Algorithms on Phylogenetic Trees

Thesis submitted to the University of Cambridge

for the degree of

Doctor of Philosophy

by

Fabio Pardi

St Catharine's College,

February 2009

Except where otherwise stated in the text, this dissertation is the result of my own work and is not the outcome of work done in collaboration.

This dissertation is not substantially the same as any I have submitted for a degree, diploma or other qualification at any other university; and no part has already been, or is currently being submitted for any degree, diploma or other qualification.

This dissertation does not exceed the permitted length.

Fabio Pardi

Contents

Summary

Algorithms on Phylogenetic Trees by Fabio Pardi

The simultaneous rise of digital computers and molecular biology fifty years ago spurred the development of several algorithms for biologically relevant problems and the birth of computational biology (e.g, [**MS57, ED66, NW70**]). Arguably the earliest among these were algorithms for inferring phylogenetic trees based on the present characteristics of species or molecules [**MS57, Sne57, SS63, ECS63, ECS64, CS65, ED66, CSE67, FM67**]. Despite this long tradition, new optimisation problems on phylogenetic trees have continued to appear, and algorithms dealing with them are an active area of research.

This dissertation presents my work on the algorithmics of phylogenetic trees in two separate directions: first, I define and solve a hierarchy of optimisation problems aiming to select a maximally diverse subset of taxa in a tree; second, I explore a recently-proposed distance-based criterion for reconstructing phylogenetic trees: balanced minimum evolution (BME).

The first direction of research is presented in Chapters 2 to 5. After introducing a measure of diversity for subsets of taxa in a tree and discussing its relevance to applications such as conservation biology, I show that a simple "greedy" algorithm is guaranteed to select maximally diverse subsets of a fixed size (Ch. 2). When instead each taxon has a "cost" and the selection is limited by a given "budget" (leading to the possibility of selecting subsets of variable size), dynamic programming algorithms become necessary (Ch. 3). A further improvement, in the context of conservation biology, is to take into account different extinction risks for different taxa (Ch. 4). Selecting taxa for conservation so as to minimise the future loss of diversity leads to a framework proposed by economists, called the Noah's Ark problem, which I generalise and tackle in Chapter 5.

The rest of the thesis deals with BME. First, I illustrate the theoretical and empirical bases supporting the use of this criterion, including a novel result regarding its robustness to noisy estimates of pairwise distances (Ch. 6). Second, I present a branch and bound algorithm for BME; this is the first practical algorithm guaranteed to construct optimal trees with respect to this criterion and I use it to investigate the margins of improvement possible for current algorithms for BME (Ch. 7).

A more detailed summary is given in Chapter 1 (Sec. 1.3), which also introduces the main themes and terminology of this dissertation.

Acknowledgements

As I am writing this, it is very late at night. I am tired, but happy. Happy because I have finally written the last word of Chapter 5 (I seem to work non-linearly even in the choice of what to do next). Happy because I have finally reached the tortoise that was constantly moving away from me (I was starting to believe that I was, not Achilles, but the tortoise!). Happy because I can now turn to new things. I feel like thanking the whole world.

However, that would be quite inaccurate — there are some people ~~who~~ to whom I feel particularly grateful ~~to~~. First and foremost my supervisor, Nick Goldman. Not only has he taught me that sentences should never finish with a preposition, but also that if you really have to use priors, optimist priors are better. He always had words of encouragement when I needed them, and of advice when it was not clear what to do next. Thanks to him, I had the fantastic opportunity of spending three months doing research in New Zealand, meeting other great people to work with and seeing wonderful places.

I would also like to thank all the people who were and have been in my research group during my time at the EBI. I have learned a great deal from them. In particular Ari, Martin, Simon and Tim, whose time here has significantly overlapped with mine. Thank you also to Nick Luscombe, Toby Gibson and David MacKay, my thesis advisory committee from the EBI and beyond, whose suggestions were always very helpful.

I am also very indebted to the beautiful people with whom I had the fortune of collaborating: Benny Chor, Mike Steel, Klaas Hartmann, Beáta Faller, Mike Hendy, David Penny, Barbara Holland, Steffen Kläre, Bui Quang Minh. Mike Steel in particular, was co-responsible for my exciting time in New Zealand.

Writing a thesis is a stressful way of using one's energies and it tends to transform the often-unaware student in a pretty peculiar creature. A special thank you goes to my friends, who managed to be nice with that creature: to Rosie, Phil and Dominic (for still inviting me to dinner), to Beatriz, Luca and Mohammed (for being such good listeners), and to my housemates Chris and Rich (for washing my dishes — sometimes — and lifting my morale — all the time). And of course to my beautiful Alyson, who met me late in the process of becoming that creature and still decided that she likes me (I think). She taught me that Zeno's paradox is wrong.

Most importantly, I would like to thank my wonderful parents Daniela and Carlo for making me what I am (yes *it is* a thank you).

CHAPTER 1

Introduction

1.1. Phylogenetics

Phylogenetics is the discipline concerned with the study of evolutionary relationships, in particular the reconstruction of phylogenetic trees from molecular or morphological data (for example DNA sequences or fossils), their analysis, and their application to yet other disciplines.

A recent application of phylogenetics has been in the choice of species, for a variety of purposes where it is important to choose species that span large portions of the underlying phylogenetic tree. For example, in biodiversity conservation it is important to concentrate conservation efforts on sets of species that are as diverse as possible, so as to avoid the loss of extensive parts of the underlying evolutionary history. Another example is genomics, where again it is desirable to sequence evolutionarily diverse sets of genomes as this allows many inferences to be drawn by comparing them. For both bioconservation and genomics, "diversity" can be mathematically defined in precisely the same way, as I will show in the next chapter. A large focus of my research (Ch. 2 – 5) has been to work on a number of optimization problems whose aim is to select from a tree a maximally diverse subset of its species.

I am also interested in the most classical question in phylogenetics: how can we infer the phylogenetic tree of a group of taxa, given their (present) characteristics? For a long time, morphological features were used for this task and it was left to the good sense of the naturalist to reconstruct phylogenies, but with the advent of molecular sequence data (and computers to analyse them) it became clear that automatic procedures for phylogenetic inference were needed. Many different methods have been developed in the past 40 – 50 years, starting from the pioneering work of R.R. Sokal [**MS57**], P.H.A. Sneath [**Sne57**], L.L. Cavalli-Sforza, A.W.F. Edwards [**ECS63, ECS64**] and others. With very few exceptions, tree reconstruction methods can be seen as optimisation algorithms looking for the tree that best fits the observed data. They essentially differ in the criterion used to measure the degree of fit between tree and data; for example parsimony, least-squares, likelihood. In this thesis, from Chapter 6 onwards, I will present my work on a relatively new criterion for tree reconstruction.

Phylogenetics involves a large amount of specialised terminology, which I briefly introduce in the rest of this section and use throughout this dissertation.

Phylogenetic trees are used to describe the historical relationships among any objects related via a process of common descent with modifications. In the context of biology, these objects can be genes, individuals, populations, species, genera, etc.; in other words any taxonomic unit of interest. As I am here concerned with the general properties of phylogenetic trees, I will usually leave unspecified what these taxonomic units are and will refer to them as *taxa*. In a phylogenetic tree, taxa are represented by its leaves (see below for a definition), and the rest of the tree represents the ancestors of these taxa and the relationships between them.

Depending on the amount of formalism needed for their investigations, some authors (e.g., [**SS03**]) introduce their works with a formal definition of phylogenetic trees, while others (e.g., [**Fel03**]) do not. Here, I adopt a very simple semi-formal approach. Throughout this thesis, a *phylogenetic tree* is a tree in the graph-theoretic sense (a collection of *branches* connecting unordered pairs of *nodes*, see, e.g. [**CLRS01**], Appendix B.5), subject to a few notes:

- a particular node may be designated as the *root* of the tree, in which case the tree is said to be *rooted*; otherwise it is *unrooted*;
- all the non-root nodes that are attached to exactly one branch are called *leaves* and their corresponding branches are called *terminal branches*; the nodes that are not leaves are called *internal nodes* and the branches that are not terminal are called *internal branches*;
- there is a one-to-one correspondence between leaves and taxa of interest; as a consequence, I will often use the words "leaves" and "taxa" interchangeably; if X is the set of taxa in a phylogenetic tree, we say that that tree is *over* X;
- branches have often associated with them real numbers representing the amount of evolution that has occurred between their two extremes (this could be time if taxa are species, or the expected number of substitutions per site if taxa are molecular sequences); these numbers are called *branch lengths*; in this thesis, trees with branch lengths are denoted with calligraphic fonts such as $\mathcal{T}$;
- a rooted tree with branch lengths, $\mathcal{T}$, is said to be *ultrametric* if the sum of the lengths of the branches on the paths from the root to any leaf is constant; note that in this case it is justified to interpret branch lengths as times;
- when a tree does not have lengths associated with its branches, that tree is called a *tree topology*; in this thesis, tree topologies are denoted with normal italic fonts such as T; a tree with branch lengths $\mathcal{T}$ is said to have topology T if T is what is obtained by stripping $\mathcal{T}$ of its branch lengths;
- two trees are effectively considered to be the same, and we write $\mathcal{T}_1 = \mathcal{T}_2$ or $T_1 = T_2$, if they represent the same information; this can be formalised by

introducing a notion of isomorphism between trees, but for simplicity I will avoid doing so;

- a tree is said to be *bifurcating* if all non-root internal nodes are attached to exactly three branches and the root is attached to either one or two branches; nodes attached to more than three branches are also called *multifurcations*; a *multifurcating* tree is one containing multifurcations;

Lastly, there is one more requirement that I only add for consistency with most literature in phylogenetics (but very few results in this thesis depend on this assumption):

- no non-root node is attached to exactly two branches.

Note that because of this last requirement, all trees are either bifurcating or multifurcating.

1.2. Problems and algorithms

Many algorithms, certainly all the ones that I deal with in this thesis, are devised in response to a well-defined *problem*. A problem is well-defined when it specifies the possible inputs it can have (its *instances*) and their corresponding desired outputs. Examples of classic problems are:

- given the distances between a number of cities, find a shortest possible tour that visits all the cities (the *traveling salesman problem*);
- given an integer number, determine whether it is prime or not;
- given a program written in a known computer language, determine whether or not this program will ever terminate execution (on an infallible computer with infinite memory and given infinite time).

Among the examples above we can distinguish between *optimisation problems* (such as the first one), where the aim is to find an optimal solution in a set of candidate solutions, and *decision problems* (such as the second and the third), where we seek a binary answer yes/no. There are other types of problems besides optimisation and decision problems. In this dissertation I will only deal with optimisation problems.

Problems are not only categorised on the basis of the types of answer they seek, but also on the basis of how complex it is to compute the answers. For some problems, there is no algorithm giving always the correct answer [**Tur36**]. For most of the problems encountered in practice, however, the complexity of their solution is measured in terms of the amount of computational resources (typically time and memory space) that algorithms need in order to compute the correct output. I now introduce the essential notions needed to discuss the computational complexity of the algorithms and problems encountered in this thesis. For more information, the interested reader may consult one of many excellent textbooks (e.g., [**CLRS01**], Part I and Ch. 34).

In this dissertation I will often use the $O(\,)$ notation to indicate the amount of resources (time, memory) used by an algorithm. This notation is standard both in

computer science and biology (e.g., [**SK88, RN93, BW98**]). Let n be a quantity describing (some aspect of) the size of the input — for example, in the case of an algorithm working on a phylogenetic tree, the number of taxa in the tree. Then, an algorithm runs in $O(f(n))$ time (or uses $O(f(n))$ memory) — where $f(n)$ is a function of n (e.g., $f(n) = n \log n$) — if there exists a constant c such that the algorithm's running time (or memory usage) is at most $cf(n)$ for every possible input corresponding to n. Note that this definition is independent of the the units used to measure time or memory space. It is also important to note that, unless otherwise specified, the notation $O(f(n))$ indicates a guarantee on the *worst-case* performance of an algorithm: for example, there are well-known sorting algorithms that run in $O(n \log n)$ time in the majority of cases, but for which some inputs may require time proportional to n^2; in this case the best we can say without making any distinction between inputs is that the sorting algorithm runs in $O(n^2)$ time.

Another useful notion for this thesis is that of NP-hardness. Many computational problems of practical relevance (and in this thesis we shall encounter a few) can be proved to be NP-hard. The importance of identifying a problem as NP-hard is that this is strong evidence that there is no polynomial-time algorithm solving it — a polynomial-time algorithm is one running in $O(p(n))$ time for some polynomial p, where n measures the amount of memory that is necessary to store the input.

In order to provide a definition of NP-hard problems, we need a number of other definitions: P is the class of decision problems that can be solved by a polynomial-time algorithm. NP is the class of decision problems for which there exists a polynomial-time randomised algorithm using a polynomial number of random bits such that: if the correct answer to the problem is *no*, then it always returns *no*; if the correct answer to the problem is *yes*, then it returns *yes* with nonzero probability. For example, it is not difficult to see that the decision version of the traveling salesman problem above, where one asks whether there is a tour shorter than a specified threshold, is in NP. A particularly important discovery was that there is a subset of NP, whose elements are called *NP-complete* problems, for which the existence of a polynomial-time solution implies that *every* problem in NP has a polynomial-time algorithm solving it [**Coo71, Kar72**], in other words that the classes P and NP coincide. After almost 40 years, no polynomial algorithm for any of these problem has been found, but neither has a proof that P $\neq$ NP been provided (and this constitutes the most important open problem in theoretical computer science).

What confers NP-complete problems their special status is that it is possible to prove that every problem in NP can be reduced in polynomial time to any NP-complete problem, where "reducing" means transforming the input of a problem into the input of another so that the desired output for the latter problem can be used to compute (again, polynomially) the output for the former. This implies that if there were a polynomial-time algorithm for a NP-complete problem, then this algorithm

could be used to solve, after an appropriate reduction, any problem in NP in polynomial time. NP-hard problems are the same as NP-complete problems, but without the requirement of being members of NP: they are *all* problems (including also optimisation problems) to which every problem in NP can be reduced in polynomial time. Clearly, in order to show that a problem is NP-hard, it suffices to show that another known NP-hard problem reduces to it in polynomial time.

Knowing that a problem is NP-hard strongly suggests that there is no polynomial algorithm solving that problem. In these cases, it becomes reasonable to devise *heuristics* — algorithms that have no guarantees on the quality of the produced solution.

1.3. Outline of the thesis

Although the first section of each chapter gives an overview of its contents, I now outline the contents of each chapter in a more concise way.

Chapter 2 introduces the phylogenetic diversity (PD) of a set of taxa, a quantitative measure of their diversity that takes into account their evolutionary relationships. PD is relevant for choosing taxa in a variety of applications, ranging from conservation biology, where the aim is to save from extinction the maximum amount of biodiversity, to comparative genomics, where sequencing a diverse set of genomes ensures good statistical power for tests aimed at finding interesting genomic features (e.g., conserved elements). Somewhat surprisingly, the problem of selecting the k taxa with maximum PD from a phylogenetic tree — which I call MaxPD — can be solved with a simple "greedy" algorithm, a nice property formally proved in the second part of Chapter 2.

Chapters 3, 4 and 5 deal with a hierarchy of optimisation problems that generalise MaxPD by introducing more realistic and complex constraints on the selection and by extending its objective function. Each problem in this hierarchy is a particular case of the ones that follow.

Chapter 3 defines and solves BMaxPD, the budgeted version of MaxPD, where different taxa require different amounts of resources (money, time, labour, etc.) for their selection (sequencing or conservation) and a limit is set on the total amount of resources available (a budget). This problem is computationally harder than the previous one, but I provide a number of dynamic programming algorithms which run efficiently when the budget is not too large.

Chapter 4 introduces the next problem in the hierarchy, which has previously been referred to as the Noah's Ark problem (NAP). The NAP more realistically models the choices that have to be made in conservation biology, as it takes into account taxon-specific extinction probabilities — clearly an important factor when selecting taxa for conservation. The objective now is to choose a subset of taxa to conserve so as to maximise the expected "future PD", that is the PD of the species that will survive until some specified future time. It turns out that in its simplest

form, where conservation can ensure survival, the NAP can be reduced to BMaxPD. In the first half of Chapter 4, I also investigate the distribution of the future PD resulting from all possible extinction scenarios.

Chapter 5 deals with the gNAP, a realistic generalisation of the NAP, where for each taxon, instead of making a binary decision between conservation and neglect, we can choose an amount of resources to allocate for its conservation, given that the probability of survival of that taxon is a known function of that amount. In other words, we aim to find the best way to distribute the available resources among the taxa under consideration, again with the objective of maximising the expected future PD. For this problem, we can extend one of the approaches described in chapter 3, but I have been unable to prove whether worst-case polynomial behaviour holds. Therefore, I show that on many typical instances of this problem the proposed algorithm is reasonably efficient.

In Chapter 6, the focus switches to the classical problem of phylogenetic tree reconstruction. In this chapter, I introduce and review a criterion for tree reconstruction based on pairwise distances between taxa: balanced minimum evolution (BME). This allows us to derive (partly novel) results that lay the ground for the following chapter. The main novel result here is about a "safety radius" for BME: if the input distances are within a neighbourhood with this radius of the correct distances, then the BME criterion is guaranteed to identify the correct tree.

Finally, Chapter 7 presents my original work on BME. In order to investigate the gains that can be obtained by improving the existing heuristics, I designed and implemented an exact algorithm for this criterion; that is, one that is guaranteed to give optimal solutions. It uses the branch and bound strategy, a typical approach to tackle problems for which a polynomial solution is not available. My experiments with the implemented program are only preliminary, as some components of the algorithm leave scope for improvement. These experiments nevertheless suggest directions for future investigations, which I discuss in detail.

Other research that I have carried out during the course of my PhD is not presented in this dissertation. A brief description and pointers to the publications this work has led to are provided in the next two sections.

1.4. Approximation algorithms for MP and AML

As I mentioned before, tree reconstruction methods essentially differ in the criteria used to measure the fit between tree and data. Interestingly, finding optimal trees with respect to some of the proposed criteria, namely maximum parsimony (MP) and ancestral maximum likelihood (AML) [**ABCH$^+$04**], can be seen as special cases of the *Steiner tree problem*, where given a set of points in a metric space one has to find a tree of minimum total length connecting all these points [**FG82, HRW92**].

The Steiner tree problem is well-known and a number of algorithms have been proposed for this problem. In the case where the space under consideration has finite

size (i.e., a weighted complete graph), *approximation algorithms* — giving solutions with a score within a guaranteed factor of the optimal solution — have been devised (e.g., [**RZ06**]).

What MP and AML have in common is that, given a set of sequences of equal length m in input, they aim to jointly reconstruct a phylogenetic tree and the sequences at the internal nodes of the tree, so as to minimise

$$\sum_{x,y} d(x,y),$$

where the sum is over all pairs of sequences x, y appearing at the extremes of a branch in the tree and $d(x,y)$ is a measure of the distance between them (MP and AML essentially differ for how $d(\,)$ is defined). This formulation makes it clear that MP and AML are special cases of the Steiner tree problem, where the underlying metric space is the set of all the possible sequences of length m with distances defined by $d(\,)$.

In 2006, I joined the then-ongoing work by Benny Chor and colleagues, who were investigating the possibility of adapting the existing approximation algorithms for the Steiner-tree problem to MP and AML tree reconstruction. This is not straightforward: these algorithms run in polynomial-time in the size of the space, which in this case is exponential in m. A closer look at these algorithms, however, reveals that all that is needed is a subroutine that can construct optimal trees for a small number of k sequences (e.g., 3 or 4) (polynomially in m, but not necessarily in k). Then an approximately-optimal tree for all the sequences in input can be obtained from these "small" optimal trees.

Constructing such a subroutine for MP is easy: it simply consists of a brute-force examination of all possible tree topologies (where calculating the parsimony score for each topology can be done with the Fitch algorithm [**Fit71**]). For AML, this is more complicated, as no equivalent of the Fitch algorithm is known for AML. This is where I gave my contribution to this work: I helped Benny Chor and collaborators to characterise solutions to AML for $k = 3$ sequences and especially for the more convoluted case $k = 4$: it turns out that optimal ancestral sequences can be found in a small set of candidate sequences.

The results of this work are approximation algorithms for MP and AML that construct trees whose score is within a guaranteed factor from the optimal score: 16/9 for AML and approaching 1.55 for MP. These results were published in *IEEE/ACM Transactions on Computational Biology and Bioinformatics* [**ACPR09**]. Since these approximation ratios are only worst-case guarantees, it remains to be seen how good the algorithms are in practice, and whether it is useful to combine them with the traditional heuristics for these problems.

1.5. Analysis of proposed ENCODE transcripts

Phylogenetic methods are extensively used in analyses comparing genomes across different species, helping to identify genomic regions with functionally important roles. Throughout the first period of my doctorate study, I was involved in the ENCODE (ENCyclopedia Of DNA Elements) project, a large collaborative effort aiming to identify "all functional elements" in the human genome [**FGG+04, BSD+07**]. The project was then in its "pilot phase", consisting of the evaluation and development of several high-throughput techniques — both computational and experimental — on a set of 44 genomic "target" regions representing about 1% of the human genome. The aim was to determine the best strategy for analysing the remaining 99%, which is the objective of the current ENCODE "production phase".

One important component of the project was the sequencing and comparative analysis of several other vertebrate genomic regions that are orthologous to the 44 human targets [**TTB+03, MBP+03, MVM+05, MCA+07**]. Our research group carried out analyses of non-neutral evolution over all proposed protein-coding transcripts in these regions. I worked first on the visualisation of the results and then, when it became clear that a significant proportion of the results were unrealistic, on trying to understand whether the results could be used to assess the quality of the data on which the analyses were based.

A DNA site is said to evolve *neutrally* if selection has no role in its evolution. Non-neutral evolution occurs either when selection eliminates new mutations in the population — in which case we say that the site is under *purifying*, or *negative*, selection — or when some new variants are actively selected and therefore increase their frequency in the population at a rate higher than what expected by random drift alone — in which case we say that the site is under *adaptive*, or *positive*, selection. Unfortunately it is difficult to distinguish the effect of selection from that of mutation: is a sequence conserved throughout evolution because of a locally low mutation rate or because of purifying selection? Is a high substitution rate due to an increased mutation rate or to adaptive selection?

In the case of protein-coding sequences, evidence for non-neutral evolution can be obtained by comparing *synonymous* and *non-synonymous* substitution rates: if a codon is evolving neutrally, it is expected to have no particular "preference" towards evolving into another synonymous codon (i.e., encoding the same amino acid) rather than into a non-synonymous one. If instead the fitness of individuals is affected by the amino acid encoded by the given codon, we can expect non-synonymous substitutions to be either over-represented (in the case of positive selection) or under-represented (negative selection), in comparison to synonymous substitutions. This is the idea at the basis of the state-of-the-art methods to detect non-neutral evolution in protein-coding sequences [**MG05, YWN05**] that our group applied to the ENCODE transcripts.

The most noticeable result of these analyses was that many exons, sometimes entire transcripts, showed very unusual selection signatures: in these exons, positive selection seems to be widespread, contrary to the common expectation that most codons in protein-coding genes are subject to strongly negative selection (with occasional isolated regions of adaptively important sites undergoing positive selection). My impression is that most of these unusual results are artefacts due to unmet assumptions of the methods used. Little is known about the sensitivity of the methods to violations of their assumptions, which are:

(1) The sequences are orthologous.
(2) The sequences are correctly aligned.
(3) *All* the sequences in the alignment are protein-coding.
(4) The reading-frame is correctly specified.
(5) Selection has not affected synonymous substitution rates.

Detecting the cases where these assumptions are not met can be relevant on one hand to improve the quality of the data used for subsequent analyses, and on the other hand to assess and compare the quality of the methods used to produce the data, for example the methods for orthology prediction, sequence alignment and gene annotation (the idea being that better methods should produce fewer examples of unusual selection signatures). Furthermore, if the unusual selection signatures we observe actually correspond to real widespread adaptive evolution in proteins, then clearly their detection is very important.

My main role in the ENCODE project was to develop an automatic system to score annotations (exons and transcripts) on the basis of how unusual their selection signatures are. Although my scores have been used by other researchers on some occasions (especially by members of the GENCODE project, the part of ENCODE focusing on the identification of protein-coding genes), I do not think this part of my work has been very successful, in my opinion because of the inability to disentangle between the various causes for the unusual selection patterns observed (see points 1–5 above). It was however very instructive for me to take part in a large collaborative project and to participate in a number of publications in prestigious journals [**MCA+07, BSD+07, TWF+08**].

CHAPTER 2

Measuring and optimising diversity in a tree: phylogenetic diversity and a greedy algorithm

2.1. Picking leaves from a tree

It is often the case that we are faced with a number of possibilities and not all of them can be pursued. A choice must be made. In biology often the choice is among a number of species, where we may need to select a subset from a collection of species for some purpose, for example for experimentation or (as will be discussed extensively) for conservation. In these cases, often the objective is to choose a subset of species that is as "diverse" as possible.

If we imagine species as leaves in the evolutionary tree of all living organisms, it is clear that this notion of diversity should somehow correspond to "spatial spread" inside this tree. This and the next three chapters will deal with a natural measure of diversity and spread in a tree and with algorithms for selecting sets of species that are optimal with respect to it. This measure is called *phylogenetic diversity* (PD), defined so that the PD of a set of species is the total length of the phylogenetic tree connecting these species (formal definitions will be given in sec. 2.5). This measure is quite natural and not surprisingly it has been independently proposed several times in the literature [**Fai92, Wei92, NM97, PG05**], with applications in a few different disciplines (discussed in secs. 2.2, 2.3 and 2.4).

The central topic of this chapter is a simple PD-optimisation problem (sec. 2.6) and a "greedy" algorithm that solves it (sec. 2.7). The problem simply requires finding a fixed number of species with maximum PD in a given tree. In computer science, greedy algorithms — which construct solutions by iteratively taking the best immediate choice — are rarely the best option to solve a problem. However, in this context, it turns out that no other strategy can do better, as the greedy algorithm always produces optimal solutions (formally proven in sec. 2.9). This is somewhat surprising, especially considering its implications when, in real life, species are chosen for comparative genomics or conservation (sec. 2.10).

Many of the results in this chapter have already been presented in a paper I published on *PLoS Genetics* [**PG05**]. Since this paper has attracted some interest after its publication, section 2.12 will summarise the work that other researchers have subsequently carried out on greedy algorithms for PD-maximisation problems.

2.2. PD in conservation biology

Human activities are currently causing an extinction event of unprecedented scale, with extinction rates up to a thousand times higher than background (see [**BGJ03**] and references therein). Conservation biologists have long realised that, faced with this process – now deemed unstoppable – the best that can be done, instead of distributing equally the already scarce resources among all the possible actions, is to concentrate on the most urgent conservation measures. In practice this means that they need to decide which populations, species or geographical sites should be prioritised for conservation. This unenviable situation has been sometimes referred to as "agony of choice" [**VWHW91, Cro92**] or the "Noah's Ark problem" [**Wei98**].

In a seminal note appearing in *Nature* in 1990, Robert May pointed out ([**May90**], p. 129):

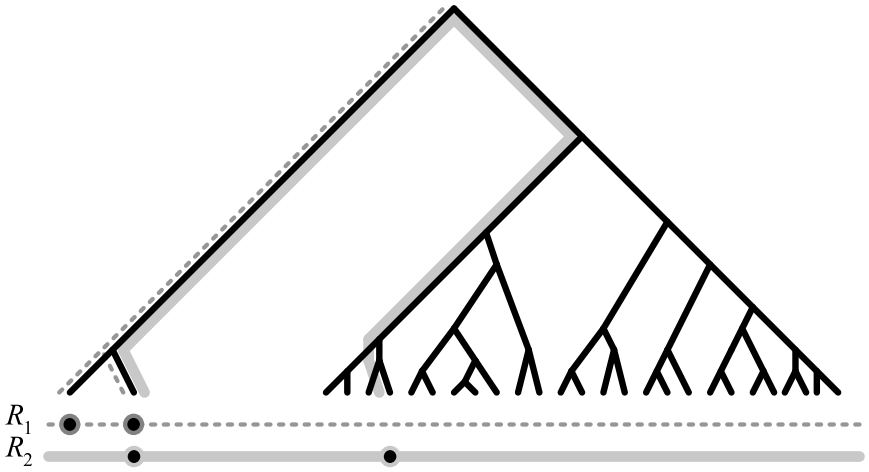
> As more and more species face extinction [...] how do we go about making choices for the ineluctably limited number of places on the ark? [...] how do we go about designing a limited set of reserves that will, in some defined sense, optimise what is saved?

May's answer to these questions (in line with the growing opinion in conservation biology) was that taxonomic relationships among species should be of primary importance in these choices. For example the tuatara, a reptile which can now be found only in a few small islands off the coast of New Zealand, is the unique survivor of a once-common order of reptiles, the Sphenodontians. It represents alone more than 200 million years of evolution and it has several features which set it aside from the other reptiles (such as a well-developed third eye in the centre of its head). Its conservation is therefore deemed to be of primary importance.

The tuatara's lesson is that the extinction of a species not closely related to other living species would cause a much greater loss in biodiversity than that of a species with closer relatives. Although it is clear that phylogenetically distinct taxa should be given priority for conservation, can we be more precise than that? Can we make this statement quantitatively precise? The need for a "calculus of biodiversity" [**May90**] motivated the definition of several mathematical indices in the years that followed May's note: the aim was to define, using phylogenetic trees, measures which could be used for prioritising species for conservation (see [**Cro97**] for an early review).

Some of these measures are simply ways of quantifying the conservation value of a species based on its position in a phylogeny [**VWHW91, AL90**], in other words phylogenetically-based weighting schemes, a research topic which has received some attention also for classical problems in bioinformatics (see sec. 5.8 in [**DEKM98**] for a review and [**ACL89**] as a particular example). The advantage of these methods, which recently have been proliferating again [**HKS08, POD05, RM06, ITC$^+$07,**

FIGURE 2.2.1. Species conservation values are not suitable for assessing the conservation value of different geographical regions.

