', S'')$ where $S'' = Str(R'')$ and $S'' \in \sigma^+$. Thus $AD(R'', S'') < AD(R, S) = ADIFF_g(P)$. Contradiction.

Therefore, S is an optimal candidate tree defined via symmetric distance. □

6.3 Alternative Formulation of Loose Supertrees

We show that $AD^+_{gr}(T,S) = AD^-(T,S)$ similar to what McMorris and Wilkinson have done with $AD^+_g(T,S) = AD^-(T,S)$. To obtain the smallest value of $AD(R,S)$ for $R \in \langle T\rangle_{gr}$, we need to graft leaves onto T in such a way that minimizes the number of clusters in R that are not in S. Since $R|\mathcal{L}_T$ displays T, we can achieve this minimum by adding missing leaves directly to clusters in T that are not in $S|L_T$. This minimum is precisely $AD(T, S|\mathcal{L}_T)$ and therefore $AD^+_{gr}(T,S) = AD^-(T,S)$ from which it follows that $AD^+_{gr}(P,S) = AD^-(P,S)$. Therefore, we have $AD^+_g(T,S) = AD^+_{gr}(T,S) = AD^-(T,S)$.

Since $AD^+_g(P,S) = AD^+_{gr}(P,S) = AD^-(P,S)$, $Loose_s$ can be defined via $AD^+_{gr}(P,S)$. Given a profile $P = (t_1,\ldots,t_k)$, let $\lambda = \{S : S \otimes P \text{ and } L_S = L_P\}$. Let λ'_s be the set of candidate supertrees S in λ for which $AD^+_{gr}(P,S)$ is minimum over λ. The *loose supertree* $Loose_s$ is the strict consensus of all trees in λ'.

We provide our alternative definition. Let $P = (t_1,\ldots,t_k)$ be a profile. A representative selection $R = (T_1,\ldots,T_k)$ where $T_i \in \langle t_i\rangle_{gr}$. Let $S = Str(R)$ and $dist(S,R)$ be symmetric difference distance. When $dist(S,R)$ is minimum, S is an optimal candidate tree. The loose supertree $Loose_s$ is the strict consensus of all optimal candidate trees.

Define $ADIFF_{gr}(P) = \min_{S'\otimes P} \min_{R'\in\langle P\rangle_{gr}}\{AD(R',S')\}$. Note that an optimal candidate tree S to generate $Loose_s$ requires that $S \otimes P$ and $AD^+_{gr}(P,S) \leq AD^+_{gr}(P,S')$ where $S' \otimes P$. Obviously $AD^+_{gr}(P,S) = ADIFF_{gr}(P)$.

The following lemma establishes that for any $S \otimes P$, there is a "bridge" representative selection R such that $T_i \in R$ displays S and t_i, and $AD(R,S) = AD^+_{gr}(P,S)$. That is, instead of two different representative selections R_1 and R_2 to achieve $S\otimes P$, that is, S and t_i are displayed by $T_i \in R_1$ where $R_1 \in \langle P\rangle_{gr}$, and $AD(R_2,S) = AD^+_{gr}(P,S)$, respectively, one R is enough, or $R = R_1 = R_2$. Moreover, if $S^* \in \lambda$, R^* is its bridge representative selection, and $AD(R^*,S^*) = AD^+_{gr}(P,S^*) \leq AD^+_{gr}(P,S)$ for any $S \in \sigma^+$, then $S^* = Str(R^*)$ and $ADIFF(P) = AD^+_{gr}(P,S^*)$.

Lemma 3. *Given a profile $P = (t_1, \ldots, t_k)$, and $S \otimes P$, there is a representative selection $R = (T_1, \ldots, T_k)$ where $T_i \in \langle t_i \rangle_{gr}$ such that*

(i) $T_i \setminus S$ has one-to-one correspondence to the splits in t_i that cannot be reduced by S.

(ii) $AD^+_{gr}(P, S) = AD(R, S)$.

(iii) $S \subseteq Str(R)$.

Moreover, for $S^* \in \lambda$ and $R^* \langle P \rangle_g r$ satisfying (i)-(iii) and $AD(R^*, S^*) = AD^+_{gr}(P, S^*) \leq AD^+_{gr}(P, S)$ for any $S \in \lambda$. $S^* = Str(R^*)$ and $ADIFF_{gr}(P) = AD^+_{gr}(P, S^*)$.

Proof. (i) Since $S \in \lambda$, S is compatible with every tree in P. There is a representative selection $R = (T_1, \ldots, T_k)$ where $S \subseteq T_i \in \langle t_i \rangle_{gr}$ and $T_i \setminus S$ has one-to-one correspondence to the splits in t_i that cannot be reduced by S.

(ii) We claim that $AD(R, S) = AD^+_{gr}(P, S)$. For any $A|B \in t_i$ that S cannot display, $T_i \setminus S \Rightarrow A|B$. $|T_i \setminus S|$ equals the number of such splits. This is the minimum number any $AD(T, S)$ can achieve where $T \in \langle t_i \rangle_{gr}$. Thus, $AD(R, S) = AD^+_{gr}(P, S)$.

(iii) Obviously $S \subseteq T_i$ and $S \subseteq Str(R)$.

We claim that $S^* = Str(R^*)$. Assume to the contrary, if $S^* \subset S' = Str(R^*)$, then $S' \in \lambda$, $AD(R^*, S') < AD(R^*, S^*) = AD^+_{gr}(P, S^*) \leq AD^+_{gr}(P, S')$. A contradiction. Hence, $S^* = Str(R^*)$.

From definition, $ADIFF_{gr}(P) = \min_{S' \in \lambda} \min_{R' \in \langle P \rangle_{gr}} \{AD(R', S')\} = \min_{S' \in \lambda} AD^+_{gr}(P, S') = AD^+_{gr}(P, S^*)$. □

This leads to the following theorem to establish the equivalence of two definitions of supertree.

Theorem 7. *Given any profile $P = (t_1, \ldots, t_k)$, $Loose_s$ defined via symmetric difference distance and asymmetric difference are identical.*

Proof. We show that an optimal candidate tree S defined via symmetric difference distance satisfies the definition via asymmetric difference, and vice versa.

Suppose that S is an optimal candidate tree via symmetric difference distance. There is a representative selection R such that $S = Str(R)$, and $dist(R, Str(R)) \leq dist(R', Str(R'))$ where $R' \in \langle P \rangle_{gr}$. Note that $S \otimes P$ as $S \subseteq T_i \in R$.

We show that S is an optimal candidate tree via asymmetric difference by showing $AD(R, S) = ADIFF_{gr}(P)$. Assume to the contrary, there is $S'' \otimes P$ with its "bridge" representative selection $R'' \in \langle P \rangle_{gr}$ as defined by Lemma 3, and $ADIFF_{gr}(P) = AD(R'', S'') < AD(R, S)$.

Since $S'' = Str(R'')$, $ADIFF_{gr}(P) = AD(R'', S'') = dist(R'', S'')$ and we have $dist(R'', S'') < AD(R, S) = dist(R, S)$, violating the fact that $dist(R, S)$ is minimum. Therefore, S must be also an optimal candidate tree via asymmetric difference.

Conversely, suppose that S is an optimal candidate tree defined via asymmetric difference. Then $S \otimes P$ and by Lemma 3 there is a "bridge" R such that $AD(R, S) = AD^+_{gr}(P, S) = ADIFF_{gr}(P)$. Moreover, $S = Str(R)$ and $AD(R, S) = dist(R, S)$.

We show that $dist(R, S) \leq dist(R', Str(R'))$ for any $R' \in \langle P \rangle_{gr}$. Assume to the contrary, there is a $R'' \in \langle P \rangle_{gr}$ such that $dist(R'', Str(R'')) < dist(R, S)$. Note that $dist(R'', Str(R'')) = AD(R'', S'')$ where $S'' \otimes P$. Thus $AD(R'', S'') < AD(R, S) = ADIFF_{gr}(P)$. Contradiction. Therefore, S is an optimal candidate tree defined via symmetric distance. □

There is another way to prove the results for the loose supertrees. Note that $S \in \lambda$ if and only if $AD^+_{gr}(S, t_i) = 0$. If $S \in \lambda$, S is compatible with every tree in P. There is a representative selection $R' = (T'_1, \ldots, T'_k)$ where $S \subseteq T'_i \in \langle t_i \rangle_{gr}$. Hence $AD^+_{gr}(S, t_i) = \min\{AD(S, T_i) : T_i \in \langle t_i \rangle_{gr}\} = AD(S, T'_i) = 0$.

The other direction follows similarly. If $AD^+_{gr}(S, t_i) = 0$, then there is a representative selection $R' = (T'_1, \ldots, T'_k)$ such that $AD^+_{gr}(S, t_i) = \min\{AD(S, T_i) : T_i \in \langle t_i \rangle_{gr}\} = AD(S, T'_i) = 0$. Then $S \subseteq T'_i$ which displays both t_i and S. Hence S is compatible with t_i and $S \in \lambda$.

Therefore, $S \in \lambda$ can be replaced by $AD^+_{gr}(S, t_i) = 0$ in the definition of loose supertrees. This is similar to use $AD^+_g(S, t_i) = 0$ in the definition of strict supertree. Thus the proof for loose supertrees can be done similarly with $_g$ changed to $_{gr}$.

Theorem 6 gives an alternative perspective on strict and loose supertrees. The use of the graft-refine span in loose supertrees implies that $Loose_s(P)$ can contain a split that is not

supported by all the input trees, as long as it is supported by one tree and compatible with all the trees. On the other hand, the use of the graft-only span in strict supertrees implies that a split can appear in $Str_s(P)$ only if, for each input tree, the split is either supported by that tree or it becomes trivial when reduced to the leaf set of that tree. These facts give an alternative justification to the observation that strict and loose supertrees *generalize* strict and loose consensus trees [McMorris and Wilkinson (2011)]. Formally:

Theorem 8. *For any profile P where the input trees have identical leaf sets, $Str_s(P) = Str(P)$ and $Loose_s(P) = Loose(P)$.*

Thus, we henceforth drop the subscript "s" from the notation for strict and loose supertrees and simply write "$Str(P)$" and "$Loose(P)$".

CHAPTER 7. PROPERTY OF CONSERVATIVE SUPERTREES

We study the properties of conservative supertrees and compare them with two other supertrees, namely the majority-rule supertree and the majority-rule (+) supertree, of which we include the definitions here.

Given any profile $P = (t_1, \ldots, t_k)$, let $R = (T_1, \ldots, T_k)$ be a representative selection of P where $T_i \in \langle P \rangle_g$. Let $S = Maj(R)$. If $dist(S, R) \leq dist(Maj(R'), R')$ for any other legal $R' \in \langle P \rangle_g$, S is an optimal candidate supertree. $Maj(P)$, the *majority-rule supertree*, is the strict consensus of all optimal S. If we change $\langle P \rangle_g$ into $\langle P \rangle_{gr}$, the definition becomes that of *majority-rule (+) supertree.*

We use Maj_c and Maj_s to denote majority-rule consensus tree/supertree and Maj_c^+ and Maj_s^+ to denote their majority-rule (+) counterparts.

Note that there is a unified perspective for strict, loose, majority, and majority-rule (+) supertrees. It depends on which span and consensus function to use, i.e., $\langle t_i \rangle_g$ or $\langle t_i \rangle_{gr}$, strict consensus or majority-rule consensus. They can be viewed as one family of supertrees.

7.1 Compatibility, Support and No conflict

Str_s and $Loose_s$ enjoy many desirable properties. First, any optimal candidate tree S is in every $T_i \in R$, Therefore, Str_s and $Loose_s$, as strict consensus of optimal candidate trees, are compatible with every input tree.

Second, every nontrivial split in Str_s ($Loose_s$) entails at least one input tree nontrivial full split. Assume to the contrary a nontrivial split $A|B \in Str_s$ ($Loose_s$) does not. As $A|B \in S$ for any optimal candidate tree S, $A|B$ $T_i \in R$ for every i where $R = (T_1, \ldots, T_k)$ and $T_i \in \langle t_i \rangle_g$ ($\langle t_i \rangle_{gr}$) . Construct $T_i' = T_i \setminus (A|B)$ and $R' = (T_1', \ldots, T_k')$. Note that the leaf set of T_i' is

$\mathcal{L}_P$ and thus T_i' entails all trivial splits in t_i. Since $A|B$ does not entail any nontrivial input tree full split, $T_i' \in \langle t_i \rangle_g$ ($\langle t_i \rangle_{gr}$) and R' is legal. Let $S' = Str(R')$. Obviously $S' = S \setminus (A|B)$. Note that $dist(S, R) = dist(S', R')$. Therefore, S' is also optimal and $A|B \notin Str_s$ ($Loose_s$), a contradiction.

This coincides with the support/conflict from Dong et al. (2010a), that is, any split in Str_s or $Loose_s$ is supported by at least one input tree and conflicted by no input trees.

7.2 Comparisons among Strict, Loose, Majority-rule and Majority-rule (+) Supertrees

Thee consensus tree is characterized by a special set of splits.

- Str_c = {All splits in every input tree}.
- $Loose_c$ = {All splits in at least one input tree, and compatible with the others}.
- Maj_c = {All splits in more than half of the input trees}.
- Maj_c^+ = {All splits in more input trees than contradicting input trees}.

Bryant (2003) and Dong et al. (2010b) show that in the consensus setting, $Str_c \subseteq Loose_c \subseteq Maj_c^+$, and $Str_c \subseteq Maj_c \subseteq Maj_c^+$. Profiles exist where inclusions become proper. The latter trees are more refined than the former trees.

However, examples show that none of the subset relationship holds in the supertree cases. In example 1, $P = (t_1, t_2, t_3)$ where $t_1 = \{ab|rcd, abc|rd\}$, $t_2 = \{ae|rcd, cd|rae\}$, and $t_3 = \{cd|rae, acd|re\}$. $Str_s = Loose_s = \{ab|rcde\}$ while $Maj_s = Maj_s^+ = \{cd|rabe\}$. Here $Str_s = Loose_s \nsubseteq Maj_s^+$.

In example 2, $P = (t_1, t_2, t_3)$ where $t_1 = \{bc|rad\}$, $t_2 = \{de|rbc\}$ and $t_3 = \{bc|rde, de|rbc\}$. $Str_s = \{de|rabc\}$, and $Maj_s = \{bc|rade, de|rabc\}$. $L_s = Maj_s^+ = \{bc|rade\}$. Thus $Str_s \nsubseteq L_s$, $Str_s \nsubseteq Maj_s^+$, and $Maj_s \nsubseteq Maj_s^+$.

In example 3, $P = (t_1, t_2, t_3)$ where $t_1 = \{ad|rbe, abd|re\}$, $t_2 = \{ad|rbc, abd|rc\}$ and $t_3 = \{bc|rad\}$. $Str_s = \{abcd|re\}$ and $L_s = \{ad|rbce, abcd|re\}$. $Maj_s = \{ad|rbce\}$. $Maj_s^+ = \{ad|rbce\}$. Hence $Str_s \nsubseteq Maj_s$, $Str_s \nsubseteq Maj_s^+$, $Loose_s \nsubseteq Maj_s$, and $Loose_s \nsubseteq Maj_s^+$.

7.3 NP-hardness Results

Theorem 9. *There is no polynomial-time algorithm to construct an optimal candidate supertree for Str_s, $Loose_s$ and Maj_s unless $P = NP$.*

Proof. We show that if there is a polynomial time algorithm to compute an optimal candidate tree, then there exists a polynomial-time algorithm for the quartet compatibility problem, which is known to be NP-complete [Steel (1992)]. The quartet compatibility problem asks whether, given a collection Q of trees on four leaves, there exists a single tree that displays them all. If the answer is "yes", we say that Q is *compatible*.

Let Q be an instance of quartet compatibility. Construct a profile P that consists of the trees in Q in some arbitrary order. We claim that Q is compatible if and only if P has an optimal candidate tree with a score of zero. Suppose first that Q is compatible and that T is any tree displaying each tree in P. Since a quartet is fully resolved, T reduces to it. Then, for every $t \in P$, $T \in \langle t \rangle_g \subseteq \langle t \rangle_{gr}$. Let R be a representative selection with k copies of T, $dist(T, R) = dist(Str(R), R) = dist(Maj(R), R) = 0$. Thus T is an optimal candidate tree. Conversely, if R has a candidate tree with zero score, it can be seen that T displays and reduces into all the quartets in Q; i.e., Q is compatible. □

7.4 Reducible Set and Pull-out Set

The basic Integer Linear Programming (ILP) formulations described in a later chapter allow us to solve supertree problems of moderate size. Here we explore two properties of the strict and loose OCT respectively that allow us to extend the range of our method significantly in practice.

Every edge e in a conservative OCT T is in every $T_i \in R$, the representative selection. it is either an edge in t_i or a pulled out edge (refinement) extended from an internal node in t_i. Either way, e cuts the t_i into two subtrees, P into two profiles – a satellite profile P_{sat} and a planet profile P_{pla}, T into two trees T_{sat} and T_{pla}. Obviously $dist(T, R) = dist(T_{sat}, R_{sat}) + dist(T_{pla}, R_{pla}) = dist(Str(R_{sat}), R_{sat}) + dist(Str(R_{pla}), R_{pla})$. Note that T is OCT if and only

if the two subtrees are be OCT over the two sub-profiles. Therefore, we can divide the original optimization problem into two optimization subproblems.

The reducible set defined previously is applicable to the strict OCT since any reduced cluster from the strict OCT has to be a cluster of the input tree that it reduces to. For loose OCT, its counterpart is a *pull-out* set. We say that $S \subseteq \mathcal{L}(P)$ with $1 < |S| < |\mathcal{L}(P)| - 1$ is a *pull-out set* if, for each $j \in K$, S is compatible with t_j.

Note that $S \cap L_j$ is compatible with t_j if and only if S is compatible with t_j. $t_j|(S \cap L_j)$ is the subtree in t_j that is "pulled out". Figure 7.1 shows a typical pull-out set which refines the multiplication at an internal node in an input tree to create an additional cluster. A pull-out set can be a reducible set.

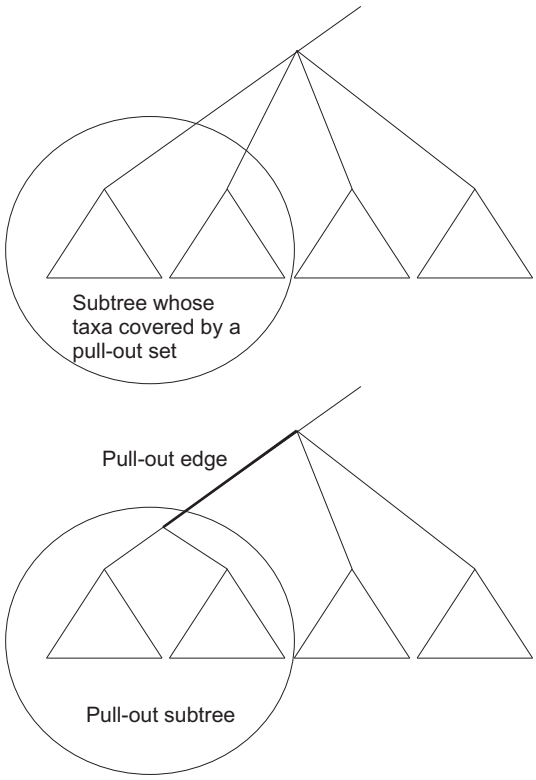

Figure 7.1 Pull-out Set

The series of concepts for substructure analysis using reducible sets can be used for pull-out sets as well. The generation of planet profile using these two sets is exactly the same. That is, if a taxon is part of the reducible or pull-out set, then it is replaced by the super taxon. When

duplicates are removed, the profile becomes the planet profile.

However, the generation of satellite profile differs. In the reducible set case, each input tree has a subtree that is cut off by the reducible set. On the contrary, in the pull-out set case, if a pull-out set cuts out several subtrees of a polytomy internal node while leaving other subtrees intact, what is cut out is not a tree! Therefore, we need to add an envelope cluster which is the union of the cut out taxa from the input tree to make a pulled-out subtree. This envelope cluster corresponds to the pull-out edge that connects the pulled-out subtree with the original internal node.

7.5 Invariant Under Remeshing

There are two meanings for *invariant under remeshing*. First, if a supertree is indeed the strict/loose OCT, then using its splits to divide it in different ways into substructures and solve again will yield the same or a different strict/loose OCT with the same objective value. Second, if a supertree is indeed the strict/loose supertree, then using its splits to divide it in different ways into substructures and solve again yields the same strict/loose supertree.

However, if a supertree is not a strict/loose OCT, and if the selected split to divide the profile is not in any of the OCT, the tree found is not optimal, that is, its objective value is higher. Hence we know what we have found is not optimum.