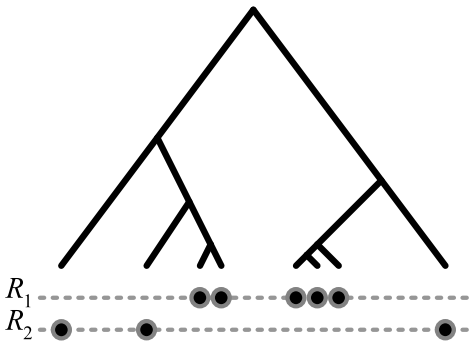


SMM07], is their simplicity and appeal to the public: their immediate use is in producing easy-to-interpret species rankings. For example EDGE, a programme of the Zoological Society of London with strong public resonance [**BBC07, Mar07**], makes public a ranking of mammalian species (on their website: www.edgeofexistence.org – "Top 100") based on how evolutionarily distinct (ED-) and globally endangered (-GE) each species is [**ITC$^+$07**].

A problem with these species-specific scores is that they are not suitable for assessing the conservation value of *sets* of species, as opposed to single species. This is an important point, because one of the main applications conservation biologists are interested in is assessing the conservation value of different geographical regions. For example, consider the theoretical scenario depicted in figure 2.2.1: we have two regions, R_1 and R_2, containing the two species indicated by the dots on the first (dashed) and second (thick) line, respectively, and we want to assess their relative conservation values. The straightforward way to obtain such values would be to sum together the conservation values of the species in each region – an approach which has been applied in the past (e.g., [**VWHW91**]). It is intuitive that the methods mentioned above, however they are defined, will all give the highest species-specific values to the two species at the extreme left of the phylogenetic tree in figure 2.2.1. (This is because, taken individually, both these species are the most phylogenetically isolated). As a consequence, R_1 will necessarily have a higher summed conservation value than R_2, and more in general than any other set of two species in this tree. However this is contrary to the common intuition that at least one species should be conserved from the largest clade on the right.

A related problem is that the ranking produced by the species-specific scores should not be taken as a programmed list of conservation projects, to be undertaken in the specified order. In the example in figure 2.2.1, the two leftmost species will

FIGURE 2.2.2. Species richness is sensitive to the definition of species.
Region R_1 contains the five species (or subspecies) indicated by the dots on the first dashed line. Region R_2 contains the three species corresponding to the dots on the second dashed line. If each of the taxonomic units in the tree is considered as a separate species, R_1 has a higher species richness than R_2 (5 species vs. 3). If we adopt a coarser definition of species, the opposite may be true, as R_1 may have as little as 2 species.



come top of the list, but once we decide to conserve one of them, conserving the other should have a much lower priority.

The problem with using species-specific conservation values to evaluate regions or to plan a number of conservation projects is that these values fail to take into account the "complementarity" [**VWHW91**] of different species: the conservation value of a species should depend on what has been conserved before. Some of the indices mentioned above (e.g., [**SMM07**]) have the potential of being updated every time a new species is conserved, but this means that any produced ranking is just provisional.

So how do we go about assessing the conservation value, in other words the biodiversity, of a set of species? Traditionally, this has been simply done by counting the number of species in the set. Regions with higher "species richness" are the ones which should be given higher priority. Although this is still a very popular approach, it has at least two disadvantages. First, it does not take into account the arguments I discussed above about the different phylogenetic distinctiveness of different species: do we really want to consider three closely related species as more biodiverse than two much more distantly related species? Second, species richness is very sensitive to the definition of species [**Fai92, MGP03, IMM04**]: depending on how "fine-grained" our concept of species is, the numbers may change considerably and so may our assessment of the relative conservation value of different collections of species or geographical regions (see fig. 2.2.2).

These considerations motivated the definition of a measure of the biodiversity of a set of species that takes into account the phylogenetic relationships among species (tree topology) and also their evolutionary distances (branch lengths). This

is the phylogenetic diversity [**Fai92**] discussed in the previous section. This measure overcomes the problems described above. In figure 2.2.1, the PD of region R_1 (the sum of the lengths of the branches with a dashed line on their side) is smaller than the one of R_2 (branches with thick light grey line), and this corresponds to the intuitive requirement that two species from the same clade should be considered as less biodiverse than two from different clades. In figure 2.2.2, the PD of the two collections of species (or subspecies) is relatively independent of the definition of species (with R_2 having slightly more PD than R_1– not shown).

I pause here to note that, despite the theoretical advantages of PD and the attention that this measure subsequently received from other researchers [**NM97, Cro97, MGP03**], species richness is still perhaps the most widespread way to measure biodiversity in conservation biology. One of the reasons for this is that it has been argued that species richness is a good surrogate measure for PD [**PCVM01, RAB$^+$04, BMdF$^+$06**] and therefore there seems to be no need to use PD directly. However, despite the obvious correlation between these two measures, it has recently been shown that important exceptions may exist: in the Cape of South Africa — a very well-known biodiversity hotspot — plant species are more numerous but phylogenetically more clustered in the west than in the east and this may result in rather different conservation decisions depending on whether we use species richness or PD to assess biodiversity: species richness gives comparatively too little importance to the conservation of the eastern parts of this region [**FGR$^+$07**]. There is thus hope that in the future PD will receive more attention from decision-makers in conservation biology.

Another reason for using PD is that, as was pointed out since its conception [**Fai92**], PD can be regarded as a predictor of feature diversity, in other words of the number of different biological features represented by the collection of conserved species. This means that phylogenetically diverse collections of species are most likely to have uses with economic value (for example to provide food or medicines) and options for future evolutionary diversifications [**FGR$^+$07**]. Section 2.11 will show a possible way of predicting the expected feature diversity resulting from a given phylogeny and how this compares to PD.

Quite interestingly, at more or less the same time as Dan Faith introduced PD [**Fai92**], a much more theoretical approach brought Martin Weitzman — an economist from Harvard who set out to found a general "theory of diversity" — to give a definition of diversity which has PD as a particular case [**Wei92**]. His definition works on collections of objects for which pairwise distances are defined — if these distances coincide with those on an ultrametric tree, then his definition of diversity coincides with that of PD. I will come back to the work of Weitzman later in this thesis.

This and the next three chapters will focus on selecting taxa (e.g., species, subspecies, populations) from a given phylogeny with the aim of maximising their PD.

As an example application, I will show which of the Madagascar lemurs should be given priority for conservation and how this decision is influenced by the various constraints that practitioners may be subject to. A possible criticism to the utility of my work for conservation biology is that conservation is usually applied to sites or regions, rather than species. However I think that the relevance of my work can be found (a) for ex-situ conservation (seedbanks, botanical gardens, zoos), where the targets are indeed species and not sites, and (b) for those cases where the taxa we wish to conserve have very little overlap in their geographic range — so that selecting taxa is the same as selecting regions. This second scenario is indeed very common: without having to evoke Darwin's finches, the typical examples are the different (geographically separated) populations within any given species. Furthermore, in this case conserving populations with maximum PD has the added advantage of preserving genetic variability and therefore the "evolutionary health" of the species as a whole.

2.3. PD in comparative genomics

Comparing homologous biological sequences has enormous potential for increasing our knowledge about their function, structure and evolution, an idea that has been applied virtually everywhere in computational biology — from protein structure prediction to the identification of functional elements in genomic sequences. The latter is one of the main purposes of sequence comparisons: looking for the distinctive evolutionary signatures of protein-coding genes, non-coding RNAs, microRNAs, cis-regulatory motifs [**SLK$^+$07**] and in general all sequences which are subject to particular selective constraints. For example, the simplest of these constraints is purifying selection, which has the effect of "slowing down" the evolution (substitutions, insertions, deletions etc.) of all functional sequences (although in different ways).

With the advent of high-throughput DNA sequencing technologies, comparative studies are increasingly performed on a genomic scale, requiring the sequencing of entire genomes (e.g., [**WLTB$^+$02, GWM$^+$04, LTWM$^+$05, MHE$^+$05, Con07**]) or significant parts of them [**TTB$^+$03, MCA$^+$07**]. Choosing the right species for sequencing is therefore crucial.

This choice is constrained by two considerations. First, the biological characteristics that one wishes to investigate determine a "maximum range" of species to include in the comparative analyses [**Sid02**]: for example, if we wish to find the genetic elements that determine mammal-specific anatomy or physiology by assessing their sequence conservation, obviously the comparison will be limited to mammalian genomes. Similarly, many aspects of human biology must be determined by genomic sequences having orthologs only in a small range of our close relatives, and it is on these species that comparative analyses aimed at human-specific traits should concentrate [**BMO$^+$03**]. The chosen range of species has been dubbed the phylogenetic [**CBP$^+$03**] or lineal [**MJP05**] scope of the analysis. Different research

communities are focusing on different scopes: for example yeasts [**KPE$^+$03**], nematodes [**SBB$^+$03**], fruit flies [**Con07, SLK$^+$07**], mammals [**MCA$^+$07**] and primates [**BMO$^+$03**].

The second constraint on the choice of species for comparative genomics is that genome sequencing is a resource-consuming activity (although this is becoming less and less true) and therefore only a limited number of species can be sequenced. Once a phylogenetic scope has been established, selecting genomes for sequencing from it should be guided by considerations regarding the effectiveness of the resulting comparative analyses, ideally assessed with some quantitative measure of their performance (e.g., sensitivity and specificity in detecting features of interest).

When the problem of choosing genomes for sequence comparisons was first posed, PD was adopted as a natural measure of the information content of a subset of the available species, although the justifications for this were initially minimal [**BMO$^+$03, CBP$^+$03, KPE$^+$03**]. Also, the link with conservation biology had not been realised — I believe this was first pointed out in [**PG05**] — and PD was variously referred to as "divergence" or "total branch length". The idea was that the statistical power in testing various evolutionary hypotheses (e.g., low substitution rates) and thus in finding interesting genomic features (e.g., protein-coding genes, noncoding functional elements) is strongly correlated with the PD of the sequences compared. Therefore, all other things being equal, *sequencing projects* (both at the genome and gene level) *should ideally target taxa with a high collective PD.*

Support for these claims came from computational experiments on the ability of different collections of genomic sequences to detect *conserved* sequences [**TTB$^+$03, MBP$^+$03**], sequences that have evolved at a slower rate than neutral. As Thomas et al. wrote ([**TTB$^+$03**], p. 791):

> An important issue relevant to future genome sequencing projects is the degree to which the detection of highly conserved sequences depends on the particular set of species being studied. [...] For the various subsets of mammalian species, there is strong correlation between total divergence of the subset (as measured by the combined branch length of the phylogenetic tree defined by the specific subset of species) and the ability to detect multi-species conserved sequences. [...] It thus seems that combined branch length will be a useful metric for guiding the selection of additional genomes to sequence.

The importance of conserved genomic sequences lies in the fact that they are likely to be functional, as the simplest explanation for their conservation is the action of purifying selection (e.g., [**WLTB$^+$02, TTB$^+$03, BMO$^+$03**]). In particular, the characterisation of non-coding conserved sequences is of interest because their function is especially intriguing (e.g., [**BPM$^+$04, BSKH05**]). Furthermore, one can hope that if a collection of genomes is suitable for detecting conservation, then

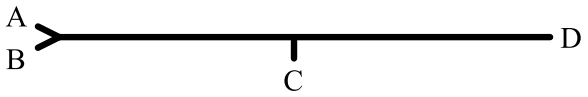
that collection will also be suitable for detecting other, more complex, distinctive evolutionary signatures.

Because of the mathematical simplicity of the methods for detecting conservation, some theoretical studies on the relationship between these methods' performance and genome choice appeared in the literature in 2005. Eddy [**Edd05**] was interested in the number of genomes necessary to detect conserved sequences as a function of the length of these sequences and of the phylogenetic divergence of the genomes. He assumed, for simplicity, a star-like phylogeny in which each genome was connected to a common ancestor with a branch of common length D. His calculations show that, among other things, the number of genomes necessary to attain any chosen sensitivity and specificity is roughly inversely proportional to D, for small values of the branch length D (note that small values of D are the most realistic, because for higher values, say for $D > 0.5$, sequence alignment becomes very problematic). In practice this means that it is roughly equivalent to use 5 genomes at a distance of 0.2 expected substitutions per site from their ancestor or 10 genomes at a distance of 0.1 or 100 genomes at a distance of 0.01. This is perfectly consistent with the view that the total PD of the species being compared (which is constant in these scenarios) determines the discriminative power of a comparative study.

A very simple scenario in which the performance of sequence comparison is precisely measured by PD is when the appearance of a feature can disprove that the sequence under consideration has a certain function. For example an insertion or a deletion with a length which is not a multiple of three is strong evidence against a sequence being protein-coding. Imagine we use the presence/absence of such a "counter-feature" to find the particular type of sequences we are interested in (in my example, coding regions) — possibly in combination with other lines of evidence. In order to maximise the information coming from this test, we would like to use genomes that maximise the probability that the counter-feature arises when a sequence is not of the wanted type. Assuming that the appearance of the counter-feature is irreversible or very unlikely to be reversed (which is realistic at least in the case of indels), then it is easy to see that the probability of observing the counter-feature must increase monotonically with PD (unless the sequence under consideration is of the wanted type). Therefore choosing a set of genomes with maximum PD also maximises the discriminative power of this test.

However, if we drop the assumption of irreversibility in my small example above, or if we allow for different trees in Eddy's analyses (larger trees and/or with a different topology), it is possible to provide examples where the PD of genomes does not coincide with their discovery power. Some examples of this were shown by McAuliffe and colleagues [**MJP05**], who insisted mostly on the fact that if the evolutionary distance between species is too large, their genome sequences will have accumulated so many changes that it will be impossible to recognise whether they have evolved neutrally or under the effect of some selective constraint. In short, saturation makes

FIGURE 2.3.1. The importance of position in choosing genomes for comparative genomics.
Sequences A and D are already available. PD is virtually indifferent between selecting B or C (perhaps it even prefers B, as its branch is slightly longer), but C is a more informative choice.



conservation and neutrality indistinguishable — more generally, it deletes whatever evolutionary signatures selection may leave behind. For example, assuming again the star topology scenario, it is possible to show that if we wish to detect sequences that evolve, say, twice as slow as neutral sites, there exists an optimal branch length D, approximately equal to 1.0 expected substitution per neutral site (fig. 2 in [**MJP05**] and fig. 3 in [**Edd05**]). The more our species deviate from this optimal distance, the smaller our ability to detect conservation.

Although this shows that in general choosing genomes with a very large PD may be a bad idea, I think these counterexamples will arise very rarely in practice as they assume very large distances between species in the given phylogenetic scope. However, one of the criteria when deciding about the phylogenetic scope should be precisely the possibility of recognising orthology and of aligning the genomic sequences: evolutionarily very wide phylogenetic scopes are unrealistic because of the difficulty of these tasks and the pooling of species with rather different biologies. Therefore, I believe that in most cases the choice will be among species that are already not too evolutionarily distant, well within the optimal values of D in the analyses of Eddy and McAuliffe and colleagues.

Another criticism which has received less attention, although it is somewhat implicit in the analyses by McAuliffe et al. [**MJP05**], is that the exact shape of the tree connecting a set of species also plays a role in determining their information content — not only the total length of this tree. I will illustrate this with an example. Consider the simple tree in figure 2.3.1. Imagine that the genomes of A and D have already been sequenced and we want to choose between B and C as the next sequence to acquire. Even without formulating the problem mathematically, it should be intuitive that the central sequence C is more informative than B: it gives us a very close estimate of the unknown ancestral sequence that lies at the node close to it, whereas the other will be very similar to A, a sequence we already know.

To see this even better, let us go back to the example where we look for "counter-features" in order to prove that a sequence does not have a certain function. It is easy to see that if we drop the assumption that the appearance of a counter-feature is irreversible — for example if we are looking at the presence/absence of stop codons

in a candidate protein-coding sequence — it is more likely to observe this counter-feature in one among A, C and D rather than in one among A, B and D, simply because the sequences of A and B will be practically identical.

So, although from the point of view of PD we can be indifferent between choosing B or C (or even slightly prefer B), C is clearly a better candidate for future sequencing — actually one could even show that if we could choose *any* position along the lineage connecting A to D for the next genome, the best choice would be the mid-point of this segment (see sec. 2.11).

One last thing to note about the example in figure 2.3.1 is that this tree strongly violates the molecular clock, that is, it implies that the rate of sequence evolution has changed a lot on some of its branches. An interesting question is whether we can construct examples such as this one — in which PD does not reflect comparative power — on an ultrametric tree (of small size – to avoid saturation). Showing that this is not possible would support the use of PD, since realistic trees are often approximately ultrametric.

I think a lot of work remains to be done on the relationship between PD and the information content, or "discovery power", of genomic sequences. Along the lines already followed by Eddy [**Edd05**] and McAuliffe and colleagues [**MJP05**], it would be interesting to tackle questions such as: under which practical circumstances is PD a bad measure to use? Is it possible to define an alternative measure that more closely reflects the comparative power of a set of sequences? Are the answers to these questions dependent on the particular evolutionary signature that we are trying to detect? Could it be that the optimal choices for detecting, say, coding sequences are not the same as those for detecting, say, RNA genes?

Despite all the criticisms described above and the questions left unanswered, the view of PD as a criterion to assess the usefulness of additional genomes for comparative genomics is still widely held [**SLK+07**]. Furthermore, as the age of individual whole-genome resequencing gets closer and closer, PD may also play a role in choosing individuals for comparative population genomics (cf. the "1000 genomes" project, www.1000genomes.org).

2.4. Other applications?

Many things can be described as "trees". Among physical objects, not only plants, but also the electric lines or the hydraulic pipes in a building, or the dendrites of a neuron are examples. Trees are also very successful as conceptual tools. Although in this thesis I will be primarily concerned with trees showing evolutionary relationships, this is just one of their many possible applications. Trees can be found, for example, in computer science (possibly the most ubiquitous data structure), linguistics (e.g. parse trees), information theory (e.g. to represent codes) and machine learning (decision trees).

Since phylogenetic diversity can be defined for any set labelling a tree, PD is likely to have other applications in addition to the ones I describe here (although I doubt anyone will ever be interested in the PD of a set of taps in a building!), in the same way as many formal concepts which were first intended for one application were then also used for another one.

In particular, a very important use of trees is to represent differences and similarities among objects (usually without the pretense of representing evolutionary relationships): data clustering techniques often produce a "dendrogram" showing a system of hierarchically structured clusters [**ELL01**]. For example, very well known in computational biology are dendrograms of genes or tissue samples constructed on the basis of the correlations in their gene expression patterns [**ESBB98**]. In this context, selecting a number of genes or samples with maximum PD may be used to choose the ones on which, say, time-consuming wet-lab experiments should be carried out. More generally, PD-maximal subsets of observations from a dendrogram may be considered representative of the whole collection and therefore can be selected for further action.

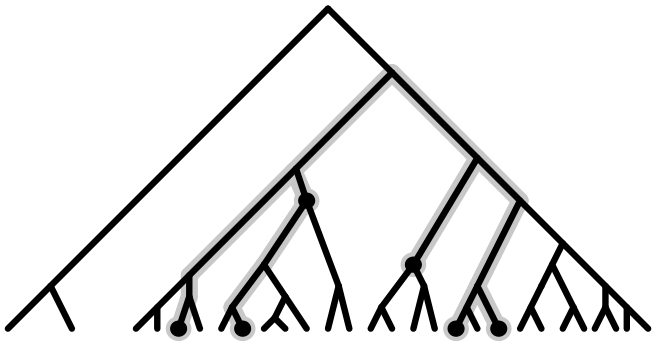
For example, selecting maximally diverse observations from a dendrogram may be useful in machine learning, when labelling the examples to input into a supervised learning system (such as an artificial neural network or a support vector machine) is tedious or resource-consuming. Imagine the goal is to be able to automatically assign biological samples to a number of discrete classes (such as tumour types). Often, obtaining these samples and expression profiles from them is quick and inexpensive, whereas understanding which class these samples belong to requires more involved experimentation. In these cases, although we may have a lot of expression data from many samples, the training set for our learning system will necessarily be of limited size and so we may want to carefully choose the samples to include in this training set. One possible way to do this would be to use the expression profiles to construct a dendrogram for the available samples and then look for a subset of these with maximum PD (under some constraints on the size of this subset or on the total cost of the experimentation required). This would ensure that the training set contains examples representative of the many different expression profiles observed.[1]

Note that the use of PD on a dendrogram for a set of species (as opposed to a phylogeny) has been already suggested by Petchey and Gaston [**PG02a, PG02b**], who aim to capture the complementarity among the species' ecological function.

Returning to the cases where the tree *does* represent evolutionary relationships, the applications of PD are probably not limited to conservation biology and comparative genomics: in the past couple of years I have had many varied suggestions, from selecting representative model strains of microbial pathogens for developing new

[1] A similar scenario can be encountered in many other applications of machine learning, for example when constructing automatic systems for discriminating among the different meanings of a word based on the contexts surrounding its occurrences. In these cases, large datasets of texts in which this word appears can be easily retrieved, but the manual annotation of these is rather tedious.

FIGURE 2.5.1. Restriction of a tree $\mathcal{T}$ on a subset S of its nodes. Highlighted is $\mathcal{T}_S$, where S is the set of the 6 nodes indicated by a black dot. The total length of this tree coincides with uPD(S).



drugs (B. Holland [**Hol01**]), to selecting individuals to genotype for SNP discovery (Junhyong Kim, personal communication).

2.5. Formal definitions: rooted and unrooted PD

In the introduction to this chapter, I defined the PD of a set of species as "the total length of the phylogenetic tree connecting these species". This section will make this statement more precise.

I will assume that we are given a tree with lengths associated with its branches. Typically, this tree will be a phylogenetic tree, but note that the results that follow are also valid for trees not representing evolutionary relationships (to allow applications outside phylogenetics – see sec. 2.4). Also note that I will tend to use the word "taxa" instead of "species", as I wish to remain as general as possible regarding the taxonomic units represented in the tree (genes, individuals, populations, subspecies, species, genera, etc.).

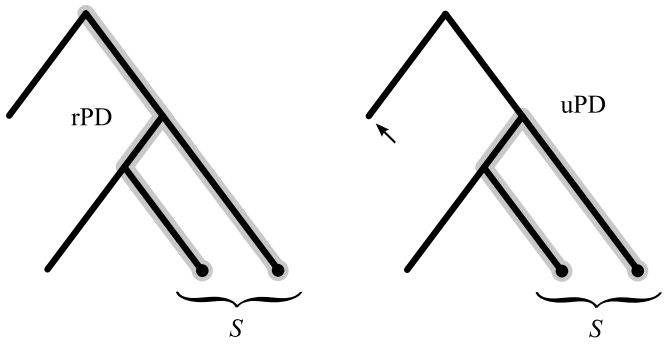
Since in the past the definition of phylogenetic diversity has been interpreted in two different ways by different authors (see, for example, the – rather acrimonious – discussion between [**FB06**] and [**CAD06**]), in order to avoid any ambiguity I will adopt a formal approach. This also allows me to clearly state the difference between the two competing definitions of PD.

DEFINITION 2.5.1. Let $\mathcal{T}$ be a tree and S a subset of the nodes in $\mathcal{T}$. The *restriction* of $\mathcal{T}$ on S, denoted $\mathcal{T}_S$, is the smallest subtree of $\mathcal{T}$ that connects all the nodes in S.

Figure 2.5.1 shows an example of a restriction of a tree on a subset of its nodes.

DEFINITION 2.5.2. Let $\mathcal{T}$ be a tree with lengths associated with its branches and S a subset of the nodes in $\mathcal{T}$. The *unrooted phylogenetic diversity* of S in $\mathcal{T}$, denoted $\text{uPD}_{\mathcal{T}}(S)$, is the sum of all the lengths of the branches in $\mathcal{T}_S$. If $\mathcal{T}$ is rooted in a

FIGURE 2.5.2. Difference between two definitions of PD. Highlighted are the branches whose lengths are summed in order to give rPD(S) (left) and uPD(S) (right). Assuming branch lengths are proportional to those in the figure, S is a subset of size 2 with maximum rPD but not maximum uPD, as exchanging any of the leaves in S with the one indicated by the arrow would increase the uPD of this set.



node ρ, the *rooted phylogenetic diversity* of S in $\mathcal{T}$, denoted $\mathrm{rPD}_{\mathcal{T}}(S)$, is the sum of all the lengths of the branches in $\mathcal{T}_{S\cup\{\rho\}}$.

Note that, for simplicity, I will sometimes drop the subscript from uPD or rPD, when the tree can be inferred from the context. The difference between the two definitions lies in whether or not the length of the path connecting the taxa to the root is included in the calculation of PD. Figure 2.5.2 shows a simple case where $\mathrm{rPD}(S) \neq \mathrm{uPD}(S)$ and also the fact that choosing taxa with maximum PD (the problem described in the next section) is dependent on the chosen definition of PD. In general, the optimisation problems associated with these two versions of PD can be of rather different difficulty, with uPD often posing more problems than rPD. Insisting on this distinction is therefore not merely a pedantic exercise.

Also note that rooted and unrooted PD are linked by this simple equation:

$$\mathrm{rPD}(S) = \mathrm{uPD}(S \cup \{\rho\}), \tag{2.5.1}$$

where ρ is the root of the underlying tree.

Historically the confusion between the two definitions of PD arises from the fact that Faith's paper introducing PD [**Fai92**] was itself somewhat ambiguous: the definition he provides seems to be equivalent to the one of uPD, but whereas one of the formal results ([**Fai92**], p. 4, last line) only holds for uPD, the only example that allows discrimination between the two interpretations ([**Fai92**], fig. 3(a), set R3) is consistent with rPD but not uPD. In later papers (e.g. [**MF98, FB06**]), Faith and colleagues clarified their preference for rPD and insisted that the original definition was indeed rooted [**FB06**], but the alternative definition for PD has become widespread (e.g. [**Cro97, Bar02, Ste05, MKvH06**]) and other authors even refer to rPD with a different name (*evolutionary history* [**NM97**]). In a way, the ambiguity

comes from the fact that if we naively define PD as "the total length of the phylogenetic tree connecting the species", this leaves open the question of how far back in the past such phylogenetic tree should extend.

In conservation biology, rPD is generally seen as a better choice than uPD [**RG02**]. A possible way to justify this (although I believe it has never been argued in these terms) is that, if we consider that the range of the conservation actions available locally is always limited to some limited portion $\mathcal{T}$ of the tree of life, the PD of all the species conserved on Earth (however it is defined – rooted or unrooted) can be expressed as

$$K + \mathrm{rPD}_{\mathcal{T}}(S),$$

where K is a constant not depending on the actions taken in $\mathcal{T}$ and S is the set of species conserved in $\mathcal{T}$. Therefore, since what we are ultimately interested in is conserving the tree of life, the objective at a local level should be to conserve rPD, rather than uPD.

In comparative genomics, on the other hand, i seems more appropriate, as most sequence comparison techniques are independent of root placement. The choice between these two measures will therefore be determined by the intended application and neither of the two can be considered intrinsically superior to the other.

In the following I will often simply write "PD" whenever a statement holds for both uPD and rPD.

2.6. A simple PD-optimisation problem

Given the potential applications of selecting taxa with large PD from a phylogeny (sec. 2.2, 2.3 and 2.4), it is natural to ask what is the best way to carry out this selection. This leads to formulating a number of optimisation problems, which will be the subject of this and the next three chapters. In the remainder of this chapter I will deal with the simplest of these problems, MaxPD, consisting of selecting a fixed number of taxa with maximum PD.

MaxPD:

Input: A tree $\mathcal{T}$ with n leaves and with non-negative lengths associated with its branches and an integer k such that $1 \leq k \leq n$.

Output: A subset S of k leaves of $\mathcal{T}$ with maximum $\mathrm{PD}_{\mathcal{T}}(S)$ (among all subsets of k leaves of $\mathcal{T}$).

Note that this actually defines two slightly different problems, depending on which of the two definitions for PD (sec. 2.5) is adopted. The example in figure 2.5.2 shows that the solutions to these two problems may be different.

An observation with practical importance is that instead of looking for an optimal set of taxa, we may want to look for an optimal way to *extend* a set of taxa we already have. In conservation biology and comparative genomics, for example, some taxa may have already been conserved or sequenced, and so, instead of asking what the

optimal choices would have been, we may look for the next best choices of taxa given the choices that have already been made. These considerations lead to reformulating MaxPD in a more general form, which I will precede with a useful definition.

Definition 2.6.1. A *k-extension* of a set S of nodes in $\mathcal{T}$ is a set S' of nodes in $\mathcal{T}$ such that $S \subseteq S'$ and $|S' - S| = k$.

MaxPD:

Input: A tree $\mathcal{T}$ with n leaves and with non-negative lengths associated with its branches, a proper subset S_0 of the leaves of $\mathcal{T}$ and an integer k such that $1 \leq k \leq n - |S_0|$.

Output: A k-extension S of S_0 with maximum $\mathrm{PD}(S)$ (among all k-extensions of S_0).

Note that the new formulation admits the old one as a particular case: just set $S_0 = \emptyset$.

2.7. A greedy algorithm

Greedy algorithms work by constructing solutions to a problem iteratively, making at each step the choice that at that stage seems optimal ([**CLRS01**], Ch. 16). This is usually not guaranteed to lead to optimal solutions: for example in the travelling salesman problem — where, given a number of cities and the distances between them, we seek the shortest round-trip route that visits each city once — a greedy algorithm may consist of constructing a route by always moving to the next closest city. However it is easy to come up with examples where this strategy actually produces very long routes, much longer than those obtained with "smarter" approaches (e.g., [**GYZ02**]).

In the case of MaxPD the obvious greedy algorithm consists of *repeatedly selecting a leaf that most increases the PD of the set of selected leaves, until k leaves have been added.*

In the next section I will give examples illustrating the functioning of this algorithm, but before then I will clarify a couple of points. In most cases, the algorithm simply consists of setting $S := S_0$ and then applying k times the greedy step below to the set of selected leaves S:

greedy step(S)
find a leaf $x \notin S$ that maximises $\mathrm{PD}(S \cup \{x\})$
$S := S \cup \{x\}$

Note again that there is a distinction between rooted and unrooted greedy steps, depending on the chosen definition for PD.