7.6 Lack of Population Invariance

In the consensus settings, strict and loose trees are population invariant. If one tree objects to certain cluster, no matter how many other trees support it, it cannot be in the consensus tree. Increasing the number of supporting trees have no effect on the consensus.

However, it is no longer true in the supertree settings. Consider the following counterexample. Let $P_1 = (t_1, \ldots, t_{17})$ where $t_1 = \cdots = t_9 = \{be|rcd, cd|rbe\}$, $t_{10} = \cdots = t_{12} = \{ac|rde\}$, and $t_{13} = \cdots = t_{17} = \{ab|rde, abe|rd\}$. The strict and loose supertree are $\{ab|rcde\}$. Let $P_2 = (t_1, \ldots, t_{16})$ where $t_1 = \cdots = t_{10} = \{be|rcd, cd|rbe\}$, $t_{11} = \cdots = t_{13} = \{ac|rde\}$, and $t_{14} = \cdots = t_{16} = \{ab|rde, abe|rd\}$. The strict and loose supertrees are fans.

Were strict and loose supertrees population invariant, P_1 and P_2 would have the same conservative supertrees, since they include the same three types of trees.

CHAPTER 8. ILP FORMULATION FOR CONSERVATIVE SUPERTREES, THE (+) WAY

8.1 Restricted Spans and Spans of Restricted Supertree

Despite the NP-completeness result, moderately large conservative supertree problems can be solved with integer linear programming, similar to what we have done in Dong et al. (2010a) for the majority-rule (+) supertree.

For $T \in \langle t \rangle_{gr}$ (or $\langle t \rangle_g$), $|Spl(T)| \geq |Spl(t)|$. Since $|Spl(T)|$ is not predetermined, the computation on $\langle t \rangle_{gr}$ or $\langle t \rangle_g$ are transformed to that on the restricted span as defined below.

Given $t \in P$, the *restricted span of a tree* t, denoted $\langle t \rangle_r$, is the set of all plenary trees T such that every nontrivial split in T extends a distinct nontrivial split in t, i.e., $|Spl(T)| = |Spl(t)|$. Note also that $\langle t \rangle_r \subseteq \langle t \rangle_g \subseteq \langle t \rangle_{gr}$. Any $T \in \langle t \rangle_r$ is a *restricted supertree* of t.

The *restricted span of a profile* $P = (t_1, \ldots, t_k)$, denoted $\langle P \rangle_r$ is the set of all $R = (T_1, \ldots, T_k)$ for P such that $T_i \in \langle t_i \rangle_r$ for $i = 1, \ldots, k$. Any $R \in \langle P \rangle_r$ is a *restricted representative selection* for P.

The restricted trees have their own spans. Let $T \in \langle t \rangle_r$. The *elaborated span of a restricted tree* T, denoted $\langle T \rangle_e$, is the set of supertrees that display T. Here only refinement is involved since T and $\langle T \rangle_e$ have the same set of taxa.

The *t-reducible span of the restricted tree* T, denoted $\langle T \rangle_t$, is the set of supertrees that display T *and* reduce to t. Obviously, $\langle T \rangle_t \subseteq \langle T \rangle_e$. Note that $\langle T \rangle_e \subseteq \langle t \rangle_{gr}$ and $\langle T \rangle_t \subseteq \langle t \rangle_g$.

Similarly, for a profile P and a restricted representative selection $R \in \langle P \rangle_r$, $\langle R \rangle_e$ is the set of representative selections that display R. $\langle R \rangle_P$ is the set of representative selections that display R *and* reduce to P.

8.2 Characterization of Various Consensus Trees

Given a profile $P = (t_1, \ldots, t_k)$, and a restrictive representative selection $R = (T_1, \ldots, T_k) \in \langle P \rangle_r$. We define two consensus trees via the distance optimality route. Let $\widehat{R} = (\widehat{T_1}, \ldots, \widehat{T_k})$ be any representative selection such that $\widehat{R} \in \langle R \rangle_P(\langle R \rangle_e)$. If $dist(Str(\widehat{R}), \widehat{R}) \leq dist(Str(\widehat{R'}), \widehat{R'})$ for any $\widehat{R'} \in \langle R' \rangle_P(\langle R' \rangle_e)$, S is an optimal candidate tree. The reducibly-loose consensus tree $RL(R)$ (the loose consensus tree, $Loose_c(R)$), is the strict consensus of all optimal S. Denote $OCT(R)$ as the set of all optimal S.

Correspondingly we define two completions of R. The *L-completion* of $R \in \langle P \rangle_r$ is $\widehat{R} = (\widehat{T_1}, \ldots, \widehat{T_k})$ where $\widehat{T_i} = T_i \cup Loose_c(R)$. The *RL-completion* of R is $\widehat{R} = (\widehat{T_1}, \ldots, \widehat{T_k})$ where $\widehat{T_i} = T_i \cup RL(R)$.

We characterize the reducibly-loose consensus and loose consensus tree similar to that of the majority-rule (+) consensus tree [Dong et al. (2010b)]. The notation in Dong et al. (2010b) are modified to reflect the reducibility requirement.

If a split $X \in T_i$, X is compatible with T_i and reducible into t_i. For notation purposes, let $K = \{1, \ldots, k\}$. Define

$$K_X(R) = \{i \in K : X \text{ is displayed by } T_i\},$$

$$K_{\overline{X}}(R) = \{i \in K : X \text{ is incompatible with } T_i\},$$

and

$$K_{\underline{X}}(R) = \{i \in K : X \text{ is incompatible with } T_i \text{ or } X \text{ is not reducible to } t_i\}.$$

Lemma 4. *For any profile P, $\{X : |K_X(P)| > |K_{\underline{X}}(P)|\}$ is compatible.*

Proof. Assume by way of contradiction that there exists a profile P and $A, B \in P$ with $|K_A(P)| > |K_{\underline{A}}(P)|$ and $|K_B(P)| > |K_{\underline{B}}(P)|$, such that A and B are not compatible. Thus, $K_A(P) \subseteq K_{\underline{B}}(P)$ and $K_B(P) \subseteq K_{\underline{A}}(P)$. So

$$|K_A(P)| > |K_{\underline{A}}(P)| \geq |K_B(P)| > |K_{\underline{B}}(P)| \geq |K_A(P)|$$

an impossibility. □

The following is our key characterization theorem, which significantly simplifies the construction of $RL(R)$ and $Loose_c(R)$.

Theorem 10. *Given profile $P = (t_1, \ldots, t_k)$ and $G = (T_1, \ldots, T_k) \in \langle P \rangle_r$,*

(i) $RL(G)$ is the set S of plenary loose splits X in G that reduce into every tree in P, i.e., $|K_X(G)| > 0$ and $|K_{\underline{X}}(G)| = 0$.

(ii) $Loose_c(G)$ is the set S of plenary loose splits X in G, i.e., $|K_X(G)| > 0$ and $|K_{\overline{X}}(G)| = 0$.

(iii) $RL(G)(Loose_c(G))$ is an optimal candidate supertree for $RL(G)$ $(Loose_c(G))$.

(iv) There is a unique representative selection R, which is the RL-completion (L-completion) of G, when $dist(RL(G), R)$ $(dist(Loose_c(G), R))$ reaches minimum.

Proof. We show (i)-(ii) simultaneously by enumerating through $RL(G)$ and $Loose_c(G)$. We first show that

1. $RL(G) \subseteq \{X : |K_X(G)| > 0 \text{ and } |K_{\underline{X}}(G)| = 0\}$,
2. $Loose_c(G) \subseteq \{X : |K_X(G)| > 0 \text{ and } |K_{\overline{X}}(G)| = 0\}$

respectively for any restricted representative selection $G = (g_1, \ldots, g_k)$ generated from any profile $P = (t_1, \ldots, t_k)$. Let $R = (T_1, \ldots, T_k)$ where $R \in \langle G \rangle_P (\langle G \rangle_e)$,$T = Str(R)$ and $dist(T, R)$ is minimum. That is, T is an optimal tree for G. Assume that there exists $A \in T$ such that $|K_A(G)| = 0$ or $|K_{\underline{A}}(G)| > 0$ $(|K_{\overline{A}}(G)| > 0)$. We simplify the assumption such that $|K_A(G)| = 0$ and $|K_{\underline{A}}(G)| = 0$ $(|K_{\overline{A}}(G)| = 0)$ since if $|K_{\overline{A}}(G)| > 0$ $(|K_{\underline{A}}(G)| > 0)$, A cannot be in every tree in R, thus $A \notin T = Str(R)$. Construct $R' = (T'_1, \ldots, T'_k)$, where $T'_i = T_i \setminus \{A\}$ if $A \in (T_i \setminus g_i)$, and $T'_i = T_i$ otherwise. Note that $K_A(R') = K_A(G)$, and clearly $|K_A(R')| = 0$ so that $A \notin Str(R')$, In fact, $Str(R') = Str(R) \setminus \{A\}$. Let $T' = Str(R')$ so $T' = T \setminus \{A\}$.

If $A \notin T_i$, then $T_i = T'_i$ and

$$d(T, T_i) = d(T', T'_i) + 1.$$

If $A \in T'_i$, then $T_i = T'_i$ and

$$d(T, T_i) = d(T', T'_i) - 1.$$

If $A \in T_i$ with $A \notin T_i'$, then

$$d(T, T_i) = d(T', T_i').$$

Thus we have

$$\sum_{i=1}^{k} d(T, T_i) - (k - |K_A(R)|) = \sum_{i=1}^{k} d(T', T_i') - |K_A(R')| = \sum_{i=1}^{k} d(T', T_i') - |K_A(G)|.$$

Since $R \in \langle G \rangle_P(\langle G \rangle_e)$, we have

1. $|K_A(R)| \leq k - |K_{\underline{A}}(G)|$,
2. $|K_A(R)| \leq k - |K_{\overline{A}}(G)|$

respectively so that

1. $|K_{\underline{A}}(G)| \leq k - |K_A(R)|$,
2. $|K_{\overline{A}}(G)| \leq k - |K_A(R)|$,

respectively.

From the above we then have

1. $\sum_{i=1}^{k} d(T, T_i) \geq \sum_{i=1}^{k} d(T', T_i') - |K_A(G)| + |K_{\underline{A}}(G)| \geq \sum_{i=1}^{k} d(T', T_i')$,
2. $\sum_{i=1}^{k} d(T, T_i) \geq \sum_{i=1}^{k} d(T', T_i') - |K_A(G)| + |K_{\overline{A}}(G)| \geq \sum_{i=1}^{k} d(T', T_i')$,

respectively.

Therefore, $R' \in \langle G \rangle_P(\langle G \rangle_e)$, $T' = Str(R')$ and $dist(T', R')$ is minimum, so that T' is an optimal tree of G, too. Since $RL(G)(Loose_c(G)) \subseteq T'$, and $A \notin T'$, $A \notin RL(G)(Loose_c(G))$. Thus we have shown

1. $RL(G) \subseteq \{X : |K_X(G)| > 1 \text{ and } |K_{\underline{X}}(G)| = 0\}$,
2. $Loose_c(G) \subseteq \{X : |K_X(G)| > 1 \text{ and } |K_{\overline{X}}(G)| = 0\}$,

respectively.

We prove an additional result here. For all optimal candidate tree T of G for the two supertrees,

1. $T \subseteq \{X : |K_X(G)| \geq 0 \text{ and } |K_{\underline{X}}(G)| = 0\}$,and

2. $T \subseteq \{X : |K_X(G)| \geq 0 \text{ and } |K_{\overline{X}}(G)| = 0\}$,

respectively. Assume to the contrary that there is $A \in T$ such that $|K_{\underline{A}}(G)| > 0 (|K_{\overline{A}}(G)| > 0)$. Then A cannot be in every tree in any representative selections. Hence $A \notin T$, contradiction.

For the reverse set inclusion, we first claim that if $R = (T_1, \ldots, T_k)$, $R \in \langle G \rangle_P (\langle G \rangle_e)$, $T = Str(R)$, and $dist(T, R)$ is minimum, then $(T_i \setminus t_i) \subseteq T$ for all $i \in K$. To see this, suppose there exists j and Y such that $Y \in (T_j \setminus t_j)$ with $Y \notin T$. Let $R' = (T'_1, \ldots, T'_k)$ be defined where $T'_j = T_j \setminus \{Y\}$ and $T'_i = T_i$ for all $i \neq j$. Thus $R' \in \langle G \rangle_P (\langle G \rangle_e)$, $Str(R') = T$, and

$$\sum_{i=1}^{k} d(T, T'_i) = \sum_{i=1}^{k} d(T, T_i) - 1$$

which contradicts the assumption that $dist(T, R)$ is minimum.

Now assume that there exists an X such that $|K_X(G)| > 1$ and

1. $|K_{\underline{X}}(G)| = 0$, $X \notin RL(G)$, and

2. $|K_{\overline{X}}(G)| = 0$, $X \notin Loose_c(G)$,

respectively. From the definition of $RL(G)$ ($Loose_c(G)$), there exists $R = (T_1, \ldots, T_k)$ such that $R \in \langle G \rangle_P (\langle G \rangle_e)$, $T = Str(R)$, $dist(T, R)$ is minimum, and $X \notin Str(R)$. Since $T = Str(R)$ and T is an optimal candidate tree, then $|K_Y(G)| \geq 0$ and $|K_{\underline{Y}}(G)| = 0 (|K_{\overline{Y}}(G)| = 0)$ for all $Y \in T$. If there exists $Y \in T$ such that X is not compatible with Y, then

1. $K_X(G) \subseteq K_{\underline{Y}}(G) = \emptyset$, and $K_Y(G) \subseteq K_{\underline{X}}(G) = \emptyset$,

2. $K_X(G) \subseteq K_{\overline{Y}}(G) = \emptyset$, and $K_Y(G) \subseteq K_{\overline{X}}(G) = \emptyset$

respectively, Therefore $|K_X(G)| = 0$, which is a contradiction. Thus T and X are compatible. From the claim proved above, $(T_i \setminus t_i) \subseteq T$ for all i so that $T_i \setminus t_i$ and X are compatible for all $i \in K$.

Define $R' = (T'_1, \ldots, T'_k)$ as follows: $T'_i = T_i \cup \{X\}$ if T_i and X are compatible and $T'_i = T_i$ otherwise. Observe that $|K_X(R')| = k$, so that $X \in Str(R') = T'$ and, noting that $X \notin T$ while

$X \in T'$, $\sum_{i=1}^{k} d(T, T_i) = \sum_{i=1}^{k} d(T', T'_i) + |K_X(G)| > \sum_{i=1}^{k} d(T', T'_i)$, which contradicts T is an optimal tree.

Now we prove (iii). Choose any $R = (T_1, \ldots, T_k)$ such that $R \in \langle G \rangle_P(\langle G \rangle_e)$, $dist(Str(R), R)$ is minimum, and let $T = Str(R)$. If $RL(G) = T(Loose_c(G) = T)$, we are done, so suppose $RL(G) \subset T(Loose_c(G) \subset T)$. Then any cluster $A \in (T \setminus RL(G))(A \in (T \setminus Loose_c(G)))$, must have $|K_A(G)| = 0$ and $|K_{\underline{A}}(G)| = 0$ ($|K_{\overline{A}}(G)| = 0$). Note that if $|K_{\underline{A}}(G)| > 0$ ($|K_{\overline{A}}(G)| > 0$), $A \notin T$, since it is not in every tree of any representative selection. Since $A \notin Str(G)$ and $A \in Str(R)$, there is an $i \in K$ such that $A \in (T_i \setminus t_i)$. Construct $T' = T \setminus \{A\}$ and $R' = (T'_1, \ldots, T'_k)$ where $T'_i = T_i \setminus \{A\}$ if $A \in (T_i \setminus t_i)$ and $T'_i = T_i$ otherwise. We have that $R' \in \langle G \rangle_P(\langle G \rangle_e)$, $T' = Str(R')$, and $A \notin T'$. Since $|K_A(R')| = 0$, $dist(T', R') \leq dist(T, R)$. Thus T' is an optimal tree of G. Applying this construction for every cluster in $T \setminus RL(G)(Loose_c(G))$, we conclude that $RL(G)(Loose_c(G))$ is an optimal candidate tree of G, respectively.

Now we prove (iv). Here we prove a more general statement. That is, for every optimal candidate tree T, there is a unique optimal representative selection R such that $R \in \langle G \rangle_P(\langle G \rangle_e)$, $T = Str(R)$, and $dist(T, R)$ is minimum.

Let $P = (t_1, \ldots, t_k)$. The existence of a $R = (T_1, \ldots, T_k)$ such that $R \in \langle G \rangle_P(\langle G \rangle_e)$, $T = Str(R)$, and $dist(T, R)$ is minimum follows from the definition of optimal candidate tree and the optimal representative selections. Recall again in the proof of (i)-(ii) we showed that since T is optimal, $(T_i \setminus t_i) \subseteq T$ for all $i \in K$. We prove that R is unique by showing that $X \notin T$ implies $K_X(R) = K_X(G)$ whereas $X \in T$ implies $K_X(R) = K$.

Suppose $X \notin T$. Then $X \notin (T_i \setminus t_i) \subseteq T$ for all $i \in K$. Since $R \in \langle G \rangle_P(\langle G \rangle_e)$, it follows that $X \in T_i$ whenever $X \in t_i$ and hence $K_X(R) = K_X(G)$.

Now let $X \in T$ and note that $K_X(R) = K$ since $T = Str(R)$. □

8.3 Strict and Loose Supertrees via Restricted Spans

We show below how to obtain conservative supertrees directly from the restricted span. An optimal candidate supertree S for a profile P is *minimal* if contracting any edge in S yields a tree that is not an optimal candidate supertree.

Theorem 11. *Let S be a minimal optimal candidate supertree for a profile P for strict (loose) supertree. Let $R \in \langle P\rangle_g(\langle P\rangle_e)$, and $S = Str(R)$. Consider any $G \in \langle P\rangle_r$ such that $R \in \langle G\rangle_P(\langle G\rangle_e)$. Then, R is the RL-completion (L-completion) of G and $S = RL(G)(Loose_c(G))$,*

Proof. We first show that S is an optimal candidate supertree for G in the two supertrees.

Assume the contrary. Then, there exists another candidate tree S' for G such that (i) $S' = Str(R')$ for some $R' \in \langle G\rangle_P(\langle G\rangle_e)$ and (ii) $s(R') < s(R)$ where $s(R) = dist(Str(R), R)$. But then, since $\langle G\rangle_P \subseteq \langle P\rangle_g(\langle G\rangle_e \subseteq \langle P\rangle_{gr})$, we have $R' \in \langle P\rangle_g(\langle P\rangle_{gr})$, and thus (ii) contradicts the optimality of S for P.

Next, we argue that S is a *minimal* optimal candidate supertree for profile G. Suppose this is not true. Then, S displays an optimal candidate supertree S' for G such that $S \neq S'$. Consider any $R' \in \langle G\rangle_P(\langle G\rangle_e)$ such that $S' = Str(R')$. Since S and S' are both optimal for G, $s(R) = s(R')$. Since R' displays P, we have $R' \in \langle P\rangle_g(\langle P\rangle_{gr})$. Hence, S' is also an optimal candidate supertree for P. This, however, contradicts the assumption that S is a minimal optimal candidate tree for P.

By theorems of last section, $RL(G)$ ($Loose_c(G)$) is an optimal candidate supertree for G, as well as the strict consensus of all optimal candidate supertrees for G. Therefore, $RL(G)$ ($Loose_c(G)$) is the only minimal optimal candidate supertree for G. Hence $T = RL(G)(Loose_c(G))$.

Due to the uniqueness of R when $dist(RL(G), R)$ ($dist(Loose_c(G), R)$) is optimum, R is the RL-completion (L-completion) of G. □

Motivated by Theorem 11, we define the *adjusted score* of a representative selection R for a profile P, denoted $\widehat{s}(R)$, to be the score of the RL-completion (L-completion) $\widehat{R}$ of R; i.e., $\widehat{s}(R) = s(\widehat{R})$.

Theorem 12. *Let P be a profile. Let $\widehat{R}$ be*

1. *RL-completion of R, or*

2. *L-completion of R,*

respectively. Let $\widehat{s}(R) = s(\widehat{R}) = dist(Str(\widehat{R}, \widehat{R}))$*, Define* $\mathcal{G} = \{G \in \langle P\rangle_r : \widehat{s}(G)$ *is minimum*$\}$ *and* $\mathcal{S} = \{T = RL(G)(Loose_c(G)) : G \in \mathcal{G}\}$*. Then,* $Str_s(P)(Loose_s(P))$ *is the strict consensus of* $\mathcal{S}$*.*

Proof. Let $\mathcal{O}$ be the set of all optimal candidate supertrees for P and let $\mathcal{M}$ be the set of all minimal optimal candidate supertrees of P. In what follows, we show that $\mathcal{M} \subseteq \mathcal{S} \subseteq \mathcal{O}$. This immediately implies the theorem, because not only is (by definition) $Str_s(P)(Loose_s(P))$ the strict consensus of $\mathcal{O}$, but it must also be the strict consensus of $\mathcal{M}$.

Suppose $T \in \mathcal{M}$. We claim that $T \in \mathcal{S}$ and, therefore, that $\mathcal{M} \subseteq \mathcal{S}$. Let R be a representative selection for P such that $T = Str(R)$. Let $G \in \langle P\rangle_r$, and $R \in \langle G\rangle_P(\langle G\rangle_e)$. By Theorem 11, $T = RL(G)(Loose_c(G))$ and R is the RL-completion (L-completion) of G. We claim that $G \in \mathcal{G}$; i.e., $\widehat{s}(G)$ is minimum. Assume, by way of contradiction, that there is another $G' \in \langle P\rangle_r$ such that $\widehat{s}(G') < \widehat{s}(G)$. Let R' be the RL-completion (L-completion) of G'. Then, $s(R') = \widehat{s}(G') < \widehat{s}(G) = s(R)$, which contradicts the assumption that T is optimal. Therefore, $\widehat{s}(G)$ is minimum and $T \in \mathcal{S}$.

Suppose $T \in \mathcal{S}$. We claim that $T \in \mathcal{O}$ and, therefore, that $\mathcal{S} \subseteq \mathcal{O}$. Let $G \in \langle P\rangle_r$ be such that $T = RL(G)(Loose_c(G))$ and the adjusted score $\widehat{s}(G)$ is minimum. Let R be the RL/L-completion of G. Assume, by way of contradiction, that $T \notin \mathcal{O}$. Then there is a $T' \in \mathcal{M}$ such that, if R' is a representative selection for P where $T' = Str(R')$, then $s(R') < s(R)$. By Theorem 11, there is a $G' \in \langle P\rangle_r$ such that $T' = RL(G')(Loose_c(G'))$ and $\widehat{s}(G') = s(R')$. Then $\widehat{s}(G') = s(R') < s(R) = \widehat{s}(G)$. This contradicts the assumption that $\widehat{s}(G)$ is minimum. □

8.4 ILP Formulations

8.4.1 Normal Profiles

The ILP formulation is similar to that in Dong et al. (2010a). Based on Theorem 12, we seek a $G \in \langle P\rangle_r$ with minimum adjusted score.

Let $P = (t_1, \ldots, t_k)$ be a profile where $|\mathcal{L}(P)| = n$. Assume t_j has m_j nontrivial splits. A *matrix representation* of t_j is a $n \times m_j$ matrix $M(t_j)$ whose columns correspond to the nontrivial splits of t_j. Suppose column i of $M(t_j)$ corresponds to split $A|B$ in t_j and let x be a taxon in

$\mathcal{L}(P)$. Then, $M_{x,i}(t_j) = 1$ if $x \in A$, $M_{x,i}(t_j) = 0$ if $x \in B$, and $M_{x,i}(t_j) =?$ otherwise. A rooted tree is converted to an unrooted one by adding a special root r. All taxa in the same block as r are assigned 1. Let $m = \sum_{j \in K} m_j$. A *matrix representation* of P, denoted $M(P)$, is a $n \times m$ matrix $M(P)$ obtained by concatenating matrices $M(t_1), M(t_2), \ldots, M(t_k)$.

A *fill-in* of $M(P)$ is a matrix representation for G. A binary *fill-in variable* F_{xi} is created if $M_{xi}(P) =?$. The ILP ensures a legal G by ensuring compatibility within each restricted supertree, and in the case of strict and majority-rule supertrees, also reducibility to underlying P.

For compatibility, we define variables C_{pq}, $1 \leq p, q \leq m$ such that $C_{pq} = 1$ precisely if columns p and q are compatible. We require that $C_{pq} = 1$ for every p, q that correspond to splits in the same input tree. The splits p and q are incompatible precisely if 00, 01, 10, and 11 all appear in some rows of columns p and q (the "four gametes condition"). The presence or absence of these patterns for columns p and q is indicated by the settings of variables $B_{pq}^{(ab)}$, $a, b \in \{0, 1\}$, where $B_{pq}^{(ab)} = 1$ if and only if there is a taxon r such that $F_{rp} = a$ and $F_{rq} = b$. The $B_{pq}^{(ab)}$s are determined from the settings of variables $\Gamma_{rpq}^{(ab)}$, where r ranges over the taxa (i.e., the rows of $M(P)$). The Γ variables satisfy $\Gamma_{rpq}^{(ab)} \Leftrightarrow ((F_{rp} = a) \wedge (F_{rq} = b))$, or

$$\begin{aligned} (-1)^a F_{rp} + (-1)^b F_{rq} + \Gamma_{rpq}^{(ab)} &\geq 1 - a - b, \\ (-1)^a F_{rp} + (-1)^b F_{rq} + 2\Gamma_{rpq}^{(ab)} &\leq 2 - a - b. \end{aligned} \tag{8.1}$$

We have that $B_{pq}^{(ab)} \Leftrightarrow \bigvee_r \Gamma_{rpq}^{(ab)}$, or

$$\begin{aligned} -\sum_r \Gamma_{rpq}^{(ab)} + B_{pq}^{(ab)} &\leq 0, \\ \sum_r \Gamma_{rpq}^{(ab)} - n B_{pq}^{(ab)} &\leq 0 \end{aligned} \tag{8.2}$$

Observe that $\neg C_{pq} \Leftrightarrow B_{pq}^{(00)} \wedge B_{pq}^{(01)} \wedge B_{pq}^{(10)} \wedge B_{pq}^{(11)}$, or

$$\begin{aligned} B_{pq}^{(00)} + B_{pq}^{(01)} + B_{pq}^{(10)} + B_{pq}^{(11)} + 4C_{pq} &\geq 4, \\ B_{pq}^{(00)} + B_{pq}^{(01)} + B_{pq}^{(10)} + B_{pq}^{(11)} + C_{pq} &\leq 4. \end{aligned} \tag{8.3}$$

We define binary variables E_{pq}, $1 \leq p, q \leq m$, where $E_{pq} = 1$ if and only if columns p and q of the filled-in matrix represent the same split. Here we have to make a distinction between

rooted and unrooted trees. In the rooted case, being equivalent translates to being identical since there exists a root taxon r such that $M_{ri}(P) = 1$ for every column i. In the unrooted tree case, there are two ways to represent the same split.

For the rooted case, $E_{pq} \Leftrightarrow \neg B_{pq}^{(01)} \wedge \neg B_{pq}^{(10)}$, or

$$\begin{aligned} B_{pq}^{(01)} + B_{pq}^{(10)} + 2E_{pq} &\leq 2, \\ B_{pq}^{(01)} + B_{pq}^{(10)} + E_{pq} &\geq 1. \end{aligned} \tag{8.4}$$

In describing the constraints for the $S^{(1)}$ and $S^{(2)}$ variables, we adopt the convention that the splits of the jth tree correspond to columns $j_1, \ldots, j_d$ of $M(P)$. Then, $S_{ij}^{(1)} \Leftrightarrow E_{ij_1} \oplus \cdots \oplus E_{ij_d}$, or

$$S_{ij}^{(1)} - \sum_{r=1}^{d} E_{ij_r} = 0. \tag{8.5}$$

On the other hand, $S_{ij}^{(2)} \Leftrightarrow C_{ij_1} \wedge \cdots \wedge C_{ij_d}$, or

$$\begin{aligned} d \cdot S_{ij}^{(2)} - \sum_{r=1}^{d} C_{ij_r} &\leq 0, \\ 1 - d - S_{ij}^{(2)} + \sum_{r=1}^{d} C_{ij_r} &\leq 0. \end{aligned} \tag{8.6}$$

For reducibility requirements of strict OCT, define *mk split-tree reducibility* binary variables $S^{(3)}$ such that $S_{ij}^{(3)} = 1$ precisely if split i is reducible into tree j as follows.

1. Split i becomes equivalent to a nontrivial split j_p in tree j, represented by $R_{ij_p} = 1$.

2. Split i becomes a trivial split in tree j, in two forms discussed below and represented by $Trivial_{ij}^{(1)}$ and $Trivial_{ij}^{(2)}$, respectively.

We have $S_{ij}^{(3)} \Leftrightarrow R_{ij_1} \oplus \cdots \oplus R_{ij_d} \oplus Trivial_{ij}^{(1)} \oplus Trivial_{ij}^{(2)}$ where j_1 and j_d are the first and last nontrivial splits in the j^{th} tree respectively. Equivalently we have

$$-S_{ij}^{(3)} + R_{ij_1} + \cdots + R_{ij_d} + Trivial_{ij}^{(1)} + Trivial_{ij}^{(2)} = 0 \tag{8.7}$$

R represents m^2 *split-split reducibility* binary variables. $R_{ij} = 1$ precisely if the reduced ith split and the jth split are equivalent.

For rooted trees, $R_{ij} = 1$ if and only if for all rows r that $M(r, j)$ is known, $M(r, i) = M(r, j)$. If in any row r, $M(r, i)$ and $M(r, j)$ are known but $M(r, j) \neq M(r, i)$, then $R_{ij} = 0$. Assume

that there are p rows where $M(r,i) =?$ and $M(r,j) = 1$, and there are $q-p$ rows where $M(r,i) =?$ and $M(r,j) = 0$. If the p rows range from r_1 to r_p and the $q-p$ rows range from r_{p+1} to r_q, we have $R_{ij} \Leftrightarrow F_{r_1 i} \wedge \cdots \wedge F_{r_p i} \wedge (1 - F_{r_{p+1} i}) \wedge \cdots \wedge (1 - F_{r_q i})$, or

$$\begin{aligned} q \cdot R_{ij} - \sum_{\ell=1}^{p} F_{r_\ell i} + \sum_{\ell=p+1}^{q} F_{r_\ell i} \leq q - p, \\ R_{ij} - \sum_{\ell=1}^{p} F_{r_\ell i} + \sum_{\ell=p+1}^{q} F_{r_\ell i} \geq 1 - p. \end{aligned} \tag{8.8}$$

For the rooted trees, define *mk Type I split-tree trivially-reducibility* variables $Trivial^{(1)}$ such that $Trivial^{(1)}_{ij} = 1$ precisely if split i reduces trivially into tree j in the forms of $\emptyset | r(L_j)$ or $x | r(L_j \setminus x)$ where L_j is the leaf set (excluding root) of tree j and x is a single taxa. Let $q = |L_j|$. $Trivial^{(1)}_{ij} \Leftrightarrow$ at least $q-1$ out of q $F_{r_p i}$ is 1, where $p = 1, \ldots, q$. In other words, we have $Trivial^{(1)}_{ij} \Leftrightarrow \sum_{p=1}^{q} F(r_p, i) \geq q - 1$, or

$$\begin{aligned} (q-1) Trivial^{(1)}_{ij} - \sum_{p=1}^{q} F_{r_p i} \leq 0, \\ 2 \cdot Trivail^{(1)}_{ij} - \sum_{p=1}^{q} F_{r_p i} \geq 2 - q. \end{aligned} \tag{8.9}$$

Define *mk Type II split-tree trivially-reducibility* variables $Trivial^{(2)}$. $Trivial^{(2)}_{ij} = 1$ precisely if split i reduces trivially into tree j in the forms of $L_j | r$, where r is the special taxon for the root of tree j. Let $q = |L_j|$. $Trivial^{(2)}_{ij} \Leftrightarrow \neg F_{r_1 i} \wedge \cdots \wedge \neg F_{r_q i}$. Equivalently we have

$$\begin{aligned} q \cdot Trivial^{(2)}_{ij} + \sum_{p=1}^{q} F_{r_p i} \leq q, \\ Trivail^{(2)}_{ij} + \sum_{p=1}^{q} F_{r_p i} \geq 1. \end{aligned} \tag{8.10}$$

With the above variables and constraints, we already ensure that the filling is legal. Next, we need another set of variables and constraints to ensure its optimality. The objective is to minimize $\widehat{s}(G)$ over all $G \in \langle P \rangle_r$. By definition, $\widehat{s}(G) = dist(RL(G), R)(dist(Loose_c(G), R))$, where $R = (\widehat{T}_1, \ldots, \widehat{T}_k)$ is the RL-completion (L-completion) of $G = (T_1, \ldots, T_k)$. We do not, however, construct the consensus trees and R explicitly. Instead we just count the related splits since all splits in the consensus trees and R are already in G.

$$dist(RL(G), R) = \sum_{j \in K} |Spl(\widehat{T}_j) \setminus Spl(RL(G))| = \sum_{i=1}^{m} (1 - w_i)$$

$$dist(Loose_c(G), R) = \sum_{j \in K} |Spl(\widehat{T}_j) \setminus Spl(Loose_c(G))| = \sum_{i=1}^{m} (1 - w_i)$$

where $w_i = 1$ precisely if the ith split is the same as a split in the consensus tree.

In the $RL(G)$ case, $w_i = 1$ if and only if $\sum_{j=1}^{k} S_{ij}^{(2)} + \sum_{j=1}^{k} S_{ij}^{(3)} = 2k$, or equivalently.

$$\begin{aligned} 2k \cdot w_i - \sum_{j=1}^{k} S_{ij}^{(2)} - \sum_{j=1}^{k} S_{ij}^{(3)} \leq 0, \\ \sum_{j=1}^{k} S_{ij}^{(2)} + \sum_{j=1}^{k} S_{ij}^{(3)} - w_i \leq 2k - 1. \end{aligned} \tag{8.11}$$

In the $Loose_c(G)$ case, $w_i = 1$ if and only if $\sum_{j=1}^{k} S_{ij}^{(2)} = k$. This is expressed by the following two constraints.

$$\begin{aligned} k \cdot w_i - \sum_{j=1}^{k} S_{ij}^{(2)} \leq 0, \\ \sum_{j=1}^{k} S_{ij}^{(2)} + 1 - k - w_i \leq 0. \end{aligned} \tag{8.12}$$

The ILP model just outlined allows us to find a $G \in \langle P \rangle_r$ corresponding to some optimal candidate supertree S. Since $Str_s(P)$ ($Loose_s(P)$) is the strict consensus of all optimal candidate supertrees, each split in $Str_s(P)$ ($Loose_s(P)$) must also be in S. Thus, once we have S, we simply need to verify which splits in S are in $Str_s(P)$ ($Loose_s(P)$) and which are not. To do this, for each split $A|B$ in S, we put additional constraints on the original ILP requiring that the optimal tree achieve an objective value equal or smaller than that of S and *not* display split $A|B$. The resulting ILP has only $O(mn)$ more variables and constraints than the original one. If the new ILP is feasible, then $A|B \notin Spl(Str_s(P))(Spl(Loose_s(P)))$; otherwise, $A|B \in Spl(Str_s(P))(Spl(Loose_s(P)))$. We have found that detecting infeasibility is generally much faster than finding an optimal solution.

The following is the detail for additional variables and constraints, as well as the objective function. First we consider rooted trees where there exists a root taxon r such that $M(r, j) = 1$ for every column j. The split to be verified is named as A. It can be viewed as an additional column of M.

Define n binary variable E_{jA}. $E_{jA} = 1$ if and only if an original column j ($j = 1, \ldots, m$) and A are identical. If in row r where $M(r, j)$ is known $M(r, j) \neq A(r)$, $E_{jA} = 0$. Assume that

there are p rows where $M(r,j) =?$ and $A(r) = 1$, and there are $q - p$ rows where $M(r,j) =?$ and $A(r) = 0$. If the p rows range from r_1 to r_p and the $q - p$ rows range from r_{p+1} to r_q, we have $E_{jA} \Leftrightarrow F_{r_1 j} \wedge \cdots \wedge F_{r_p j} \wedge (1 - F_{r_{p+1} j}) \wedge \cdots \wedge (1 - F_{r_q j})$, or

$$\begin{aligned} q \cdot E_{jA} - \sum_{\ell=1}^{p} F_{r_\ell j} + \sum_{\ell=p+1}^{q} F_{r_\ell j} &\leq q - p, \\ E_{jA} - \sum_{\ell=1}^{p} F_{r_\ell j} + \sum_{\ell=p+1}^{q} F_{r_\ell j} &\geq 1 - p. \end{aligned} \tag{8.13}$$

In the new model, if a filled column is identical with column A, then it cannot be part of the optimal candidate tree. That is, if $E_{jA} = 1$, $w_j = 0$. Equivalently we have $w_j + E_{jA} - 1 \leq 0$.

Given the solved objective value $s(G)$, the additional constraint for the objective function is $\sum_{j=1}^{m} 1 - w_j \leq s(G)$.

8.4.2 Profiles with Trivial Trees

A normal profile is made of nontrivial trees, which are biologically meaningful. The ILP formulation so far does not consider special trees as input. However, as we will describe later, when substructure analysis is used, the original profile is divided into a satellite profile and a planet profile where each input tree in these sub-profiles is a subtree of the original input tree. These new input trees can have special trees as follows. For rooted trees, they are empty trees, singletons, cherries, and fans of at least 3 taxa excluding the root.

These trivial trees are compatible with any supertree. Thus for loose OCT we create a reduced profile that excludes these trivial trees. However, for the strict OCT, we need to satisfy the reducibility requirements.

In the rooted trees case, for empty trees, singletons, and cherries, any supertree split reduces trivially to them. A *3-fan* is a fan with at least three leaves excluding the root. If any supertree split can reduce to a smaller fan inside it, and thus violating the reducibility requirement. Additional constraints are need to prevent them.

For instance, the induced subtree for a 3-fan could contain extra clusters without additional qualification constraints. Suppose a 3-fan contains four taxa a, b, c, d, and the induced subtree

contains a cherry ab. Even though compatible with the 3-fan, ab is not part of it. The supertree does not qualify a strict candidate tree.

Let the original profile be P and the reduced profile with normal trees be P'. Suppose there are u 3-fans in P. A special profile P'' is created for these 3-fans.

Introduce m by u binary variables $S_{ip}^{(4)}$ where $i = 1, \ldots, m$ for m columns in M and $p = 1, \ldots, u$ for the u 3-fans in P''. $S_{ip}^{(4)} = 1$ if and only if the i^{th} split is trivially reducible to the 3-fan $t_p \in P''$.

Similar to $Trivial^{(1)}$ and $Trivial^{(2)}$, $Tri^{(1)}$ and $Tri^{(2)}$ are defined to handle the two types 3-fan trivial reduction conditions. Note that the p in $Trivial_{ip}^{(1)}$ denotes the p^{th} tree in P while the p in $Tri_{ip}^{(1)}$ denotes the p^{th} 3-fan in P''. $S_{ip}^{(4)} \Leftrightarrow Tri_{ip}^{(1)} \oplus Tri_{ip}^{(2)}$, or

$$S_{ip}^{(4)} = Tri_{ip}^{(1)} + Tri_{ip}^{(2)} \tag{8.14}$$

$Tri(1)_{ip} = 1$ if and only if split i reduces trivially into tree $t_p \in P''$ in the form $\emptyset | r(L_p)$ or $x | r(L_p \backslash x)$ where L_p is the leaf set (excluding the root) of t_p and x is a single taxon. Let $q = |L_p|$. $Tri_{ip}^{1} = 1$ precisely if at least $q - 1$ out of q $F_{r_\ell i}$ are 1, where $\ell = 1, \ldots, q$. In other words, we have $Tri_{ip}^{(1)} \Leftrightarrow \sum_{\ell=1}^{q} F_{r_\ell i} \geq q - 1$, or

$$\begin{aligned} (q-1) \cdot Tri_{ip}^{(1)} - \sum_{\ell=1}^{q} F_{r_\ell i} \leq 0 \\ 2 \cdot Tri_{ip}^{(1)} - \sum_{\ell=1}^{q} F_{r_\ell i} \geq 2 - q \end{aligned} \tag{8.15}$$

Similarly, $Tri_{ip}^{(2)} = 1$ if and only if split i reduces trivially into tree p in the form of $L_p | r$, where r is the special taxon for root of t_p. Let $q = |L_p|$, we have $Tri_{ip}^{(2)} \Leftrightarrow \neg F_{r_1 i} \wedge \cdots \wedge \neg F_{r_q i}$. In other words, we have $Tri_{ip}^{(2)} \Leftrightarrow \sum_{\ell=1}^{q} F_{r_\ell i} = 0$, or

$$\begin{aligned} q \cdot Tri_{ip}^{(2)} + \sum_{\ell=1}^{q} F_{r_\ell i} \leq q \\ Tri_{ip}^{(2)} + \sum_{\ell=1}^{q} F_{r_\ell i} \geq 1 \end{aligned} \tag{8.16}$$

Define m binary variables w' such that w'_i corresponds to column i of the M matrix. $w'_i = 1$ precisely if the i^{th} column of M is compatible with every tree in G, the restricted span, and is reducible to every tree in P, that is, reducible every tree in to P' and trivially reducible to

every 3-fan in P''. $w_i = 1$ precisely if the i^{th} column of M is compatible with every tree in G, and is reducible to every tree in P', the reduced profile.