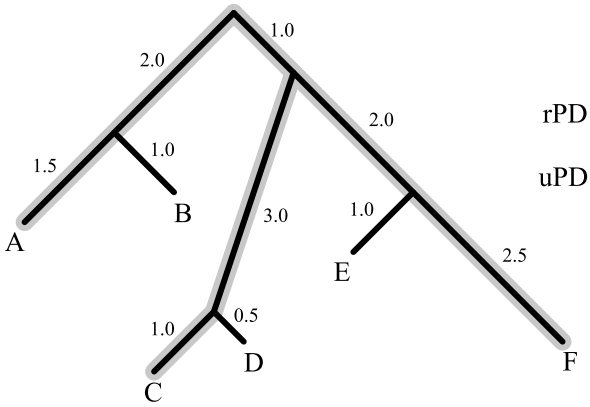
The reason why I wrote "in most cases" in the description above is that in the unrooted case, if $S_0 = \emptyset$, the greedy algorithm should be specified in a slightly different way. In this case it is easy to see that, for example, two unrooted greedy

steps applied to $\emptyset$ do not necessarily result in two leaves with maximum uPD: the first choice will be entirely arbitrary as for any leaf x, $\text{uPD}(\{x\}) = 0$. To avoid this problem, the first step of the algorithm for the unrooted case with $S_0 = \emptyset$ should be to look for the two leaves x and y with maximum $\text{uPD}(\{x, y\})$ and set $S = \{x, y\}$. Then the greedy algorithm will carry on as in the other cases, that is, the greedy step is applied until k taxa have been selected (i.e. it is applied $k - 2$ times).

The interesting fact about this greedy algorithm is that, despite its simplicity and despite its inherent "short-sightedness", it is guaranteed to construct optimal solutions for MaxPD: for whatever alternative algorithm we devise, for example had we – by brute force – considered all the possible choices of k leaves, no better solution than that provided by the greedy algorithm can be obtained. This is exemplified in the next section and formally proven in section 2.9.

As for the exact implementation of this algorithm and computational complexity considerations, I will not enter into these issues here, as this has been the subject of other research [**MKvH06, SNM08**]. In particular Andreas Spillner and colleagues [**SNM08**] present an optimal $O(n)$ time implementation.

FIGURE 2.7.1. Toy example illustrating the behaviour of the greedy algorithm.
On the left, an instance of MaxPD. Branch lengths are indicated by the numbers labelling the branches. On the right, a table with the leaves that are selected at the various stages of the greedy algorithm in the rooted case (top row), in the unrooted case (bottom row), with an empty initial set (left column), with {B,C} as initial set. Each element of the table shows the sequence of taxa that would be selected by the greedy steps in each of the four scenarios. Highlighted in gray is the solution for $k = 3$ and $S_0 = \emptyset$ (the same both in the rooted and unrooted case).



	$S_0 = \varnothing$	$S_0 = \{B,C\}$
rPD	F, C, A, B, E, D	F, A, E, D
uPD	{A,F}, C, B, E, D	F, A, E, D

2.8. Examples

Figure 2.7.1 shows a simple example of the behaviour of the greedy algorithm.

In the rooted case with an empty initial set, the greedy algorithm starts by selecting F, as this is the leaf with the largest rPD (5.5). Then C is taken, as this is

the leaf that adds the most rPD (4.0). Then the algorithm takes A (+3.5), followed by B (+1.0) and E (+1.0) — both orders are possible since these two taxa add the same rPD — and finally D (+0.5). The leaf subsets thus constructed ({F}, {F,C}, {F,C,A}, ...) all have maximum rPD among the subsets of their size, a property which not only holds in this particular example but in general, as will be formally proven in the next section.

In the unrooted case with an empty initial set, the greedy algorithm starts by selecting A and F, as this is the pair of leaves with the largest uPD (9.0). Then C is selected (+4.0), followed by B (+1.0), E (+1.0) and D (+0.5), in this order. Again, the constructed subsets ({A,F}, {A,F,C}, {A,F,C,B}, ...) have maximum uPD among the leaf subsets of their size.

As an example of the case where we have a non-empty initial set S_0, if $S_0 =$ {B,C} the algorithm — both in the rooted and unrooted case — then selects F (+4.5), followed by A (+1.5), E (+1.0) and D (+0.5). Again, the subsets thus constructed ({B,C,F}, {B,C,F,A}, ...) have maximum PD among the extensions of {B,C} of their size.

For an example with some practical relevance, consider figure 2.8.1. Here the greedy algorithm starts by taking Mra and Dma (in this order in the rooted case, together in the unrooted case). Then it takes Led, Iin, Pfu el, Hgr gr, Cme, Ala, Vva ru and Atr. At this point the desired number of taxa ($k = 10$) has been collected and the algorithm stops. The 10 resulting taxa have maximum PD among all possible subsets of 10 taxa.

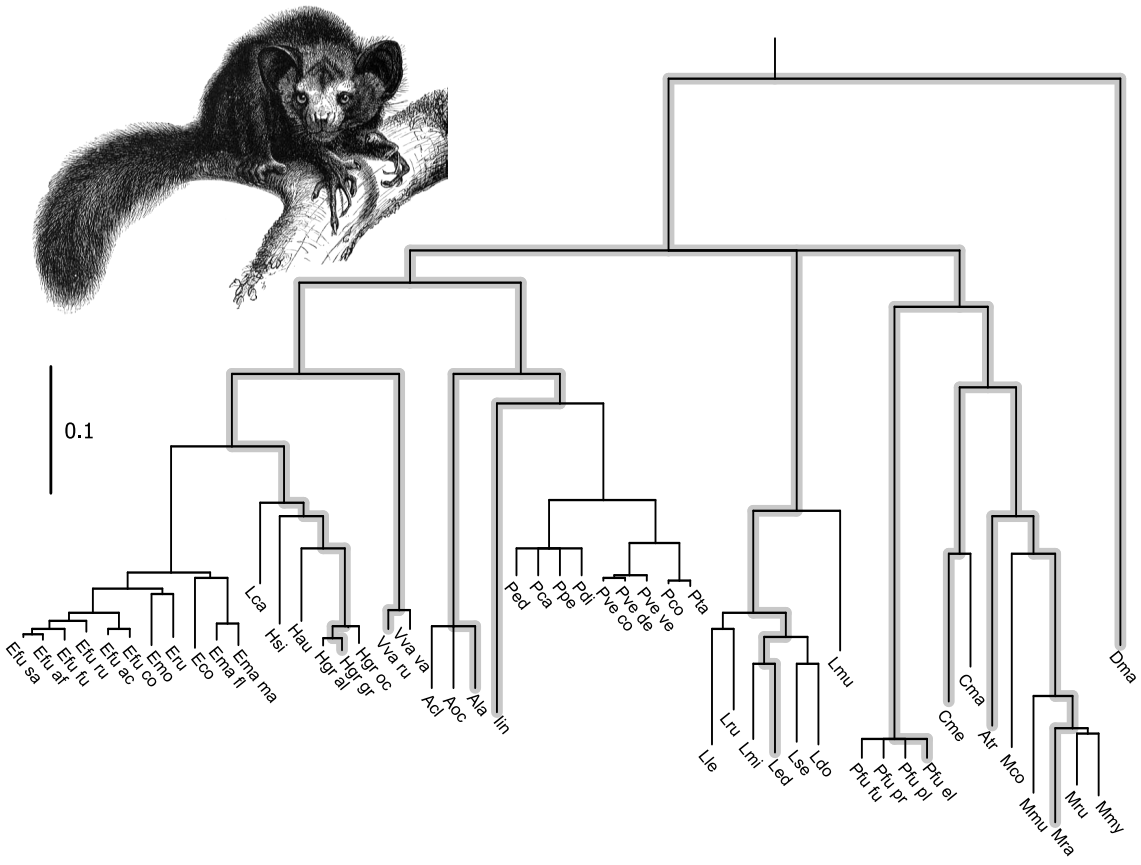
2.9. The algorithm's correctness

In this section, I will prove that the greedy algorithm described above is correct for MaxPD, that is, it produces optimal solutions to this problem. This proof appears, in a slightly different form and for the unrooted case only, in a paper I published with Nick Goldman in 2005 [**PG05**]. At more or less the same time, Mike Steel independently proved, again in a very similar way, the correctness of the greedy algorithm for the unrooted case with $S_0 = \emptyset$ [**Ste05**].

The idea of the proof is the following. I first prove (Theorem 2.9.2) that applying a greedy step to further extend an already optimally extended set of taxa always results in another optimally extended set of taxa. Since the first step of the greedy algorithm necessarily results in an optimally extended set, subsequent steps will construct only other optimally extended sets (Corollary 2.9.3). The greedy algorithm can therefore be used to construct optimal extensions of any desired size. The hard part lies in Theorem 2.9.2, which first needs a graph-theoretic property of tree restrictions (Lemma 2.9.1) in order to be proved.

In all the propositions that follow, $\mathcal{T}$ is a tree with non-negative lengths associated with its branches and $V(\mathcal{T})$ is the set of nodes of $\mathcal{T}$. Given a subtree $\mathcal{T}'$ and a node x of $\mathcal{T}$, the *path from $\mathcal{T}'$ to x* is the sequence with the fewest number of

FIGURE 2.8.1. Application of the greedy algorithm to the conservation of Madagascar lemurs.
A tentative phylogenetic tree of 52 lemurs (species and subspecies) was drawn on the basis of some recent publications (e.g., **[Yod97, YRG+00, PME+01, PFM02, RSZ04, AFR+06]**). The correspondence between taxa and abbreviations is reported in the Appendix. Branch lengths are drawn in scale. The solution of (rooted and unrooted) MAXPD on this tree with $k = 10$ consists of taking the taxa at the leaves of the subtree highlighted in gray.

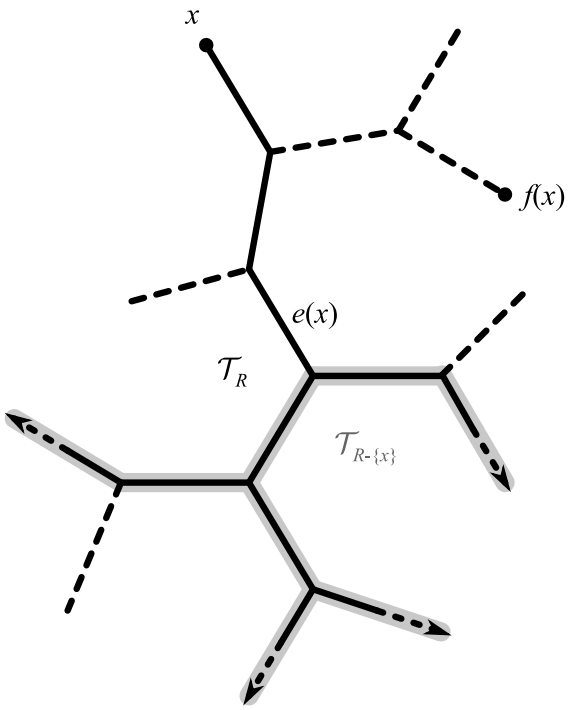


branches in $\mathcal{T}$ such that (i) consecutive branches are adjacent, (ii) the first branch has one of its ends in $\mathcal{T}'$, (iii) one of the ends of the last branch is x. The *length* of a path is the sum of the lengths of its branches.

LEMMA 2.9.1. *Let S and R be subsets of $V(\mathcal{T})$ and let $2 \leq |S| < |R|$. Then there exists $x \in R - S$ such that the path from $\mathcal{T}_S$ to x contains the path from $\mathcal{T}_{R-\{x\}}$ to x.*

PROOF. Suppose the contrary. That is, suppose that for every $x \in R - S$ the path from $\mathcal{T}_S$ to x does not contain the path from $\mathcal{T}_{R-\{x\}}$ to x. Then, for every $x \in R - S$, the path from $\mathcal{T}_{R-\{x\}}$ to x cannot be empty — because otherwise it would be contained in the path from $\mathcal{T}_S$ to x — so let us call $e(x)$ the first branch in this path, the one attached to $\mathcal{T}_{R-\{x\}}$ (see figure 2.9.1). The central observation

FIGURE 2.9.1. Illustration of the reasoning in Lemma 2.9.1.
A relevant portion of $\mathcal{T}_R$ is represented in solid black, whose part in $\mathcal{T}_{R-\{x\}}$ is highlighted in gray. That is, the path from $\mathcal{T}_{R-\{x\}}$ to x is composed of the non-highlighted solid branches. The dashed branches are from $\mathcal{T}$. If all the elements of S were on the same side of $e(x)$ as $\mathcal{T}_{R-\{x\}}$, then clearly the path from $\mathcal{T}_S$ to x would contain the one from $\mathcal{T}_{R-\{x\}}$ to x. Therefore some node $f(x) \in S$ must be on the same side as x and it is also easy to see that $f(x) \notin R$.



here is that $e(x)$ splits $\mathcal{T}$ in two parts and at least one node $f(x) \in S - R$ must lie on the same side of $e(x)$ as x — otherwise it is easy to see that the path from $\mathcal{T}_S$ to x would contain the path from $\mathcal{T}_{R-\{x\}}$ to x.

Note that, since $|R| \geq 3$, all the paths from $\mathcal{T}_{R-\{x\}}$ to some $x \in R - S$ must be disjoint. Therefore the $f(x)$ nodes are all distinct. Since every $x \in R-S$ corresponds to a different $f(x) \in S - R$, we have $|S - R| \geq |R - S|$. But this is equivalent to $|S| \geq |R|$, which contradicts the lemma's assumptions. □

I now prove that the application of a greedy step to a uPD-optimal extension (S) of an initial set (S_0) necessarily results in another uPD-optimal extension (S^+). Informally, I show that however any extension (R) with the same size as S^+ is constructed, a set that has at least the same uPD as R can be obtained from S by adding one taxon in R to S. Therefore the greedy step, which can add any taxon to S — not only those in R — will necessarily lead to a total uPD in S^+ that is at least as great as that in R. S^+ therefore has maximum uPD among all its equally sized extensions of the initial set.

THEOREM 2.9.2. *Let $S_0 \subseteq S \subseteq V(\mathcal{T})$, where S is an h-extension of S_0 with maximum* uPD *and $|S| \geq 2$. Then applying an unrooted greedy step to S results in an $(h+1)$-extension of S_0 with maximum* uPD.

PROOF. Let R be any $(h+1)$-extension of S_0. By Lemma 2.9.1, there exists at least one $x \in R - S$ such that the path from $\mathcal{T}_S$ to x contains the path from $\mathcal{T}_{R-\{x\}}$ to x (see figure 2.9.2). Note that $x \notin S_0$. Therefore,

$$\text{uPD}(R - \{x\}) \leq \text{uPD}(S) \tag{2.9.1}$$

as $R-\{x\}$ and S are both h-extensions of S_0 and S has maximum uPD among such sets. Also,

$$\text{length of the path from } \mathcal{T}_{R-\{x\}} \text{ to } x \leq \text{length of the path from } \mathcal{T}_S \text{ to } x \tag{2.9.2}$$

as the second path contains the first. Thus,

$$\text{uPD}(R) \leq \text{uPD}(S \cup \{x\})$$

by combining (2.9.1) and (2.9.2). Now denote by S^+ a set obtained by applying an unrooted greedy step to S. Then,

$$\text{uPD}(S \cup \{x\}) \leq \text{uPD}(S^+)$$

and therefore

$$\text{uPD}(R) \leq \text{uPD}(S^+).$$

Since the last inequality holds for any $(h+1)$-extension R of S_0, S^+ is an $(h+1)$-extension of S_0 with maximum uPD. □

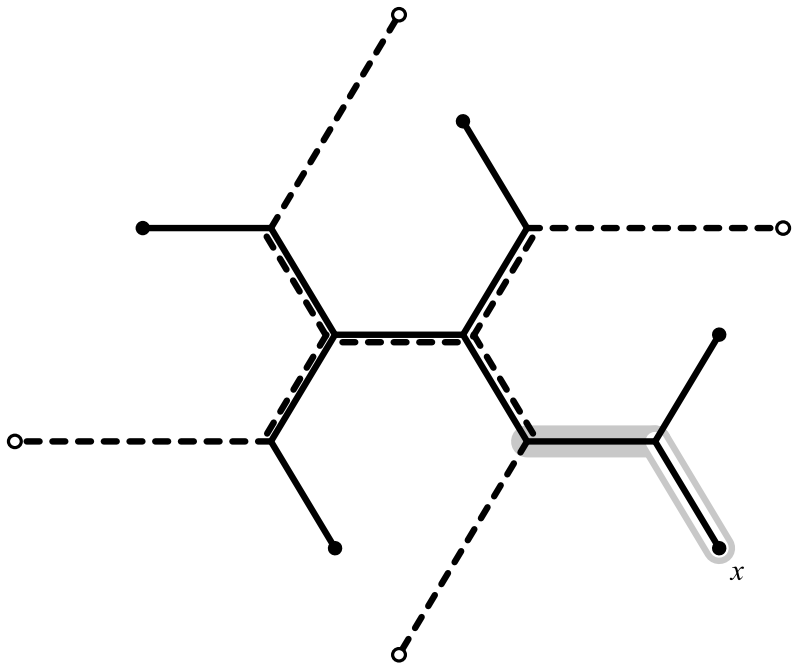
Note the assumption that $|S| \geq 2$ in Theorem 2.9.2. This ensures that either S_0 is nonempty or $h > 1$. In fact, if we have both $S_0 = \emptyset$ and $h = 1$, the theorem is not true: in this case, any S containing a single leaf has maximum $\text{uPD}(S) = 0$, but applying a greedy step to S does not necessarily result in two leaves with maximum uPD.

COROLLARY 2.9.3. *Let S_0 be such that $\emptyset \neq S_0 \subseteq V(\mathcal{T})$. Applying k unrooted greedy steps to S_0 results in a k-extension of S_0 with maximum* uPD.

PROOF. By induction on k. For $k = 0$ or 1 the proposition is either trivial or follows trivially from the definition of greedy step. For $k \geq 2$, by the inductive hypothesis, applying $k-1$ unrooted greedy steps to S_0 results in a set S which is a $(k-1)$-extension of S_0 with maximum uPD. Note that, because $S_0 \neq \emptyset$ and $|S - S_0| = k - 1 \geq 1$, we have $|S| \geq 2$. Therefore, by Theorem 2.9.2, applying one more greedy step to S results in a k-extension of S_0 with maximum uPD. □

COROLLARY 2.9.4. *Let S_0 be a set of $h \geq 2$ leaves of $\mathcal{T}$ with maximum* uPD. *Applying k unrooted greedy steps to S_0 results in a set of $h+k$ leaves with maximum* uPD.

FIGURE 2.9.2. Illustration of the reasoning in Theorem 2.9.2.
A possible scenario for $S_0 = \emptyset$ and $h = 4$. Empty and filled circles represent elements of S and R, respectively. Dashed and solid lines represent paths in $\mathcal{T}_S$ and $\mathcal{T}_R$, respectively. As predicted by Lemma 2.9.1, there is a $x \in R$ such that the path from $\mathcal{T}_S$ to x (grey) contains the one from $\mathcal{T}_{R-\{x\}}$ (highlighted in white on grey background). It should be visually intuitive that $\text{uPD}(R - \{x\}) \leq \text{uPD}(S)$ implies that $\text{uPD}(R) \leq \text{uPD}(S \cup \{x\})$.



PROOF. By induction on k. For $k = 0$, the resulting set is S_0 itself which has maximum uPD among the sets of h leaves. For $k \geq 1$, by the inductive hypothesis applying $k-1$ unrooted greedy steps to S_0 results in a set S of $h+k-1$ leaves with maximum uPD. In other words, S is a $(h+k-1)$-extension of $\emptyset$ with maximum uPD. Note that $|S| = h + k - 1 \geq 2$. Therefore, by Theorem 2.9.2, applying one more greedy step to S results in an $(h+k)$-extension of $\emptyset$ with maximum uPD, that is, a set of $h+k$ leaves with maximum uPD. □

The propositions above are concerned uniquely with the unrooted case. The following results, in which I assume that $\mathcal{T}$ is a tree rooted in a node ρ, apply to the rooted case.

LEMMA 2.9.5. *Let $S_0 \subseteq S \subseteq V(\mathcal{T})$. S is an h-extension of S_0 with maximum* rPD *if and only if $S \cup \{\rho\}$ is an h-extension of $S_0 \cup \{\rho\}$ with maximum* uPD.

PROOF. By equation (2.5.1), the following two propositions are equivalent:

(i) S is an h-extension of S_0 and for every h-extension S' of S_0, $\text{rPD}(S') \leq \text{rPD}(S)$;

(ii) $S \cup \{\rho\}$ is a h-extension of $S_0 \cup \{\rho\}$ and for every h-extension $S' \cup \{\rho\}$ of $S_0 \cup \{\rho\}$, $\text{uPD}(S' \cup \{\rho\}) \leq \text{uPD}(S \cup \{\rho\})$. □

COROLLARY 2.9.6. *Let S_0 be a set of leaves of $\mathcal{T}$ (possibly empty). Applying k rooted greedy steps to S_0 results in a k-extension of S_0 with maximum* rPD.

PROOF. A simple consequence of Lemma 2.9.5 (for $h = 1$) is that applying one rooted greedy step to a set of leaves S_0 results in selecting the same leaf (or leaves in case of ties) as applying one unrooted greedy step to $S_0 \cup \{\rho\}$. Therefore, if applying k rooted greedy steps to S_0 results in a set S, then applying k unrooted greedy steps to $S_0 \cup \{\rho\}$ can result in $S \cup \{\rho\}$. Therefore, by Corollary 2.9.3, $S \cup \{\rho\}$ is a k-extension of $S_0 \cup \{\rho\}$ with maximum uPD. But by Lemma 2.9.5 this is the same as saying that S is a k-extension of S_0 with maximum rPD, which is what we set out to prove. □

The three corollaries above prove the correctness of the greedy algorithm for MAXPD in these three scenarios: for the unrooted case with $S_0 \neq \emptyset$ (Corollary 2.9.3), for the unrooted case with $S_0 = \emptyset$ (Corollary 2.9.4), and for the rooted case with any S_0 (Corollary 2.9.6).

2.10. The moral of the story: being greedy works

Proving that a greedy algorithm is sufficient for some task is always an interesting result. In fact, because these algorithms are usually efficient and easy to implement, general theories on their applicability have been developed [**KLS91**].

In the case of our optimisation problem, the result I have proven is not only of algorithmic interest but potentially has consequences for real-life strategies for genome sequencing and biodiversity conservation. Imagine, for example, that given our current progress in the sequencing of mammalian genomes, a number of sequencing centres, each having the resources to sequence one more mammal, are in the process of deciding about the next genome they are going to take on. How should they behave in order to ensure that the resulting sequenced species have a large collective PD? Is some sort of cooperation necessary?

Clearly, openness regarding each group's decision is necessary, since if one decides to sequence, say, the cat, the others must avoid sequencing this or any other closely related feline. Note that, had PD been a main guiding criterion, the choice to sequence the rat [**GWM$^+$04**] soon after the mouse [**WLTB$^+$02**] would not have been made. But apart from communicating their intentions, is real cooperation among the groups necessary? Applying the result described above, it is apparent that the answer is *no*. If every group selfishly ("greedily") decides to sequence the genome that at the moment of choice is the most "appealing" — i.e., adds the most PD to the set of species already sequenced or previously chosen by the other groups — then the best possible outcome is guaranteed. Another practical consequence of the optimality of the greedy algorithm is that no planning is needed, either.

Specifically, no consideration of next (or any future) year's resources is necessary when determining priorities for this year's expenditure.

Although in this discussion I have focused on genomics, exactly the same considerations are valid for conservation biology, where often geographically separated conservation groups have to make decisions regarding taxa from the same phylogenetic scope.

A possible criticism of these considerations is that PD is rarely the only guiding criterion in choosing species — both for sequencing and conservation. A way to deal with this criticism [**Ste05**] is to imagine that the real objective function for these choices is some linear combination of PD and other taxon-specific scores — quantifying for example medical relevance, amenability to laboratory experimentation or any other application-specific factor (see [**PG05**] for a list of important factors in genomics). It is easy to see that such an objective function is equivalent to PD on a transformed version of the original phylogenetic tree, one where the terminal branches' lengths are extended to reflect the taxon-specific scores. Therefore, if the different groups choose taxa greedily with respect to this objective function — i.e., always selecting the taxon that increases it the most — it is still guaranteed that the resulting set is optimal with respect to it.

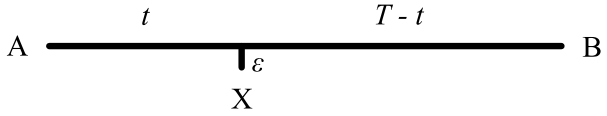
In our 2005 paper ([**PG05**] p. 674), Nick Goldman and I concluded:

> If genome or conservation scientists follow a seemingly short-term strategy — involving neither planning nor cooperation in the choice of future genomes for sequencing or species for conservation — then, provided they are open about their choices, they are guaranteed the best long-term strategy.

This was a deliberately provocative statement, and our conclusion is subject to assumptions that are often violated in practice: as I will show in the next three chapters, adding other factors, such as the different costs of selecting different taxa, makes the greedy algorithm non-optimal for the resulting problem. However, subsequent research (see sec. 2.12) showed that greedy algorithms applied to PD maximisation problems are still surprisingly successful: even when they are not guaranteed optimality, sometimes they can be shown to return approximate solutions [**BS08**].

The interest in greedy strategies is also due to their usefulness in those scenarios where planning is difficult or impossible. Available resources may fluctuate, funding may be interrupted or extended, so even though at a first stage it may seem that the selection of k taxa is feasible, it may turn out later that $k' \neq k$ taxa can actually be taken. If a greedy strategy is indeed effective for the problem at hand, one can compile a ranked list of taxa and select them in the specified order. Independently from the amount of resources that turn out to be available, wherever one stops selecting in this list, the resulting subset of k' taxa will be a good one.

FIGURE 2.11.1. 3-taxon tree with X moving arbitrarily along AB. Branch lengths are t, $T-t$ and ε. The behaviour of GD($\{A, B, X\}$) as a function of t is studied in the main text (T and ε are assumed to be constants).



2.11. Beyond PD: Genetic Diversity and Feature Diversity

As I mentioned before, especially in section 2.3, I believe that improvements over PD are possible. This is a direction of research which has not been explored very much. In this section, I wish to provide a glimpse of some possible future developments.

One measure of diversity on a phylogeny alternative to PD is *genetic diversity* (GD) [**Cro92, Cro97**], defined as "the probability of there being two or more character states" in the set of taxa under consideration [**Cro92**]. Clearly some model of character evolution on the given phylogenetic tree must be assumed to make this definition precise. Originally, it was implicitly assumed that no reversal to a previously held state was possible: calling p_e the probability of a character change along a branch e, genetic diversity was simply given by

$$\mathrm{GD}(S) = 1 - \prod_{e \in \mathcal{T}_S} (1 - p_e). \tag{2.11.1}$$

This version of GD turns out to be strictly related to uPD: assuming an infinite-states Markov model of character evolution (with characters changing at rate κ), p_e and the corresponding branch length t_e are simply related through $p_e = 1 - e^{-\kappa t_e}$, and so

$$\mathrm{GD}(S) = 1 - \prod_{e \in \mathcal{T}_S} \exp(-\kappa t_e) =$$

$$= 1 - \exp(-\kappa \sum_{e \in \mathcal{T}_S} t_e) = 1 - \exp(-\kappa \cdot \mathrm{uPD}(S)).$$

Therefore, GD grows monotonically with uPD and maximising GD is equivalent to maximising uPD. This result was suggested without proof by Crozier ([**Cro97**] p. 252) and the simple derivation above appears elsewhere [**PG07**].

It is important, however, to realise that the equivalence above does not hold for other models of character evolution. For example, if we assume the Jukes-Cantor model (JC) [**JC69**], equation (2.11.1) is not true anymore, as the probability that no change is observed in S must also account for the possibility that some change occurs in the interior of the tree, but is subsequently reversed. I will not attempt to deal with a general expression for GD under JC, but instead will show its particular behaviour on a simple 3-taxon tree, as this is particularly instructive.

Consider the tree in figure 2.11.1, where taxon X is attached with a branch of constant length ε onto the lineage (of constant length T) between A and B. It is interesting to consider the behaviour of GD as we move X along this lineage, i.e., as a function of branch length t in figure 2.11.1. Let 1, 2 and 3 correspond to the three branches in this tree and let $p_i = \frac{3}{4}(1 - e^{-\kappa t_i})$ be the associated probabilities of change under the JC model. Then the probability of observing more than one character state is one minus the probability of having no changes at all or of having three identical changes on each of the three branches:

$$\mathrm{GD}(\{\mathrm{A},\mathrm{B},\mathrm{X}\}) = 1 - (1-p_1)(1-p_2)(1-p_3) - \frac{p_1 p_2 p_3}{9},$$

where the $1/9$ factor is because we need to ensure that the three changes all end up giving the same nucleotide. Some more algebra shows that

$$\begin{aligned}\mathrm{GD} &= 1 - \frac{1}{16}\left(1 + 3(e^{-\kappa(t_1+t_2)} + e^{-\kappa(t_1+t_3)} + e^{-\kappa(t_2+t_3)}) + 6e^{-\kappa(t_1+t_2+t_3)}\right) = \\ &= 1 - \frac{1}{16}\left(1 + 3(e^{-\kappa T} + e^{-\kappa(t+\varepsilon)} + e^{-\kappa(T-t+\varepsilon)}) + 6e^{-\kappa(T+\varepsilon)}\right) = \\ &= 1 - \frac{1}{16}\left(1 + 3e^{-\kappa T} + 3e^{-\kappa\varepsilon}(e^{-\kappa t} + e^{-\kappa(T-t)}) + 6e^{-\kappa(T+\varepsilon)}\right) \qquad (2.11.2)\end{aligned}$$

It is easy to see that this quantity is maximised for the values of t that minimise $e^{-\kappa t} + e^{-\kappa(T-t)}$, i.e., for $t = T/2$.

This is an interesting result, first of all because it shows that, under JC, maximising GD is not the same as maximising PD: as we move X along AB, the PD is constant but GD has a maximum as X reaches the mid-point of this lineage. Second, having an optimum for $t = T/2$ precisely corresponds to our intuition of how an ideal measure of diversity should behave (see figure 2.3.1 and the discussion at page 24).

Although GD has been proposed in the conservation biology literature [**Cro92**], it also has some relevance for choosing sequences for comparative analyses: recall the discussion on using counter-features to disprove that a sequence belongs to a certain functional class (page 23). An extreme example of this is when looking for invariant sequences (or even sites), sequences that do not change, most arguably because of selective — thus functional — constraints. In this case, any variation of the sequence acts as a counter-feature. The discriminative power of a set of genomes, i.e., the probability of observing a useful counter-feature, coincides here with the probability of observing different versions of a variable sequence, in other words with the genomes' GD.