To accommodate the 3-fans in P'', for the i^{th} column in M, we have $w'_i \Leftrightarrow w_i \wedge S^{(4)}_{i1} \wedge \cdots \wedge S^{(4)}_{iu}$, or

$$\begin{aligned} (u+1) \cdot w'_i - w_i - \sum_{\ell=1}^{u} S^{(4)}_{i\ell} &\leq 0 \\ w'_i - w_i - \sum_{\ell=1}^{u} S^{(4)}_{i\ell} &\geq -u \end{aligned} \tag{8.17}$$

The objective function becomes

$$\text{minimize} \qquad \sum_{i=1}^{m} 1 - w'_i \tag{8.18}$$

8.4.3 Profile with Splits to Avoid

To seek optimal solutions where certain splits are not part of the solution, constraints similar to that of verification purposes can be added to the ILP, except that we do not impose any constraints on the objective values.

8.4.4 Size of Practically Solvable Models

We have some empirical results to gauge the practically solvable ILP model. With the primates (large set) and seabirds experience, it seems when the product of the number of taxa and the number of splits is not larger than 1500, the (+) model can be solved pretty quickly by CPLEX in several seconds to several minutes. Beyond that, it could be longer than desirable.

Another possibility is to study the number of unknowns in the M matrix. It might be able to serve as a size limit as well.

CHAPTER 9. ILP FORMULATION FOR GOOD SPLITS, The (-) WAY

The (-) route ILP is based on the asymmetric difference definitions. If the whole OCT is treated as unknown, our experience shows the size of problems it can handle are usually smaller than that of the basic (+) ILP, due to the fact that it has to be assumed a binary tree and numerous constraints are required to ensure the filling represents a tree. However, the (-) ILP is perfect to provide just a few splits for large problems so that we can divide and conquer the problem with the reducible and pull-out sets. If only one split is sought, constraints are much fewer.

9.1 Binary Fill-in Vector

Let $P = (t_1, \ldots, t_k)$ be a profile where $|\mathcal{L}(P)| = n$. We define a n binary vector M representing a nontrivial split of the OCT. M is a single unfilled column, except that the row for the root is filled with 1 in the rooted tree case. The ILP associates a *fill-in variable* F_x with each M_x. F_x will be assigned a value of 0 or 1, representing an assignment of taxon x to one of the two sides of a split.

As M only contains nontrivial split, it must contain at least two 0's and two 1's. For rooted trees we have

$$\begin{aligned} \sum_{r=1}^{n-1} F_r &\geq 1, \\ \sum_{r=1}^{n-1} F_r &\leq n-3. \end{aligned} \tag{9.1}$$

n includes the root and $F_n \equiv 1$.

9.2 Representations of Input and Induced Subtrees

Each input tree t_i is represented by an n by $|Spl(i)|$ binary matrix Mt_i. $Mt_i(p,q) = 0$ if the p^{th} taxon is in the non-root block of the q^{th} split. $Mt_i(p,q) = 1$ if it is in the root block. $Mt_i(p,q) =?$ if the p^{th} taxon is not in t_i.

We need k induced subtrees represented by IM_i. IM_i is essentially M. However, only those rows where $Mt_i \neq ?$ are considered in IM_i.

9.3 Column Identity

To compute the distance between an input tree and its induced subtree, we need to establish column identity between Mt_i and IM_i with binary variables EE_p^i where $p = 1, \ldots, |Spl(t_i)|$. $EE_p^i = 1$ precisely if a known column p in Mt_i is equivalent with IM_i. We assume that t_i has distinct columns.

For rooted trees, there are u rows where $Mt_i(r,p) = 1$ and $IM_i(r)$ is unknown, and $|\mathcal{L}_i|-1-u$ rows where $Mt_i(r,p) = 0$ and $IM_i(r)$ is unknown. $\mathcal{L}_i$ includes the root r. Note that the last rows of Mt_i and IM_i are filled with 1 to represent the roots. Name the u rows from r_1 to r_u and the $|\mathcal{L}_i|-1-u$ rows from $r_{(u+1)}$ to $r_{(|\mathcal{L}_i|-1)}$. We have $EE_p^i \Leftrightarrow F_{r_1} \wedge \cdots \wedge F_{r_u} \wedge (1 - F_{r_{(u+1)}}) \wedge \cdots \wedge (1 - F_{r_{(|\mathcal{L}_i|-1)}})$, or

$$\begin{aligned} (|\mathcal{L}_i| - 1) \cdot EE_p^i - \sum_{\ell=1}^{u} F_{r_\ell} + \sum_{\ell=u+1}^{|\mathcal{L}_i|-1} F_{r_\ell} \leq |\mathcal{L}_i| - 1 - u, \\ EE_p^i - \sum_{\ell=1}^{u} F_{r_\ell} + \sum_{\ell=u+1}^{|\mathcal{L}_i|-1} F_{r_\ell} \geq 1 - u. \end{aligned} \tag{9.2}$$

Note that $p = 1, \ldots, |Spl(t_i)|$.

9.4 Constraints for a Strict Good Split

9.4.1 Normal Profile

For a good split T in a strict candidate tree, its k induced subtrees IM_i, $i = 1, \ldots, k$, satisfy the following condition. For rooted trees, $Spl(T|\mathcal{L}_i) \subseteq Spl(t_i)$. Hence, IM_i must either have an equivalent column in Mt_i or reduces to a trivial split in t_i.

1. Reduces to an equivalent column p in Mt_i, represented by $EE_p^i = 1$.

2. Reduces to a trivial split in t_i, in two forms discussed below and represented by $Trivial_i^{(1)}$ and $Trivial_i^{(2)}$, respectively.

We have $\bigoplus_{p=1}^{|Spl(t_i)|} EE_p^i \oplus Trivial_i^{(1)} \oplus Trivial_i^{(2)}$, or

$$\sum_{p=1}^{|Spl(t_i)|} EE_p^i + Trivial_i^{(1)} + Trivial_i^{(2)} = 1 \tag{9.3}$$

For rooted trees, define binary variables $Trivial^{(1)}$ such that $Trivial_i^{(1)} = 1$ precisely if it reduces trivially into input tree t_i in the forms of $\emptyset|\mathcal{L}_i$ (all 1's in IM_i) or $x|(\mathcal{L}_i \setminus x)$ (one 0 and others 1's in IM_i). Here $\mathcal{L}_i$ is the leaf set (including root) of tree t_i and x is a single taxon other than the root.

Note that we have $Trivial_i^{(1)} \Leftrightarrow \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \geq |\mathcal{L}_i| - 2$, or

$$\begin{aligned} (|\mathcal{L}_i| - 2)Trivial_i^{(1)} - \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \leq 0, \\ 2 \cdot Trivail_i^{(1)} - \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \geq 3 - |\mathcal{L}_i|. \end{aligned} \tag{9.4}$$

Define binary variables $Trivial^{(2)}$ such that $Trivial_i^{(2)} = 1$ if and only if the split reduces trivially into t_i in the forms of $(\mathcal{L}_j \setminus r)|r$, where r is the root (all 0's in IM_i except the root row).

Note that $Trivial_i^{(2)} \Leftrightarrow \bigwedge_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} = 0$. Then $Trivial_i^{(2)} \Leftrightarrow \sum_{p=1}^{|\mathcal{L}_i|-1} F(r_p) = 0$, or

$$\begin{aligned} (|\mathcal{L}_i| - 1) \cdot Trivial_i^{(2)} + \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \leq |\mathcal{L}_i| - 1, \\ Trivial_i^{(2)} + \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \geq 1. \end{aligned} \tag{9.5}$$

9.4.2 Profiles with Trivial Trees

Trivial trees have only trivial clusters that are not included in Mt. Indeed the Mt for a trivial tree is an empty matrix. Special handling is arranged.

First, a reduced profile P' is created to include only nontrivial trees in the original profile P. The normal ILP applies to P'. Second, a special profile P'' is created to hold the u 3-fans in P. The following additional variables and constraints handle the special situation.

Let $\mathcal{L}_i$ $(i = 1, \ldots, u)$ be the leaf set of the 3-fan $t_i \in P''$. Note that the induced subtree IM_i automatically satisfies the triviality condition for empty tree, singleton, and cherry for the rooted trees. For 3-fans, IM_i should to constrained to reduce into a trivial split in $t_i \in P''$.

$Tri^{(1)}$ and $Tri^{(2)}$ are defined to handle the two types 3-fan trivial reduction conditions. Note that the i in $Trivial_i^{(1)}$ denotes the i^{th} tree in P while the i in $Tri_i^{(1)}$ denotes the i^{th} 3-fan in P''. To meet the condition, we have $Tri_i^{(1)} \oplus Tri_i^{(2)}$. Equivalently, we have

$$Tri_i^{(1)} + Tri_i^{(2)} = 1 \tag{9.6}$$

For rooted trees, define u binary variables $Tri^{(1)}$ such that $Tri_i^{(1)} = 1$ if and only if the split reduces trivially into t_i in the forms of $\emptyset|L_i$ (all 1's in IM_i) or $x|(\mathcal{L}_i \setminus x)$ (one 0 and others 1's in IM_i). Here $\mathcal{L}_i$ is the leaf set (including root) of t_i and x is a single taxon other than the root. Note that we have $Tri_i^{(1)} \Leftrightarrow \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \geq |\mathcal{L}_i| - 2$, or

$$\begin{aligned} (|\mathcal{L}_i| - 2)Tri_i^{(1)} - \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \leq 0, \\ 2 \cdot Tri_i^{(1)} - \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \geq 3 - |\mathcal{L}_i|. \end{aligned} \tag{9.7}$$

Define u binary variables $Tri^{(2)}$ such that $Tri_i^{(2)} = 1$ if and only if the split reduces trivially into t_i in the forms of $(\mathcal{L}_i \setminus r)|r$, where r is the root (all 0's in IM_i except the root row). Note that $Tri_i^{(2)} \Leftrightarrow \bigwedge_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} = 0$. In other words, we have $Tri_i^{(2)} \Leftrightarrow \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} = 0$, or

$$\begin{aligned} (|\mathcal{L}_i| - 1) \cdot Tri_i^{(2)} + \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \leq |\mathcal{L}_i| - 1, \\ Tri_i^{(2)} + \sum_{p=1}^{|\mathcal{L}_i|-1} F_{r_p} \geq 1. \end{aligned} \tag{9.8}$$

9.5 Constraints for a Loose Good Split

A loose candidate supertree T must be compatible with t_i. The following theorem transforms compatibility of T and t_i to compatibility of $T|\mathcal{L}_i$ and t_i.

Theorem 13. *Let t be an input tree and T be a supertree where $\mathcal{L}_t \subseteq \mathcal{L}_T$. t and T are compatible if and only if t and $T|\mathcal{L}_t$ are compatible.*

Proof. If t and T are compatible, there is an S that displays t and T. S also displays any induced subtree of T. Hence S displays $T|\mathcal{L}_t$. Therefore, t and $T|\mathcal{L}_t$ are compatible.

Conversely, if t and $T|\mathcal{L}_t$ are compatible, define T' such that $Spl(T') = Spl(T|\mathcal{L}_t) \cup Spl(t)$. Every edge in $T|\mathcal{L}_t$ can be mapped to a different edge in T' so that their corresponding splits are the same. Restore every edge in $T|\mathcal{L}_t$ to its original path with degree-2 internal nodes in T and restore the pruned taxa and their related edges. Let the resulting tree be S. Obviously S displays T and t. Hence T and t are compatible. □

Define $BB_p^{(ab)i}$, $a, b \in \{0, 1\}$, where $BB_p^{(ab)i} = 1$ precisely if there is a taxon r such that $Mt_{rp}^i = a$ and $F_r = b$. The $BB_p^{(ab)i}$s are determined from the settings of variables $\Gamma\Gamma_{rp}^{(ab)i}$, where r ranges over the taxa in Mt_i. The $\Gamma\Gamma$ variables satisfy $\Gamma\Gamma_{rp}^{(ab)i} \Leftrightarrow \big((Mt_{rp}^i = a) \wedge (F_r = b)\big)$, or

$$\begin{aligned} (-1)^b F_r + \Gamma\Gamma_{rp}^{(ab)i} &\geq 1 - a - b - (-1)^a Mt_{rp}^i, \\ (-1)^b F_r + 2\Gamma\Gamma_{rp}^{(ab)i} &\leq 2 - a - b - (-1)^a Mt_{rp}^i. \end{aligned} \tag{9.9}$$

We have that $BB_p^{(ab)i} \Leftrightarrow \bigvee_r \Gamma\Gamma_{rp}^{(ab)i}$, or

$$\begin{aligned} -\sum_r \Gamma\Gamma_{rp}^{(ab)i} + BB_p^{(ab)i} &\leq 0, \\ \sum_r \Gamma\Gamma_{rp}^{(ab)i} - n \cdot BB_p^{(ab)i} &\leq 0 \end{aligned} \tag{9.10}$$

To ensure compatibility, observe that $\neg(BB_p^{(00)i} \wedge BB_p^{(01)i} \wedge BB_p^{(10)i} \wedge BB_p^{(11)i})$, or

$$BB_p^{(00)i} + BB_p^{(01)i} + BB_p^{(10)i} + BB_p^{(11)i} \leq 3. \tag{9.11}$$

where p is the p^{th} column in Mt_i.

In the loose good split case, no constraints are needed for trivial trees.

9.6 Distance Computation

For each Mt_i, define $|Spl(t_i)|$ binary variables δ_p^i where $p = 1, \ldots, |Spl(t_i)|$. δ_p^i records a p^{th} column in Mt_i that is not part of the IM_i subtree. The constraints are $\neg\delta_p^i \Leftrightarrow EE_p^i$, or

$$\delta_p^i + EE_p^i = 1. \tag{9.12}$$

9.7 Objective Function

The objective function is

$$\text{minimize} \quad \sum_{i=1}^{k} \sum_{p=1}^{|Spl(t_i)|} \delta_p^i. \tag{9.13}$$

9.8 Artificial Clusters and Constraints

The greedy iterative ILP can generate clusters nonexistent for the basic (+) ILP. One type of the nonexistent clusters we observed is the merger of non-overlapping clusters from different input trees. An *artificial* cluster is a good cluster whose reduced clusters in input trees form at least two non-intersecting components when any reduced clusters sharing at least one taxon are joined together.

Mathematically sound, the combination is meaningless biologically. Moreover, the following theorems guarantee that when artificial clusters are avoided, it is possible to find a strict/loose candidate tree with equal or less total distance.

Theorem 14. *Let $P = (t_1, \ldots, t_k)$ be a profile and T be a strict candidate tree of P. If an artificial cluster $A \in T$, there is another strict candidate T' such that $A \notin T'$ where $AD^-(P, T') = AD^-(P, T)$.*

Proof. Assume that $A|\mathcal{L}_i = B_i$ where B_i is a cluster in t_i $(i = 1, \ldots, k)$. $B_1, \ldots, B_k$ join into p non-overlapping sets $C_1, \ldots, C_p$ where $p \geq 2$. Replace the subtree $T|A$ with new subtrees $T|C_1$, $T|C_2$ to $T|C_p$. Shrink the edge between $T|A$ and its parent u and connect the roots of $T|C_1$,$T|C_2$ to $T|C_p$ to u. The new tree is T'.

Note that every B_j $(j = 1, \ldots, k)$ can now be obtained by reducing one of C_i's $(i = 1, \ldots, p)$ to B_j. Other C_ℓ where $\ell \neq i$ has no taxa common to L_j and vanishes during reduction. Thus $AD^-(P, T') = AD^-(P, T)$. □

Theorem 14 shows that we can find strict OCT without artificial clusters.

Theorem 15. *Let $P = (t_1, \ldots, t_k)$ be a profile and T be a loose candidate tree of P. If an artificial cluster $A \in T$, there is another loose candidate T' such that $A \notin T'$ where $AD^-(P, T') \leq AD^-(P, T)$.*

Proof. If A is a reducible cluster (i.e., $A|\mathcal{L}_i \in t_i$ where $i = 1, \ldots, k$), the proof is the same as that of the strict supertree case. Let A be a pull-out cluster but not a reducible cluster. That is, there is an i such that $A|\mathcal{L}_i = D_1 \cup \cdots \cup D_q$ where $D_j \in t_i$ for $j = 1, \ldots, q$ but their union $D_1 \cup \cdots \cup D_q \notin t_i$. $D_1, \ldots, D_q$ and the projected clusters in other input trees form p non-intersecting sets $C_1, \ldots, C_p$ where $p \geq 2$ and $A = C_1 \cup \cdots \cup C_p$. Replace the subtree $T|A$ with new subtrees $T|C_1, T|C_2, \ldots, T|C_p$. Shrink the edge between $T|A$ and its parent u and connect the roots of $T|C_1, T|C_2, \ldots, T|C_p$ to u. The new tree is T'.

Note that $C_\ell|\mathcal{L}_i$ $(\ell = 1, \ldots, p)$ is compatible with $D_j \in t_i$ for $j = 1, \ldots, q$, because it is the union of some of D_j. Since none of D_j $(j = 1, \ldots, q)$ is canceled by $A|\mathcal{L}_i$, while it is possible that $D_j = C_\ell|\mathcal{L}_i$ $(\ell = 1, \ldots, p)$, $AD^-(P, T') \leq AD^-(P, T)$. □

Theorem 15 shows that if T is a loose OCT, then T' is also a loose OCT. Hence we can always find a loose OCT without artificial clusters.

Avoiding artificial clusters gives us better OCT, as artificial clusters create faulty combinations that prevent the formation of correct clusters down the road.

We have developed two methods to handle artificial clusters. One is by adding additional constraints when a solved cluster is artificial. The other is to provide a set of constraints to prevent artificial clusters from happening. The first method is suitable when there are no or very few artificial constraints. Usually it happens when the number of trees are way more than the number of taxa involved. It generates smaller size ILP models. The second method is suitable when there are a lot of artificial constraints. Usually it happens when the number of taxa is way more than the number of trees. It generates larger ILP models.

9.8.1 Constraints after Finding Artificial Clusters

For rooted trees, let $A|B$ be an artificial split that is solved. $A|B$ is n by 1. M has $M(n) = 1$ to represent the root and has $n - 1$ F variables. Suppose corresponding to these variables, u rows of $A|B$ have values 1 and $n - 1 - u$ rows of $A|B$ have values are 0. The constraint can be

written as $\neg(F_{r_1} \wedge \cdots \wedge F_{r_u} \wedge (1 - F_{r_{(u+1)}}) \wedge \cdots \wedge (1 - F_{r_{(n-1)}}))$, or

$$\sum_{\ell=1}^{u} F_{r_\ell} - \sum_{\ell=u+1}^{n-1} F_{r_\ell} \leq u - 1. \tag{9.14}$$

9.8.2 Constraints to Suppress Artificial Clusters

The following constraints apply to rooted trees only. First, we define t binary variables dd for all distinct clusters, $dd_1, \ldots, dd_t$. $dd_i = 1$ precisely if dd_i's corresponding cluster is a subset of the good cluster. In other words, all of its F's are 0. $dd_p \Leftrightarrow \neg F_{i_1} \wedge \neg F_{i_2} \wedge \cdots \wedge \neg F_{i_q}$, where $i_1 \ldots i_q$ are its q taxa.

$$\begin{aligned} q \cdot dd_p + \sum_{j=1}^{q} F_{i_j} &\leq q \\ dd_p + \sum_{j=1}^{q} F_{i_j} &\geq 1 \end{aligned} \tag{9.15}$$

Second, we define $n-1$ binary variables $LTLI_1 \ldots LTLI_{n-1}$. $LTLI$ stands for *Linked-To-a-Larger-Index*. $LTLI_i = 1$ if and only if

1. a cluster p, $dd_p = 1$, has a taxon whose index is i, and another taxon whose index is larger than i; or

2. two overlapped clusters p and q, $dd_p = dd_q = 1$, i corresponds to the largest taxon index in p, and q has a taxon whose index is larger than i.