Back with conservation biology, one of the aims of PD is to predict the feature diversity of the species being conserved [**Fai92**], that is, the number of different (morphological, physiological, behavioural) characteristics appearing in this set. But in this case, why not calculate the feature diversity itself? We may define *feature diversity* (FD) as the expected number of character states that will be observed in the set of taxa under consideration. Contrast this with GD, which was defined as the

probability that this number be greater than one. Like GD, FD depends on a model, but perhaps in this case the natural choice is an infinite-states model, as the features we are interested in (such as morphological ones) are better thought as coming from an infinite, continuous set.

It is again instructive to consider the 3-taxon tree. In this case, the expected number of character states is given by

$$\begin{aligned}\mathrm{FD} &= 1 \cdot (1-p_1)(1-p_2)(1-p_3) + \\ &+ 2 \cdot [p_1(1-p_2)(1-p_3) + (1-p_1)p_2(1-p_3) + (1-p_1)(1-p_2)p_3] \\ &+ 3 \cdot [p_1p_2(1-p_3) + p_1(1-p_2)p_3 + (1-p_1)p_2p_3 + p_1p_2p_3] = \\ &= 1 + p_1 + p_2 + p_3 - p_1p_2p_3.\end{aligned}$$

Then, assuming $p_i = 1 - e^{-\kappa t_i}$, we get

$$\mathrm{FD} = 3 + e^{-\kappa(t_1+t_2+t_3)} - e^{-\kappa(t_1+t_2)} - e^{-\kappa(t_1+t_3)} - e^{-\kappa(t_2+t_3)}.$$

For the case $t_1 = t$, $t_2 = T - t$, $t_3 = \varepsilon$, where T and ε are constant (figure 2.11.1), it is easy to see that this function has a very similar behaviour to that for GD derived in (2.11.2). In particular, the optimal value for t is again $T/2$.

Because of their definitions — more directly linked to their intended applications — and because of their behaviour — more consistent with our intuitive expectations — GD and FD may turn out to be better than PD as measures of diversity on a phylogeny. They probably constitute the first step in this interesting avenue of research.

2.12. Related work

An alternative measure of the informativeness for comparative analyses of a set of sequences related by a tree has been provided by Lee Newberg and colleagues [**NL04, New07**]. Their aim is to measure the expected utility of the sequences for constructing a nucleotide equilibrium probability distribution for each multiply aligned DNA sequence position, in other words for constructing accurate sequence profiles. A similar approach has been developed by Goldman, Massingham and collaborators [**Gol98, MG00, Geu07**], whose focus, however, is on phylogenetic inference rather than comparative genomics.

A more general analysis of diversity, not only limited to elements of a tree, was prompted by Weitzman's early work [**Wei92**] and is receiving increasing attention in economics [**NP02, NP03, NP04**].

The correctness proofs for the greedy algorithm for MaxPD [**Ste05, PG05**] have spurred further research on this algorithm. Efficient implementations were devised first by Minh et al. [**MKvH06**], who describe an $O(n \log k)$ time algorithm and provide a computer program implementing it (PDA, available online), and then by Spillner et al. [**SNM08**], who reduce this to $O(n)$ (and it is easy to see that this is the best possible asymptotic worst-case complexity). In the rooted case, the

crucial step in the latter algorithm is to determine the increase in rPD that each leaf would contribute if the greedy algorithm were run until all leaves are taken. These quantities can be calculated with two traversals (first bottom-up and then top-down) in $O(n)$ time. Then the k leaves with the k largest increases are selected in $O(n)$ time ([**CLRS01**], sec. 9.3).

The performance of greedy algorithms on other problems related to MaxPD has also been investigated: Hartmann and Steel [**HS06**] have shown their correctness for a few special cases of the Noah's Ark problem [**Wei98**] (a generalisation of MaxPD, focused on conservation biology, taking into account the different extinction risks of different taxa — I will deal with this problem in Chapters 4 and 5). In a similar vein, Moulton et al. [**MSS07**] defined a number of generalisations of MaxPD with relevance to conservation (e.g., taking into account ecological dependencies between species) and showed (rather restrictive) sufficient conditions for the correctness of greedy algorithms. More recently, Bordewich and Semple [**BS08**] showed that a polynomial-time greedy-like algorithm applied to the "nature reserve selection" problem — where one would like to select geographical regions so as to maximise the PD of the species appearing in them — is a $(1 - 1/e)$-approximation algorithm for this problem, i.e., it is guaranteed to produce a solution with at least $1 - 1/e \simeq 63\%$ the PD of the optimal solution. (They also proved that getting a better approximation algorithm for this problem is impossible, unless P=NP.)

Another related direction of research has been to apply MaxPD to split networks, a generalisation of phylogenetic trees allowing simultaneous representation of several alternative phylogenetic histories (which is useful both when we are uncertain about the true tree and when evolution is inherently non-treelike — see [**HB06**] for a review). Minh et al. [**MKvH08**] described a dynamic programming algorithm for MaxPD on circular split systems, a particular type of split network. This result has been recently extended by Spillner et al. [**SNM08**], who show a more efficient algorithm for this problem and propose other algorithms for other special classes of split networks, while also proving the NP-hardness of the general problem. A reason to be interested in MaxPD on split networks is that this is equivalent to the problem of maximising the average PD across a collection of alternative trees. The hardness result therefore extends to this problem when the number of trees is unbounded. Interestingly, if the collection is limited to three trees, the problem is still NP-hard [**SNM08**], but not if the collection has just two trees [**BSS07**].

CHAPTER 3

Including costs: the budgeted MaxPD problem and dynamic programming solutions

3.1. "All animals are equal, but some animals are more equal than others"

– George Orwell [**Orw45**].

In the previous chapter, I have shown how the selection of taxa should take into account their different degree of phylogenetic distinctness, when this is the guiding criterion for the selection. In this and the next chapter, I will deal with additional factors that discriminate among taxa: their different costs (this chapter) and their different extinction risks (which is something obviously relevant to conservation only — next chapter).

For all the applications I discussed in the previous chapter, the selection of taxa is somehow constrained by the limited availability of some underlying resource, often (but not always) money. This is why MaxPD includes the specification of a parameter k indicating the number of taxa that it is feasible to take. However it should be noted that this parameter can only be determined if the taxa require (roughly) the same amount of resources. In reality this is usually not true: in bioconservation, for example, where we may be designing a nature reserve of fixed area, different species will typically require different amounts of land. Similarly, in the case of sequencing, different genomes have different sizes and so will have different costs, not only in terms of money, but also in terms of time and instruments required for sequencing. Potentially, a choice will have to be made between selecting few "expensive" taxa or many "cheap" ones.

Formalising this scenario is easy: we just need to quantify the "cost" of each taxon (whatever the limited resource, e.g., money, time, labour, machinery, land, etc.) and the total amount of available resources, which we may call the "budget". We then aim to find the subset of taxa with maximum PD among all those with total cost at most equal to the budget (see sec. 3.2).

After showing that this is a computationally hard problem (sec. 3.3), this chapter will provide dynamic programming algorithms solving this problem in the rooted case (sec. 3.4 and 3.5) and in the unrooted case (sec. 3.6). Again, examples of their application will be presented (sec. 3.7) and the related literature discussed (sec. 3.8).

3.2. The budgeted MaxPD problem

Along the lines described at page 38, a possible approach to limit the amount of resources consumed when picking taxa on the basis of their PD would be to modify the input tree by shortening each terminal branch by an amount related to the cost of its taxon, so that the selection of costly taxa would be discouraged. This is not very satisfying: for example it does not seem possible to guarantee that the selected taxa will have maximum PD among all the other choices of taxa with the same total cost. The natural way to deal with costs is a different one:

BMaxPD:

Input: A tree $\mathcal{T}$ with non-negative lengths t_e associated with each branch e, and non-negative costs c_s associated with each leaf s. A non-negative budget B.

Output: A subset S of the leaves of $\mathcal{T}$ that

$$\text{maximises } \mathrm{PD}(S),$$
$$\text{subject to } \sum_{s \in S} c_s \leq B.$$

Note, once again, that the two different definitions of PD correspond to two different versions, rooted and unrooted, of this problem. Unlike in the previous chapter, my approaches to solve these two cases are not straightforward variations of one another. Therefore I will first deal with the rooted case (sec. 3.4 and 3.5) and then show how the resulting algorithms can be adapted to the unrooted case (sec. 3.6).

The following terminology will be used throughout this chapter:

Definition 3.2.1. A *candidate solution* for a tree $\mathcal{T}$ (with branch lengths and leaf costs) is simply a subset of its leaves. A *feasible solution* for $\mathcal{T}$ and budget B is a candidate solution S for $\mathcal{T}$ with total cost not exceeding the budget, i.e. such that $\sum_{s \in S} c_s \leq B$. An *optimal solution*, or more simply a *solution*, for $\mathcal{T}$ and B is a feasible solution with maximum PD.

In the rest of this chapter, I will assume that all costs and the budget are integers. This is a necessary assumption for the algorithms I will present in the next sections, and it is not unrealistic, since real-life resources are inherently discrete (e.g. currencies) and costs and budgets will often be known not with great precision, but only approximately.

Finally note that the formulation of BMaxPD can also be used to express the scenario whereby a number of taxa have already been taken: a maximally diverse extension of an initial set of leaves S_0 can be obtained by solving BMaxPD with all the costs for the leaves in S_0 set to 0.

3.3. Computational complexity

BMaxPD is a NP-hard problem, as the knapsack problem, another well-known NP-hard problem [**CLRS01**], is simply its special case for star-shaped trees [**HS06**]. In order to make this more explicit, I restate the knapsack problem and show its connection to BMaxPD.

Knapsack:

Input: A set of n objects with values $v_1, v_2, \ldots, v_n$ and costs $c_1, c_2, \ldots, c_n$ and a budget B.

Output: A subset $S \subseteq \{1, 2, \ldots, n\}$ of the given objects with maximum total value $\sum_{i \in S} v_i$ among those subject to $\sum_{i \in S} c_i \leq B$.

The input for Knapsack can be easily transformed into an input for BMaxPD: just construct a star tree with the given objects as its leaves and where each branch has a length equal to the value of the object it leads to (and leave the costs of the leaves and the budget unaltered). Then it is easy to see that solving BMaxPD on this input would provide a subset of leaves which is also a solution for Knapsack. This shows the polynomial-time reducibility of Knapsack to BMaxPD and therefore BMaxPD is NP-hard.

In the following three sections, I will show a number of algorithms for BMaxPD that run in polynomial time in the size of the tree and in the budget B. Technically, these algorithms are examples of *pseudo-polynomial* time algorithms, as one of the factors determining their running time, B, may itself grow exponentially in the size of the input (see [**GJ79**] for further discussion on pseudo-polynomial algorithms).

In practice, the fact that the computational complexity depends on B may indeed seem problematic — especially for the algorithm in the next section whose time complexity is quadratic in B. For example, if the underlying resource is money, the budget may be a very large number (of the order of the millions, if measured in common currencies), which would make the proposed algorithms inefficient. To avoid this, all costs should be preprocessed and expressed as multiples of a large unit (such as their greatest common divisor — efficiently found with a generalisation of Euclid's algorithm ([**CLRS01**], sec. 31.2) — or any large unit which leaves the costs expressed to an acceptable degree of accuracy). For example, in the case of budgets of the order of millions of euros, we may be content to express everything in multiples of €10,000 or €100,000. This is probably precise enough to satisfy the accountants, makes B of the order of the hundreds or even less and permits the algorithms to run very quickly.

3.4. An $O(B^2n)$-time dynamic programming algorithm

It is not difficult to see that rPD has a recursive structure. As a consequence, it is possible to devise dynamic programming algorithms that solve BMaxPD in

the rooted case. This section and the next one describe two such algorithms. The contents of this section are largely based on a paper published on *Systematic Biology* in 2007 [**PG07**].

The key observation that allows the use of a dynamic programming algorithm is that optimal solutions to BMAXPD can be simply decomposed into optimal solutions to its subproblems — technically, BMAXPD is said to have *optimal substructure* ([**CLRS01**], sec. 15.3). As a consequence, optimal solutions to BMAXPD can be constructed by first tackling and solving its subproblems and then combining the optimal solutions thus found.

In order to see what this means in practice, I introduce some concepts that allow us to apply BMAXPD to smaller portions of the input tree. I define a *clade* of a tree $\mathcal{T}$ as any subtree of $\mathcal{T}$ consisting of a branch e and everything else below e in $\mathcal{T}$ (note that I imagine $\mathcal{T}$ with its root at the top); a clade should be thought of as rooted at the top of its top branch e. In a bifurcating tree, every clade is composed of a root branch and, possibly, by two other clades, which I call its *subclades*. (For example, in figure 3.4.1, left, $\mathcal{T}$ is a clade of the input tree; $\mathcal{T}$ has itself two subclades, $\mathcal{L}$ and $\mathcal{R}$, whereas $\mathcal{L}$ is composed of one terminal branch and no subclades.)

In the rest of this section, I will denote the input tree by $\mathcal{T}_0$, so as to distinguish it from $\mathcal{T}$, which will be used to denote a generic clade of $\mathcal{T}_0$. I will assume that $\mathcal{T}_0$ is rooted at the top of a root branch (possibly of zero length), so that $\mathcal{T}_0$ is one of its own clades. Also, I will assume that $\mathcal{T}_0$ is a bifurcating tree. This is not a limitation, as any multifurcating tree $\mathcal{T}_1$ can be transformed into an equivalent bifurcating tree $\mathcal{T}_0$ — i.e., one such that $\mathrm{PD}_{\mathcal{T}_0}(S) = \mathrm{PD}_{\mathcal{T}_1}(S)$ for every subset of leaves S — by substituting each multifurcation with a number of bifurcations separated by new internal branches with length 0.

A way to decompose BMAXPD into smaller subproblems is to ask what is the best way to invest any given part of the budget into each clade of $\mathcal{T}_0$. By assuming that costs are integer, we can also assume that the part of the budget used to take any subset of taxa will also be integer. Therefore, we seek optimal solutions to BMAXPD for every combination of (i) a clade $\mathcal{T}$ of the input tree $\mathcal{T}_0$ and (ii) an integer "sub-budget" $b \in \{0, 1, \ldots, B\}$. It turns out that we can do this incrementally, starting from the clades consisting of only a terminal branch and then using the solutions already found to construct solutions for larger clades. Note that one of these subproblems (the one with $\mathcal{T} = \mathcal{T}_0$ and $b = B$) coincides with the global problem we set out to solve.

For ease of description, I will imagine that optimal solutions to these subproblems (or more precisely some sufficient information about them, as I will show later) are stored in a table (the *solutions table*) whose rows correspond to all the different clades and whose columns correspond to all the sub-budgets $0, 1, ..., B$ (see figure 3.4.1). Clearly, position $(\mathcal{T}, b)$ will contain (information about) an optimal solution

for $\mathcal{T}$ and b. I will show that this table can be completed one row at a time, starting from the bottom and going up.

First, the solutions for the clades consisting of only a terminal branch (leading to, say, leaf s) are simply either the empty set $\emptyset$ or $\{s\}$, depending on whether the corresponding taxon is too expensive to be taken with the available sub-budget (i.e. $c_s > b$), or not ($c_s \leq b$), respectively. Therefore the rows of the solutions table corresponding to terminal branches (in figure 3.4.1, the 3rd, 4th, 6th and the last two) can be filled without looking at the content of any other row.

Second, when instead a clade $\mathcal{T}$ contains two subclades ($\mathcal{L}$ and $\mathcal{R}$), an optimal solution S for $\mathcal{T}$ and sub-budget b will simply amount to the union $S = S_\mathcal{L} \cup S_\mathcal{R}$ of two optimal solutions for two other subproblems: $S_\mathcal{L}$ will be optimal for $\mathcal{L}$ and some integer sub-budget $i \leq b$, whereas $S_\mathcal{R}$ will be optimal for $\mathcal{R}$ and $b - i$. For example, in figure 3.4.1, an optimal solution for $\mathcal{T}$ and $b = 4$ is $\{\mathrm{C}, \mathrm{E}\}$, where $\{\mathrm{C}\}$ is optimal for $\mathcal{L}$ and sub-budget 2, and $\{\mathrm{E}\}$ is optimal for $\mathcal{R}$ and 2. This is the optimal substructure property mentioned above and its proof is rather trivial.[1]

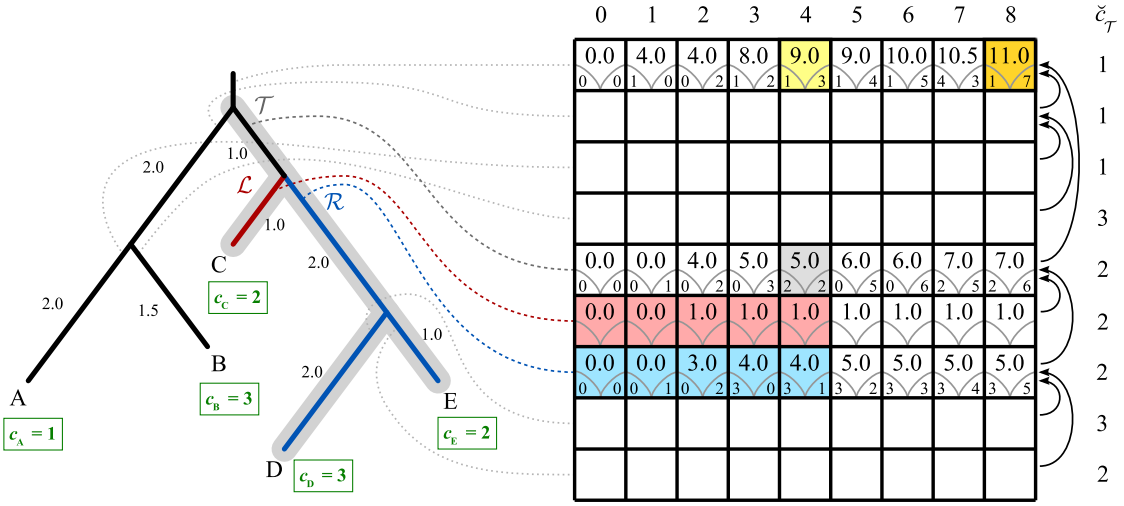
As a consequence of this property, if we have already calculated and stored optimal solutions for $\mathcal{L}$ and $\mathcal{R}$ and for all sub-budgets, it becomes possible to find a solution for $\mathcal{T}$ and any given b: denoting by $S_\mathcal{L}^{(i)}$ the solution stored for $\mathcal{L}$ and i (and similarly for $S_\mathcal{R}^{(j)}$), just compare all the subsets $S_\mathcal{L}^{(0)} \cup S_\mathcal{R}^{(b)}$, $S_\mathcal{L}^{(1)} \cup S_\mathcal{R}^{(b-1)}, \ldots,$ $S_\mathcal{L}^{(b)} \cup S_\mathcal{R}^{(0)}$ and take the one with the largest rPD. This is guaranteed to find an optimal solution for $\mathcal{T}$ and b, because the possibility of decomposing an optimal solution S into $S_\mathcal{L} \cup S_\mathcal{R}$ — where $S_\mathcal{L}$ is optimal for $\mathcal{L}$ and some $i \in \{0, 1, \ldots, b\}$, and $S_\mathcal{R}$ is optimal for $\mathcal{R}$ and $b - i$ — implies that $\mathrm{rPD}(S) = \mathrm{rPD}(S_\mathcal{L} \cup S_\mathcal{R}) =^2 \mathrm{rPD}(S_\mathcal{L}^{(i)} \cup S_\mathcal{R}^{(b-i)})$ and therefore $S_\mathcal{L}^{(i)} \cup S_\mathcal{R}^{(b-i)}$ — which is one of the sets considered above — is also optimal.

Consequently, we can fill the entire solutions table one row at a time: because the content of the row for a clade $\mathcal{T}$ can be completely derived from the content of the rows for its subclades $\mathcal{L}$ and $\mathcal{R}$, we just need to make sure that, whenever we fill the row for a clade, the rows for its subclades have already been filled. This can be achieved by dealing with the clades in a bottom-up order: let the clades be ordered according to, say, a preorder traversal of all the branches in $\mathcal{T}_0$ [**CLRS01**] and let

[1] The reason for this is simple: S is naturally partitioned into $S_\mathcal{L}$ and $S_\mathcal{R}$, containing the taxa in $\mathcal{L}$ and $\mathcal{R}$, respectively. Calling i the total cost of $S_\mathcal{L}$, if either $S_\mathcal{L}$ were not optimal for $\mathcal{L}$ and i or $S_\mathcal{R}$ were not optimal for $\mathcal{R}$ and $b - i$, then we could replace this suboptimal choice of taxa with a better one and therefore improve also S, but this would contradict the fact that S is optimal.

[2] There is a special case in which this equality is not true: imagine that there are multiple optimal solutions for $\mathcal{L}$ and i (the same argument would hold for $\mathcal{R}$ and $b-i$), some of which are empty and some of which are not — which is possible if there are paths of zero length from the root of $\mathcal{L}$ to some of its taxa. Now suppose $S_\mathcal{L} \neq \emptyset$ and $S_\mathcal{R} = \emptyset$, whereas $S_\mathcal{L}^{(i)} = S_\mathcal{R}^{(b-i)} = \emptyset$. In this case we may have $\mathrm{rPD}(S_\mathcal{L} \cup S_\mathcal{R}) > 0$, whereas $\mathrm{rPD}(S_\mathcal{L}^{(i)} \cup S_\mathcal{R}^{(b-i)}) = \mathrm{rPD}(\emptyset \cup \emptyset) = 0$. Therefore $S_\mathcal{L}^{(i)} \cup S_\mathcal{R}^{(b-i)} = \emptyset$ will not be not optimal.
To avoid this we must ensure that, whenever we can derive multiple optimal solutions for a clade $\mathcal{T}$ and a budget b, and some of these are empty while some others are not, a nonempty solution will be stored in $S_\mathcal{T}^{(b)}$.

FIGURE 3.4.1. An instance of BMAXPD (rooted case) and its solution.
On the left, an input tree for BMAXPD. Branch lengths are indicated by the numbers to the side of the branches. Taxon costs are in the boxes next to the leaves (green) and the budget is $B = 8$. The following clades of the input tree are highlighted: $\mathcal{T}$ (broad grey branches), $\mathcal{L}$ (red branches), $\mathcal{R}$ (blue branches). On the right, the corresponding solutions table (the content of some rows is omitted for clarity). The correspondence between clades and rows is indicated by dotted lines (coloured in the cases of $\mathcal{T}$, $\mathcal{L}$ and $\mathcal{R}$). Rows are ordered according to a top-down visit of all clades: 1: ((A,B),(C,(D,E))); 2: (A,B); 3: A; 4: B; 5: (C,(D,E)); 6: C; 7: (D,E); 8: D; 9: E. Columns correspond to sub-budgets $b = 0, \ldots, 8$. Each cell indicates the corresponding value of $\lambda_{\mathcal{T}}(b)$ (top), of $\iota_{\mathcal{T}}(b)$ (bottom left) and of $b - \iota_{\mathcal{T}}(b)$ (bottom right). To the right of the table are the $\check{c}_{\mathcal{T}}$ values. See text for details about these values. The arrows on the right indicate the dependencies amongst the rows. The top-right cell (shaded orange) indicates the optimal rPD (11, achieved by selecting taxa {A, C, D, E}).



the rows in the table reflect this ordering (as illustrated in figure 3.4.1); then, fill the rows from the last to the first. Once the entire table has been filled, the optimal solution we sought (the one for $\mathcal{T}_0$ and B) is available from its top-right corner.

3.4.1. Actual implementation of the algorithm. Although explicitly storing entire subproblem solutions in the solutions table is feasible, this is certainly not efficient. As I will show, instead of storing solutions, it is sufficient to retain their rPD and the way the sub-budget should be partitioned between the two subclades (if there are any). More precisely, let $S_{\mathcal{T}}^{(b)}$ be the solution found for clade $\mathcal{T}$ and sub-budget b; instead of storing $S_{\mathcal{T}}^{(b)}$, we will store two quantities: (a) the maximum rPD achievable in $\mathcal{T}$ with sub-budget b, which I call $\lambda_{\mathcal{T}}(b)$ and is equal to $\mathrm{rPD}_{\mathcal{T}}(S_{\mathcal{T}}^{(b)})$; and (b) (only when $\mathcal{T}$ has two subclades) the sub-budget that solution $S_{\mathcal{T}}^{(b)}$ allocates to $\mathcal{T}$'s left subclade, which I call $\iota_{\mathcal{T}}(b)$ — obviously this determines also the sub-budget to allocate to the right subclade, $b - \iota_{\mathcal{T}}(b)$. Note that this does not mean that the

actual expenditures in the subclades need equal $\iota_{\mathcal{T}}(b)$ and $b - \iota_{\mathcal{T}}(b)$, but rather that the left part of $S_{\mathcal{T}}^{(b)}$ is optimal for $\iota_{\mathcal{T}}(b)$, and its right part for $b - \iota_{\mathcal{T}}(b)$. In figure 3.4.1, each cell in the solutions table shows $\lambda_{\mathcal{T}}(b)$ at the top and, when appropriate, in the two bottom corners, $\iota_{\mathcal{T}}(b)$ on the left and (for illustrative purposes) $b - \iota_{\mathcal{T}}(b)$ on the right, which indicates the way the sub-budget should be partitioned between the two subclades.

Note that it is precisely thanks to storing $\iota_{\mathcal{T}}(b)$ that solutions need not be memorised; this information (once available) allows us to reconstruct the optimal solution found for any given $\mathcal{T}$ and b: just visit all the clades in $\mathcal{T}$ in a top-down fashion always using the $\iota_{\mathcal{T}}$ values to find out what part of b should be devoted to each clade; in the end, b will have been broken down into all the expenditures to allocate to each taxon in $\mathcal{T}$; a taxon s should be included in the solution only if the expenditure for s is greater or equal to its cost c_s (see below, algorithm 3, RECONSTRUCT$(\mathcal{T}, b)$, for a formalisation of this procedure). For example, imagine that we wish to reconstruct the solution for clade $\mathcal{T}$ and sub-budget 4 in figure 3.4.1; the two values at the bottom corners of the table entry for $\mathcal{T}$ and 4 (shaded gray) indicate that we should allocate 2 to $\mathcal{L}$ and 2 to $\mathcal{R}$. $\mathcal{L}$ only contains taxon C, whose cost is exactly 2 and therefore should be selected. $\mathcal{R}$ has two subclades, one containing D and the other E; the table entry for $\mathcal{R}$ and 2 indicates that nothing should be spent in the left subclade (so D should not be selected, as its cost is greater than 0) and that 2 should be assigned to the right subclade (so we do select E, as its cost is exactly 2); therefore the solution for $\mathcal{T}$ and 4 is $\{\mathrm{C}, \mathrm{E}\}$.

The solutions table is filled with $\lambda_{\mathcal{T}}(b)$ and $\iota_{\mathcal{T}}(b)$ values in a way analogous to the one we described above in terms of whole solutions. The clades are visited in a bottom-up order. For the clades only consisting of a terminal branch (leading to, say, taxon s), the $\lambda_{\mathcal{T}}(b)$ values are set to 0 for entries with b up to (but not including) c_s and to the length of the terminal branch for the remaining entries; the $\iota_{\mathcal{T}}(b)$ are left undefined, as they have no meaning for these clades. For example, see the solutions table row for $\mathcal{L}$ (shaded red) in figure 3.4.1.

When instead we visit a clade $\mathcal{T}$ composed of a branch (e) and two subclades ($\mathcal{L}$ and $\mathcal{R}$), we need to consider two cases. First, for all the entries with b smaller than the minimum cost among the taxa in $\mathcal{T}$ (which I denote by $\check{c}_{\mathcal{T}}$), $\lambda_{\mathcal{T}}(b)$ is set to 0, as clearly the sub-budget b is not enough to cover the cost of any of the taxa in $\mathcal{T}$; note also that for these entries $\iota_{\mathcal{T}}(b)$ can be set to any $i = 0, 1, \ldots, b$, as any of these values will lead to reconstructing the empty solution (for example, see the first two entries in the table row for $\mathcal{T}$ in figure 3.4.1). Second, for the remaining entries (those with $b \geq \check{c}_{\mathcal{T}}$) $\lambda_{\mathcal{T}}(b)$ is set to the maximum value among $t_e + \lambda_{\mathcal{L}}(0) + \lambda_{\mathcal{R}}(b)$, $t_e + \lambda_{\mathcal{L}}(1) + \lambda_{\mathcal{R}}(b-1), \ldots, t_e + \lambda_{\mathcal{L}}(b) + \lambda_{\mathcal{R}}(0)$ (where t_e represents the length of e), as this is the rPD of the best possible combination of complementary solutions for $\mathcal{L}$ and $\mathcal{R}$; the corresponding $\iota_{\mathcal{T}}(b)$ is set to a value of $i \in \{0, 1, \ldots, b\}$ that maximises $\lambda_{\mathcal{L}}(i) + \lambda_{\mathcal{R}}(b-i)$. For example, suppose we aim to fill the entry for $\mathcal{T}$ and 4 in figure

3.4.1. Since we are proceeding in a bottom-up fashion, the rows corresponding to $\mathcal{L}$ and $\mathcal{R}$ have already been filled, and the $\lambda_\mathcal{R}$ and $\lambda_\mathcal{L}$ values are all available. Therefore, $\lambda_\mathcal{T}(4) = \max\{1.0 + \lambda_\mathcal{L}(0) + \lambda_\mathcal{R}(4), 1.0 + \lambda_\mathcal{L}(1) + \lambda_\mathcal{R}(3), \ldots, 1.0 + \lambda_\mathcal{L}(4) + \lambda_\mathcal{R}(0)\} = \max\{5.0, 5.0, 5.0, 2.0, 2.0\} = 5.0$. This corresponds to consideration of combining the first entry shaded in red with the fifth in blue, the second in red with the fourth in blue, and so on; in the end we check which of these combinations has given the largest rPD. The value of $\iota_\mathcal{T}(4)$ is set accordingly: in this case there are three equivalent combinations and $\iota_\mathcal{T}(4)$ could be set to 0, 1 or 2 (leading to two different but equally good solutions).