In order to describe the second case, we introduce a set of binary variables bb_{pq} such that $bb_{pq} = 1$ precisely if clusters p and q are overlapped, and $dd_p = dd_q = 1$. $bb_{pq} \Leftrightarrow dd_p \wedge dd_q$, or

$$\begin{aligned} 2 \cdot bb_{pq} - dd_p - dd_q &\leq 0 \\ dd_p + dd_q - bb_{pq} &\leq 1 \end{aligned} \tag{9.16}$$

Now we establish the constraints for $LTLI_i$ (i=1,...,n-1) with dd and bb. $\bigcup_{p'} dd_{p'} \cup \bigcup_{pq} bb_{pq} \Leftrightarrow LTLI_i$, or

$$\begin{aligned} LTLI_i - \sum_{j=1}^{t} dd_{p_j} - \sum_{uv} bb_{uv} &\leq 0 \\ \sum_{j=1}^{t} dd_{p_j} + \sum_{uv} bb_{uv} - G \cdot LTLI_i &\leq 0 \end{aligned} \tag{9.17}$$

where G is the number of items of $\sum_{j=1}^{t} dd_{p_j} + \sum_{uv} bb_{uv}$. Note that the largest taxon in the leaf set always has $LTLI_{n-1} = 0$.

Third, the projected clusters forming a connected component is equivalent to the sum of $LTLI_i$ being exactly one less than the number of taxa in the good cluster. In other words, every taxon of the cutter cluster is linked to a larger index taxon except for the largest one. If there are two non-overlapping components, there is another taxon j with $LTLI_j = 0$ and it cannot link to the taxa in the other component. Equivalently we have the following constraints.

$$\sum_{i=1}^{n-1} F_i + \sum_{i=1}^{n-1} LTLI_i = n - 2 \tag{9.18}$$

Since when $F_i = 1$, $LTLI_i = 0$; and for all $F_i = 0$,only the largest i can have $LTLI_i = 1$, other $LTIL_j = 0$ for any $j \neq i$.

To reduce the number of constraints, singletons can be removed without affecting the value of $LTLI$, because they do not meet the first case requirement. For the second case, i is the index for the single taxon in cluster p. Due to overlapping, cluster q must also have this taxon. it can be simplified as to the first case of $dd_q = 1$. Hence removal of cluster p, the singleton, does not matter.

In implementation, we first generate unique clusters without singletons. After defining dd variables and the $dd - F$ constraints, we loop through i to define $LTLI$ variables as well as involved bb variables, $dd - bb$ constraints, and $LTLI - dd - bb$ constraints.

9.9 Number of Variables and Constraints

We discuss the variables and constraints for both strict and loose OCT first. There are $O(n)$ for F variables, $O(1)$ for F constraints, $O(m)$ for EE, δ variables and $EE - F$, $\delta - EE$ constraints.

For strict good split pairwise distance computation. we have takes $O(k)$ for $Trivial^{(1)}$, $Trivial^{(2)}$, $Tri^{(1)}$, $Tri^{(2)}$ variables, and $EE - Trivial^{(1)} - Trivial^{(2)}$, $Trivial^{(1)} - F$, $Trivial^{(2)} - F$, $Tri^{(1)} - Tri^{(2)}$, $Tri^{(1)} - F$, $Tri^{(2)} - F$ constraints.

For loose good split compatibility between $T|\mathcal{L}_i$ and t_i, we have $O(m)$ for BB variables, and BB, BB-$\Gamma\Gamma$ constraints, $O(mn)$ for $\Gamma\Gamma$ and $\Gamma\Gamma - F$ constraints.

It takes $O(1)$ to add anti-artificial cluster F related constraints using the first method. For the second method, we have $O(m)$ dd variables and $dd - F$ constraints, $O(m^2)$ bb variables and $bb - dd$ constraints, $O(n)$ $LTLI$ variables and $LTLI - bb - dd$ constraints, and $O(1)$ $F - LTLI$ constraints.

Therefore, using the first method to apply anti-artificial cluster constraints, $O(m)$ is the bound for the variables and constraints for the strict good split. $O(mn)$ is the bound for variables and constraints for the loose good split.

Using the second method to apply anti-artificial cluster constraints, $O(m^2)$ is the bound for both strict and loose good split. Here we assume $m > n$.

CHAPTER 10. ALGORITHMS

10.1 Divide and Conquer

From previous chapters we know that reducible and pull-out sets are properties of the conservative supertrees. The iterative (-) ILP provides good splits to divide the original profile into planet and satellite profiles. The basic (+) ILP provides exact solutions for small models.

DivideAndConquer is a program that employs the recursive routine *myDivideAndConquer* to obtains the objective value and the strict/loose optimal candidate tree. OBJ and OCT stand for the original objective value and the optimal candidate tree found by *DivideAndConquer*. $Profile$, $Profile_{sat}$ and $Profile_{pla}$ are the original, satellite and planet profiles. OBJ_{sat} and OBJ_{pla} are satellite and planet OBJ's, while OCT_{sat} and OCT_{pla} are satellite and planet OCT's.

Algorithm 1 [OBJ, OCT] = DivideAndConquer

1: Input data and generate $Profile$.
2: $Avoid = \emptyset$.
3: $[OBJ, OCT] = myDivideAndConquer(Profile, Avoid)$.

10.2 Verification

$VerifyHelper$ is a program that calls the recursive $Verify$ routine. It stops when two consecutive runs return identical supertrees. We could make it more strict by requiring more runs of identically returned supertrees. A *category* number is assigned to each edge: 0 for an edge in OCT but not in the true supertree; 1 for an external edge (trivial cluster); 2 for other edges. $LastST$ is the supertree found by $Verify$ in the last round. ST is the supertree found by $Verify$ in this round. $VerifyHelper$ works as follows. $LastST$ is initially set to OCT. All

Algorithm 2 [OBJ, OCT] = myDivideAndConquer(Profile, Avoid)

if Solvable with the basic (+) ILP **then**
  Solve it and return OBJ and OCT.
else
  Solve for a good split A with the iterative (-) ILP.
  if Infeasible, i.e., A cannot be found **then**
    OBJ = the sum of all nontrivial splits in $Profile$.
    OCT = a fan for the profile leaf set.
  else
    Create the satellite and planet profiles with A.
    $[OBJ_{sat}, OCT_{sat}] = myDivideAndConquer(Profile_{sat}, Avoid_{sat})$.
    $[OBJ_{pla}, OCT_{pla}] = myDivideAndConquer(Profile_{pla}, Avoid_{pla})$.
    $OBJ = OBJ_{sat} + OBJ_{pla}$.
    Combine OCT_{sat} and OCT_{pla} to create OCT.
return OBJ and OCT

the edges are set to category 2 (unclear state) except for external edges whose category number are 1. ST is the supertree returned by $Verify$ whose input are $Profile$, OBJ, OCT, and the category numbers. With ST, the new category numbers, some determined in this round, are also returned. If $LastST$ and ST are different, this verification round did find some new edges that belong to OCT but not the true supertree. Since the returned supertree might still have some unfound edges in OCT but not in the true supertree, another round of verification is done after $LastST$ is updated with ST. If $LastST$ and ST are identical, this round finds no more edges in OCT but not in the true supertree. The program terminates.

Algorithm 3 ST = VerifyHelper(Profile,OBJ,OCT)

Set category value for each edge/cluster: trivial clusters 1, others 2
$LastST = OCT$
while 1 **do**
  $[ST, category] = Verify(Profile, OBJ, OCT, category)$.
  if $ST = LastST$ **then**
    return ST
  else
    $LastST = ST$.;

The recursive algorithm $Verify$ checks which edge (cluster) in OCT is in conservative supertree. It works as follows. In a while loop, if the tree consists of only trivial splits or splits that in an OCT but not in the true supertree, then the trivial splits are returned (a fan

is returned). This is a base case in some final stages of substructure analysis. If the OCT has edges (clusters) that are undetermined (category numbers = 2), pick the first one (A) for verification by putting it into *Avoid* to recompute the scores. There are three outcomes.

1. If it is the same as the original OBJ, A is likely in OCT but not in the true supertree. Therefore, we assign 0 to its category number. Also we use the newly solved OCT^1 to intersect the old OCT so that more edges in OCT but not in the true supertree can be determined. Then we move to the next edge in the while loop.

2. If it is lower than the original OBJ, then the OCT is not optimum in the first place. The program stops and the OCT needs to be resolved, probably with a random restart process that randomly cut the OCT and restart the *DivideAndConquer*.

3. If it is larger than the original OBJ, or it is infeasible (A is in *Avoid* and no other nontrivial cutters can be found), then it is possible that A is in the true supertree, if all the previous cuts to compute this OBJ^1 are in the true supertree. But it is also possible that some of the previous cuts to compute this OBJ^1 are in the true supertree while the others are in OCT but not in the true supertree. When some of the edges that are in OCT but not in the true supertree must be kept and another A must be avoided, it is possible that the OBJ^1 is higher than OBJ. So we cannot guarantee that A is in the true supertree. Hence, we still keep A's category number as 2. Nevertheless, we use A as a cutter to divide the $Profile$, *Avoid*, OBJ and OCT to recursively apply $Verify$ to subproblems. Once solved, it is assembled and returned. Note that in the returned values, many of the category numbers are still 2. But those edges whose category numbers are 0 are cleared.

Inside $Verify$, *Recompute* calculates the new objective value and the new OCT with *Avoid*. It works as follows. If *Avoid* is empty for the profile involved, then the original OBJ and OCT for the profile are returned. If the verification problem is solvable by the basic (+) ILP, we invoke basic ILP with $Profile$ and *Avoid* to solve it and return the new OBJ^1 and OCT^1. If the problem is too big, then there are two cases to consider.

Algorithm 4 [ST,OCT,category] = Verify(Profile,OBJ,OCT,category)

1: **while** 1 **do**
2: **if** No edges are of category 2 **then**
3: ST = all clusters in OCT whose category values are 1.
4: **return**
5: Pick the first cluster A in OCT whose category value is 2.
6: Put A into *Avoid*
7: $[OBJ^1, OCT^1] = Recompute(Profile, Avoid, OBJ, OCT, category)$
8: **if** $OBJ^1 >= 0$, $OBJ >= 0$ and $OBJ^1 == OBJ$ **then**
9: $category(A) = 0$, since $A \in OCT \setminus ST$
10: Compare OCT^1 and OCT to determine more category values for other edges
11: **else if** $OBJ^1 >= 0$ and $OBJ^1 < OBJ$ **then**
12: OBJ and OCT are not optimum, stop
13: **else if** $OBJ >= 0$ and $OBJ^1 > OBJ$, or infeasible ($OBJ^1 = -1$) **then**
14: Use A to generate $Profile_{pla}$, $Avoid_{pla}$, OBJ_{pla}, OCT_{pla}, and $category_{pla}$.
15: $[ST_{pla}, OCT_{pla}, category_{pla}] = Verify(Profile_{pla}, OBJ_{pla}, OCT_{pla}, category_{pla})$.
16: Use A to generate $Profile_{sat}$,$Avoid_{sat}$, OBJ_{sat}, OCT_{sat},$category_{sat}$.
17: $[ST_{sat}, OCT_{sat}, category_{sat}] = Verify(Profile_{sat}, OBJ_{sat}, OCT_{sat}, category_{sat})$.
18: Assemble ST, OCT, and $category$ from substructure analysis.
19: **return**

1. If there is only one nontrivial split left, it is in *Avoid* and unsolvable by the basic (+) ILP, we call *myDivideAndConquer* to solve it. It either finds another way to divide and solve it, or it shows that there are no other nontrivial splits. The result is returned as the new OBJ^1 and OCT^1.

2. If there are more than one nontrivial split left, we randomly pick a category 2 cluster B that is not in *Avoid* to use it as the cutter to divide the $Profile$, *Avoid*, OBJ and OCT, and recursively call *Recompute* to calculate new OBJ^1 and OCT^1 for the sub-profiles. If any one is infeasible, *Recompute* passes the infeasibility information to its caller. Otherwise, it combines the results from two sub-profiles and return the new OBJ^1 and OCT^1.

$VerifyHelper$ is fast since in *Recompute*, the initial OCT edges are used as cuts to divide the profiles recursively. Previously these cuts were found by running *myDivideAndConquer* repeatedly. Now most of the computation is the basic (+) ILP. Occasionally a call to *DivideAndConquer* is used when the problem is too big and cannot be cut further.

Algorithm 5 [OBJ1,OCT1] = Recompute(Profile,Avoid,OBJ,OCT,category)

1: **if** $Avoid = \emptyset$ **then**
2: $OBJ^1 = OBJ$, $OCT^1 = OCT$
3: **return**
4: **if** Solvable by the basic (+) ILP **then**
5: Invoke basic (+) ILP with $Profile$ and $Avoid$.
6: **else**
7: **if** Only one nontrivial split left, in $Avoid$, and unsolvable by the basic (+) ILP **then**
8: $[OBJ^1, OCT^1] = myDivideConquer(Profile, Avoid)$.
9: **return**
10: Randomly pick a category 2 cluster B that is not in $Avoid$.
11: Use B to generate $Profile_{pla}$,$Avoid_{pla}$, OBJ_{pla}, OCT_{pla}, $category_{pla}$.
12: $[OBJ^1_{pla}, OCT^1_{pla}] = Recompute(Profile_{pla}, Avoid_{pla}, OBJ_{pla}, OCT_{pla}, category_{pla})$
13: **if** $OBJ^1_{pla} = -1$, i.e., infeasible **then**
14: $OBJ^1 = -1$, $OCT^1 = \emptyset$
15: **return**
16: Use B to generate $Profile_{sat}$, $Avoid_{sat}$, OBJ_{sat}, OCT_{sat} and $category_{sat}$
17: $[OBJ^1_{sat}, OCT^1_{sat}] = Recompute(Profile_{sat}, Avoid_{sat}, Obj_{sat}, OCT_{sat}, category_{sat})$;
18: **if** $OBJ^1_{sat} = -1$, i.e., infeasible **then**
19: $OBJ^1 = -1$; $OCT^1 = \emptyset$
20: **return**
21: Get OBJ^1 and OCT^1 by combining their substructure counterparts

The shortcoming of this verification algorithm is that, given an edge to verify, it is possible that during a series of cuts in *Recompute*, some cutting edges may be in OCT but not in the true supertree. Thus a high recomputed objective value does not ensure the edge to verify is in the supertree. However, if the recomputed objective value is the same, it guarantees the edge to verify is NOT in the true supertree and ST, assuming the OCT is a true optimal candidate tree.

When we use divide and conquer, it is always possible that the cutters computed by *myDivideAndConquer* is OCT but not in the true supertree, even if we do not use *Recompute* but just use *myDivideAndConquer* all the way through. Then using the current algorithm saves enormous time. During experiment we have seen *myDivideAndConquer* found edges in the original OCT repeatedly. So a lot of time is wasted to compute them repeatedly.

To mitigate the impact, we randomize the picking of cutters during *Recompute* and try multiple sequential runs until the supertrees from consecutive runs converge. We can run many times to make sure all supertrees returned in these runs are exactly the same. Ideally in each run, some edges in OCT but not in the true supertree are found and marked. In the next run they cannot serve as cutters. With a higher percentage of edges in ST, it is more likely to identify edges that are in OCT but not in the true supertree since more cutters will be in the supertree.

This verification algorithm is imperfect. The supertree it finds is still possible to contain some edges in the OCT but not in the true supertree. However, in our experiments most of them converge in very few runs.

CHAPTER 11. EXPERIMENTAL RESULTS

We developed a prototype implementation of the techniques described in the previous sections, and used it to conduct a series of computational tests. Our system is based on a collection of MATLAB scripts to automatically generate the appropriate ILPs directly from the input profiles. These scripts invoke CPLEX to solve the ILPs exactly or to apply the divide-and-conquer heuristics.

Our first tests were aimed at finding the limits of the applicability of the exact ILP formulation. As expected, this approach only allowed us to solve rather small problems. The critical parameter is the product of n, the number of taxa, and m, the total number of nontrivial splits in all the input trees. Generally speaking, problems with $nm \geq 2000$ are difficult to solve, while problems with $nm \leq 1000$ are quickly solvable. Thus, we chose $nm = 1000$ as a threshold to use divide and conquer. Note that this is just a rule of thumb, as there are problems with $nm > 1000$ that are solvable, such as the smaller of the primates datasets in Ranwez et al. (2007), which has $n = 33$, $m = 48$, and $nm = 1584$.

We analyzed five large data sets using the divide-and-conquer heuristic: the larger of the two primate data sets in Ranwez et al. (2007), seabirds [Kennedy and Page (2002)], placental mammals [Beck et al. (2006)], legumes [Wojciechowski et al. (2000)] and marsupials [Cardillo et al. (2004)]. The current version of our software assumes a common outgroup among the input trees (i.e., that the trees are rooted). Table 11.1 summarizes the results of our analysis. All experiments were run on an Intel Core 2 64 bit quad-core processor (2.83GHz). Times are given in seconds or hours. Solution and verification times are listed separately. When the solution was a fan, no verification was needed.

The supertrees obtained appear quite reasonable. Interested readers can contact the author for the PDF file that includes these supertrees.

Table 11.1 Summary of Experimental Results

			Strict OCT		Loose OCT	
Data set	n	k	Soln.	Verif.	Soln.	Verif.
Primates	72	24	199 s.	171 s.	232 s.	1568 s.
Seabirds	121	7	251 s.	755 s.	625 s.	817 s.
Placental Mammals	116	726	561 s.	322 s.	28.7 h.	22.2 h.
Legume	571	22	31 s.	fan	28.7 h.	3.15 h.
Marsupial	267	158	0.92 h.	0.96 h.	4.76 h.	8.62 h.

For instance, all 26 of the clusters in the strict marsupial supertree are in the published paper [Cardillo et al. (2004)]. The loose marsupial supertree contains all clusters in the strict supertree along with 38 additional clusters, most of which are in the published supertree. Out of the 64 clusters in this loose supertree, 61 are in the published paper.

We note that many data sets encountered in practice — in particular, the above-mentioned marsupial and placental mammal data sets — include a "scaffold" tree (also called a "backbone" or "seed" tree); i.e., a tree that covers a broad span of taxa, without necessarily being comprehensive. For instance, the marsupial dataset in Cardillo et al. (2004) includes the tree implied by the marsupial classification of Wilson and Reeder (1993) "because it is currently widely accepted as a taxonomic reference for mammals, and because its low resolution means it can easily be overruled by more resolved phylogenies, minimizing its influence on the final supertree." However, a low-resolution scaffold has different effects on strict and loose trees. By the definition of strict supertrees, if a set of taxa appears as a fan in any of the input trees, it must remain as a fan in the strict supertree, no matter what the other trees say. Thus, a low-resolution scaffold can lead to a low-resolution strict supertree. On the other hand, again by definition, scaffolds do not prevent loose supertrees from being well-resolved. Our experimental results — e.g., on the marsupial data set — confirm this observation.

Figure 11.1 shows that the strict and loose supertrees for the primates small data set. This is obtained via the basic ILP.

Figure 11.2 shows that the strict and loose supertrees for the primates small data set. This is obtained via the divide and conquer approach.

Figure 11.3 show that the strict consensus of MRP, majority-rule (+), strict and loose supertrees for the Seabirds data set. The strict and loose supertrees are obtained via the divide and conquer approach.