In summary, the λ and ι values are calculated with the following recursions (which assume that $\mathcal{T}$ is composed of branch e and, possibly, subclades $\mathcal{L}$ and $\mathcal{R}$):

$$\lambda_\mathcal{T}(b) = \begin{cases} 0 & \text{if } b < \breve{c}_\mathcal{T}, \\ t_e & \text{if } e \text{ is terminal and } b \geq \breve{c}_\mathcal{T}, \\ t_e + \max\limits_{i \in \{0,1,\ldots,b\}} \{\lambda_\mathcal{L}(i) + \lambda_\mathcal{R}(b-i)\} & \text{otherwise.} \end{cases}$$

$$\iota_\mathcal{T}(b) = \begin{cases} \text{undefined} & \text{if } \mathcal{T} \text{ has no subclades,} \\ \operatorname{argmax}\limits_{i \in \{0,1,\ldots,b\}} \{\lambda_\mathcal{L}(i) + \lambda_\mathcal{R}(b-i)\} & \text{otherwise.} \end{cases}$$

The above argmax term indicates the value of i that maximises the expression on its right; when there are several such values, it indicates any one of them (e.g. chosen randomly) or, in the special case where $b \geq \breve{c}_\mathcal{T}$ and $\lambda_\mathcal{L}(i) + \lambda_\mathcal{R}(b-i) = 0$ for all i, any value of i such that the resulting $\iota_\mathcal{T}(b)$ will cause reconstruction of a non-empty solution (this can be achieved by taking either $i = \breve{c}_\mathcal{L}$ if $\breve{c}_\mathcal{L} \leq b$, or $i = b - \breve{c}_\mathcal{R}$ if $\breve{c}_\mathcal{R} \leq b$). Implicitly storing the empty solution would either be wrong (if $t_e > 0$, any optimal solution is certainly non-empty) or (if $t_e = 0$) might lead to constructing a non-optimal solution further up in the tree (see discussion in footnote 2).

The various minimum costs $\breve{c}_\mathcal{T}$ for all clades should also be derived and stored, as they are used in the recursion for $\lambda_\mathcal{T}(b)$ above. For a given $\mathcal{T}$, this can be done when we set about filling its row, by taking either c_s (if $\mathcal{T}$ is a terminal branch leading to s) or the minimum between $\breve{c}_\mathcal{L}$ and $\breve{c}_\mathcal{R}$ (if $\mathcal{T}$ contains subclades $\mathcal{L}$ and $\mathcal{R}$). In figure 3.4.1, the $\breve{c}_\mathcal{T}$ values are reported on the right of the solutions table.

Once all the λ and ι values have been derived in the solutions table, the solution implicitly found for the top-right entry of the table is a solution to the global problem and it can be reconstructed using the $\iota_\mathcal{T}$ values in the way described above. However, there may be more than one optimal solution to BMaxPD, all equally good with respect to rPD, but possibly involving different overall expenditures. It is not guaranteed that the solution reconstructed is the cheapest among them. Although this was not required by the original problem formulation, this is clearly a desirable property and easy to achieve: by looking at the solutions found for smaller budgets, we can check if the budget can be reduced without affecting the optimal rPD. This corresponds to scanning the solutions table from its top-right entry towards the left,

Algorithm 1 BOTTOM-UP($\mathcal{T}$)

if $\mathcal{T}$ contains subclades $\mathcal{L}$ and $\mathcal{R}$ **then**
    BOTTOM-UP($\mathcal{L}$)
    BOTTOM-UP($\mathcal{R}$)
end if
CALCULATE($\mathcal{T}$)

Algorithm 2 CALCULATE($\mathcal{T}$)

if $\mathcal{T}$ only consists of a terminal branch e ending in leaf s **then**
    $\breve{c}_{\mathcal{T}} = c_s$
    for $b = 0, \ldots, \breve{c}_{\mathcal{T}} - 1$ **do** $\lambda_{\mathcal{T}}(b) = 0$
    for $b = \breve{c}_{\mathcal{T}}, \ldots, B$ **do** $\lambda_{\mathcal{T}}(b) = t_e$
end if
if $\mathcal{T}$ consists of an internal branch e and subclades $\mathcal{L}$ and $\mathcal{R}$ **then**
    $\breve{c}_{\mathcal{T}} = \min\{\breve{c}_{\mathcal{L}}, \breve{c}_{\mathcal{R}}\}$
    for $b = 0, \ldots, \breve{c}_{\mathcal{T}} - 1$ **do** $\lambda_{\mathcal{T}}(b) = 0$, $\iota_{\mathcal{T}}(b) = 0$
    for $b = \breve{c}_{\mathcal{T}}, \ldots, B$ **do**
        $\iota_{\mathcal{T}}(b) = \underset{i \in \{0,1,\ldots,b\}}{\operatorname{argmax}} \{\lambda_{\mathcal{L}}(i) + \lambda_{\mathcal{R}}(b-i)\}$
        $\lambda_{\mathcal{T}}(b) = t_e + \lambda_{\mathcal{L}}(\iota_{\mathcal{T}}(b)) + \lambda_{\mathcal{R}}(b - \iota_{\mathcal{T}}(b))$
        if $\lambda_{\mathcal{L}}(\iota_{\mathcal{T}}(b)) + \lambda_{\mathcal{R}}(b - \iota_{\mathcal{T}}(b)) = 0$ **then**
            if $\breve{c}_{\mathcal{L}} \leq b$ **then** $\iota_{\mathcal{T}}(b) = \breve{c}_{\mathcal{L}}$ **else** $\iota_{\mathcal{T}}(b) = b - \breve{c}_{\mathcal{R}}$
        end if
    end for
end if

until a reduction in $\lambda_{\mathcal{T}_0}(b)$ is observed. The solution for the last (i.e. least) b with $\lambda_{\mathcal{T}_0}(b) = \lambda_{\mathcal{T}_0}(B)$ is a minimal-cost rPD-optimal solution, and can be reconstructed in the usual way. In some cases it may even be of interest to derive all (minimal-cost) optimal solutions, which could be achieved by storing multiple $\iota_{\mathcal{T}}(b)$ values and using all of them in the reconstruction at the end. However, I note that the number of solutions to reconstruct may grow exponentially in the size of the problem.

This concludes the description of this dynamic programming algorithm for maximising rPD subject to costs constraints. Another description, less verbose and directly convertible into computer code, is given by the pseudocode in algorithms 1, 2 and 3. BMAXPD applied to $\mathcal{T}_0$ and B is simply solved by a call to BOTTOM-UP($\mathcal{T}_0$) (which fills up the solutions table) followed by a call to RECONSTRUCT($\mathcal{T}_0, B$), which returns an optimal solution. The procedure described in the previous paragraph for finding the cheapest optimal solution is also easily formalised. Additionally, this dynamic programming algorithm could be modified to solve BMAXPD for trees with branches of any length (i.e., where negative lengths are allowed). Since the algorithm I will present in the next section can be more easily modified to deal with negative branch lengths, I do not describe this here.

Regarding the computational complexity of this algorithm, the calculation of a single entry in the solutions table requires $O(b) = O(B)$ time, as all possible ways

Algorithm 3 RECONSTRUCT$(\mathcal{T}, b)$

if $\mathcal{T}$ only consists of a terminal branch ending in leaf s **then**
  if $b \geq c_s$ **return** $\{s\}$
  if $b < c_s$ **return** $\emptyset$
end if
if $\mathcal{T}$ contains subclades $\mathcal{L}$ and $\mathcal{R}$ **then**
  return RECONSTRUCT$(\mathcal{L}, \iota_{\mathcal{T}}(b)) \cup$ RECONSTRUCT$(\mathcal{R}, b - \iota_{\mathcal{T}}(b))$
end if

to split sub-budget b need to be examined. This must be repeated for each of the $(2n-1)(B+1) = O(Bn)$ subproblems (where n is the number of leaves), giving a total of $O(B^2 n)$ operations for filling the solutions table. The reconstruction of an optimal solution from the top-right entry only takes $O(n)$ time, as it consists of a top-down traversal of all the $2n - 1 = O(n)$ clades of $\mathcal{T}_0$, in which each clade can be dealt with in constant time. Therefore the entire algorithm has time complexity $O(B^2 n)$. Memory complexity is dominated by the size of the solutions table, and so is $O(Bn)$.

Finally, I note that BMAXPD could also be formulated as an integer linear programming (ILP) problem (in a way analogous to that of [**RG02**]) and solved with standard off-the-shelf techniques. However, there are no guarantees that the running time of these algorithms would be better than exponential in n.

3.5. A $O(Bn \log n)$-time dynamic programming algorithm

This section[3] presents an alternative dynamic programming algorithm for BMAXPD, whose running time grows linearly with B, instead of quadratically. Its description is somewhat more complicated, and requires the introduction of new formalisms. I denote by $\mathcal{T}^A$, where A is a subset of the nodes in $\mathcal{T}$, the tree obtained from $\mathcal{T}$ by setting to 0 all the lengths of the branches on the paths from the root to the nodes in A. Assuming that the leaves in $\mathcal{T}$ are numbered $1, 2, \ldots, i, \ldots, n$, for simplicity I will write $\mathcal{T}_i$ instead of $\mathcal{T}_{\{1,2,\ldots,i\}}$, i.e., $\mathcal{T}_i$ denotes the restriction of $\mathcal{T}$ on $\{1, 2, \ldots, i\}$ (see definition 2.5.1). Also, I will write $\mathcal{T}_i^A$ instead of $(\mathcal{T}^A)_i$ and $\mathcal{T}^a$ instead of $\mathcal{T}^{\{a\}}$. See figure 3.5.2 for an example of such tree.

I assume that the leaves of the input tree $\mathcal{T}$ are *contiguously numbered*, i.e., we can imagine drawing the tree and then assigning 1 to the leftmost leaf, 2 to the second leaf on the left, and so on until assigning n to the rightmost leaf. To ease the description of the algorithm, I also assume that another leaf, $n+1$, is added to $\mathcal{T}$ — directly attached to its root with a branch of length 0.

The rationale behind these definitions is that $\mathcal{T}_i^J$, where $J \subseteq \{i+1, ..., n+1\}$, contains all the phylogenetic diversity that remains to be gained in $\mathcal{T}_i$ if the taxa in J are selected. It turns out (Proposition 3.5.1 below) that the notation $\mathcal{T}_i^J$ is

[3]The contents of this section also largely appear in a paper written in collaboration with Arndt von Haeseler's group [**MPKvH09**], but they are the result of my own work.

somewhat superfluous as this tree does not depend on the whole collection of taxa in J, but only on the taxon j with minimum index in this set, i.e. $\mathcal{T}_i^J = \mathcal{T}_i^j$ (with $j = n+1$ if $J = \emptyset$). In brief, the algorithm I will describe consists of incrementally solving the problem for all trees of the form $\mathcal{T}_i^j$, with $1 \leq i < j \leq n+1$, and for all the budgets $b \in \{0, 1, \ldots, B\}$ (i and j will be implicitly assumed to be integers throughout). This can be done recursively, by using solutions derived for previously encountered trees and budgets. The exact recursions will be described in the next subsection: we will see that, when $i > 1$, the solution for $\mathcal{T}_i^j$ and budget b will either be obtained from the solution for $\mathcal{T}_{i-1}^i$ and $b - c_i$, or from the one for $\mathcal{T}_{i-1}^j$ and b. Note that the solution to BMaxPD applied to the input tree $\mathcal{T}$ and budget B is the one obtained for $\mathcal{T}_n^{n+1} = \mathcal{T}$ and $b = B$.

Leaving the description of how and why this procedure works to the next section, I now show that the number of $\mathcal{T}_i^j$ trees that need to be taken into account can be reduced from $O(n^2)$ — seemingly implied by the $\mathcal{T}_i^j$ notation — to $O(n \log n)$, if an appropriate preprocessing of the input tree is applied. The following proposition is rather intuitive, and including a proof would be tedious:

Proposition 3.5.1. *Let $\mathcal{T}$ be contiguously numbered, $i \in \{1, 2, \ldots, n\}$ one of its leaves, and J a subset of its leaves such that $\emptyset \neq J \subseteq \{i+1, ..., n+1\}$ and such that $j = \min J$. Then*

$$\mathcal{T}_i^J = \mathcal{T}_i^j = \mathcal{T}_i^{\mathrm{mrca}(i,j)},$$

where $\mathrm{mrca}(i, j)$ *denotes the most recent common ancestor node of i and j.*

The collection of trees we are interested in can therefore be re-expressed in this way:

$$\{\mathcal{T}_i^j : 1 \leq i < j \leq n+1\} =$$
$$\{\mathcal{T}_i^a : 1 \leq i \leq n,\, a = \mathrm{mrca}(i, j) \text{ for some } j > i\} =$$
$$\bigcup_{i=1}^{n} \Theta_i,$$

where $\Theta_i = \{\mathcal{T}_i^a : a = \mathrm{mrca}(i, j) \text{ for some } j > i\} = \{\mathcal{T}_i^j : i < j \leq n+1\}$.

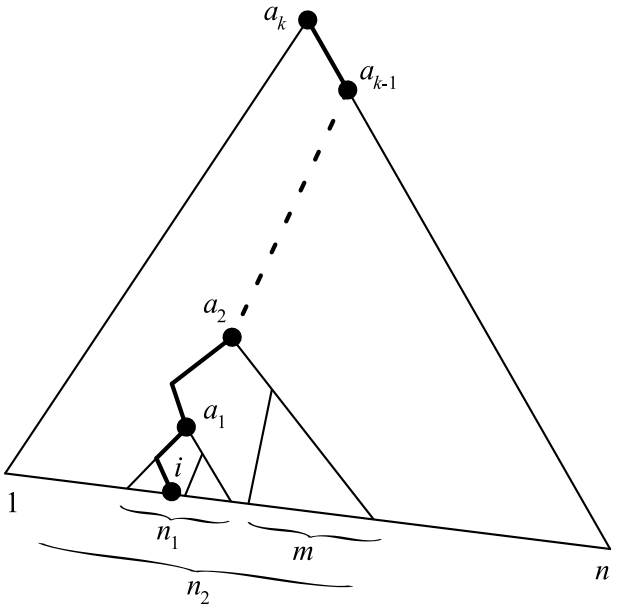
It is important to realise that the size of these sets depends on the chosen numbering of the leaves in $\mathcal{T}$. In fact, I will show below (Prop. 3.5.2) that if we number the leaves so that large clades come towards the end, then we can guarantee that $|\Theta_i|$ does not exceed $1 + \log_2 n$. Therefore, we have that at most

$$\left| \bigcup_{i=1}^{n} \Theta_i \right| \leq n(1 + \log_2 n) = O(n \log n)$$

trees need to be dealt with by the algorithm.

The numbering that ensures this result can be obtained in the following way: imagine that the input tree is drawn with its root at the top; for each internal node compare the clades rooted in that node ("clades" in the sense of the previous section) and then swap these clades so that the largest one (i.e., the one with most leaves)

FIGURE 3.5.1. Illustration of the proof of Proposition 3.5.2.



becomes the rightmost one. At the end of this procedure, label the leaves of the tree with 1 to n, from the leftmost to the rightmost, and then attach the additional leaf $n+1$ to the root.

PROPOSITION 3.5.2. *Let $\mathcal{T}$ be contiguously numbered with the procedure described above. Then, for any $i \in \{1, 2, \ldots, n\}$, $|\Theta_i| \leq 1 + \log_2 n$.*

PROOF. Let $a_1, a_2, \ldots, a_k$ be the ancestor nodes of i such that $a = \text{mrca}(i, j)$ for some $j > i$. Let them be ordered from the closest to i to the furthest (see fig. 3.5.1), with a_k therefore coinciding with the root of $\mathcal{T}$. Clearly $k = |\Theta_i|$. Let $n_1, n_2, \ldots, n_k$ denote the number of descendant leaves of each of these ancestors. Clearly $n_1 \geq 2$, as $a_1 = \text{mrca}(i, j)$ for two distinct leaves i and j.

Now consider the rightmost clade rooted in a_2. Because of the way the taxon numbering was defined, the number of leaves in this clade, m, must be at least the same as that in any other clade rooted in a_2. In particular we will have $m \geq n_1$. Therefore $n_2 \geq n_1 + m \geq 2n_1 \geq 2^2$. Similarly we can prove that $n_3 \geq 2n_2 \geq 2^3$, that $n_4 \geq 2^4, \ldots$, and that $n_{k-1} \geq 2^{k-1}$. Taking logarithms in the last inequality, we obtain $k - 1 \leq \log_2 n_{k-1} \leq \log_2 n$ and therefore $|\Theta_i| = k \leq 1 + \log_2 n$. □

3.5.1. The recursion and its correctness. I now show how to derive optimal solutions for BMAXPD applied to any tree in $\Theta := \bigcup_i \Theta_i = \{\mathcal{T}_i^j : 1 \leq i < j \leq n+1\}$ and any budget in $\{0, 1, \ldots, B\}$. I denote by $S(\mathcal{T}_i^j, b)$ an optimal solution for $\mathcal{T}_i^j$ and b. Recall that the notations $\mathcal{T}_i^{j_1}$ and $\mathcal{T}_i^{j_2}$ with same subscript i but different superscripts $j_1 \neq j_2$ need not correspond to different trees.

It easy to see that for any j, $S(\mathcal{T}_1^j, b)$ should be set to $\emptyset$ or $\{1\}$ depending on whether taxon 1 (the only taxon in $\mathcal{T}_1^j$) is too expensive to be taken with the available budget (i.e., $c_1 > b$), or not (i.e., $c_1 \leq b$), respectively.

As concerns $S(\mathcal{T}_i^j, b)$ for $i > 1$, we need to consider two cases: either $S(\mathcal{T}_i^j, b)$ will include taxon i or not. If not, then we can just ignore i: $S(\mathcal{T}_i^j, b)$ must be optimal for $\mathcal{T}_{i-1}^j$ and b, and will therefore be equal to $S(\mathcal{T}_{i-1}^j, b)$.

If $S(\mathcal{T}_i^j, b)$ will include i, then the remaining part of the solution, $S(\mathcal{T}_i^j, b) - \{i\}$, should take into account the fact that the path from the root to i will be covered by i. That means that $S(\mathcal{T}_i^j, b) - \{i\}$ must be optimal with respect to the remaining budget $b - c_i$ and with respect to $\mathcal{T}_{i-1}^{\{i,j\}}$, the tree that is obtained from $\mathcal{T}_{i-1}^j$ by setting to 0 the lengths of all the branches that are in the path from the root to i. As a consequence of proposition 3.5.1, we have that $\mathcal{T}_{i-1}^{\{i,j\}} = \mathcal{T}_{i-1}^i$, and therefore $S(\mathcal{T}_i^j, b)$ can be obtained as $\{i\} \cup S(\mathcal{T}_{i-1}^i, b - c_i)$.

The two cases above show that $S(\mathcal{T}_i^j, b)$ must be one of two candidate solutions. In order to find which one, we just need to evaluate both and take the best. These informal considerations are summarised by the following recursion:

(3.5.1)

$$S(\mathcal{T}_i^j, b) = \begin{cases} \emptyset & \text{if } i = 1 \text{ and } b < c_1, \\ \{1\} & \text{if } i = 1 \text{ and } b \geq c_1, \\ S(\mathcal{T}_{i-1}^j, b) & \text{if } i > 1 \text{ and } b < c_i, \\ \text{best}_{\mathcal{T}_i^j} \left(S(\mathcal{T}_{i-1}^j, b),\ \{i\} \cup S(\mathcal{T}_{i-1}^i, b - c_i) \right) & \text{if } i > 1 \text{ and } b \geq c_i, \end{cases}$$

where $\text{best}_{\mathcal{T}}(S_1, S_2)$ denotes the set, among S_1 and S_2, that has maximum $\text{rPD}_{\mathcal{T}}$ (and can be either one when these have the same $\text{rPD}_{\mathcal{T}}$).

This recursion is used to first obtain all the $S(\mathcal{T}_1^j, b)$ solutions; then these solutions are used to derive all the $S(\mathcal{T}_2^j, b)$, which in turn are used to obtain all the $S(\mathcal{T}_3^j, b)$, and so on. The procedure ends when all the $S(\mathcal{T}_n^{n+1}, b)$ solutions have been derived and returns $S(\mathcal{T}_n^{n+1}, B)$ as an optimal solution to the global problem.

Figure 3.5.2 illustrates the behaviour of this algorithm on the same instance we saw in figure 3.4.1. The rows in the solutions table correspond to different $\mathcal{T}_i^j$ trees and are filled starting from the top (ignore for the moment their contents, as these will be explained later below): the two top rows, corresponding to the $\mathcal{T}_1^j$ trees, are filled by using the first two cases in recursion (3.5.1). Then, the third row, which corresponds to the $\mathcal{T}_2^j$ trees — actually all the same tree — gets filled by using the other two cases in (3.5.1). Next, the fourth and fifth rows, corresponding to the two trees in $\Theta_3 = \{\mathcal{T}_3^j : 3 < j \leq n+1\}$, are filled by using again the third and fourth case in (3.5.1) and only using information already stored in the third row. In general, note that the rows corresponding to trees in Θ_i are filled by using only solutions derived for trees in Θ_{i-1}. This goes on until all the table has been filled and the solution to the global problem is available at the bottom-right corner of the table. A

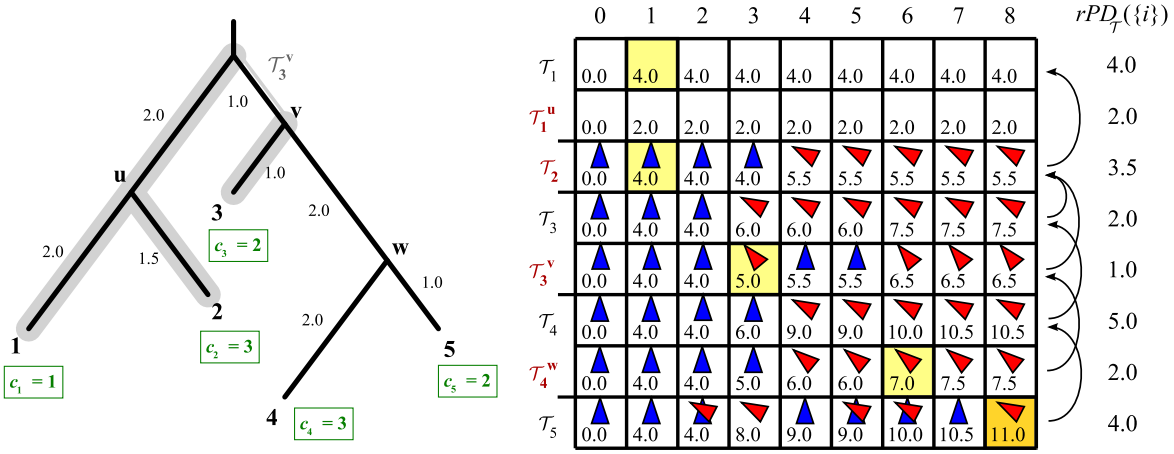
more detailed description of the algorithm's behaviour on this example is provided in section 3.7.1.

Interestingly, $\Theta_1, \Theta_2, \ldots, \Theta_n$ can be thought of as "generations" of trees: the solutions for any tree $\mathcal{T}_i^j \in \Theta_i$ are derived from the solutions for two other trees from the previous generation, $\mathcal{T}_{i-1}^i$ and $\mathcal{T}_{i-1}^j$, which I will call the *father* and *mother* of $\mathcal{T}_i^j$, respectively. (Funnily enough, all the trees in the same generation have the same father).

Because of the relative complexity of this algorithm, compared to the one in the previous section, this time I will include a proof of its correctness.

PROPOSITION 3.5.3. *For any $\mathcal{W} \in \Theta$ and $b \in \{0, 1, \ldots, B\}$, the solution $S(\mathcal{W}, b)$ calculated with the algorithm described above is optimal for $\mathcal{W}$ and b.*

FIGURE 3.5.2. An instance of BMAXPD (rooted case) and its solution.
On the left, an instance of the problem. Branch lengths are indicated to the side of the branches. Taxon costs are in the boxes next to the leaves (green) and the budget is $B = 8$. Tree $\mathcal{T}_3^{\mathrm{v}} = \mathcal{T}_3^{\mathrm{w}} = \mathcal{T}_3^4 = \mathcal{T}_3^5$ is highlighted in grey; one of its branches has a thinner highlight, to indicate that its length in $\mathcal{T}_3^{\mathrm{v}}$ is zero. On the right, a table containing the data derived to solve this instance. Rows correspond to the trees in Θ while column headings 0–8 indicate budgets b. Each cell thus corresponds to a tree $\mathcal{T}_i^j \in \Theta$ and a budget b and displays: (1) at the bottom, the value of $\lambda_{\mathcal{T}_i^j}(b)$ and (2) at the top, an arrow specifying whether this value was obtained as $\lambda_{\mathcal{M}}(b)$ (blue vertical arrow, $\mathcal{M} = \mathcal{T}_{i-1}^j$) or as $\mathrm{rPD}_{\mathcal{T}_i^j}(\{i\}) + \lambda_{\mathcal{F}}(b - c_i)$ (red oblique arrow, $\mathcal{F} = \mathcal{T}_{i-1}^i$). (The presence of both arrows indicates that this value can obtained in both ways and the absence of arrows indicates that one of the first two cases from recursion (3.5.1) was used.) Note that a red oblique arrow implies that taxon i should be included in the solution for $\mathcal{T}_i^j$ and b. On the right of the table are arrows specifying the mother of each tree. The father tree in each generation is indicated in red. The cells in orange and yellow are the ones that are examined when reconstructing the optimal solution for this instance (see main text).



PROOF. By induction on the number i of leaves in $\mathcal{W}$. If $i = 1$, $S(\mathcal{W}, b)$ is obtained with one of the first two cases in recursion (3.5.1) and the thesis is trivially true. If $i > 1$, $S(\mathcal{W}, b)$ must have been obtained either as

$$S(\mathcal{M}, b),$$

(third case in recursion (3.5.1)) or as

$$\underset{\mathcal{W}}{\text{best}} \left(S(\mathcal{M}, b), \{i\} \cup S(\mathcal{F}, b - c_i) \right).$$

(fourth case in recursion (3.5.1)), where $\mathcal{M}$ and $\mathcal{F}$ are the mother and father tree of $\mathcal{W}$ respectively. Since both these trees have $i - 1$ leaves, I apply the inductive hypothesis and assume that $S(\mathcal{M}, b)$ and $S(\mathcal{F}, b - c_i)$ (if $b \geq c_i$) are optimal for their trees and budgets. Whichever of the two cases above was used to derive $S(\mathcal{W}, b)$, it is easy to see that, since $S(\mathcal{M}, b)$ and $S(\mathcal{F}, b - c_i)$ (assuming $b \geq c_i$) are feasible for their budgets, $S(\mathcal{W}, b)$ must be feasible for b.

Let S' be any other feasible solution for $\mathcal{W}$ and b. I will show that $\text{rPD}_{\mathcal{W}}(S') \leq \text{rPD}_{\mathcal{W}}(S(\mathcal{W}, b))$, which implies the optimality of $S(\mathcal{W}, b)$. Throughout this proof, the $\text{rPD}_{\mathcal{W}}$ of an optimal solution for $\mathcal{W}$ and b is denoted by $\lambda_{\mathcal{W}}(b)$.

If the third case in recursion (3.5.1) was used, then we have $\text{rPD}_{\mathcal{W}}(S(\mathcal{W}, b)) = \text{rPD}_{\mathcal{W}}(S(\mathcal{M}, b)) = \text{rPD}_{\mathcal{M}}(S(\mathcal{M}, b)) = \lambda_{\mathcal{M}}(b)$. If instead the fourth case in recursion (3.5.1) was used, then we have

$$\begin{gathered}\text{rPD}_{\mathcal{W}}(S(\mathcal{W}, b)) = \max \left\{ \text{rPD}_{\mathcal{W}} \left(S(\mathcal{M}, b) \right), \text{rPD}_{\mathcal{W}} \left(\{i\} \cup S(\mathcal{F}, b - c_i) \right) \right\} = \\ \max \left\{ \text{rPD}_{\mathcal{M}} \left(S(\mathcal{M}, b) \right), \text{rPD}_{\mathcal{W}}(\{i\}) + \text{rPD}_{\mathcal{F}} \left(S(\mathcal{F}, b - c_i) \right) \right\} = \\ \max \left\{ \lambda_{\mathcal{M}}(b), \text{rPD}_{\mathcal{W}}(\{i\}) + \lambda_{\mathcal{F}}(b - c_i) \right\}.\end{gathered}$$

In both cases we have that $\text{rPD}_{\mathcal{W}}(S(\mathcal{W}, b))$ is greater or equal to $\lambda_{\mathcal{M}}(b)$ and also to $\text{rPD}_{\mathcal{W}}(\{i\}) + \lambda_{\mathcal{F}}(b - c_i)$, if $b \geq c_i$.

Regarding S', there are two cases, depending on whether it contains i or not. If it does not contain i, then $\text{rPD}_{\mathcal{W}}(S') = \text{rPD}_{\mathcal{M}}(S') \leq \lambda_{\mathcal{M}}(b) \leq \text{rPD}_{\mathcal{W}}(S(\mathcal{W}, b))$, where the first inequality comes from the fact that S' is a feasible solution for $\mathcal{M}$ and b. If S' does contain i, then

$$\text{rPD}_{\mathcal{W}}(S') = \text{rPD}_{\mathcal{W}}(\{i\}) + \text{rPD}_{\mathcal{F}}(S' - \{i\}) \leq \text{rPD}_{\mathcal{W}}(\{i\}) + \lambda_{\mathcal{F}}(b - c_i) \leq \text{rPD}_{\mathcal{W}}(S(\mathcal{W}, b)),$$

where the first inequality comes from the fact that, as the total cost of S' is at most b, $S' - \{i\}$ must be feasible for $\mathcal{F}$ and $b - c_i$. In both cases, $\text{rPD}_{\mathcal{W}}(S') \leq \text{rPD}_{\mathcal{W}}(S(\mathcal{T}, b))$. □

COROLLARY 3.5.4. *$S(\mathcal{T}_n^{n+1}, B)$ is optimal for the input tree and budget.*

3.5.2. Actual implementation of the algorithm. For simplicity, the algorithm above has been described in terms of optimal sets of taxa to store for each subproblem. As for the previous algorithm, however, it is sufficient to store their rPD and some information that allows their reconstruction. Let $\mathcal{T}$ now denote a

generic tree in Θ_i and $\mathcal{F}$ and $\mathcal{M}$ its father and mother tree, if any. As in the previous section, $\lambda_{\mathcal{T}}(b)$ denotes the rPD of an optimal solution for $\mathcal{T}$ and b. The following recursion (a simple translation of (3.5.1)) allows us to calculate $\lambda_{\mathcal{T}}(b)$ for all $\mathcal{T} \in \Theta$ and $b \in \{0, 1, \ldots, B\}$:

$$\lambda_{\mathcal{T}}(b) = \begin{cases} 0 & \text{if } i = 1 \text{ and } b < c_1, \\ \mathrm{rPD}_{\mathcal{T}}(\{1\}) & \text{if } i = 1 \text{ and } b \geq c_1, \\ \lambda_{\mathcal{M}}(b) & \text{if } i > 1 \text{ and } b < c_i, \\ \max\{\, \lambda_{\mathcal{M}}(b),\, \mathrm{rPD}_{\mathcal{T}}(\{i\}) + \lambda_{\mathcal{F}}(b - c_i) \,\} & \text{if } i > 1 \text{ and } b \geq c_i. \end{cases} \tag{3.5.2}$$

Once the PD of an optimal solution to the global problem has been derived, one (or even all) optimal solution(s) can be generated by tracing back the choices that have led to this value. In practice one can store a flag specifying whether a solution was derived from the one for the mother tree $\mathcal{M}$ or from the one for the father tree $\mathcal{F}$ (see fig. 3.5.2).

Also, it is important to note that the dynamic programming algorithm described above needs to be preceded by a preprocessing phase that determines: (1) the contiguous numbering of the taxa, (2) a memory location where to store the necessary information for each tree in Θ, (3) for each tree in $\Theta - \Theta_1$, pointers to the memory locations for its mother and father tree, (4) for each tree $\mathcal{T} \in \Theta_i$, the value of $\mathrm{rPD}_{\mathcal{T}}(\{i\})$. I will not describe how these can be done here, but it suffices to know that computationally the time needed for the preprocessing will be dominated by the remaining operations.

Such operations basically consist in deriving all the $\lambda_{\mathcal{T}}(b)$ values (and associated flags), and then reconstructing an optimal set of leaves. There are $O(Bn \log n)$ values for $\lambda_{\mathcal{T}}(b)$ and each value can be computed in constant time (see recursion (3.5.2)). The reconstruction of an optimal solution takes $O(n)$ time. Therefore the time complexity of the whole procedure is $O(Bn \log n)$.

Finally, note that in order to deal with negative branch lengths as well, the algorithm described above can be simply modified by altering the way the solutions for trees in Θ_1 are derived: the only change we need to make to recursion (3.5.1) is to set the optimal solution to $\mathrm{best}_{\mathcal{T}_i^j}(\emptyset, \{1\})$ in the second case (i.e., for $i = 1$ and $b \geq c_i$). For recursion (3.5.2), just change that same case into $\lambda_{\mathcal{T}}(b) = \max\{0, \mathrm{rPD}_{\mathcal{T}}(\{1\})\}$.

3.6. The unrooted case

I now turn to solving the unrooted version of BMaxPD, i.e., the one with PD defined as uPD[4]. An example is given in figure 3.6.1. At first glance, an obstacle to its solution seems to be that there is no evident way to break it into smaller problems: even if we solve the unrooted problem on portions of the input tree — which I will denote once again by $\mathcal{T}_0$ — this does not tell us much about the solution for the whole tree.

[4]Again, the contents of this section are based on the *Systematic Biology* 2007 paper [**PG07**].

The key observation here is that a solution to the unrooted problem, if not the empty set, is equal to $\{s\} \cup R$, where R is a solution to the rooted problem with budget $B - c_s$ applied to $\mathcal{T}_0^s$, which in this section will denote the version of $\mathcal{T}_0$ rooted in leaf s. (For example in figure 3.6.1, {A,C,D,E}, the optimal solution to the unrooted problem, can be written as {A}∪{C,D,E}, where {C,D,E} is a solution to the rooted problem on $\mathcal{T}_0^{\mathrm{A}}$ with budget $B - c_{\mathrm{A}} = 8 - 1 = 7$.)

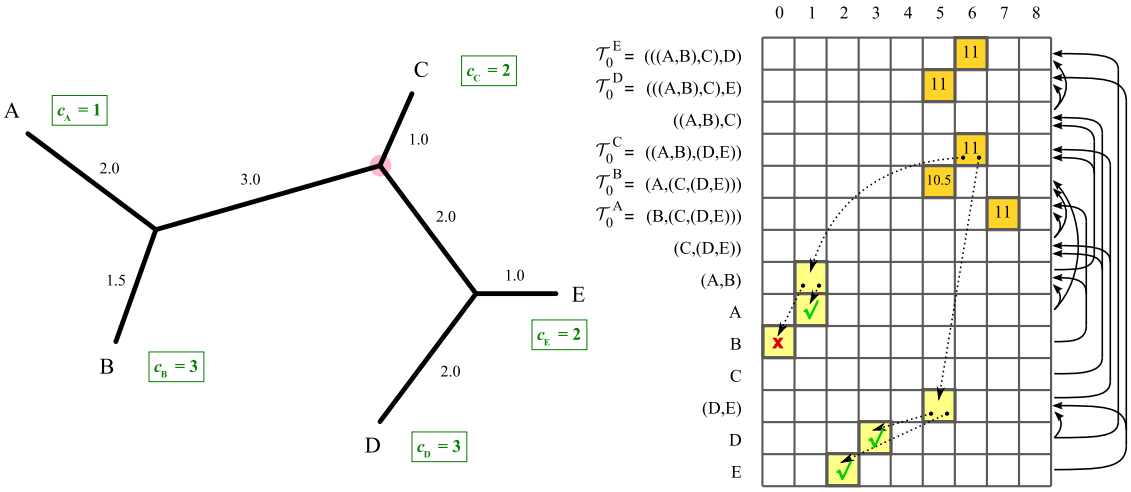
This allows us to reduce the unrooted problem to a number of related rooted problems: we could iteratively root $\mathcal{T}_0$ in each of its leaves s and calculate a solution R_s to the rooted problem with budget $B - c_s$ (or skip s if $c_s > B$); any of the subsets $\{s\} \cup R_s$ with the largest uPD (or, equivalently, any of the ones with the largest rPD(R_s)) is a solution to the unrooted problem. This procedure involves repeating any of the rooted algorithms described in the previous two sections once for every leaf, thus requiring $O(B^2n^2)$ or $O(Bn^2 \log n)$ time, depending on whether we use the algorithm of section 3.4 or that of section 3.5, respectively.

However there is a better way to adapt the algorithm of section 3.4, one that ends up requiring only $O(B^2n)$ time. The rest of this section is devoted to describing this approach. In brief, it consists of extending the definition of a clade so that it includes all the different leaf-rooted trees $\mathcal{T}_0^s$; then, as before, we solve the rooted subproblems for all the clades in $\mathcal{T}_0$ and all possible sub-budgets. Since this now includes additional clades not present before, we devise a new ordering of the clades so that we can derive new solutions from the previously calculated ones. In the end, we compare the rooted solutions found for the various $\mathcal{T}_0^s$; any of the best ones will provide us with an optimal solution to the unrooted problem.

In this section I define a *clade* of $\mathcal{T}_0$ as any rooted subtree $\mathcal{T}$ of $\mathcal{T}_0$ consisting of (a) a branch e, with one of its ends being the root of $\mathcal{T}$, and (b) everything else in $\mathcal{T}_0$ that lies on the other (i.e. non-root) side of e. This definition includes all the clades in the sense of section 3.4, but whereas before each branch identified one single clade, now for each branch there are two clades, each rooted by one of its ends. For example the branch of length 3.0 in figure 3.6.1 is at the 'top' of both clades (A,B) and (C,(D,E)). Assuming that $\mathcal{T}_0$ is originally unrooted (the position of any root has no relevance for uPD) and bifurcating (as in sec. 3.4), there are exactly $2(2n - 3)$ clades in $\mathcal{T}_0$. In particular, note that the $\mathcal{T}_0^s$, the versions of $\mathcal{T}_0$ rooted in each leaf s, are now clades.

Although the collection of clades of $\mathcal{T}_0$ is now non-hierarchical, the fundamental properties of a single clade have not changed from section 3.4: a clade is still a rooted tree with branch lengths and leaf costs. Therefore we can apply the rooted version of BMaxPD to all the clades $\mathcal{T}$ of $\mathcal{T}_0$ and sub-budgets $b \in \{0, 1, \ldots, B\}$. We are particularly interested in the solutions R_s for the leaf-rooted clades $\mathcal{T}_0^s$ and sub-budgets $B - c_s$, because, as observed above, some of them provide us with solutions $\{s\} \cup R_s$ to the unrooted problem.

FIGURE 3.6.1. An instance of BMAXPD (unrooted case) and its solution.
On the left, an instance of the problem. Branch lengths are indicated by the numbers to the side of the branches. Taxon costs are in the boxes next to the leaves (green) and the budget is $B = 8$. On the right, the corresponding solutions table (partially filled, for clarity). Rows correspond to the clades specified to the left, while column headings 0–8 indicate sub-budgets b. Arrows to the right of the table indicate the dependencies amongst the rows; see the main text for an explanation of the row ordering. Orange cells correspond to solutions to the rooted subproblem for clades $\mathcal{T}_0^s$ and sub-budgets $B - c_s$. Dotted arrows and yellow cells show the reconstruction of the solution {A, C, D, E} to the unrooted problem (visited taxa are marked with a green tick or a red cross, depending on whether they are selected of not).



All these subproblems will again be implicitly solved by incrementally deriving $\lambda_{\mathcal{T}}(b)$ and $\iota_{\mathcal{T}}(b)$ for all $\mathcal{T}$ and b. As we showed before, the calculation of these values is either straightforward or directly obtainable from the corresponding values for $\mathcal{T}$'s subclades. It is therefore necessary to tackle the clades in an order that guarantees that subclades are met before their "superclades". Whereas before this was trivially satisfied by a bottom-up traversal of all clades, now a slightly more complex approach is needed. First, $\mathcal{T}_0$ is rooted in an arbitrarily chosen node (different choices lead to different orderings of the clades, but all produce the same final results). This determines a way to picture the tree (imagine it redrawn with the root at the top and all branches descending from there) and we can then classify clades into *downward* clades — those consisting of a branch and everything below it — and and *upward* clades — the remaining ones. For example, in figure 3.6.1 the root is set to the position highlighted in pink; as a result, (A,B) is a downward clade and (C,(D,E)) an upward clade.

The λ and ι values are calculated for downward clades first, going in the same bottom-up order as in section 3.4. In figure 3.6.1, where again we imagine that all

the results are stored in a solutions table, this corresponds to filling the bottom half of the table, starting from the bottom row and going upwards. The results for the upward clades are then derived in a top-down fashion: we can imagine that a preorder traversal of all the branches in $\mathcal{T}_0$ is carried out [**CLRS01**] and each time a branch is visited, the λ and ι values for the corresponding upward clade are computed. In figure 3.6.1, upward clades are visited in the following order: 1: (C,(D,E)), 2: (B,(C,(D,E))), 3: (A,(C,(D,E))), 4: ((A,B),(D,E)), 5: ((A,B),C), 6: (((A,B),C),E), 7: (((A,B),C),D), which corresponds to filling the top half of the solutions table from bottom to top. This ordering of the clades guarantees that whenever we derive solutions for a clade $\mathcal{T}$ containing subclades $\mathcal{L}$ and $\mathcal{R}$, the solutions for $\mathcal{L}$ and $\mathcal{R}$ have already been calculated.

Once all the λ and ι values have been calculated, we turn our attention to the values $\lambda_{\mathcal{T}_0^s}(B - c_s)$ for all leaves s. By definition, these are equal to $\mathrm{rPD}(R_s)$, where R_s is a solution to the rooted problem with budget $B - c_s$ applied to $\mathcal{T}_0^s$, and therefore also equal to the uPD of the candidate solutions $\{s\} \cup R_s$ to the unrooted problem. Clearly, the leaves s that maximise $\lambda_{\mathcal{T}_0^s}(B - c_s)$ are those contained in an optimal solution to the unrooted problem (in figure 3.6.1, all of them except B). If we pick any one of these leaves s and reconstruct R_s by using the $\iota_{\mathcal{T}}$ values starting from $\iota_{\mathcal{T}_0^s}(B - c_s)$ — as described in the previous section or, equivalently, with a call to RECONSTRUCT$(\mathcal{T}_0^s, B - c_s)$ — we therefore also obtain an optimal solution $\{s\} \cup R_s$ to the unrooted problem. For example, if in figure 3.6.1 we pick taxon C, following the $\iota_{\mathcal{T}}$ values (indicated in the figure by dotted arrows) leads to rooted solution {A,D,E} and therefore {A,C,D,E} is a solution to the unrooted problem.

Note that whereas in this example {A,C,D,E} coincides with the set of leaves that maximise $\lambda_{\mathcal{T}_0^s}(B - c_s)$, in general the latter will not necessarily coincide with an optimal solution but rather with the *union* of all optimal solutions to the unrooted problem.

A more concise description of this algorithm for the unrooted version of BMaxPD is given by the pseudocode in algorithms 1–5. An optimal solution is simply obtained by a call to DOUBLE TRAVERSAL$(\mathcal{T}_0)$ (where $rev(\mathcal{T})$ denotes the clade rooted in the same branch as $\mathcal{T}$, but oriented towards the opposite side). The cheapest optimal solution can be obtained in a similar way to that described in section 3.4: just try to reduce the budget below $B - c_s$ for all leaves s that maximise $\lambda_{\mathcal{T}_0^s}(B - c_s)$; one of the leaves that lead to the largest reduction should be used as starting point for the reconstruction of the minimal-cost uPD-optimal set of leaves. This is achieved by straightforward modification of the code in DOUBLE TRAVERSAL$(\mathcal{T})$.

The algorithm just described for the unrooted problem is practically equivalent in computational complexity to the one in the last section. The solutions table has now $2(2n - 3)$ rows, but this is still $O(n)$. Reconstructing an optimal solution, including searching for the leaf to start the reconstruction from, still takes just $O(n)$ time.

Algorithm 4 DOUBLE TRAVERSAL($\mathcal{T}$)

do root $\mathcal{T}$ in any one of its leaves
BOTTOM-UP($\mathcal{T}$)
TOP-DOWN($rev(\mathcal{T})$)
let s be a leaf of $\mathcal{T}$ that maximises $\lambda_{\mathcal{T}^s}(B - c_s)$
return $\{s\} \cup$ RECONSTRUCT($\mathcal{T}^s, B - c_s$)

Algorithm 5 TOP-DOWN($\mathcal{T}$)

CALCULATE($\mathcal{T}$)
if $\mathcal{T}$ is a subclade of two other clades $\mathcal{F}$ and $\mathcal{M}$ **then**
    TOP-DOWN($\mathcal{F}$)
    TOP-DOWN($\mathcal{M}$)
end if

Therefore this algorithm also has time complexity $O(B^2 n)$ and memory complexity $O(Bn)$.

A simple improvement to this algorithm — and also to those in sections 3.4 and 3.5 — can be obtained by noting that the rows of the solutions table do not always need to be filled up until their last column. If a subtree $\mathcal{T}$ (a clade or a $\mathcal{T}_i^j$ subtree in the case of the algorithm in sec. 3.5) is such that the total sum of the costs of its leaves, which we may denote by $C_{\mathcal{T}}$, is smaller than B, then the solutions for $\mathcal{T}$ and b need only be derived for $b < C_{\mathcal{T}}$: the solutions for $b \geq C_{\mathcal{T}}$ simply consist of all leaves in $\mathcal{T}$. Procedures CALCULATE($\mathcal{T}$) and RECONSTRUCT($\mathcal{T}, b$) can be modified accordingly. This results in some saving of running time, although it does not affect the time complexity of the algorithms presented.

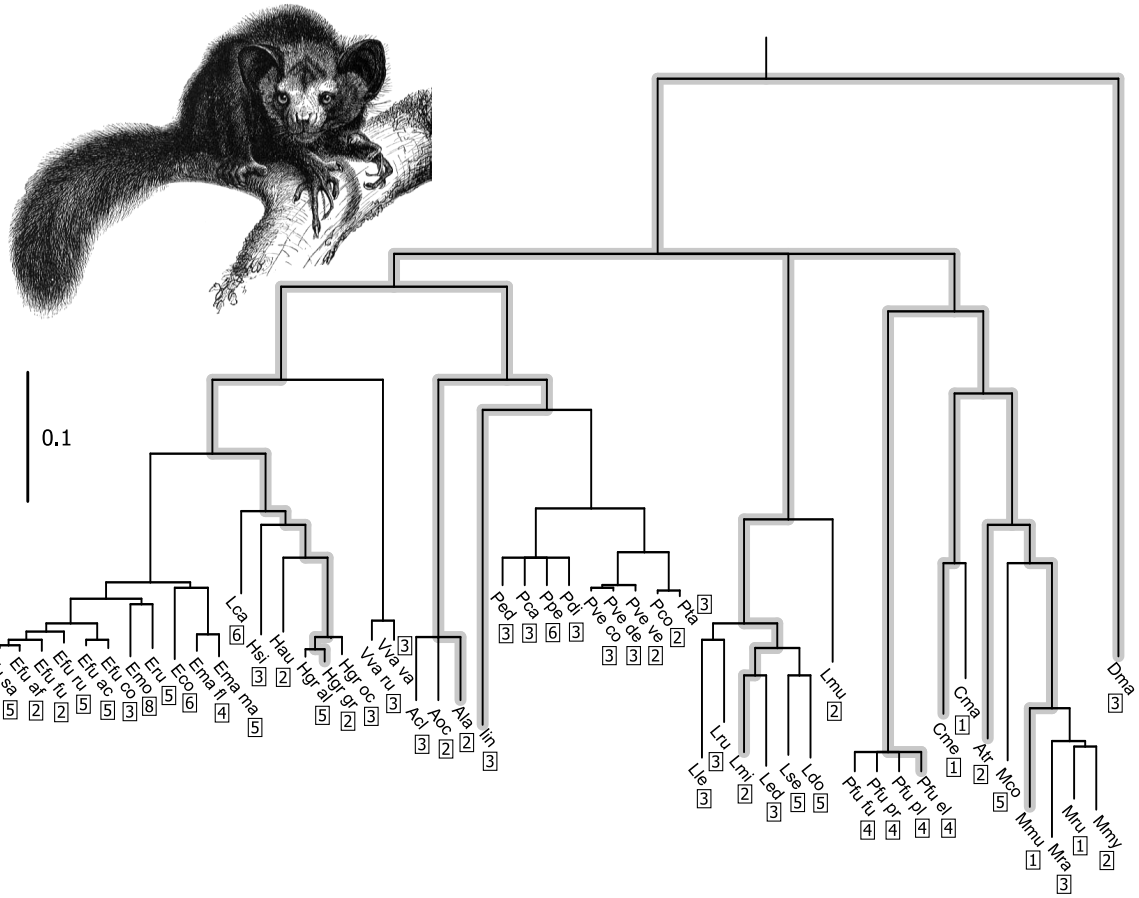
3.7. Examples

Sections 3.4 and 3.6 already provide toy examples of the behaviour of the algorithms I presented there. For the algorithm in section 3.5, a detailed example is provided in the next subsection.

These examples are also designed to show the limitations of greedy algorithms on BMAXPD. There are several possible greedy algorithms for this problem, but the most naïve among them consists of always selecting the leaf that most increases PD among those that can be selected with the remaining budget. It is easy to see that this greedy algorithm would not work on these examples: for instance, in the case pictured in figure 3.4.1, D would be selected first, followed by A and then, considering the remaining budget ($B - c_{\mathrm{D}} - c_{\mathrm{A}} = 8 - 3 - 1 = 4$), B would be selected next, leading to a suboptimal total rPD of 10.5. Another greedy algorithm — consisting of always selecting the leaf with the highest ratio between increase in PD and cost c_s [**HS06**] — would actually produce an optimal solution for the instance in figure 3.4.1. It would however fail if we set the budget to $B = 4$: whereas the optimal solution for this budget is {A, D}, with an rPD of 9 (yellow cell in fig. 3.4.1), this

FIGURE 3.7.1. An instance of BMAXPD relevant to the conservation of Madagascar lemurs.

The phylogenetic tree is the same as in fig. 2.8.1. Taxon conservation costs (in the boxes next to the leaves) were estimated from information available from the IUCN Red List [**IUC06**], and can be considered as expressed in terms of the underlying resource of limited availability, e.g., as millions of euros. The solution of this instance of BMAXPD for a budget $B = 20$ consists of taking the taxa at the leaves of the subtree highlighted in gray.



greedy algorithm would initially select A and then E, thus leading to a suboptimal solution.

For a more realistic example, consider figure 3.7.1. The solution to this instance of BMAXPD for $B = 20$, both in the rooted and unrooted case, consists of the taxa in {Hgr gr, Ala, Iin, Lmi, Pfu el, Cme, Atr, Mmu, Dma}. It should be noted that this solution cannot be obtained from the one for $B = 19$ ({Hgr gr, Ala, Iin, Lmi, Pfu el, Cme, Cma, Mmu, Dma}) through a greedy step (i.e., a simple addition), but needs exchange of one taxon (Cma) for another (Atr).

This and other examples were solved with a program I wrote that implements the algorithms published in 2007 [**PG07**] (C++ code and executables available via http://www.ebi.ac.uk/goldman/rats). Note that a brute-force approach to solve

BMaxPD is not possible even on a moderate-size phylogenetic tree such as the one I have just shown, where the number of possible subsets of taxa is $2^{52} > 10^{15}$. My program can comfortably solve even much larger instances than this one. I have experimented with instances with B of the order of the thousands and the program implementing the algorithm of section 3.4 only takes a few seconds per clade (1.7 GHz Pentium processor with 512 MB of RAM). Note that these are the times needed to process one clade and that they do not depend on the size of the clades (because each nontrivial clade will require the same $O(B^2)$ operations to be processed). The total running time is then obtained by multiplying these times by the number of clades in the input tree and therefore depends on the number of taxa but is independent of the tree shape.

3.7.1. A toy example for the algorithm in section 3.5. I will now illustrate the behaviour of the algorithm described in section 3.5 on the example in figure 3.5.2.

The preprocessing phase results in the following: (1) it makes sure that for each internal node the rightmost clade rooted in that node has maximum size among the clades rooted in that node (see Appendix): for nodes **u** and **w**, both clades have the same size; for node **v** the right clade has two taxa whereas the left one only has one taxon, so this requirement is satisfied. The numbering $(1, 2, 3, 4, 5)$ ensues. (2) Each tree in Θ gets allocated a row in the table on the right; the tree corresponding to each row is indicated on its left (in red the fathers in each generation). (3) For each $\mathcal{T} \in \Theta$, a pointer to its mother tree is derived and indicated to the right of the table. The pointer to the father tree is not explicitly indicated, as the father tree is the first one whose identifier is in red as we go up from $\mathcal{T}$'s row. (4) The $\mathrm{rPD}_{\mathcal{T}}(\{i\})$ values are derived and indicated on the right of the table.

The contents of the table are explained in the legend of figure 3.5.2. The algorithm works by deriving the $\lambda_{\mathcal{T}}(b)$ values one row at a time, from the top to the bottom row. For the first two rows, this is obtained by applying the first two cases of recursion (3.5.2): in both cases $\lambda_{\mathcal{T}}(0)$ is set to 0, as $0 < 1 = c_1$. For the remaining entries of these rows, $\lambda_{\mathcal{T}}(b)$ is set to $\mathrm{rPD}_{\mathcal{T}}(\{1\})$, which is 4.0 for $\mathcal{T} = \mathcal{T}_1$ and 2.0 for $\mathcal{T} = \mathcal{T}_1^{\mathrm{u}}$.

For the other rows, the third and fourth case of (3.5.2) are used. I will show how this is done for the row relative to tree $\mathcal{T}_4$ (the third from the bottom): first of all note that the mother of $\mathcal{T}_4$ (which can be looked up by following the arrow on the right of the table) is $\mathcal{M} = \mathcal{T}_3$, whereas its father (which can be found by looking for the first tree indicated in red above $\mathcal{T}_4$) is $\mathcal{F} = \mathcal{T}_3^{\mathrm{v}}$. Because $c_4 = 3$, $\lambda_{\mathcal{T}_4}(0)$, $\lambda_{\mathcal{T}_4}(1)$, $\lambda_{\mathcal{T}_4}(2)$ are all obtained with the third case of (3.5.2), i.e., they are set to $\lambda_{\mathcal{T}_3}(0) = 0.0$, $\lambda_{\mathcal{T}_3}(1) = 4.0$, $\lambda_{\mathcal{T}_3}(2) = 4.0$, respectively. The remaining entries in this

row are obtained with the fourth case of (3.5.2):

$$\lambda_{\mathcal{T}_4}(3) = \max\{\,\lambda_{\mathcal{T}_3}(3),\, \mathrm{rPD}_{\mathcal{T}_4}(\{4\}) + \lambda_{\mathcal{T}_3^{\mathrm{v}}}(0)\,\} = \max\{\,6.0,\, 5.0 + 0.0\,\} = 6.0,$$
$$\lambda_{\mathcal{T}_4}(4) = \max\{\,\lambda_{\mathcal{T}_3}(4),\, \mathrm{rPD}_{\mathcal{T}_4}(\{4\}) + \lambda_{\mathcal{T}_3^{\mathrm{v}}}(1)\,\} = \max\{\,6.0,\, 5.0 + 4.0\,\} = 9.0,$$
$$\lambda_{\mathcal{T}_4}(5) = \max\{\,\lambda_{\mathcal{T}_3}(5),\, \mathrm{rPD}_{\mathcal{T}_4}(\{4\}) + \lambda_{\mathcal{T}_3^{\mathrm{v}}}(2)\,\} = \max\{\,6.0,\, 5.0 + 4.0\,\} = 9.0,$$
$$\lambda_{\mathcal{T}_4}(6) = \max\{\,\lambda_{\mathcal{T}_3}(6),\, \mathrm{rPD}_{\mathcal{T}_4}(\{4\}) + \lambda_{\mathcal{T}_3^{\mathrm{v}}}(3)\,\} = \max\{\,7.5,\, 5.0 + 5.0\,\} = 10.0,$$
$$\lambda_{\mathcal{T}_4}(7) = \max\{\,\lambda_{\mathcal{T}_3}(7),\, \mathrm{rPD}_{\mathcal{T}_4}(\{4\}) + \lambda_{\mathcal{T}_3^{\mathrm{v}}}(4)\,\} = \max\{\,7.5,\, 5.0 + 5.5\,\} = 10.5,$$
$$\lambda_{\mathcal{T}_4}(8) = \max\{\,\lambda_{\mathcal{T}_3}(8),\, \mathrm{rPD}_{\mathcal{T}_4}(\{4\}) + \lambda_{\mathcal{T}_3^{\mathrm{v}}}(5)\,\} = \max\{\,7.5,\, 5.0 + 5.5\,\} = 10.5.$$

Once all the $\lambda_{\mathcal{T}}(b)$ values have been derived, the value of $\lambda_{\mathcal{T}_5}(8) = 11.0$ (orange cell) indicates the rPD of an optimal subset for the global problem. In order to reconstruct such an optimal subset S, we need to backtrack the sequence of choices that led to this value. Because $\lambda_{\mathcal{T}_5}(8)$ was obtained as $\mathrm{rPD}_{\mathcal{T}_5}(\{5\}) + \lambda_{\mathcal{T}_4^{\mathrm{w}}}(6) = 4.0 + 7.0 = 11.0$, then $5 \in S$. Then we look at $\lambda_{\mathcal{T}_4^{\mathrm{w}}}(6)$ (yellow): because it was obtained as $\mathrm{rPD}_{\mathcal{T}_4^{\mathrm{w}}}(\{4\}) + \lambda_{\mathcal{T}_3^{\mathrm{v}}}(3) = 2.0 + 5.0 = 7.0$, then also $4 \in S$. Next we look at $\lambda_{\mathcal{T}_3^{\mathrm{v}}}(3)$ (yellow): because it was obtained as $\mathrm{rPD}_{\mathcal{T}_3^{\mathrm{v}}}(\{3\}) + \lambda_{\mathcal{T}_2}(1) = 1.0 + 4.0 = 5.0$, then also $3 \in S$. $\lambda_{\mathcal{T}_2}(1)$ was instead obtained from $\mathcal{T}_2$'s mother as $\lambda_{\mathcal{T}_1}(1) = 4.0$, so $2 \notin S$. Finally, since the remaining budget (1) is enough to cover the cost of taxon 1, then also $1 \in S$. Therefore we have that the optimal subset is $\{1, 3, 4, 5\}$.

3.8. Related work

Another algorithm solving BMaxPD can be obtained by adapting the algorithm of Minh et al. [**MKvH08**]. (This is the other main result in the paper written in collaboration with von Haeseler's group [**MPKvH09**].) Although the computational complexity of the resulting algorithm — $O(Bn^2)$ and $O(Bn^3)$ time in the rooted and unrooted case, respectively — is not better than that of the algorithms I presented (see in particular sec. 3.5), this approach has the benefit of being applicable not only to phylogenetic trees, but also, more generally, to circular split systems (as already discussed in sec. 2.12).

BMaxPD can also be seen as a particular case of the Noah's Ark problem [**Wei98**], which will not be discussed here, but extensively dealt with in the next two chapters. A different generalisation of BMaxPD was considered by Bordewich and Semple [**BS08**]. This is the "budgeted nature reserve selection problem", where the selection is applied to *groups* of species, instead of single species, and where the objective is to maximise the PD of the *union* of the selected groups. In the context of conservation biology, each group typically represents the subset of species living in a candidate geographical site for a nature reserve. A budget and costs for the various groups are also given. For this problem (and therefore also for BMaxPD, which is its particular case for groups containing only one taxon each), Bordewich and Semple show a polynomial-time approximation algorithm [**BS08**]. The version of this problem with constant costs was also considered by Rodrigues and Gaston

[**RG02**], who formulated it as an integer linear program, and by Moulton et al. [**MSS07**], who showed its NP-hardness.

Optimisation problems aiming to maximise the diversity of conserved species subject to budgetary constraints have also been the subject of a lot of research in conservation biology (e.g., [**CPSC96, PCS$^+$00, CM01, RG02, Öna04, ACH$^+$04**]), although the focus there has been almost exclusively on selecting geographical sites for conservation (rather than species) and often diversity is simply measured in terms of species richness (but see discussion at page 19). The techniques proposed to solve these problems are rarely sophisticated: often they involve the application of some greedy strategy (not guaranteeing optimality) or of standard techniques if the problem can be formulated as an (integer) linear program (therefore not guaranteeing a polynomial-time resolution).