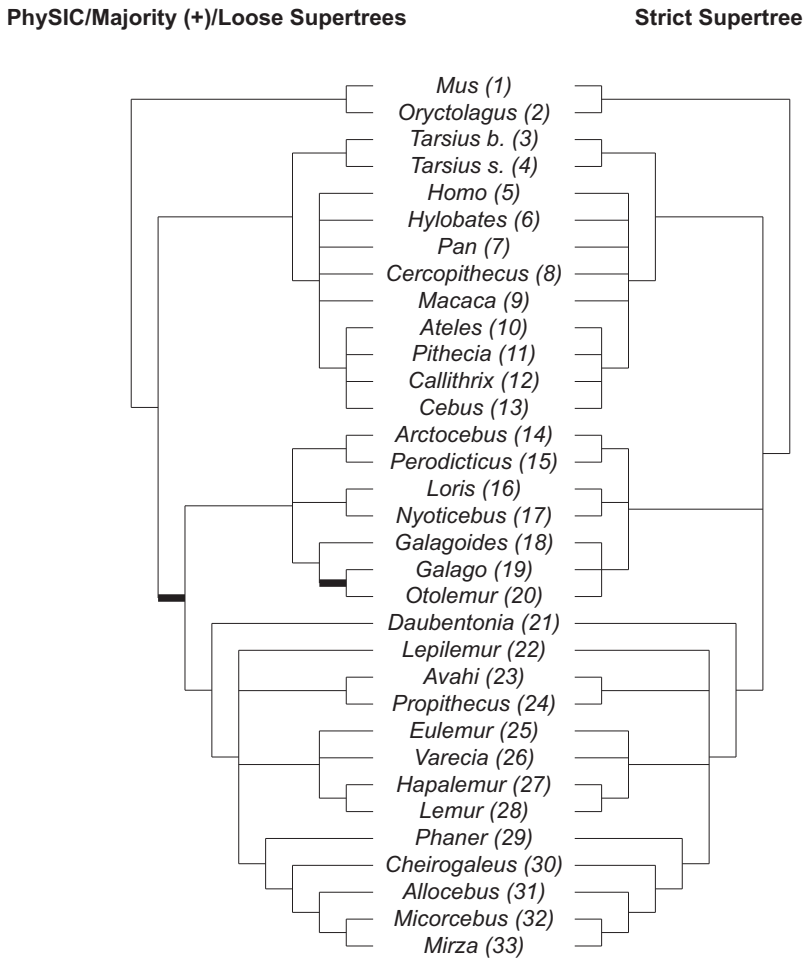

Figure 11.1 Strict and Loose Supertrees for Primates Small Data Set

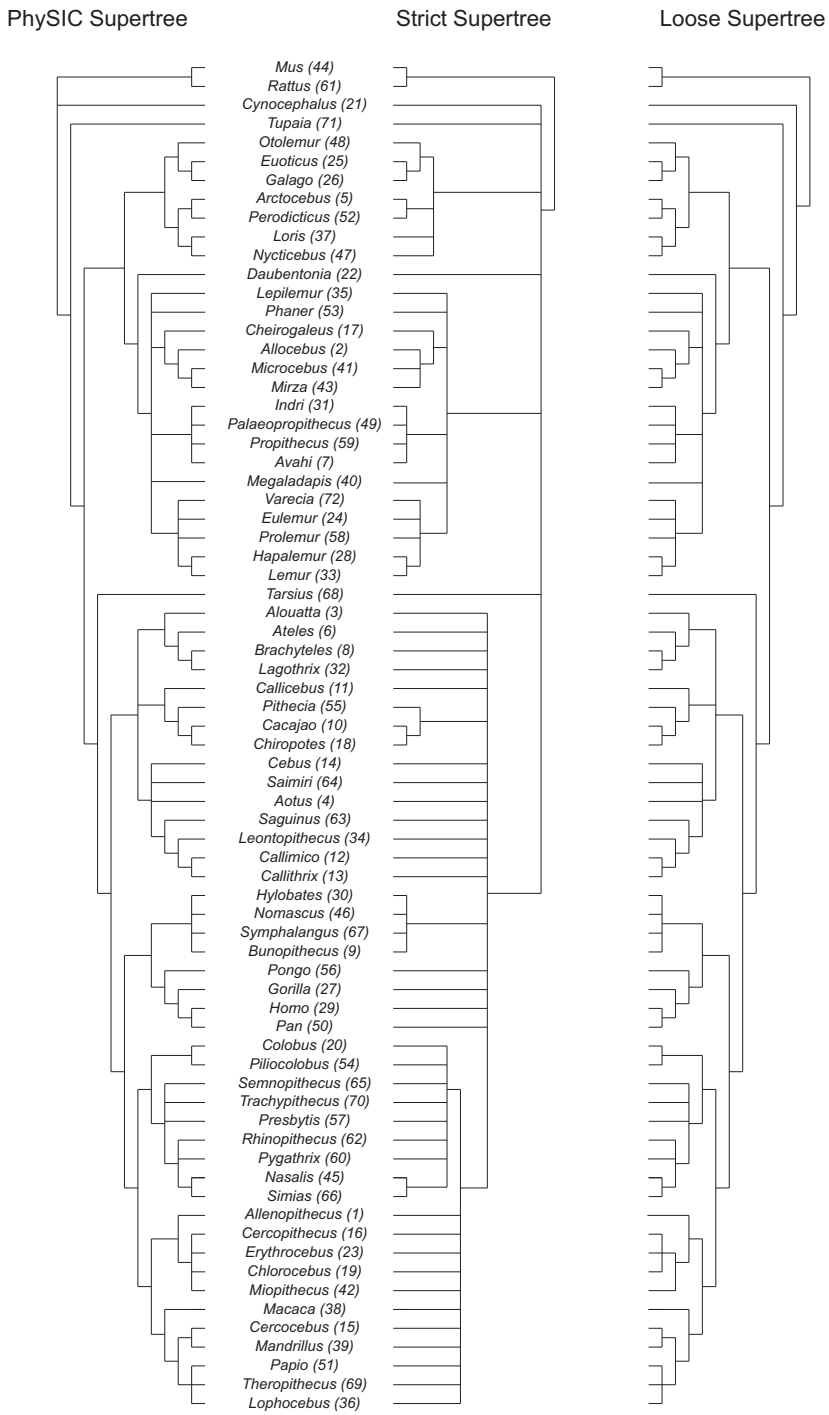

Figure 11.2 Strict and Loose Supertrees for Primates Large Data Set

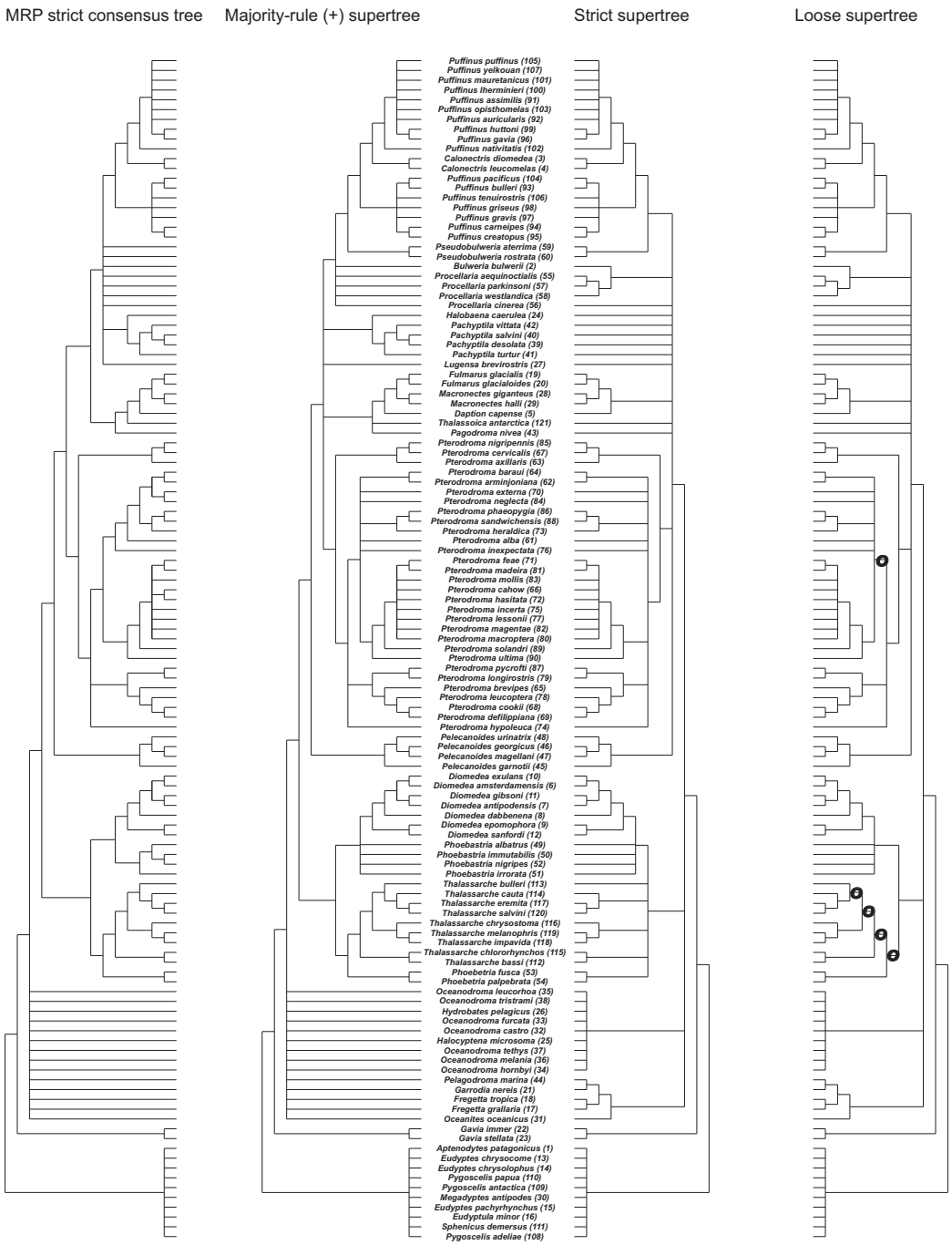

Figure 11.3 Strict Consensus of MRP, Majority-rule (+), Strict and Loose Supertrees for Seabirds Data Set

CHAPTER 12. ILP FOR REDUCED STRICT CONSENSUS TREES

In the next three chapters, the ILP formulations for reduced consensus trees will be developed. We define the problem before describing the controlling matrices, variables and constraints, followed by the objective values and the sizes of ILP. Some definitions are common for three or two reduced consensus trees, while others are unique to the specified reduced consensus tree. We will follow the sequence of reduced strict, loose and majority-rule consensus trees.

Let $P = (t_1, \ldots, t_k)$ be a profile of rooted trees with identical leaves and $\mathcal{L}(P)$ be its leaf set. Let $X \subset \mathcal{L}(P)$. An *induced sub-profile* $P' = (t'_1, \ldots, t'_k)$ is a profile of induced subtrees such that $t'_i = t_i|X$ for $i = 1, \ldots, k$. A *candidate reduced strict consensus tree* T is the strict consensus tree of P'. An *optimal reduced strict consensus tree* T^* is a candidate strict consensus tree whose number of nontrivial clusters is maximum among all choices of X.

12.1 Binary Matrices for Profile and Split-Tree Relationship

Let $P = (t_1, \ldots, t_k)$ be a profile where $\mathcal{L}(P) = n$. n includes a special root taxon. t_j is represented by matrix $M(t_j)$ whose m_j columns correspond to the ***unique*** nontrivial splits of t_j that have not appeared in trees before t_j. Let $m = \sum_{j=1}^{k} m_j$. Suppose the column i of $M(t_j)$ corresponds to the split $A|B$ in t_j and let x be a taxon in $\mathcal{L}(P)$. Then, $M_{x_i}(t_j) = 1$ if $x \in A$, $M_{x_i}(t_j) = 0$ if $x \in B$. All taxa in the same block as the root are assigned 1. $M(P)$ is a matrix obtained by concatenating matrices $M(t_1), \ldots, M(t_k)$.

The first tree is special for reduced strict consensus tree because all reduced splits or clusters can be found in the first reduced tree. This is not true for loose or majority-rule reduced consensus trees because the reduced split/cluster might come from another tree.

Define an m by k binary matrix $Intree$. $Intree(i, j) = 1$ if and only if the i^{th} reduced split

is in the j^{th} reduced tree. If the i^{th} original split is in the j^{th} original tree, so are their reduced forms. Hence, $Intree(i,j) = 1$ for those original splits found in original trees. However, even if the i^{th} original split is not in the original j^{th} tree, when reduced, the reduced i^{th} split could still be in the reduced j^{th} tree because the reduced i^{th} split might be trivial, or it becomes identical with a reduced split (either trivial or nontrivial) in the reduced j^{th} tree. Therefore, if $Intree(i,j) \neq 1$ for original split/original tree, we put a question mark to $Intree(i,j)$. We will use the unknown $Intree(i,j)$ to define U_{ij} variables later.

12.2 Variables and Constraints

For the induced subtree, we provide $n-1$ S binary variables to select which taxa are included. $S_r = 1$ if and only if the r^{th} row (taxon) is in the induced subtree.

For the taxa in the selected rows, the patterns of induced splits i and j are 00, 01, 10, and 11. The pattern 11 always appears since every split has the root filling as 1. The presence or absence of these patterns for columns i and j is indicated by the settings of variables $B_{ij}^{(ab)}$, where $a, b \in \{0,1\}$, $i = 1, \ldots, m_1$, $j = 1, \ldots, m$, and $i < j$.

$B_{ij}^{(ab)} = 1$ if and only if there is selected a taxon r such that $S_r = 1$ and $M_{ri} = a$ and $M_{rj} = b$. M being known, $B_{ij}^{(ab)}$s are determined by the related S_rs. We have that $B_{ij}^{(ab)} \Leftrightarrow \bigcup_{q=1}^{p} S_{r_q}$, or

$$\begin{aligned} -S_{r_1} - S_{r_2} - \cdots - S_{r_p} + B_{ij}^{(ab)} &\leq 0 \\ S_{r_1} + S_{r_2} + \cdots + S_{r_p} - n \cdot B_{ij}^{(ab)} &\leq 0 \end{aligned} \tag{12.1}$$

when ab are 01 and 10, as E_{ij} required later.

Note that some $B_{ij}^{(ab)} \equiv 0$ if the pattern ab does not exist in all the rows of splits i and j. Thus in implementation we do not define them and their related constraints.

We define $O(m_1 \cdot m)$ binary variables E_{ij} where $i = 1, \ldots, m_1$, $j = 1, \ldots, m$ and $i < j$ to represent identity across all distinct splits when they are reduced. Observe that $E_{ij} \Leftrightarrow \neg B_{ij}^{(01)} \wedge \neg B_{ij}^{(10)}$, or

$$\begin{aligned} B_{ij}^{(01)} + B_{ij}^{(10)} + 2 \cdot E_{ij} &\leq 2, \\ B_{ij}^{(01)} + B_{ij}^{(10)} + E_{ij} &\geq 1. \end{aligned} \tag{12.2}$$

Define m_1 δ^0 and m_1 δ^1 binary variables to represent nontrivial clusters in the reduced strict consensus tree. Note that a reduced nontrivial cluster must have at least two 0s and two 1s. Hence, for split j, the sum of S_r corresponding to $M(r,j) = 0$ must be at least 2, and the sum of S_r corresponding to $M(r,j) = 1$ must be at least 1, excluding the root, which is another 1. For $1 \leq j \leq m_1$, we have $\delta_j^0 \Leftrightarrow \sum_{q=1}^{p} S_{r_q} \geq 2$, or

$$\begin{aligned} 2 \cdot \delta_j^0 - \sum_{q=1}^{p} S_{r_q} &\leq 0 \\ \sum_{q=1}^{p} S_{r_q} - n \cdot \delta_j^0 &\leq 1 \end{aligned} \tag{12.3}$$

Similarly we have $\delta_j^1 \Leftrightarrow \sum_{q=1}^{p} S_{r_q} \geq 1$, where $1 \leq r_q \leq n-1$, or

$$\begin{aligned} \delta_j^1 - \sum_{q=1}^{p} S_{r_q} &\leq 0 \\ \sum_{q=1}^{p} S_{r_q} - n \cdot \delta_j^1 &\leq 0 \end{aligned} \tag{12.4}$$

Based on the unknown elements of $Intree$, define at most $m_1 \cdot (k-1)$ binary variables U_{ij} where $i = 1, \ldots, m_1$ and $j = 2, \ldots, k$. $U_{ij} = 1$ if and only if the i^{th} split is in the j^{th} tree. For those undetermined U_{ij}, we have the following for i^{th} split and the j^{th} tree, $U_{ij} \Leftrightarrow \bigcup_{\ell \in M(t_j)} E_{i\ell} \cup \neg\delta_i^0 \cup \neg\delta_i^1$, or

$$\begin{aligned} U_{ij} + \delta_i^0 + \delta_i^1 - E_{i\ell_1} - \cdots - E_{i\ell_{m_j}} &\leq 2 \\ E_{i\ell_1} + \cdots + E_{i\ell_{m_j}} - \delta_i^0 - \delta_i^1 - (m_j + 2) \cdot U_{ij} &\leq -2 \end{aligned} \tag{12.5}$$

Note that $U_{i1} \equiv 1$ and are constants.

Define m_1 binary variables W where $W_i = 1$ precisely if the i^{th} split is in every other trees. That is, $W_i \Leftrightarrow \sum_{j=2}^{k} U_{ij} = k-1$, or

$$\begin{aligned} (k-1) \cdot W_i - \sum_{j=2}^{k} U_{ij} &\leq 0, \\ \sum_{j=2}^{k} U_{ij} - W_i &\leq k-2. \end{aligned} \tag{12.6}$$

Define $m_1 - 1$ D binary variables to represent duplicated clusters. Observe that $D_1 \equiv 0$ and that, for $1 < p \leq m_1$, $D_p \Leftrightarrow \bigcup_{i=1}^{p-1} E_{ip}$, or

$$\begin{aligned} D_p - \sum_{i=1}^{p} E_{ip} &\leq 0 \\ \sum_{i=1}^{p} E_{ip} - p \cdot D_p &\leq 0 \end{aligned} \tag{12.7}$$

Define m_1 V binary variables to represent unique clusters in the reduced strict consensus tree. For $1 \leq p \leq m_1$, we have

$$V_p \Leftrightarrow W_p \wedge \neg D_p \wedge \delta_p^0 \wedge \delta_p^1 \tag{12.8}$$

Equivalently we have

$$\begin{aligned} 4 \cdot V_p - W_p + D_p - \delta_p^0 - \delta_p^1 &\leq 1 \\ W_p - D_p + \delta_p^0 + \delta_p^1 - V_p &\leq 2 \end{aligned} \tag{12.9}$$

12.3 Objective Function

The objective value is the number of unique reduced strict splits (clusters). Define objective variable obj.

$$obj - \sum_{p=1}^{m_1} V_p = 0 \tag{12.10}$$

The objective function is

$$\text{maximize} \quad obj \tag{12.11}$$

12.4 Number of Variables and Constraints

The size of variables and constraints are determined by n, m_1, m and k. For variables, we have

1. S: $O(n)$
2. B, E: $O(m_1 m)$
3. U: $O(m_1 k)$

4. δ^0, δ^1, W,D,V: $O(m_1)$

5. obj: $O(1)$

Therefore, the ILP variables is $\max(O(m_1k), O(m_1m))$.

For constraints, we have

1. $B - S$, $E - B$: $O(m_1m)$

2. $U - E - \delta^0 - \delta^1$: $O(m_1k)$

3. $\delta^0 - S, \delta^1 - S$,$W - U$, $D - E$, $V - D - W - \delta^0 - \delta^1$: $O(m_1)$

Therefore, the ILP constraints is $\max(O(m_1k), O(m_1m))$.

Note that $m_1 = O(n)$. The key here is m. When the clusters within P are highly heterogeneous, m approaches $k \cdot n$. When the clusters within P are highly homogeneous, m approaches n. Rogue taxa could be a reason to affect whether it is homogeneous or heterogeneous. Hence for both ILP variables and constraints, the upper bound is $O(k \cdot n^2)$. But depends on its heterogeneity, the running varies wildly.

CHAPTER 13. ILP FOR REDUCED LOOSE CONSENSUS TREES

Let $P = (t_1, \ldots, t_k)$ be a profile of rooted trees with identical leaves and $\mathcal{L}(P)$ be its leaf set. Let $X \subset \mathcal{L}(P)$. An *induced sub-profile* $P' = (t'_1, \ldots, t'_k)$ is a profile of induced subtrees such that $t'_i = t_i|X$ for $i = 1, \ldots, k$. A *candidate reduced loose consensus tree* T is the loose consensus tree of P'. An *optimal reduced loose consensus tree* T^* is a candidate loose consensus tree whose number of nontrivial clusters is maximum among all choices of X.

13.1 Binary Matrix for Profile

Let $P = (t_1, \ldots, t_k)$ be a profile where $\mathcal{L}(P) = n$. n includes a special root taxon. P is represented by matrix $M(P)$ whose m columns correspond to ***unique*** nontrivial splits of P. $M(P)$ starts with $M(t_1)$, and more distinct nontrivial splits from subsequent $M(t_i)$s are concatenated to form $M(P)$. Suppose the column i of $M(P)$ corresponds to the split $A|B$ in t_j and let x be a taxon in $L(P)$. Then, $M_{x_i}(t_j) = 1$ if $x \in A$, $M_{x_i}(t_j) = 0$ if $x \in B$. All taxa in the same block as the root are assigned 1.

The reduced loose consensus tree clusters must be in one tree and be compatible with every other clusters. In real problems, it is expected input trees share a large percentage of common clusters. The succinct arrangement reduces the size of ILP model.

13.2 Variables and Constraints

For the induced subtree, we provide $n - 1$ S binary variables to select which taxa are included. $S_r = 1$ if and only if the r^{th} row (taxon) is in the induced subtree.

For the taxa in the selected rows, the patterns of induced splits i and j are 00, 01, 10, and 11. The pattern 11 always appears since every split has the root filling as 1. The presence or

absence of these patterns for columns i and j is indicated by the settings of variables $B_{ij}^{(ab)}$, where $a, b \in \{0, 1\}$, $i, j = 1, \ldots, m$, number of unique clusters in all trees and $i < j$. Note that the clusters within the first tree ($i, j = 1, \ldots, m_1$) are compatible with each other. Their induced clusters are also mutually compatible.