CHAPTER 4

Including survival probabilities: future PD and the Noah's Ark problem

4.1. The missing ingredient in conservation biology: extinction risks

In this chapter — and the next one — the focus will shift to the topic phylogenetic diversity was originally intended for: conservation biology. In this context, an important factor that should be taken into account is that not all taxa are equally likely to become extinct. Clearly, if a species is not endangered, then expending resources to conserve this species (or a very closely related one, with very little additional diversity) is not a good choice. Conservation efforts should concentrate on the most endangered taxa.

Usually extinctions cannot be predicted with certainty, but often the probability of their happening can be estimated. Assuming that a probabilistic model for the survival of the taxa until some future time is given (usually in the form of taxon-specific probabilities), the PD of the taxa that will survive — which I call the *future* PD — is a random variable, with a well-defined distribution. The future PD should be thought of as a measure of the future remaining biodiversity. Its distribution and dependence on human actions are the subject of this chapter.

The distribution of the future PD is important as it gives us an understanding of the range of possible consequences of extinctions on biodiversity. Section 4.2 will provide a mathematical result showing that if the tree has enough taxa and its branch lengths and taxon-associated probabilities satisfy mild regularity conditions, then it is reasonable to expect that the future PD has an approximately normal distribution, with an easy-to-compute mean and variance. Furthermore, section 4.3 will describe an efficient algorithm to derive the exact distribution of the future PD on any given tree, which can be useful when the number of taxa is moderate or the mild conditions of section 4.2 are not satisfied.

A different problem from deriving its distribution is that of deciding the appropriate actions to ensure that the future PD is large. Here the problem can be formalised in several different ways. First, we need to decide how to model the effect of human actions. Can conservation guarantee a taxon's survival? (Cf. secs. 4.4 and 4.6.) What is the relationship between conservation expenditure and survival probabilities? (Cf. next chapter.) Do conservation actions only increase the survival probability of a single taxon at a time? (Cf. sec. 4.7.)

Second, we need to decide what the objective of conservation actions should be, as the future PD is unpredictable and therefore the aim cannot simply be to maximise it. One possibility, perhaps the simplest, is to aim to maximise the mean of this random variable. This gives rise to different variants of what is known as the Noah's Ark problem [**Wei98, HS06**] and will be dealt with extensively in this chapter from section 4.4 onwards and in the next chapter. A different possibility, which has not been well explored, is that of minimising the probability that the future PD will fall below some given critical value. I will discuss this possible direction for related work in section 4.8, together with the literature relevant to the topics introduced in this chapter.

4.2. A simple extinction model and asymptotic distribution of PD

In a paper that appeared in *Science* in 1997 ([**NM97**], fig. 3), Sean Nee and Robert May showed that the distribution of the PD of 12 taxa sampled randomly from a tree with 64 taxa is strongly dependent on the shape of the tree: as is intuitive, for balanced trees we can expect that a larger portion of the total PD is preserved in this sample than for unbalanced ones (which means that unbalanced phylogenies are more prone to biodiversity losses). However one thing that these distributions had in common was their shape: in both cases presented, they looked remarkably normal.

Motivated by these observations, Beáta Faller and Mike Steel were interested in understanding their generality. On this project I collaborated with them, working mostly on the algorithm I will describe in the next section. Both this algorithm and the theoretical result I will state and discuss in this section appear in a paper we recently published on the *Journal of Theoretical Biology* [**FPS08**].

Nee and May's experiment simulates the effect of a large extinction event in which each taxon's extinction is equally probable and independent of the others. This simple "field of bullets" model [**Rau92, NM97, VG98**], however, fails to take into account the different extinction risks of different taxa (see sec. 4.1). In the model I will consider, which we called the *generalised field of bullets model* [**FPS08**], each taxon s survives until some future time t with a given probability p_s, and again the survivals (and therefore extinctions) of different taxa are treated as independent events (an assumption I will discuss in sec. 4.7). Note that t is a fixed time — for example 5 years or 100 years from now — rather than a continuous variable. The taxa correspond to the leaves in a tree $\mathcal{T}$ with branch lengths. The ones that survive form a random set of leaves whose $\mathrm{PD}_{\mathcal{T}}$ is what I call the *future PD* and denote by φ (or $\varphi_{\mathcal{T}}$ if the tree needs to be explicitly indicated). In order to distinguish whether I am referring to the rooted or unrooted definition of this variable, I will use the variants φ^r and φ^u, respectively.

The key observation to study the distribution of the future PD is that this random variable should be seen as the sum of many (non-independent) random variables, one

for each branch of $\mathcal{T}$. The random variable associated with a branch e contributes to this sum either with its length t_e or with 0, depending on whether or not this branch is part of the restriction of $\mathcal{T}$ on the surviving taxa (plus the root, in the rooted case). Using the fact that the expectation of the sum of random variables is equal to the sum of their expectations, it is not hard to see that [**FPS08**]

$$\mathbb{E}[\varphi^r] = \sum_e t_e \cdot \mathbb{P}[\mathcal{S}_{C(e)}], \tag{4.2.1}$$

$$\mathbb{E}[\varphi^u] = \sum_e t_e \cdot \mathbb{P}[\mathcal{S}_{C(e)}] \cdot \mathbb{P}[\mathcal{S}_{D(e)}],$$

where the sums are over all branches of $\mathcal{T}$ and the following formalisms have been used: if $\mathcal{T}$ is rooted, $C(e)$ denotes the set of taxa that are separated from the root of $\mathcal{T}$ by e; if $\mathcal{T}$ is unrooted, the two sets of taxa on the two sides of e are denoted by $C(e)$ and $D(e)$ (it does not matter which is which). Finally $\mathcal{S}_Y$, where Y is a subset of taxa, denotes the event that at least one taxon in this subset survives. Its probability is easily calculated:

$$\mathbb{P}[\mathcal{S}_Y] = 1 - \prod_{s \in Y} (1 - p_s). \tag{4.2.2}$$

As for the variance of φ, we have that [**FPS08**]

$$\mathrm{Var}[\varphi^r] = \sum_e t_e^2 \cdot \mathbb{P}[\mathcal{S}_{C(e)}] \cdot (1 - \mathbb{P}[\mathcal{S}_{C(e)}]) + 2 \sum_{e,f : C(e) \subset C(f)} t_e \cdot t_f \cdot \mathbb{P}[\mathcal{S}_{C(e)}] \cdot (1 - \mathbb{P}[\mathcal{S}_{C(f)}]),$$

and that

$$\begin{array}{c} \mathrm{Var}[\varphi^u] = \sum_e t_e^2 \cdot \mathbb{P}[\mathcal{S}_{C(e)}] \cdot \mathbb{P}[\mathcal{S}_{D(e)}] \cdot (1 - \mathbb{P}[\mathcal{S}_{C(e)}] \cdot \mathbb{P}[\mathcal{S}_{D(e)}]) + \\ + \sum_{e \neq f} t_e \cdot t_f \cdot \mathbb{P}[\mathcal{S}_{X(e,f)}] \cdot \mathbb{P}[\mathcal{S}_{X(f,e)}] \cdot (1 - \mathbb{P}[\mathcal{S}_{X - X(e,f)}] \cdot \mathbb{P}[\mathcal{S}_{X - X(f,e)}]), \end{array} \tag{4.2.3}$$

where X denotes the set of all the taxa in $\mathcal{T}$ and $X(e,f)$ is the set of taxa in $\mathcal{T}$ that are separated from f by e.

Since the expression for $\mathrm{Var}[\varphi^u]$ has not been described anywhere else in the literature, I will now include its proof.

Proof of Equation (4.2.3). Let $\mathbb{I}[\mathcal{S}_Y]$ denote the random variable indicating whether or not event $\mathcal{S}_Y$ happens, i.e., let $\mathbb{I}[\mathcal{S}_Y] = 0$ if no taxon in Y survives and $\mathbb{I}[\mathcal{S}_Y] = 1$, otherwise. Then $\varphi^u = \sum_e t_e \cdot \mathbb{I}[\mathcal{S}_{C(e)}] \cdot \mathbb{I}[\mathcal{S}_{D(e)}]$. The following derivation uses the fact that if $e \neq f$ then the simultaneous occurrence of the events $\mathcal{S}_{C(e)}$, $\mathcal{S}_{D(e)}$, $\mathcal{S}_{C(f)}$ and $\mathcal{S}_{D(f)}$ is equivalent to the occurrence of events $\mathcal{S}_{X(e,f)}$ and $\mathcal{S}_{X(f,e)}$.

$$\mathrm{Var}[\varphi^u] = \mathbb{E}[(\varphi^u)^2] - \mathbb{E}[\varphi^u]^2 =$$

$$\mathbb{E}\left[\sum_{e,f} t_e \cdot t_f \cdot \mathbb{I}[\mathcal{S}_{C(e)}] \cdot \mathbb{I}[\mathcal{S}_{D(e)}] \cdot \mathbb{I}[\mathcal{S}_{C(f)}] \cdot \mathbb{I}[\mathcal{S}_{D(f)}]\right] -$$

$$\sum_{e,f} t_e \cdot t_f \cdot \mathbb{P}[\mathcal{S}_{C(e)}] \cdot \mathbb{P}[\mathcal{S}_{D(e)}] \cdot \mathbb{P}[\mathcal{S}_{C(f)}] \cdot \mathbb{P}[\mathcal{S}_{D(f)}] =$$

$$\sum_{e,f} t_e \cdot t_f \cdot \left(\mathbb{P}[\mathcal{S}_{C(e)}, \mathcal{S}_{D(e)}, \mathcal{S}_{C(f)}, \mathcal{S}_{D(f)}] - \mathbb{P}[\mathcal{S}_{C(e)}] \cdot \mathbb{P}[\mathcal{S}_{D(e)}] \cdot \mathbb{P}[\mathcal{S}_{C(f)}] \cdot \mathbb{P}[\mathcal{S}_{D(f)}]\right) =$$

$$\sum_e t_e^2 \left(\mathbb{P}[\mathcal{S}_{C(e)}, \mathcal{S}_{D(e)}] - \mathbb{P}[\mathcal{S}_{C(e)}]^2 \cdot \mathbb{P}[\mathcal{S}_{D(e)}]^2\right) +$$
$$\sum_{e \neq f} t_e \cdot t_f \cdot \left(\mathbb{P}[\mathcal{S}_{X(e,f)}, \mathcal{S}_{X(f,e)}] - \mathbb{P}[\mathcal{S}_{X(e,f)}] \cdot \mathbb{P}[\mathcal{S}_{X(f,e)}] \cdot \mathbb{P}[\mathcal{S}_{X-X(e,f)}] \cdot \mathbb{P}[\mathcal{S}_{X-X(f,e)}]\right) =$$
$$\sum_e t_e^2 \cdot \mathbb{P}[\mathcal{S}_{C(e)}] \cdot \mathbb{P}[\mathcal{S}_{D(e)}] \cdot (1 - \mathbb{P}[\mathcal{S}_{C(e)}] \cdot \mathbb{P}[\mathcal{S}_{D(e)}]) +$$
$$\sum_{e \neq f} t_e \cdot t_f \cdot \mathbb{P}[\mathcal{S}_{X(e,f)}] \cdot \mathbb{P}[\mathcal{S}_{X(f,e)}] \cdot (1 - \mathbb{P}[\mathcal{S}_{X-X(e,f)}] \cdot \mathbb{P}[\mathcal{S}_{X-X(f,e)}]).$$

Now that we have formulas for the mean and variance of the future PD, it is natural to wonder about the full distribution of this variable. Faller and Steel were aiming to understand whether this distribution can be approximated with a Gaussian and the conditions under which this is justifiable. They did this by proving an asymptotic result on the normality of the future PD, i.e., in loose terms, by showing that if we imagine a sequence of increasingly large trees with associated branch lengths and taxon survival probabilities satisfying some regularity conditions, then the sequence of distributions corresponding to these trees must become more and more normal-like. As a consequence, for large trees (say, with more than 50 taxa) we expect the future PD to be approximately normally distributed with the mean and variance given above. The precise statement is given by the following theorem.

THEOREM 4.2.1. *Consider an infinite sequence of trees $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n, \ldots$ where, for each $n \geq 1$, $\mathcal{T}_n$ has n leaves. The branches of these trees have non-negative lengths satisfying condition (L) below and the leaves have probabilities satisfying (P). These probabilities are to be interpreted as taxon-specific survival probabilities, defining a generalised field of bullets model for each tree. If we denote by φ_n the future PD of $\mathcal{T}_n$, then the sequence of random variables*

$$\frac{\varphi_n - \mathbb{E}[\varphi_n]}{\sqrt{\mathrm{Var}[\varphi_n]}}$$

converges in distribution to $N(0,1)$ as $n \to \infty$, where $N(0,1)$ denotes a standard normally distributed random variable.

The conditions mentioned in this theorem are:

(P): let $p_s^{(n)}$ denote the probability associated with taxon s in tree $\mathcal{T}_n$. For some $\epsilon > 0$ and for each n, we have

$$\epsilon \leq p_s^{(n)} \leq 1 - \epsilon$$

for all taxa s in $\mathcal{T}_n$ except for at most An^α of them, where A and α are constants, with $0 \leq \alpha < 1/2$.

(L): let $t_e^{(n)}$ denote the length of branch e in tree $\mathcal{T}_n$ and $T(n)$ the length of the longest branch in $\mathcal{T}_n$. Furthermore, let Te_n denote the set of terminal branches in $\mathcal{T}_n$. Then, for each n we have

$$\sum_{e \in \mathrm{Te}_n} \left(\frac{t_e^{(n)}}{T(n)}\right)^2 \geq Bn^\beta$$

for some constants $B > 0$, $\beta > 2\alpha$.

Despite their descriptions being somewhat technical, the ideas behind these conditions are simple. (P) requires that most taxa survive with probabilities away from 0 and 1 (with the exception of a number of them that grows not too rapidly in n). The reason why we need this requirement is intuitive: taxa that are certainly going to become extinct or certainly survive contribute to the future PD with a constant term. Constant terms, however, do not contribute towards producing a normal distribution, which is usually obtained as a sum of a large number of random terms. As for (L), loosely speaking this condition constrains the terminal branches to be, on average, not too short compared to the longest branch in the tree. The reason why we need to impose this is less intuitive, but still understandable: if a small number of branches are much longer than the rest, the future PD will be largely determined by which of these branches will still be present in the tree connecting the surviving taxa, and this will produce a multimodal distribution: for example it is intuitive that if most of the tree length is concentrated in one branch e, the distribution of future PD will have two peaks corresponding to whether or not event $\mathcal{S}_{C(e)}$ occurs. It can be shown that condition (L) is expected to hold for many realistic trees, for example those produced by a Yule model [**FPS08**].

As for the proof of Theorem 4.2.1, we have described it in detail elsewhere [**FPS08**]. Because of its length and because I did not give a significant contribution towards it, I will not report it here. The main idea is to partition $\mathcal{T}_n$ into a number of subtrees. One of them comprises all the branches e with a large $C(e)$ — large enough to be practically certain that at least one taxon from $C(e)$ will survive. The contribution to the future PD coming from this part of the tree converges to a non-random function of n. The remaining subtrees contribute to the future PD each with an independent random component and although these are not identically distributed, it is possible to a apply a standard central limit theorem that shows that their sum converges in distribution to a normal.

4.3. An algorithm to derive the distribution of PD

Theorem 4.2.1 reduces the problem of computing the distribution of the future PD to that of determining just two parameters — its mean and variance — which can be easily computed with the formulae shown in the last section. However this theorem is an asymptotic result and so normality can only be expected for large trees. Given the increasing trend in biology towards constructing and analysing phylogenetic trees that contain large numbers of taxa, this is not particularly restrictive. Another limitation of this theorem is its requirement that conditions (P) and (L) be satisfied. Although, again, there are no strong reasons to expect these conditions to be violated in practice, it may be useful to have an efficient algorithm for calculating the distribution of φ on any given tree. Of course, an approximation to this distribution can also be obtained by simulation — by repeatedly sampling random subsets

of taxa, according to the given generalised field of bullets model, and recording their PD.

This section provides an exact algorithm for the distribution of the future PD. I make the simplifying assumption that the branch lengths are non-negative integer-valued, which implies that φ can only have values in the set $\{0, 1, \ldots, L\}$, where $L = \sum_e t_e$ is the total branch length of the input tree. This assumption is not problematic in practice, as we can rescale all the branch lengths so that they are arbitrarily close to integer multiples of some small value (in doing so we can approximate the correct distribution within any desired precision, as detailed at the end of this section).

As in section 3.4, I assume that the input tree is rooted with exactly one branch adjacent to its root and that the tree is bifurcating. These assumptions do not affect the generality of the algorithm, as any tree (including unrooted trees) can be modified to satisfy them without changing the distribution for φ: one can resolve multifurcations arbitrarily and possibly move the root at the top of a new branch, always assigning length 0 to the newly introduced branches.

Let e be an arbitrary branch of the input tree. In the following φ_e^r denotes the contribution to φ^r that comes from e and the branches separated from the root by e (i.e., the contribution from the clade rooted in e). Similarly, φ_e^u denotes the uPD of the taxa surviving in $C(e)$. Then, for any integer k, define

$$\begin{aligned} f_e(k) &= \mathbb{P}[\varphi_e^r = k,\, \mathcal{S}_{C(e)}], \\ f'_e(k) &= \mathbb{P}[\varphi_e^u = k,\, \mathcal{S}_{C(e)}], \\ q_e = \mathbb{P}[\overline{\mathcal{S}}_{C(e)}] &= \prod_{s \in C(e)} (1 - p_s), \end{aligned}$$

where $C(e)$ and $\mathcal{S}_{C(e)}$ are defined as before and $\overline{\mathcal{S}}_{C(e)}$ denotes the complement of $\mathcal{S}_{C(e)}$, i.e., the event that none of the taxa in $C(e)$ survives.

The objective of the algorithm is to derive these quantities for the root branch ρ (and the necessary values of k; see below), as they are all that is needed to describe the distribution of φ, both in the rooted and unrooted case. In fact, simply observe that the distributions of φ^r and φ^u are given by:

$$\mathbb{P}[\varphi^r = k] = \mathbb{P}[\varphi_\rho^r = k,\, \mathcal{S}_{C(\rho)}] + \mathbb{P}[\varphi_\rho^r = k,\, \overline{\mathcal{S}}_{C(\rho)}] = f_\rho(k) + q_\rho \cdot I_{k=0},$$

and

$$\mathbb{P}[\varphi^u = k] = \mathbb{P}[\varphi_\rho^u = k,\, \mathcal{S}_{C(\rho)}] + \mathbb{P}[\varphi_\rho^u = k,\, \overline{\mathcal{S}}_{C(\rho)}] = f'_\rho(k) + q_\rho \cdot I_{k=0},$$

where $I_{k=0}$ equals 0 or 1 depending on the proposition displayed as subscript, in this case $k = 0$, being false or true, respectively.

The algorithm consists of doing a bottom-up traversal of all the branches, so that each time an branch e is visited, one calculates the values of q_e, of $f_e(k)$ in the range $k \in \{t_e, t_e + 1, \ldots, L\}$, and, if we are also interested in the distribution of φ^u, of $f'_e(k)$ in the range $k \in \{0, 1, \ldots, L\}$. Note that outside the specified ranges $f_e(k)$

and $f'_e(k)$ are equal to 0. The calculation of these values is performed in a recursive way, by using the values that have been calculated for previously visited branches. For q_e this is trivial, and for $f_e(k)$ and $f'_e(k)$ the recursions described in the next two subsections must be used. Once these values are derived for the root branch, the distributions of φ^r and φ^u can be calculated in the way described above.

4.3.1. Recursion for $f_e(k)$.

- If e leads into leaf s, then

$$f_e(k) \;=\; \mathbb{P}[\varphi_e^r = k,\, \mathcal{S}_{\{s\}}] \cdot I_{k=t_e} \;=\; p_s \cdot I_{k=t_e}.$$

- If the non-root side of e is an internal node, let c and d be the other two branches adjacent to it. Then

$$f_e(k) = \sum_{i=t_c}^{k-t_e-t_d} f_c(i) \cdot f_d(k-t_e-i) + q_d \cdot f_c(k-t_e) + q_c \cdot f_d(k-t_e). \tag{4.3.1}$$

Note that if $t_e \le k < t_c + t_e$, the term $f_c(k-t_e)$ used in (4.3.1) has not been calculated before; the same applies to $f_d(k-t_e)$, if $t_e \le k < t_d + t_e$. In these cases, the algorithm assumes that these values are 0, which ensures that there is no need to calculate and store $f_e(k)$ for k outside the range $\{t_e, t_e+1, \ldots, L\}$. In fact this range can be restricted further, as described in subsection 4.3.3.

Equation (4.3.1) is easily proved: we have

$$\begin{aligned} f_e(k) \;&=\; \mathbb{P}[\varphi_e^r = k,\, \mathcal{S}_{C(c)},\, \mathcal{S}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi_e^r = k,\, \mathcal{S}_{C(c)},\, \overline{\mathcal{S}}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi_e^r = k,\, \overline{\mathcal{S}}_{C(c)},\, \mathcal{S}_{C(d)}] \\ &=\; \mathbb{P}[\varphi_c^r + \varphi_d^r = k - t_e,\, \mathcal{S}_{C(c)},\, \mathcal{S}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi_c^r = k - t_e,\, \mathcal{S}_{C(c)},\, \overline{\mathcal{S}}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi_d^r = k - t_e,\, \overline{\mathcal{S}}_{C(c)},\, \mathcal{S}_{C(d)}]. \end{aligned}$$

Thus, using the independence between the survival events in $C(c)$ and $C(d)$,

$$\begin{aligned} f_e(k) \;&=\; \sum_{i=0}^{k-t_e} \mathbb{P}\left[\varphi_c^r = i,\, \mathcal{S}_{C(c)}\right] \cdot \mathbb{P}\left[\varphi_d^r = k - t_e - i,\, \mathcal{S}_{C(d)}\right] + \\ &\quad + \mathbb{P}[\varphi_c^r = k - t_e,\, \mathcal{S}_{C(c)}] \cdot \mathbb{P}[\overline{\mathcal{S}}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi_d^r = k - t_e,\, \mathcal{S}_{C(d)}] \cdot \mathbb{P}[\overline{\mathcal{S}}_{C(c)}] \\ &=\; \sum_{i=t_c}^{k-t_e-t_d} f_c(i) \cdot f_d(k-t_e-i) + q_d \cdot f_c(k-t_e) + q_c \cdot f_d(k-t_e). \end{aligned}$$

Note that the range of the sum is reduced from $i \in \{0, 1, \ldots, k-t_e\}$ to $i \in \{t_c, t_c+1, \ldots, k-t_e-t_d\}$ as for the remaining values of i, either $f_c(i) = 0$ or $f_d(x-t_e-i) = 0$.

4.3.2. Recursion for $f'_e(k)$.

- If e leads into leaf s, then

$$f'_e(k) \;=\; p_s \cdot I_{k=0}.$$

- If the non-root side of e is an internal node, let c and d be the other two branches adjacent to it. Then

$$f'_e(k) = \sum_{i=t_c}^{k-t_d} f_c(i) \cdot f_d(k-i) + q_d \cdot f'_c(k) + q_c \cdot f'_d(k), \tag{4.3.2}$$

which is proved in a way similar to (4.3.1):

$$\begin{aligned} f'_e(k) &= \mathbb{P}[\varphi^u_e = k,\, \mathcal{S}_{C(c)},\, \mathcal{S}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi^u_e = k,\, \mathcal{S}_{C(c)},\, \overline{\mathcal{S}}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi^u_e = k,\, \overline{\mathcal{S}}_{C(c)},\, \mathcal{S}_{C(d)}] \\ &= \mathbb{P}[\varphi^r_c + \varphi^r_d = k,\, \mathcal{S}_{C(c)},\, \mathcal{S}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi^u_c = k,\, \mathcal{S}_{C(c)},\, \overline{\mathcal{S}}_{C(d)}] \\ &\quad + \mathbb{P}[\varphi^u_d = k,\, \overline{\mathcal{S}}_{C(c)},\, \mathcal{S}_{C(d)}] \\ &= \sum_{i=t_c}^{k-t_d} f_c(i) \cdot f_d(k-i) + q_d \cdot f'_c(k) + q_c \cdot f'_d(k). \end{aligned}$$

4.3.3. Efficiency considerations. Let L_e be the sum of the lengths of all the branches separated from the root by branch e, including e itself. Clearly the maximum values that φ^r_e and φ^u_e can attain are L_e and $L_e - t_e$, respectively. A more efficient version of the algorithm than that outlined above can be obtained by restricting the calculation of $f_e(k)$ to the values of $k \in \{t_e, t_e+1, \ldots, L_e\}$, and those of $f'_e(k)$ to the values of $k \in \{0, 1, \ldots, L_e - t_e\}$. Note that the sum in (4.3.1) can then be further restricted to the values of i such that $i \le L_c$ and $k - t_e - i \le L_d$ and the sum in (4.3.2) to the values of i such that $i \le L_c$ and $k - i \le L_d$.

In order to analyse the computational complexity of this improved algorithm, it helps to think of the $f_e(k)$ and $f'_e(k)$ values as organised in branch-associated vectors, which I denote by $\mathbf{f}_e$ and $\mathbf{f}'_e$. If e is a terminal branch, these vectors have length 1 and their calculation can be done in constant time. Otherwise, if we call c and d the branches attached to the non-root extreme of e, then $\mathbf{f}_e$ and $\mathbf{f}'_e$ are obtained from the corresponding vectors for c and d in the way described by equations (4.3.1) and (4.3.2). It is not hard to see that the summations in (4.3.1) and (4.3.2) mean that each pair of elements from $\mathbf{f}_c$ and $\mathbf{f}_d$ are multiplied together to contribute towards some elements of $\mathbf{f}_e$ and $\mathbf{f}'_e$ — which amounts to $O(L_c L_d)$ multiplications. The remaining operations to calculate $\mathbf{f}_e$ and $\mathbf{f}'_e$ — the ones involving the terms containing q_c and q_d in (4.3.1) and (4.3.2) — can be done in a time proportional to the length of these vectors — that is in $O(L_e - t_e) = O(L_c + L_d)$ time. Therefore, for any internal branch

e, the calculation of $\mathbf{f}_e$ and $\mathbf{f}'_e$ can be done in $O(L_c L_d)$ time. Proposition 4.3.1 below shows that the sum of all the $L_c L_d$ terms, for all sister branches c and d, is bounded above by L^2. This quantity, combined with the constant times needed to process each branch, gives a total running time for the entire procedure of $O(n + L^2)$. Since typically every pair of taxa in the tree is separated by at least one branch of positive length, we have that $n = O(L)$ and therefore the running time above is equivalent to $O(L^2)$.

PROPOSITION 4.3.1. *Adopting the notations and assumptions of the rest of this section, and indicating by $c(e)$ and $d(e)$ the two branches that are adjacent to the non-root side of e,*

$$\sum_e L_{c(e)} L_{d(e)} \leq L^2,$$

where the sum is over all internal branches of the input tree (including ρ, the branch adjacent to its root).

PROOF. By induction on n, the number of leaves in the input tree.
If $n = 2$, then, trivially, $\sum_e L_{c(e)} L_{d(e)} = L_{c(\rho)} L_{d(\rho)} \leq L^2$.
If $n > 2$, then

$$\sum_e L_{c(e)} L_{d(e)} \leq L_{c(\rho)} L_{d(\rho)} + L_{c(\rho)}^2 + L_{d(\rho)}^2 \leq (L_{c(\rho)} + L_{d(\rho)})^2 \leq L^2,$$

where the first inequality is obtained by applying the inductive hypothesis to the subtrees rooted in $c(\rho)$ and in $d(\rho)$. □

Regarding memory requirements, note that each time we calculate the information relative to e (namely q_e, $\mathbf{f}_e$ and $\mathbf{f}'_e$), the information relative to the branches on its non-root side (if any) can be deleted, as it will never be used again. So, at any given moment the information of at most n "active" branches needs to be stored. (In practice the maximum number of active branches could be brought down to $O(\log n)$ by organising the bottom-up traversal so that for each branch its larger subtree is always traversed first, but I do not describe this here.) If we use the range restriction just described, the sizes of the $\mathbf{f}_e$ and $\mathbf{f}'_e$ vectors for all the active branches sum to a number bounded above by $2(n + L)$, and therefore the algorithm requires $O(n + L)$ space, equivalent to $O(L)$ in practice.

Regarding the assumption that the branch lengths be expressed as integer multiples of some fixed unit, note that in some cases this unit may need to be very small, thus potentially causing L to be very large and the algorithm very inefficient. In these cases it is possible to produce instead an arbitrarily precise *approximation* of the distribution of the future PD: if we round each branch length to the nearest integral multiple of x/n, then

$$|\varphi' - \varphi| \leq \sum_e |t'_e - t_e| \leq \sum_e \frac{x}{2n} < x,$$

where t'_e is the rounded length of branch e and φ' is the rounded future PD. In other words, we can achieve any desired precision x by re-expressing each branch length as a multiple of x/n. Since L grows linearly with n/x, the algorithm is still polynomial, both in n and $1/x$.

4.4. A simple problem where conservation ensures survival and its solution

A different problem from deriving the distribution of the future phylogenetic diversity is that of deciding appropriate actions to preserve it. As I will show in the rest of this chapter and in the next one, there are several ways to formalise this problem. Roughly speaking, I will assume a generalised field of bullets model (see sec. 4.2) in which the probabilities of survival p_s associated with each taxon s can be raised at some cost. The simplest scenario is that each taxon s can be conserved at a given cost c_s, and this *ensures* the survival of s. If a taxon is not conserved, its probability of survival remains at a specified a_s. In other words, the probability of survival of each taxon s can be raised from an initial a_s to 1 at a cost of c_s. This scenario is relatively realistic, describing for example the situation that taxa can be saved by simply conserving a few of their individuals, e.g., in a seed bank or zoo.

The effect on the future PD caused by raising the probabilities of survival of some taxa can be summarised by looking at the increase in the expected value of the future PD. In the rest of this section, $\mathbb{E}[\varphi|S]$ denotes the expectation of the future PD under the assumption that S is the set of conserved taxa, in other words under the generalised field of bullets model where the probabilities of survival are given by

$$p_s = \begin{cases} a_s & \text{if } s \notin S, \\ 1 & \text{if } s \in S. \end{cases} \tag{4.4.1}$$

In the following problem — and most of the ones that will be discussed later — the aim is to make the choices that maximise $\mathbb{E}[\varphi|S]$. In summary (the rationale behind the problem's name will become clear later in the thesis):

$a_s \xrightarrow{c_s} 1$ **NAP:**

Input: A tree $\mathcal{T}$ with non-negative lengths t_e associated with each branch e. Integer costs c_s and probabilities a_s associated with each leaf s ($c_s \geq 0$ and $0 \leq a_s \leq 1$). A budget $B \geq 0$.