$B_{ij}^{(ab)} = 1$ if and only if there is selected a taxon r such that $S_r = 1$ and $M_{ri} = a$ and $M_{rj} = b$. Because M is known, $B_{ij}^{(ab)}$s are determined by the related S_rs. We have that $B_{ij}^{(ab)} \Leftrightarrow \bigcup_{q=1}^{p} S_{r_q}$, or

$$\begin{aligned} -S_{r_1} - S_{r_2} - \cdots - S_{r_p} + B_{ij}^{(ab)} &\leq 0 \\ S_{r_1} + S_{r_2} + \cdots + S_{r_p} - n \cdot B_{ij}^{(ab)} &\leq 0 \end{aligned} \tag{13.1}$$

when ab are 00, 01 and 10. Note that $B_{ij}^{(00)}$ where $1 \leq i, j \leq m_1$ are not needed, as well as its corresponding $B - S$ constraints.

Note that some $B_{ij}^{(ab)}$ variables can be predetermined to be 0 if pattern ab does not exist in all the rows of splits i and j. Thus in implementation we do not define them and their related constraints.

We define binary variables E_{ij} where $i, j = 1, \ldots, m$ and $i < j$ to represent identity across all tree splits. Observe that $E_{ij} \Leftrightarrow \neg B_{ij}^{(01)} \wedge \neg B_{ij}^{(10)}$. We have

$$\begin{aligned} B_{ij}^{(01)} + B_{ij}^{(10)} + 2 \cdot E_{ij} &\leq 2, \\ B_{ij}^{(01)} + B_{ij}^{(10)} + E_{ij} &\geq 1. \end{aligned} \tag{13.2}$$

We define binary variables C_{ij} where $i = 1, \ldots, m$, $j = m_1 + 1, \ldots, m$, and $i < j$ to represent compatibility across all tree splits and the rest tree splits. The compatibility of first tree splits are self evident. Observe that $\neg C_{ij} \Leftrightarrow B_{ij}^{(00)} \wedge B_{ij}^{(01)} \wedge B_{ij}^{(10)} \wedge B_{ij}^{(11)}$. Since $B_{ij}^{(11)} = 1$ for rooted trees, we have

$$\begin{aligned} B_{ij}^{(00)} + B_{ij}^{(01)} + B_{ij}^{(10)} + 4C_{ij} &\geq 3, \\ B_{ij}^{(00)} + B_{ij}^{(01)} + B_{ij}^{(10)} + C_{ij} &\leq 3. \end{aligned} \tag{13.3}$$

where $1 \leq i \leq m$ and $m_1 + 1 \leq j \leq m$.

If two original clusters, cluster i and cluster j are compatible, their induced clusters on the same set of taxa are compatible. Hence $C_{ij} \equiv 1$. The variable C_{ij} and the constraints 13.3

are not needed. However, if cluster i and cluster j are incompatible, their induced clusters could be incompatible or compatible, depending on what taxa are selected. Hence C_{ij} and the constraints 13.3 cannot be omitted.

Define m binary variables W where $W_i = 1$ precisely if the ith cluster is compatible with every other cluster. That is, $W_i \Leftrightarrow \sum_{j=1,j\neq i}^{m} C_{ij} = m-1$, or

$$\begin{aligned} (m-1)\cdot W_i - \sum_{j=1,j\neq i}^{m} C_{ij} \leq 0, \\ \sum_{j=1,j\neq i}^{m} C_{ij} - W_i \leq m-2. \end{aligned} \tag{13.4}$$

When $1 \leq i,j \leq m_1$, $C_{ij} \equiv 1$. In other ranges where $j < i$, $C_{ij} \equiv C_{ji}$, which was defined earlier and will be used.

Define $m-1$ D binary variables to represent duplicated clusters. Observe that $D_1 \equiv 0$ and is a constant. For $1 < p \leq m$, $D_p \Leftrightarrow \bigcup_{i=1}^{p-1} E_{ip}$, or

$$\begin{aligned} D_p - \sum_{i=1}^{p} E_{ip} \leq 0 \\ \sum_{i=1}^{p} E_{ip} - p\cdot D_p \leq 0 \end{aligned} \tag{13.5}$$

Define m δ^0 and m δ^1 binary variables to represent nontrivial clusters in the reduced loose consensus tree. Note that a reduced nontrivial cluster must have at least two 0s and two 1s. Hence, for split j, the sum of S_r corresponding to $M(r,j) = 0$ must be at least 2, and the sum of S_r corresponding to $M(r,j) = 1$ must be at least 1, excluding the root, which is another 1. For $1 \leq j \leq m_1$, we have $\delta_j^0 \Leftrightarrow \sum_{q=1}^{p} S_{r_q} \geq 2$, or

$$\begin{aligned} 2\cdot\delta_j^0 - \sum_{q=1}^{p} S_{r_q} \leq 0 \\ \sum_{q=1}^{p} S_{r_q} - n\cdot\delta_j^0 \leq 1 \end{aligned} \tag{13.6}$$

Similarly we have $\delta_j^1 \Leftrightarrow \sum_{q=1}^{p} S_{r_q} \geq 1$ where $1 \leq r_q \leq n-1$, or

$$\begin{aligned} \delta_j^1 - \sum_{q=1}^{p} S_{r_q} \leq 0 \\ \sum_{q=1}^{p} S_{r_q} - n\cdot\delta_j^1 \leq 0 \end{aligned} \tag{13.7}$$

Define m V binary variables to represent unique clusters in the reduced loose consensus tree. For $1 \leq p \leq m$, we have $V_p \Leftrightarrow W_p \wedge \neg D_p \wedge \delta_p^0 \wedge \delta_p^1$, or

$$\begin{aligned} 4 \cdot V_p - W_p + D_p - \delta_p^0 - \delta_p^1 &\leq 1 \\ W_p - D_p + \delta_p^0 + \delta_p^1 - V_p &\leq 2 \end{aligned} \tag{13.8}$$

13.3 Objective Function

The objective value is the number of unique reduced loose clusters. Define objective variable obj.

$$obj - \sum_{i=1}^{m} V_i = 0 \tag{13.9}$$

The objective function is

$$\text{maximize} \quad obj \tag{13.10}$$

13.4 Number of Variables and Constraints

The size of variables and constraints are determined by n and m. For variables, we have

1. S: $O(n)$
2. B,E, C: $O(m^2)$
3. W,D,δ^0, δ^1,V: $O(m)$
4. obj: $O(1)$

Therefore, the ILP variables is $O(m^2)$.

For constraints, we have

1. $B - S$, $E - B$, $C - B$: $O(m^2)$
2. $W - C$, $D - E$, $\delta^0 - S, \delta^1 - S$, $V - D - W - \delta^0 - \delta^1$: $O(m)$

Therefore, the ILP constraints is $O(m^2)$.

When the clusters within P are highly heterogeneous, m approaches kn. When the clusters within P are highly homogeneous, m approaches n. Rogue taxa could be a reason to affect

whether it is homogeneous or heterogeneous. Hence for both ILP variables and constraints, the upper bound is $O(k^2n^2)$.

CHAPTER 14. ILP FOR REDUCED MAJORITY-RULE CONSENSUS TREES

Let $P = (t_1, \ldots, t_k)$ be a profile of rooted trees with identical leaves and $\mathcal{L}(P)$ be its leaf set. Let $X \subset \mathcal{L}(P)$. An *induced sub-profile* $P' = (t'_1, \ldots, t'_k)$ is a profile of induced subtrees such that $t'_i = t_i|X$ for $i = 1, \ldots, k$. A *candidate reduced majority-rule consensus tree* T is the majority-rule consensus tree of P'. An *optimal reduced majority-rule consensus tree* T^* is a candidate majority-rule consensus tree whose number of nontrivial clusters is maximum among all choices of X.

14.1 Binary Matrices for Profile and Split-Tree Relationship

Let $P = (t_1, \ldots, t_k)$ be a profile where $\mathcal{L}(P) = n$. n includes a special root taxon. t_j is represented by matrix $M(t_j)$ whose m_j columns correspond to the ***unique*** nontrivial splits of t_j that have not appeared in trees before t_j. Let $m = \sum_{j=1}^{k} m_j$. Suppose the column i of $M(t_j)$ corresponds to the split $A|B$ in t_j and let x be a taxon in $\mathcal{L}(P)$. Then, $M_{x_i}(t_j) = 1$ if $x \in A$, $M_{x_i}(t_j) = 0$ if $x \in B$. All taxa in the same block as the root are assigned 1. $M(P)$ is a matrix obtained by concatenating matrices $M(t_1), \ldots, M(t_k)$.

Define an m by k binary matrix $Intree$. $Intree(i, j) = 1$ if and only if the i^{th} reduced split is in the j^{th} reduced tree. If the i^{th} original split is in the j^{th} original tree, so will their reduced forms be. Hence, $Intree(i, j) = 1$ for those original splits found in original trees. However, even the i^{th} original split is not in the original j^{th} tree, when reduced, the reduced i^{th} split could be in the reduced j^{th} tree because the reduced i^{th} split might be trivial, or it becomes identical with a reduced split (either trivial or nontrivial) in the reduced j^{th} tree. Therefore, if $Intree(i, j) \neq 1$ for original split/original tree, we put a question mark to $Intree(i, j)$. We will

use the unknown $Intree(i,j)$ to define U_{ij} variables later.

For the induced subtree, we provide $n-1$ S binary variables to designate which taxa to include. $S_r = 1$ precisely if the r^{th} row (taxon) is in the induced subtree.

For the taxa in the selected rows, the patterns of induced columns i and j are 00, 01, 10, and 11. The pattern 11 always appears since every cluster has the root filling as 1. The presence or absence of these patterns for columns i and j is indicated by the settings of variables $B_{ij}^{(ab)}$, where $a, b \in \{0,1\}$, $i, j = 1, \ldots, m$, and $i < j$.

$B_{ij}^{(ab)} = 1$ if and only if there is selected a taxon r such that $S_r = 1$ and $M_{ri} = a$ and $M_{rj} = b$. M being known, $B_{ij}^{(ab)}$ can be determined by the related S_r. We have that $B_{ij}^{(ab)} \Leftrightarrow \bigcup_{q=1}^{p} S_{r_q}$, or

$$\begin{aligned} -S_{r_1} - S_{r_2} - \cdots - S_{r_p} + B_{ij}^{(ab)} &\leq 0 \\ S_{r_1} + S_{r_2} + \cdots + S_{r_p} - n \cdot B_{ij}^{(ab)} &\leq 0 \end{aligned} \tag{14.1}$$

when ab are 01 and 10, as needed in E_{ij} later.

Note that some $B_{ij}^{(ab)} \equiv 0$ if the pattern ab does not exist in all the rows of splits i and j. Thus in implementation we do not need to define them and their related constraints.

We define $\frac{m(m-1)}{2}$ binary variables E_{ij} where $i, j = 1, \ldots, m$, $i < j$ to represent identity across all reduced clusters. Observe that $E_{ij} \Leftrightarrow \neg B_{ij}^{(01)} \wedge \neg B_{ij}^{(10)}$, or

$$\begin{aligned} B_{ij}^{(01)} + B_{ij}^{(10)} + 2 \cdot E_{ij} &\leq 2, \\ B_{ij}^{(01)} + B_{ij}^{(10)} + E_{ij} &\geq 1. \end{aligned} \tag{14.2}$$

Define m δ^0 and m δ^1 binary variables to represent nontrivial clusters in the reduced strict consensus tree. Note that a reduced nontrivial cluster must have at least two 0s and two 1s. Hence, for split j, the sum of S_r corresponding to $M(r,j) = 0$ must be at least 2, and the sum of S_r corresponding to $M(r,j) = 1$ must be at least 1, excluding the root, which is another 1. For $1 \leq j \leq m$, we have $\delta_j^0 \Leftrightarrow \sum_{q=1}^{p} S_{r_q} \geq 2$, or

$$\begin{aligned} 2 \cdot \delta_j^0 - \sum_{q=1}^{p} S_{r_q} &\leq 0 \\ \sum_{q=1}^{p} S_{r_q} - n \cdot \delta_j^0 &\leq 1 \end{aligned} \tag{14.3}$$

Similarly we have $\delta_j^1 \Leftrightarrow \sum_{q=1}^{p} S_{r_q} \geq 1$ where $1 \leq r_q \leq n-1$, or

$$\begin{aligned} \delta_j^1 - \sum_{q=1}^{p} S_{r_q} &\leq 0 \\ \sum_{q=1}^{p} S_{r_q} - n \cdot \delta_j^1 &\leq 0 \end{aligned} \tag{14.4}$$

Based on the unknown elements of $Intree$, define at most $m \cdot (k-1)$ binary variables U_{ij} where $i = 1, \ldots, m$ and $j = 1, \ldots, k$. $U_{ij} = 1$ if and only if the i^{th} cluster is in the j^{th} tree. That is, $U_{ij} \Leftrightarrow \bigcup_{\ell \in M(t_j)} E_{i\ell} \cup \neg\delta_i^0 \cup \neg\delta_i^1$, or

$$\begin{aligned} U_{ij} + \delta_i^0 + \delta_i^1 - E_{i\ell_1} - \cdots - E_{i\ell_{m_j}} &\leq 2 \\ E_{i\ell_1} + \cdots + E_{i\ell_{m_j}} - \delta_i^0 - \delta_i^1 - (m_j + 2) \cdot U_{ij} &\leq -2 \end{aligned} \tag{14.5}$$

Note that the i^{th} cluster is always in its own tree. $U_{ip} \equiv 1$ if the i^{th} cluster is originally in t_p. If $i \geq \ell$, $E_{\ell i}$ can be replaced by $E_{i\ell}$, which was defined earlier.

Define m binary variables W where $W_i = 1$ precisely if the i^{th} cluster is in more than half of the trees. That is, $W_i \Leftrightarrow \sum_{j=1}^{k} U_{ij} > \frac{k}{2}$. When k is odd, it becomes $W_i \Leftrightarrow \sum_{j=1}^{k} U_{ij} \geq \frac{k+1}{2}$. When k is even, it becomes $W_i \Leftrightarrow \sum_{j=1}^{k} U_{ij} \geq \frac{k}{2} + 1$. When k is odd, it is expressed as

$$\begin{aligned} \sum_{j=1}^{k} U_{ij} - \frac{k-1}{2} \cdot W_i &\geq 1, \\ \sum_{j=1}^{k} U_{ij} - \frac{k+1}{2} \cdot W_i &\leq \frac{k-1}{2}. \end{aligned} \tag{14.6}$$

When k is even, it is expressed as

$$\begin{aligned} \sum_{j=1}^{k} U_{ij} - \frac{k}{2} \cdot W_i &\geq 1, \\ \sum_{j=1}^{k} U_{ij} - \frac{k}{2} \cdot W_i &\leq \frac{k}{2}. \end{aligned} \tag{14.7}$$

Define $m-1$ D binary variables to represent duplicated clusters in the reduced profile. Duplicated clusters can be in the same tree or different trees. Observe that $D_1 \equiv 0$ and is a constant. For $1 < p \leq m$, we have $D_p \Leftrightarrow \bigcup_{i=1}^{p-1} E_{ip}$, or

$$\begin{aligned} D_p - \sum_{i=1}^{p} E_{ip} &\leq 0 \\ \sum_{i=1}^{p} E_{ip} - (p-1) \cdot D_p &\leq 0 \end{aligned} \tag{14.8}$$

Define m V binary variables to represent unique clusters in the reduced majority-rule consensus tree. For $1 \leq p \leq m$, we have $V_p \Leftrightarrow W_p \wedge \neg D_p \wedge \delta_p^0 \wedge \delta_p^1$, or

$$\begin{aligned} 4 \cdot V_p - W_p + D_p - \delta_p^0 - \delta_p^1 &\leq 1 \\ W_p - D_p + \delta_p^0 + \delta_p^1 - V_p &\leq 2 \end{aligned} \tag{14.9}$$

14.2 Objective Function

The objective value is the number of unique reduced majority clusters. Define objective variable obj.

$$obj - \sum_{i=1}^{m} V_i = 0 \tag{14.10}$$

The objective function is

$$\text{maximize} \quad obj \tag{14.11}$$

14.3 Number of Variables and Constraints

The size of variables and constraints are determined by n, m_1, m and k. For variables, we have

1. S: $O(n)$
2. B, E: $O(m^2)$
3. U: $O(mk)$
4. $\delta^0, \delta^1, W, D, V$: $O(m)$
5. obj: $O(1)$

Therefore, the ILP variables is $\max(O(mk), O(m^2))$.

For constraints, we have

1. $B - S$,$E - B$: $O(m^2)$
2. $U - E - \delta^0 - \delta^1$: $O(mk)$
3. $\delta^0 - S, \delta^1 - S$,$W - U$,$D - E$,$V - D - W - \delta^0 - \delta^1$: $O(m)$

Therefore, the ILP constraints is $\max(O(mk), O(m^2))$.

When the clusters within P are highly heterogeneous, m approaches kn. When the clusters within P are highly homogeneous, m approaches n. Rogue taxa could be a reason to affect whether it is homogeneous or heterogeneous. Hence for both ILP variables and constraints, it is $O(k^2n^2)$.

CHAPTER 15. EXPERIMENTAL RESULTS FOR REDUCED CONSENSUS TREES

Our data sets were derived from those used in a previous study by Pattengale et al. (2011). There are 16 sets of bootstrapped trees constructed from single-gene and multi-gene DNA sequences, with 125-2,554 taxa. We refer to the sets by the number of sequences present in them. For each set of trees, we extracted a set of 50 trees on 20 leaves. For this, we randomly selected the required number of trees and restricted them on a random set of leaves of the required size.

We compared our ILP formulation with the rogue taxon solution proposed by Pattengale et al. (2011). The latter tries to uncover new internal edges by merging bipartitions that are sufficiently similar that the merged bipartition becomes frequent enough to be included in the consensus tree on the reduced leafset. Though quite fast, the heuristic cannot guarantee an optimal solution.

We computed the strict and majority reduced consensus trees with 16 20-taxon-50-tree input files. All experiments were run on an Intel Core 2 64 bit quad-core processor (2.83GHz). The ILPs were solved using CPLEX (CPLEX is a trademark of IBM). We used MATLAB to generate the input files for CPLEX and to post-process the output files. Table 15.1 and Table 15.2 show the cases where the ILP approach improved over the heuristic method. The columns gives the serial number, total running time, original internal edges (OIE), new internal edges after rogue taxon removal (NIE), and total leaf loss (LL) for our solution. Also included are the heuristic NIE and LL results (HNIE) and (HLL).

We can make some preliminary observations, based on our experimental results. First, the local search heuristics does not seem as effective when the removed rogue taxa account for a high percentage of the total number of taxa. On the other hand, in most cases, the improvement

Table 15.1 ILP and Heuristic Results for Strict Reduced Consensus Trees

Dataset	Time(sec)	OIE	NIE	HNIE	LL	HLL
150	334	4	**7**	4	10	0
218	78	5	**8**	7	6	3
500	243	5	**8**	6	8	3
1288	79	7	**9**	8	7	1
1512	1897	6	**7**	6	3	0
2000	20	7	**10**	9	5	2
2554	22	8	**9**	8	2	0
628	9	9	9	9	0	0
1908	163	8	8	8	1	0

Table 15.2 ILP and Heuristic Results for Majority-rule Reduced Consensus Trees

Dataset	Time(sec)	OIE	NIE	HNIE	LL	HLL
150	1366	15	15	15	0	0
218	1762	15	**16**	15	1	0
500	62	18	18	18	0	0
1288	6929	14	16	16	1	1
1512	7659	15	**16**	15	1	0
2000	1417	15	**16**	15	2	0
2554	28	18	18	18	0	0
628	46	15	**16**	15	1	0
1908	30550	14	**15**	14	1	0

achieved by the exact algorithm over the heuristic method is at most one, showing that the heuristic method is quite accurate.

Second, the ILP maximizes NIE only and does not minimize the leaf loss. Thus our ILP seems to lose more leaves than the heuristic method, which attempts to minimize the leaf loss as well as maximize NIE. It is straightforward to use an ILP solver to find all optimal solutions and choose the one with minimum leaf loss. Alternatively, it is easy to modify the objective function of the ILPs described in the previous section so that it becomes a weighted sum of the number of leaves and the number of internal edges. No new variables need to be introduced, since the sum of the S_i's gives us the size of the selected set X. An interesting phenomenon occurs when the number of leaves and the number of internal edges have the same weight, which is effectively what is done in Pattengale et al. (2011). Here, we find that the ILP is reluctant to remove taxa to make gain on internal edges. Indeed, it would appear that the

method is too conservative in this case.

Third, we found that reduced majority consensus trees tend to lose many fewer leaves than the reduced strict consensus trees; sometimes no leaves are lost at all. This might be explained as follows. In the strict reduced consensus case, if there is a rogue taxon that is placed incorrectly in just one tree but correctly in others, it has to be removed in order for all of the trees to have common clusters so that they are included in the strict consensus. If it is not removed, it could lead to a highly unresolved strict consensus tree. Removing it might lead to loss of internal edges, but the gain is large enough to justify the removal. On the contrary, for reduced majority-rule consensus, as long as the taxon is correctly placed in more than half of the trees, the answer is correct. If removing it leads to loss of internal edges, the resolution actually decreases.

Fourth, the reduced consensus problems for strict and majority-rule consensus are independent. That is one might be able to improve resolution by one method, but not by the other. For instance, there are instances where the resolution of the majority-rule consensus tree can be improved by rogue taxon removal but the strict cannot. This happens because two clusters that were non-majority previously, after the removal of rogue taxon become the same and hence the combined frequency crosses the threshold value. However, the combined frequency is still not high enough (i.e., 100%) for the cluster to appear in reduced strict consensus. In passing, we should note that in general the majority-rule reduced consensus ILP takes longer to solve than the strict reduced consensus ILP.

Finally, the ILP is much slower than the heuristic method. The latter typically ran within a fraction of a seconds in most of the cases we studied. In contrast, as the numbers of taxa and trees increase, the exact algorithm quickly becomes impractical. Despite their drawbacks, however, our exact solutions give a good benchmark against which to evaluate and explore the limitations of heuristic methods.

CHAPTER 16. CONCLUSIONS

Our results indicate that the majority-rule (+) method produces biologically reasonable phylogenies (i.e., phylogenies with no unsupported groups), and that the method is practical for medium-scale problems. Unfortunately, while polynomial, the size of our ILP is quadratic in the total number of splits in the input trees. This, together with the fact that solving the ILP takes exponential time in the worst case limits the range of applicability of the basic ILP formulation. It also explains in part why the addition of a single tree to a data set can convert a tractable problem into an intractable one. More extensive tests are needed to assess the limitations of the basic ILP approach accurately. In any event, our computational experience shows that the technique does handle some real, biologically significant, problems nicely. Moreover, our results suggest that the ILP approach, in combination with our data reduction heuristic is a promising way to tackle larger problems.

Loose and strict supertrees provide a rigorous approach for combining phylogenetic information. We have shown that it is possible to construct such supertrees for datasets on the scale of those encountered in practice. From a mathematical standpoint, it would be interesting to elucidate the relationship between our approach and the triplet based approach of PhySIC [Ranwez et al. (2007)], as well as possible quartet-based versions of the latter. From a practical standpoint, a more detailed performance analysis of loose and strict supertrees is needed to truly evaluate their scalability. To conduct such a study, we first need to improve the efficiency of our prototype implementation. There are a number of ways to do this. Something as simple as using a compiled version of the software should increase the speed notably in small problems.

The exact algorithms for the reduced consensus trees serve as a benchmark for evaluating various approximation algorithms. They also enable us to study more properties which could

form the basis of more efficient approach. For instance, detailed analysis of the exact solutions when we gradually increase the number of lost leaves reveals two trends. One is to achieve optimality by expanding the same subset of discarded leaves. The other is to shift the subset of leaves at certain point. Another example is that in the strict reduced consensus trees, we observed a leaf loss number beyond which the reduced trees from various input trees are the same (convergence point) followed by a stage in which the number of internal edges decreases by 1 whenever leaf loss increases by 1. Close examination reveals that the reduced trees at and beyond convergence point are binary. Thus removing one more taxon creates one more degree-2 node and reduces one more internal edge. Without exact solution, we are not able to do this type of study.

In general, this dissertation is about exact solutions, in majority-rule (+) supertrees, conservative supertrees, and rogue taxa problems. Its methodology is quite different from popular approximation methods in computational biology. It is an interesting research direction on how to utilize our technique to help improve the approximating algorithms. By providing deeper understanding and better heuristics, similar to the pioneering work in the Traveling Salesman Problem [Applegate et al. (2006)], we could make more contribution in tackling these large and hard biology problems.

BIBLIOGRAPHY

Adams III, E. N. (1972). Consensus techniques and the comparison of taxonomic trees. *Systematic Zoology*, 21(4):390–397.

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

Amenta, N., Clarke, F., and St. John, K. (2003a). A linear-time majority tree algorithm. In *Proc. 3rd Workshop Algs. in Bioinformatics (WABI'03)*, volume 2812 of *Lecture Notes in Computer Science*, pages 216–226. Springer-Verlag.

Amenta, N., Clarke, F., and St. John, K. (2003b). A linear-time majority tree algorithm. In *Proc. 3rd Workshop Algs. in Bioinformatics (WABI'03)*, pages 216–227. SpringerVerlag.

Amir, A. and Keselman, D. (1994). Maximum agreement subtree in a set of evolutionary trees. *SIAM Journal on Computing*, 26:758–769.

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2006). *The Traveling Salesman Problem*. Princeton University Press, Princeton, NJ.

Bansal, M., Burleigh, J. G., Eulenstein, O., and Fernández-Baca, D. (2010). Robinson-Foulds supertrees. *Algorithms for Molecular Biology*, 5(1):18.

Barthélemy, J., McMorris, F. R., and Powers, R. C. (1992). Dictatorial consensus functions on n-trees. *Mathematical Social Sciences*, 25:59–64.

Barthélemy, J. P. and McMorris, F. R. (1986). The median procedure for n-trees. *Journal of Classification*, 3:329–334.

Baum, B. and Ragan, M. (2004). The MRP method. In Bininda-Emonds, O., editor, *Phylogenetic supertrees: Combining Information to Reveal the Tree of Life*, pages 17–34. Kluwer Academic, Dordrecht.

Baum, B. R. (1992). Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, 41:3–10.

Beck, R. M. D., Bininda-Emonds, O. R. P., Cardillo, M., Liu, F. G. R., and Purvis, A. (2006). A higher-level MRP supertree of placental mammals. *BMC Evol. Biol.*, 6:93.

Berry, V. and Semple, C. (2006). Fast computation of supertrees for compatible phylogenies with nested taxa. *Systematic Biology*, 55:270–288.

Bininda-Emonds, O. R. P., editor (2004). *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, volume 4 of *Series on Computational Biology*. Springer, Berlin.

Bininda-Emonds, O. R. P., Cardillo, M., Jones, K. E., MacPhee, R. D. E., Beck, R. M. D., Grenyer, R., Price, S. A., Vos, R. A., Gittleman, J. L., and Purvis, A. (2007). The delayed rise of present-day mammals. *Nature*, 446:507–512.

Bininda-Emonds, O. R. P. and Sanderson, M. J. (2001). Assessment of the accuracy of matrix representation with parsimony analysis supertree construction. *Syst. Biol.*, 50:565–579.

Bremer, K. (1990). Combinable component consensus. *Cladistics: The International Journal of the Willi Hennig Society*, 6:369–372.

Brooke, M. d. L. (2002). Seabird systematics and distribution: a review of current knowledge. In Schreiber, E. A. and Burger, J., editors, *Biology of Marine Birds*, pages 57–85. CRC press, Boca Raton, Florida.

Brown, D. G. and Harrower, I. M. (2006). Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 3(2):141–154.

Bryant, D. (1997). *Building trees, hunting for trees, and comparing trees: Theory and methods in phylogenetic analysis*. PhD thesis, Department of Mathematics, University of Canterbury, New Zealand.

Bryant, D. (2003). A classification of consensus methods for phylogenetics. In Janowitz, M., Lapointe, F.-J., McMorris, F., B. Mirkin, B., and Roberts, F., editors, *Bioconsensus*, volume 61 of *Discrete Mathematics and Theoretical Computer Science*, pages 163–185. American Mathematical Society, Providence, RI.

Cardillo, M., Bininda-Emonds, O. R. P., Boakes, E., and Purvis, A. (2004). A species-level phylogenetic supertree of marsupials. *J. Zool.*, 264(11–31).

Chaudhary, R., Burleigh, J. G., and Fernández-Baca, D. (2011). Fast local search for unrooted robinson-foulds supertrees. In *Proceedings of the 7th international conference on Bioinformatics research and applications*, ISBRA'11, pages 184–196, Berlin, Heidelberg. Springer-Verlag.

Chen, D., Diao, L., Eulenstein, O., Fernández-Baca, D., and Sanderson, M. (2003). Flipping: A supertree construction method. In Janowitz, M., Lapoint, F.-J., McMorris, F. R., and Roberts, F. S., editors, *Bioconsensus*, volume 61 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 135–160. American Mathematical Society, Providence, RI.

Chi, Y., Muntz, R., Nijssen, S., and Kok, J. (2004). Frequent subtree mining — an overview. *Fundamenta Informaticae*, 66(1-2):161–198.

Constantinescu, M. and Sankoff, D. (1995). An efficient algorithm for supertrees. *Journal of Classification*, 12:101–112.

Cotton, J. A. and Page, R. D. M. (2004). Tangled trees from molecular markers: reconciling conflict between phylogenies to build molecular supertrees. In Bininda-Emonds, O. R. P., editor, *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, volume 4 of *Series on Computational Biology*, pages 107–125. Springer, Berlin.

Cotton, J. A. and Wilkinson, M. (2007). Majority-rule supertrees. *Systematic Biology*, 56:445–452.

Cranston, K. and Rannala, B. (2007). Summarizing a posterior distribution of trees using agreement subtrees. *Systematic Biology*, 56(4):578.

Danna, E., Fenelon, M., Gu, Z., and Wunderling, R. (2007). Generating multiple solutions for mixed integer programming problems. In Fischetti, M. and Williamson, D. P., editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *LNCS*, pages 280–294. SPRINGER, Berlin.

Day, W. and McMorris, F. (2003). *Axiomatic Consensus Theory in Group Choice and Biomathematics*. SIAM Frontiers in Mathematics, Philadelphia, PA.

Deepak, A., Dong, J., and Fernández-Baca, D. (2012). Identifying rogue taxa through reduced consensus: Np-hardness and exact algorithms. In *ISBRA 2012*.

Delsuc, F., Brinkmann, H., Chourrout, D., and Philippe, H. (2006). Tunicates and not cephalochordates are the closest living relatives of vertebrates. *Nature*, 439:965–968.

Dong, J. and Fernández-Baca, D. (2009). Properties of majority-rule supertrees. *Systematic Biology*, 58(3):360–367.

Dong, J. and Fernández-Baca, D. (2011). Constructing large conservative supertrees. In *WABI*, pages 61–72.

Dong, J., Fernández-Baca, D., and McMorris, F. R. (2010a). Constructing majority-rule supertrees. *Algorithms in Molecular Biology*, 5(2).

Dong, J., Fernández-Baca, D., McMorris, F. R., and Powers, R. C. (2010b). Majority-rule (+) consensus trees. *Math. Biosci.*, 228(1):10–15.

Dong, J., Fernández-Baca, D., McMorris, F. R., and Powers, R. C. (2011). An axiomatic study of majority-rule (+) and associated consensus functions on hierarchies. *Discrete Applied Mathematics*, 159:2038–2044.

Eulenstein, O. (2005). Consensus trees and supertrees. In Aluru, S., editor, *Handbook of Computational Molecular Biology*, volume 9 of *Computer & Information Science*. Chapman & Hall/CRC, Boca Raton, FL.

Eulenstein, O., Chen, D., Burleigh, J., Fernández-Baca, D., and Sanderson, M. (2004). Performance of flip supertrees with a heuristic algorithm. *Systematic Biology*, 53(2):299–308.

Farach, M., Przytycka, T. M., and Thorup, M. (1995). On the agreement of many trees. *Inf. Process. Lett.*, 55(6):297–301.

Finden, C. and Gordon, A. (1985). Obtaining common pruned trees. *Journal of Classification*, 2(1):255–276.

Foulds, L. R. and Graham, R. L. (1982). The Steiner problem in phylogeny is NP-complete. *Adv. Appl. Math.*, 3:43–49.

Goloboff, P. (2005). Minority rule supertrees? MRP, compatibility, and minimum flip may display the least frequent groups. *Cladistics*, 21:282–294.

Goloboff, P. A. and Pol, D. (2005). Semi-strict supertrees. *Cladistics*, 18(5):514–525.

Gordon, A. D. (1986). Consensus supertrees: The synthesis of rooted trees containing overlapping sets of labelled leaves. *J. Classification*, 9:335–348.

Gusfield, D. (2003). Haplotype inference by pure parsimony. In *CPM*, pages 144–155.

Gusfield, D. (2009). The multi-state perfect phylogeny problem with missing and removable data: Solutions via integer-programming and chordal graph theory. In *RECOMB*, pages 236–252.

Gusfield, D., Frid, Y., and Brown, D. (2007). Integer programming formulations and computations solving phylogenetic and population genetic problems with missing or genotypic data. In Lin, G., editor, *Computing and Combinatorics, 13th Annual International Conference, COCOON 2007, Banff, Canada, July 16-19, 2007, Proceedings*, volume 4598, pages 51–64. Springer.

Henzinger, M. R., King, V., and Warnow, T. (1996). Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. In *Proc. 7th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 333–340.

Kennedy, M. and Page, R. D. M. (2002). Seabird supertrees: combining partial estimates of procellariiform phylogeny. *The Auk*, 119(1):88–108.

Lee, C.-M., Hung, L.-J., Chang, M.-S., Shen, C.-B., and Tang, C.-Y. (2005). An improved algorithm for the maximum agreement subtree problem. *Information Processing Letters*, 94(5):211–216.

Margush, T. and McMorris, F. R. (1981). Consensus n-trees. *Bulletin of Mathematical Biology*, 43:239–244.

McMorris, F. R. and Wilkinson, M. (2011). Conservative supertrees. *Syst. Biol.*, 60(2):232–238.

Meacham, C. A. (1982). The loose consensus of evolutionary histories. In *16th Numerical Taxonomy Conference*, Notre Dame, IN.

Nadler, S., Carreno, R., Mejía-Madrid, H., Ullberg, J., Pagan, C., Houston, R., and Hugot, J. (2007). Molecular phylogeny of clade III nematodes reveals multiple origins of tissue parasitism. *Parasitology*, 134(10):1421–1442.

Ng, M. and Wormald, N. (1996). Reconstruction of rooted trees from subtrees. *Discrete Applied Mathematics*, 69(1–2):19–31.

Nunn, G. B. and Stanley, S. E. (1998). Body size effects and rates of cytochrome b evolution in tube-nosed seabirds. *Molecular Biology and Evolution*, 15:1360–1371.

Page, R. (2002). Modified mincut supertrees. In Guigó, R. and Gusfield, D., editors, *Algorithms in Bioinformatics: Second International Workshop, WABI 2002*, volume 2452 of *LNCS*, pages 537–551. SPRINGER.

Pattengale, N., Aberer, A., Swenson, K., Stamatakis, A., and Moret, B. (2011). Uncovering hidden phylogenetic consensus in large datasets. *IEEE/ACM Journal of Computational Biology and Bioinformatics*, 8(4):902–11.

Pattengale, N. D., Gottlieb, E. J., and Moret, B. M. E. (2007). Efficiently computing the robinson-foulds metric. *Journal of Computational Biology*, 14(6):724–735.

Pisani, D. and Wilkinson, M. (2002). MRP, taxonomic congruence and total evidence. *Systematic Biology*, 51:151–155.

Purvis, A. (1995). A modification to Baum and Ragan's method for combining phylogenetic trees. *Systematic Biology*, 44:251–255.

Ragan, M. A. (1992). Phylogenetic inference based on matrix representation of trees. *Molecular Phylogenetics and Evolution*, 1:53–58.

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

Redelings, B. (2009). Bayesian phylogenies unplugged: Majority consensus trees with wandering taxa.

Robinson, D. F. and Foulds, L. R. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147.

Rodrigo, A. G. (1996). On combining cladograms. *Taxon*, 45:267–274.

Ross, H. A. and Rodrigo, A. G. (2004). An assessment of matrix representation with compatibility in supertree construction. In Bininda-Emonds, O. R. P., editor, *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, volume 4 of *Series on Computational Biology*, pages 35–63. Springer.

Scornavacca, C. (2009). *Supertree methods for phylogenomics*. PhD thesis, Univ. of Montpellier II, Montpellier, France.

Scornavacca, C., Berry, V., Douzery, E. J. P., and Ranwez, V. (2008). PhySIC IST: cleaning source trees to infer more informative supertrees. *BMC Bioinformatics*, 9:413.

Semple, C. (2003). Reconstructing minimal rooted trees. *Discrete Applied Mathematics*, 127:489–503.

Semple, C. and Steel, M. (2000). A supertree method for rooted trees. *DAM*, 105:147–158.

Semple, C. and Steel, M. (2003). *Phylogenetics*. Oxford Lecture Series in Mathematics. Oxford University Press, Oxford.

Sibley, C. G. and Ahlquist, J. E. (1990). *Phylogeny and Classification of Birds: A Study in Molecular Evolution.* Yale University Press, New Haven, Connecticut.

Sokal, R. R. and Rohlf, F. J. (1981). Taxonomic congruence in the leptopodomorpha. *Systematic Zoology*, 30:309–325.

Sridhar, S., Lam, F., Blelloch, G. E., Ravi, R., and Schwartz, R. (2008). Mixed integer linear programming for maximum-parsimony phylogeny inference. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5(3):323–331.

Steel, M. A. (1992). The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116.

Sullivan, J. and Swofford, D. (1997). Are guinea pigs rodents? The importance of adequate models in molecular phylogenetics. *Journal of Mammalian Evolution*, 4(2):77–86.

Swenson, K., Chen, E., Pattengale, N., and Sankoff, D. (2011). The kernel of maximum agreement subtrees. In *Proc. International Symposium on Bioinformatics Research and Applications*, pages 123–135. Springer.

Swofford, D. L. (1991). When are phylogeny estimates from molecular and morphological data incongruent? In Miyamoto, M. and Cracraft, J., editors, *Phylogenetic Analysis of DNA Sequences*, chapter 14, pages 295–333. Oxford University Press, New York.

Thomson, R. and Shaffer, H. (2010). Sparse supermatrices for phylogenetic inference: taxonomy, alignment, rogue taxa, and the phylogeny of living turtles. *Systematic Biology*, 59(1):42.

Wilkinson, M. (1994). Common cladistic information and its consensus representation: reduced Adams and reduced cladistic consensus trees and profiles. *Systematic Biology*, 43(3):343.

Wilkinson, M. (1995). More on reduced consensus methods. *Systematic Biology*, 44(3):435.

Wilkinson, M. (1996). Majority-rule reduced consensus trees and their use in bootstrapping. *Molecular Biology and Evolution*, 13(3):437.

Wilkinson, M., Cotton, J. A., Lapointe, F.-J., and Pisani, D. (2007). Properties of supertree methods in the consensus setting. *Systematic Biology*, 56:330–337.

Wilkinson, M., Pisani, D., Cotton, J. A., and Corfe, I. (2005). Measuring support and finding unsupported relationships in supertrees. *Systematic Biology*, 54(5):823–831.

Wilkinson, M., Thorley, J. L., Pisani, D. E., Lapointe, F.-J., and McInerney, J. O. (2004). Some desiderata for liberal supertrees. In Bininda-Emonds, O. R. P., editor, *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, volume 4 of *Series on Computational Biology*, pages 65–85. Springer.

Willson, S. J. (2004). Constructing rooted supertrees using distance. *Bulletin of Mathematical Biology*, 66:1755–1783.

Wilson, D. and Reeder, D., editors (1993). *Mammal Species of the World: A Taxonomic and Geographic Reference*. Smithsonian Institution Press, Washington & London, 2nd edition.

Wojciechowski, M., Sanderson, M., Steele, K., and Liston, A. (2000). Molecular phylogeny of the "Temperate Herbaceous Tribes" of Papilionoid legumes: a supertree approach. In Herendeen, P. and Bruneau, A., editors, *Advances in Legume Systematics*, volume 9, pages 277–298. Royal Botanic Gardens, Kew.

Xiao, Y. and Yao, J. (2003). Efficient data mining for maximal frequent subtrees. In *Proc. IEEE International Conference on Data Mining*, pages 379–386. IEEE.

Zaki, M. (2005). Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE Trans. on Knowl. and Data Eng.*, 17(8):1021–1035.

Zhang, S. and Wang, J. T. L. (2008). Discovering frequent agreement subtrees from phylogenetic data. *IEEE Trans. on Knowl. and Data Eng.*, 20:68–82.