Output: A subset S of the leaves of $\mathcal{T}$ that

$$\text{maximises } \mathbb{E}[\varphi|S],$$
$$\text{subject to } \sum_{s \in S} c_s \leq B.$$

An example of the rooted $a_s \xrightarrow{c_s} 1$ NAP will be discussed in section 4.5.

The key observation to solve the rooted version of this problem is given by Proposition 4.4.1: for any instance of the $a_s \xrightarrow{c_s} 1$ NAP, it is possible to transform the input

tree $\mathcal{T}$ into a new tree $\mathcal{T}'$ so that the gain in $\mathbb{E}[\varphi^r]$ obtained by conserving the taxa in S coincides with the rPD of S in $\mathcal{T}'$, for any subset S of taxa. As a consequence, any $a_s \xrightarrow{c_s} 1$ NAP is equivalent to a BMaxPD applied to the transformed tree $\mathcal{T}'$. We can solve the latter with any of the algorithms described in the previous chapter.

Proposition 4.4.1. *Given a tree $\mathcal{T}$ with branch-associated lengths t_e and leaf-associated probabilities a_s defining the generalised field of bullets model specified in 4.4.1, we have, for any subset S of leaves*

$$\mathbb{E}[\varphi^r|S] - \mathbb{E}[\varphi^r|\emptyset] \;=\; \mathrm{rPD}_{\mathcal{T}'}(S),$$

where $\mathcal{T}'$ is a tree with the same topology as $\mathcal{T}$ and branch lengths t'_e such that $t'_e = t_e \prod_{s\in C(e)}(1-a_s)$.

Proof. Using (4.2.1) and (4.2.2), it is easy to see that

$$\mathbb{E}[\varphi^r|S] - \mathbb{E}[\varphi^r|\emptyset] \;=$$

$$\sum_{e:C(e)\cap S=\emptyset} t_e \left(1 - \prod_{s\in C(e)}(1-a_s)\right) + \sum_{e:C(e)\cap S\neq\emptyset} t_e \;- \sum_e t_e \left(1 - \prod_{s\in C(e)}(1-a_s)\right) =$$

$$\sum_{e:C(e)\cap S\neq\emptyset} t_e \prod_{s\in C(e)}(1-a_s) \;=\; \mathrm{rPD}_{\mathcal{T}'}(S).$$

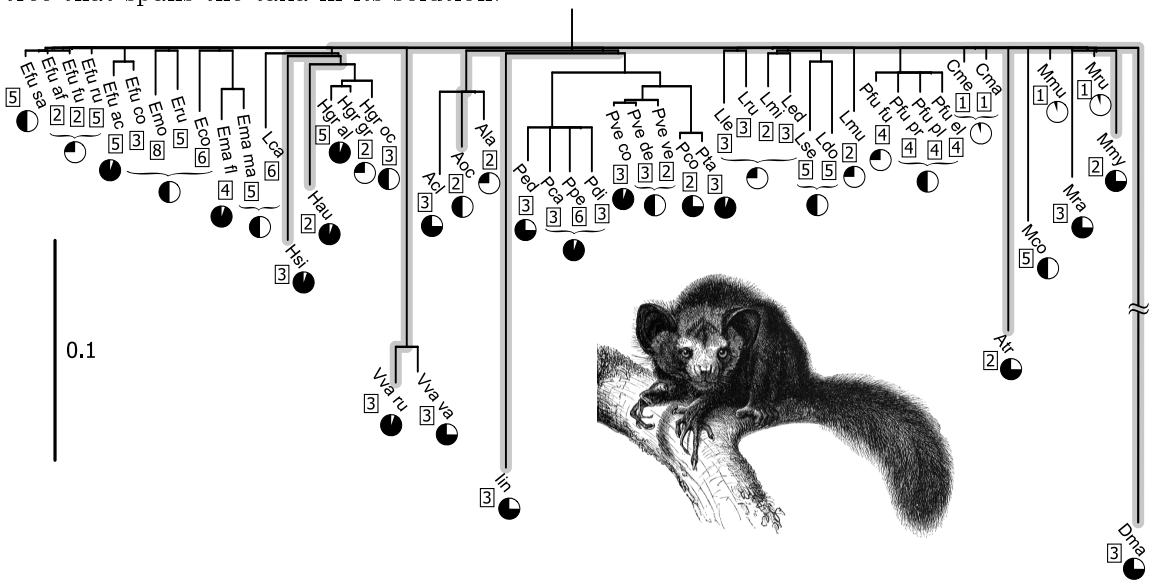
□

For example, figure 4.4.1 shows the tree $\mathcal{T}'$ that is obtained from the tree $\mathcal{T}$ in figure 3.7.1 by taking into account the survival probabilities indicated in the circles in figure 4.4.1 (see legend).

The transformation result therefore proves that any subset of taxa S that maximises $\mathbb{E}[\varphi^r|S]$ on the input tree $\mathcal{T}$ subject to any given constraint also maximises rPD on $\mathcal{T}'$ subject to the same constraint, and vice versa. Therefore, if the definition of PD is rooted, any $a_s \xrightarrow{c_s} 1$ NAP can be solved by first multiplying each branch length t_e by a factor equal to $\prod_{s\in C(e)}(1-a_s)$ and then applying one of the algorithms described in the previous chapter to the BMaxPD with the same costs and budget.

If instead the definition of PD is unrooted, the $a_s \xrightarrow{c_s} 1$ NAP can be solved using the same idea as in section 3.6: its solution is either empty or equal to $\{r\} \cup R$, where r is a particular taxon and R is a solution to the rooted $a_s \xrightarrow{c_s} 1$ NAP on the input tree rooted in r and with budget $B - c_r$. Therefore, the problem can be solved by rooting the tree in each taxon in turn; for each rooting, transform the tree in the way described in Proposition 4.4.1 (this transformation is dependent on the position of the root, so we will get a different tree each time) and solve the resulting rooted BMaxPD with budget $B-c_r$. Denoting by R_i the solution obtained when rooting in taxon r_i, then a solution to the unrooted $a_s \xrightarrow{c_s} 1$ NAP is among $\{r_1\} \cup R_1, \{r_2\} \cup R_2, \ldots, \{r_n\} \cup R_n$ and can be found by simply comparing their $\mathbb{E}[\varphi^u]$. Note that the time complexity of this algorithm will now be $O(n^2B^2)$, as BMaxPD will have to be solved on n different trees.

FIGURE 4.4.1. Phylogenetic tree obtained from the one in fig. 3.7.1 by applying the transformation described in Proposition 4.4.1.
For each taxon s, its cost c_s is indicated by the adjacent box and its probability of survival a_s by the white area in the adjacent circle. The probabilities of survival were derived from the IUCN Red List classifications [**IUC06**]: taxa classified in risk categories CR, EN, VU, NT, LC were given probabilities 5%, 25%, 50%, 75%, and 95%, respectively. The tree in fig. 3.7.1 and the probabilities indicated here define a $a_s \overset{c_s}{\to} 1$ NAP (the budget is assumed to be $B = 20$). Highlighted is the tree that spans the taxa in its solution.



4.5. Examples

Figure 4.4.1 shows the tree $\mathcal{T}'$ that is obtained from the tree $\mathcal{T}$ in figure 3.7.1, by applying the transformation described in the previous section. It is interesting to reflect on the intuitive meaning of this transformation. The new tree is obtained by multiplying each branch length t_e by $\prod_{s \in C(e)}(1 - a_s)$, which is the probability that, as a result of extinctions, that branch will disappear from the tree connecting the surviving taxa, if no conservation action is taken. Therefore, we can think of this as a form of weighting that gives more importance to those parts in the tree that are more likely to get lost. For example, in the tree in figure 4.4.1, most of the internal branches have a length that is very close to 0. This is because these branches have many taxa below them and are therefore unlikely to be lost. Contrast this with the internal branch leading to the two Vva subspecies: this branch is relatively long, as both Vva ru and Vva va are likely to become extinct.

The relevance of the tree in figure 4.4.1 lies in the fact that — assuming that the given probabilities of survival are realistic — if the aim is to preserve the future PD then Madagascar lemurs should be chosen for conservation on the basis of their PD in *this* tree, rather than in their real phylogenetic tree (without branch rescaling). If we seek a solution with maximum expected future PD, the $a_s \overset{c_s}{\to} 1$ NAP on the

tree of figure 3.7.1 is equivalent to BMaxPD on the tree in figure 4.4.1. Solving this problem (with budget $B = 20$) with one of the algorithms of the last chapter leads to taking the taxa in the set {Hsi, Hau, Vva ru, Aoc, Iin, Atr, Mmy, Dma}. Again, this solution cannot be obtained through a greedy step from the one for $B = 19$, but needs exchange of one taxon (Mra) for two other taxa (Aoc and Mmy).

4.6. The Noah's Ark problem

When I first proposed the optimisation problems discussed so far (and devised algorithms for them), I was unaware that a more general problem formalising the issues of choice in biodiversity conservation had already been introduced in the economics literature by Martin Weitzman [**Wei98**]. This problem, which he suggestively dubbed the "Noah's Ark problem" (NAP), is receiving increasing attention especially in environmental and ecological economics [**SMG+03, RMSG+03, vdHvdBvI05**].

The NAP is more general than the problems above in two respects. The first of these is the definition of biodiversity that it uses. In the original formulation of the NAP [**Wei98**], each species identifies a collection of biological characteristics that that species exhibits; the diversity of a set of species is defined as the total number of characteristics that are represented in that set. (Weitzman likes to compare species to libraries; the diversity of a set of libraries is the total number of different books present in at least one of those libraries.) If we assume that the characteristics are acquired (at most once and never lost) via an evolutionary process following a phylogenetic tree $\mathcal{T}$, the diversity thus defined is equal to the rPD on $\mathcal{T}$, where each branch length is set to the number of characteristics acquired along that branch. Incidentally, if instead we do not make these assumptions, it is still possible to see Weitzman's diversity as coinciding with the rPD on a suitably defined phylogenetic network (see [**MKvH08, SNM08**] for a definition of PD on networks). In the following, however, because differences and similarities among species are naturally represented by a tree, I do assume that the Weitzman diversity of a set of species is equal to its rooted PD on a given tree. As a consequence, the objective function of the NAP — originally the expected diversity of the surviving species — coincides here with the expected future rooted PD as defined in section 4.2. In the following I will drop the superscript r from φ^r and assume, unless otherwise stated, a rooted definition for φ.

The second, more important, aspect that makes the NAP more general than the approaches I have presented is that conservation does not ensure survival. In Weitzman's original formulation the probability of survival can be raised from a_s to b_s ($0 \leq a_s \leq b_s \leq 1$) at a specified cost c_s. More precisely, for each species s, the probability of survival p_s increases as a linear function of the conservation expenditure x_s on s from a minimum of a_s (for $x_s = 0$) to a maximum of b_s (for

$x_s \geq c_s$). That is, for $0 \leq x_s \leq c_s$, the following holds:

$$p_s = a_s + \frac{x_s}{c_s}(b_s - a_s). \tag{4.6.1}$$

In this formulation, the problem is to determine the optimal expenditures $\{x_s\}$ for all species s, not a subset of species to conserve. However, it turns out that optimal solutions to the NAP are "extreme", i.e., for every species s (with the possible exception of one species), either nothing or the whole c_s is spent on it [**Wei98**]. In order to see this, I express the expected future PD as a function of the expenditures on two species s and r. The result can be shown to have a very simple form:

$$\mathbb{E}[\varphi|x_s, x_r] = K_0 + K_1 x_s + K_2 x_r - K_3 x_s x_r, \tag{4.6.2}$$

where the K_i are non-negative constants.

If the optimal solution to the NAP involved two species s and r having $x_s \notin \{0, c_s\}$ and $x_r \notin \{0, c_r\}$, it is easy to see from (4.6.2) that expenditure could be moved from one species to the other until either $x_s \in \{0, c_s\}$ or $x_r \in \{0, c_r\}$, so that the new value for $\mathbb{E}[\varphi|x_s, x_r]$ would be greater or equal to the old one. This shows that at least one optimal solution to the NAP must be extreme.

As a consequence, it becomes reasonable to cast the NAP as a subset selection problem, an approach which also has the advantage of making the problems discussed so far — MaxPD, BMaxPD and the $a_s \xrightarrow{c_s} 1$ NAP — particular cases of the problem I will formulate. It has to be noted, however, that the formulation I will give here — also given previously in the literature [**HS06, PG07**] — is not entirely equivalent to that of Weitzman, an issue that will be discussed in more detail at the end of this section. In practice, however, it is reasonable to expect the solutions to the two problems to be rather similar in most realistic instances.

The formulation is very similar to that given for the $a_s \xrightarrow{c_s} 1$ NAP: again $\mathbb{E}[\varphi|S]$ denotes the expected future PD under the assumption that S is the set of conserved taxa, but now a "conserved" taxon survives with a given probability b_s. In other words I assume a generalised field of bullets model where the probabilities of survival are given by

$$p_s = \begin{cases} a_s & \text{if } s \notin S, \\ b_s & \text{if } s \in S. \end{cases}$$

$a_s \xrightarrow{c_s} b_s$ **NAP:**

Input: A tree $\mathcal{T}$ with non-negative lengths t_e associated with each branch e. Integer costs c_s and probabilities a_s, b_s associated with each leaf s ($c_s \geq 0$ and $0 \leq a_s \leq b_s \leq 1$). A budget $B \geq 0$.

Output: A subset S of the leaves of $\mathcal{T}$ that

$$\text{maximises } \mathbb{E}[\varphi|S],$$

$$\text{subject to } \sum_{s \in S} c_s \leq B.$$

It should now be clear why I named the $a_s \xrightarrow{c_s} 1$ NAP (sec. 4.4) in that way: it is simply the special case of the $a_s \xrightarrow{c_s} b_s$ NAP where $b_s = 1$ for every taxon s. Similarly, BMAXPD is the special case of the $a_s \xrightarrow{c_s} b_s$ NAP where $a_s = 0$ and $b_s = 1$ for every s. This is because if we set p_s to 1 or 0 depending on whether $s \in S$ or not, respectively, then we have $\varphi = \mathrm{PD}(S)$ and therefore also the objective function $\mathbb{E}[\varphi|S] = \mathrm{PD}(S)$. BMAXPD therefore coincides with the $0 \xrightarrow{c_s} 1$ NAP. Finally, it is easy to see that MAXPD (chapter 2) coincides with the $0 \xrightarrow{1} 1$ NAP. More generally, the $0 \xrightarrow{c} 1$ NAP — where all costs are equal to a constant c — is equivalent to MAXPD once we set the parameter k to the integer part of B/c.

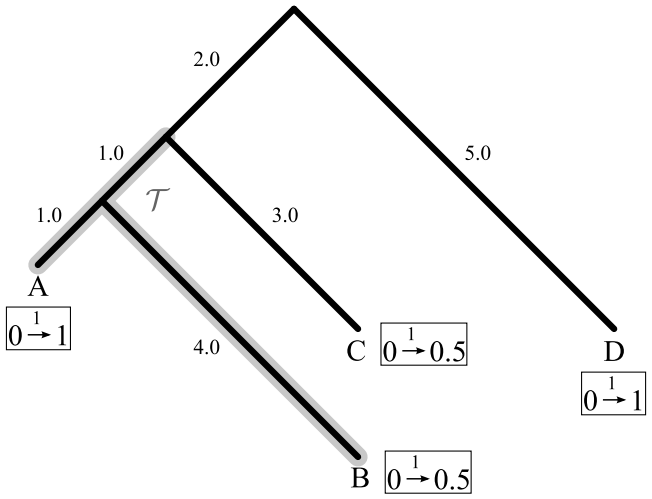
The problems I have dealt with so far can then be seen as belonging to a hierarchy of subproblems of the $a_s \xrightarrow{c_s} b_s$ NAP. Chapter 2 showed that the $0 \xrightarrow{c} 1$ NAP can be solved with a greedy algorithm; Chapter 3 presented a dynamic programming algorithm for the $0 \xrightarrow{c_s} 1$ NAP; and this chapter (sec. 4.4) showed that the $a_s \xrightarrow{c_s} 1$ NAP can also be solved with dynamic programming techniques via transformation into the $0 \xrightarrow{c_s} 1$ NAP. The transformation idea in section 4.4 also means that the $a_s \xrightarrow{c} 1$ NAP can be solved with greedy techniques: an instance of this problem on a tree $\mathcal{T}$ is equivalent to an instance of the $0 \xrightarrow{c} 1$ NAP on the transformed tree $\mathcal{T}'$ of Proposition 4.4.1, which is solvable with the greedy algorithm of chapter 2.

And what about the $a_s \xrightarrow{c_s} b_s$ NAP itself?

The next chapter will show an approach that can be used to solve a further generalisation of this problem. For the moment I would like to note that the transformation idea of section 4.4 can be adapted to reduce the $a_s \xrightarrow{c_s} b_s$ NAP to a $0 \xrightarrow{c_s} b'_s$ NAP. Constructing $\mathcal{T}'$ as before, a similar proof (omitted) to that of Proposition 4.4.1 shows that solving any $a_s \xrightarrow{c_s} b_s$ NAP on $\mathcal{T}$ is equivalent to solving the corresponding $0 \xrightarrow{c_s} (b_s - a_s)/(1 - a_s)$ NAP on $\mathcal{T}'$. Therefore, all the difficulty of the $a_s \xrightarrow{c_s} b_s$ NAP is somehow already present in the $0 \xrightarrow{c_s} b_s$ NAP.

It is also worth noting that it does not seem straightforward to apply a dynamic programming approach to the $a_s \xrightarrow{c_s} b_s$ NAP (or the $0 \xrightarrow{c_s} b_s$ NAP), as the optimal substructure property described for BMAXPD in section 3.4 does not hold for this problem. As an example of this, consider the instance in figure 4.6.1. It is easy to check (see figure legend) that the solution here is {A,D}. If the optimal substructure property were valid, the part of this solution relative to any clade, for example the $\mathcal{T}$ highlighted in gray, should also be optimal with respect to that clade and the expenditure in that clade. That is, {A} should be optimal for $\mathcal{T}$ and the budget $b = 1$. However this subproblem has {B} as its solution instead: in $\mathcal{T}$, $\mathbb{E}[\varphi|\{\mathrm{A}\}] = 2$ whereas $\mathbb{E}[\varphi|\{\mathrm{B}\}] = 2.5$. This shows that global solutions to the $a_s \xrightarrow{c_s} b_s$ NAP may be composed of solutions that are suboptimal for their clades (in this case $\mathcal{T}$). Intuitively, this happens here because, rather than having a large future PD in $\mathcal{T}$, it is more important that at least one taxon in $\mathcal{T}$ will survive, as this means that the length of the branch from the root of the input tree to the root of $\mathcal{T}$ (length 2.0) will

FIGURE 4.6.1. The $a_s \overset{c_s}{\to} b_s$ NAP does not have optimal substructure.
An instance of the problem. The budget is $B = 2$. Branch lengths are indicated by the numbers to the side of the branches. The box near each taxon s indicates its associated cost and probabilities in the format $a_s \overset{c_s}{\to} b_s$. Clade $\mathcal{T}$ is highlighted in gray. The optimal solution is {A,D}, as $\mathbb{E}[\varphi|\{A,D\}] = 9$, and the other feasible solutions have a smaller value of the objective function: $\mathbb{E}[\varphi|\{A,B\}] = 6$, $\mathbb{E}[\varphi|\{A,C\}] = 5.5$, $\mathbb{E}[\varphi|\{B,C\}] = 5.5$, $\mathbb{E}[\varphi|\{B,D\}] = 8.5$, $\mathbb{E}[\varphi|\{C,D\}] = 7.5$.



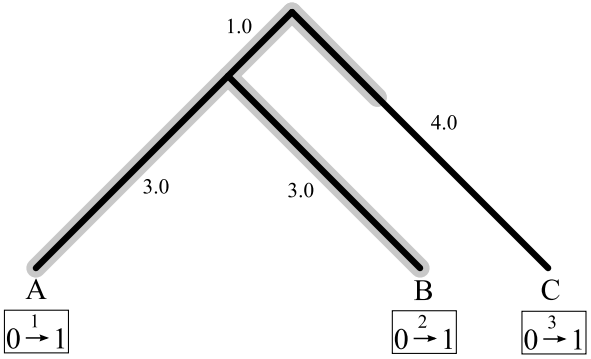
count towards the global future PD. Had this branch been shorter (with a length smaller than 1), the solution to this instance would have been {B,D}.

The tradeoff between ensuring a large contribution to φ coming from a clade and ensuring a high probability of survival of some taxon in that clade will be a central concept of the next chapter.

Finally I note that one of the aspects under which the formulation of the $a_s \overset{c_s}{\to} b_s$ NAP differs from that of Weitzman is that the objective function I adopted does not include a species' utility term quantifying "how much we like or value the existence of a species" [**Wei98**]. However the generality of the problem is not affected by this omission, as species' utilities can be taken into account in the $a_s \overset{c_s}{\to} b_s$ NAP by increasing the length of each terminal branch by a quantity equal to the utility of the species it leads to (as already discussed in section 2.10). The real difference between my formulation and the original one is the subject of the next subsection.

4.6.1. Relation between the $a_s \overset{c_s}{\to} b_s$ NAP and Weitzman's NAP. The difference between the formulation given here for the NAP — the $a_s \overset{c_s}{\to} b_s$ NAP above — and the original formulation given by Weitzman [**Wei98**] is essentially that the $a_s \overset{c_s}{\to} b_s$ NAP does not allow the spending of intermediate amounts of resources between 0 and c_s on a taxon s. On the other hand, Weitzman's NAP defines taxon-specific expenditures x_s that translate into survival probabilities in the way described by (4.6.1). In this section, Weitzman's NAP will be called the *fractional* NAP, as it

FIGURE 4.6.2. The fractional and non-fractional solutions to the NAP are not simply related.
An instance of the problem. The budget is $B = 4$. Branch lengths are indicated by the numbers to the side of the branches. The box near each taxon s indicates its associated cost and probabilities in the format $a_s \overset{c_s}{\to} b_s$. Highlighted in gray is a representation of the optimal solution for the fractional problem: since C survives with probability 1/3, only a third of the branch leading to C is highlighted.



allows a taxon to be conserved "fractionally" by neither spending the minimum nor the maximum allowed; conversely the $a_s \overset{c_s}{\to} b_s$ NAP will be called the *non-fractional* NAP.

An apparent link between the two versions of the NAP is that, as I have already shown, optimal solutions to the fractional NAP must have all expenditures x_s, except (possibly) one, at the extremes of the allowed interval $[0, c_s]$. In other words they are inherently non-fractional, with the exception of one taxon which is conserved fractionally. Because of this, one may be tempted to think that the solutions to the fractional and non-fractional problem will simply differ by the taxon conserved fractionally. However, this is not the case. Consider the instance in figure 4.6.2, where the budget is $B = 4$. The solution to the fractional problem consists of spending 1 on A, 2 on B and 1 on C; the resulting probabilities of survival are 1, 1 and 1/3, respectively, resulting in an expected future PD of $8\frac{1}{3}$. On the other hand the solution to the non-fractional problem consists of taking A and C, which has $\mathbb{E}[\varphi|\{\text{A,C}\}] = 8$. This example shows that the non-fractional solution cannot be simply obtained by eliminating the taxon fractionally conserved in the fractional solution, and, equivalently, that the fractional solution cannot be obtained in general by extending the non-fractional solution.

Not only can the fractional and non-fractional NAP have different solutions not necessarily related to each other in a straightforward way, but they have different algorithmic properties. As a start, whereas it is easy to see that the non-fractional problem is NP-hard (as it is a generalisation of BMAXPD, which I showed to be NP-hard in sec. 3.3), the computational complexity of the fractional NAP has not been established yet, to my knowledge. Also, algorithms may be effective for one problem but not for the other.

For an example of this, consider the fractional NAP with $a_s = 0$ and $b_s = 1$ for all taxa s, applied to an ultrametric tree (see ch. 1). Hartmann and Steel [**HS06**] show that a greedy algorithm can solve the fractional NAP in this scenario. This algorithm consists of repeatedly taking the taxon s that maximises

$$\frac{\mathbb{E}[\varphi|S \cup \{s\}] - \mathbb{E}[\varphi|S]}{c_s}, \tag{4.6.3}$$

where S is the set of taxa previously taken. In words, this is the taxon that gives the best increase in expected future PD per spending unit. This goes on until the total cost of S exceeds the budget. The last taxon added is then only fractionally conserved to bring the total cost to the budget.

A simple greedy algorithm for the non-fractional version of this problem — i.e. BMaxPD applied to an ultrametric tree — can be adapted from the one above: repeatedly take the taxon s that maximises (4.6.3), a*mong those that, given the current unused budget, it is still feasible to take.* The algorithm stops as soon as the unused budget is insufficient to select any remaining taxon. Whereas the greedy algorithm for the fractional problem is guaranteed to produce optimal solutions, this adaptation for the non-fractional case is not. Consider again figure 4.6.2. The greedy algorithm for the fractional problem selects here A, then B and finally selects fractionally C, leading to the optimal solution for the fractional NAP, as expected since the tree is ultrametric and $a_s = 0$ and $b_s = 1$ for all s [**HS06**]. The greedy algorithm for the non-fractional case selects A, then B and then stops, as the remaining budget is insufficient to select C. Since the optimal solution here is to select A and C, this shows that the greedy algorithm does not guarantee optimality in this case.

4.7. Extinction interdependencies and the nature reserve problem

An important shortcoming of the generalised field of bullets model and the Noah's Ark problems above is that the taxa are assumed to survive or become extinct independently of one another, whereas in reality strong interdependencies may exist: for example, between predators and prey [**WL95, WTL00, vdHvdBvI05**]. These dependencies can in principle be formalised, but this would make the dynamic programming algorithms presented here inapplicable, both the one for calculating the distribution of the future PD and the ones for the NAP. This is because, in order for a problem to have optimal substructure, a degree of independence between its parts is needed. Non-exact approaches such as simulations for the distribution of φ and hill-climbing or evolutionary algorithms for improved NAP-type problems may be preferable.

Perhaps, for the NAP, the most problematic aspect of taking into account ecological interdependencies is not the probabilistic dependency between the survivals/extinctions of different species — after all, the form of $\mathbb{E}[\varphi]$ in (4.2.1) remains correct, as the expectation of a sum of random variables is equal to the sum of their expectations, regardless of any dependence between them — but the fact that in

practice it is very rare that we can raise the probability of survival of one species without affecting that of other species. This may be justified for *ex-situ* conservation — removal and protection of a small number of individuals from their environment does not affect the survival of other taxa — but clearly not for *in-situ* conservation [**vdHvdBvI05**]. In particular, when protection is applied to a geographical area rather than to a species or population, survival probabilities are usually raised simultaneously for the entire group of taxa that lives in that area. Interestingly, this leads quite naturally to a generalization of the NAP in which conservation is applied to a number of potential nature reserves, each defining a set of changes in survival probabilities. I call this the *nature reserve problem* (NRP). In the rest of this section I propose a formulation for this problem and point out its connections with other optimisation problems previously proposed.

In the NRP the objective is to select a subset of a collection of candidate reserve sites, where for each site r the cost of conserving it is c_r, and a budget B is given. For each reserve site r and species s, the probability that s is present in r (at some future time) is given by b_{rs} or a_{rs} depending on whether site r is conserved or not, respectively. (This can model both the survival of s in r and our uncertainty regarding the actual occurrence of s in r [**PCS$^+$00**].) These probabilities define a generalised field of bullets model, where, assuming that R is the subset of conserved sites, the species' probabilities of survival are given by

$$p_s = 1 - \prod_{r \in R}(1 - b_{rs}) \prod_{r \notin R}(1 - a_{rs}).$$

In words, a species survives if and only if it will be present in at least one site. Note that I am here assuming independence between events in different sites. $\mathbb{E}[\varphi|R]$ will now denote the expected future PD resulting from conserving the reserve sites in R.

$a_{rs} \xrightarrow{c_r} b_{rs}$ **NRP:**

Input: A tree $\mathcal{T}$ with non-negative lengths t_e associated with each branch e. A set $\Re$ of candidate reserve sites. For each site r and leaf s of $\mathcal{T}$, two probabilities a_{rs} and b_{rs}. Integer costs c_r associated with each site $r \in \Re$ and a budget $B \geq 0$.

Output: A subset $R \subseteq \Re$ of reserve sites that

$$\text{maximises } \mathbb{E}[\varphi|R],$$

$$\text{subject to } \sum_{r \in R} c_r \leq B.$$

Note that I do not require that $a_{rs} \leq b_{rs}$, as conserving a site may, in some cases, actually *reduce* the probability of survival of a species in that site (think, for example, of species benefiting from human exploitation of the territory or of species which are preyed upon by other species benefiting from conservation).

It is easy to see that when each reserve has at most one species with non-zero probabilities, the NRP reduces to the $a_s \xrightarrow{c_s} b_s$ NAP.

Designing nature reserves, including prioritizing geographical sites, with the aim of preserving as much biodiversity as possible, is a common theme in conservation biology (e.g. [**Dia75, HU80, PHM$^+$93, DRRW97, ACPS98, HVD$^+$98, MP00, RAB$^+$04**]), and recently there has been a lot of interest in defining and solving (often heuristically) optimisation problems for reserve selection (e.g. [**MNP88, CB89, Und94, CPSC96, CSD96, CPW$^+$97, PCS$^+$00, CM01, RG02, OB03**]). Many (if not most) of the problems proposed in the past — not least the NAP — are special cases of the NRP. The PD-maximization problem of Rodrigues and Gaston [**RG02**] can be seen as the $0 \xrightarrow{1} 0/1$ NRP — where by writing 0/1 on the right of the arrow I mean that survival probabilities b_{rs} are constrained to equal 0 or 1. Similarly, the budgeted nature reserve problem of Bordewich and Semple [**BS08**] coincides with the $0 \xrightarrow{c_r} 0/1$ NRP and the expected species richness maximisation problem of Polasky et al. [**PCS$^+$00**] coincides with the $0 \xrightarrow{1} b_{rs}$ NRP applied to an ultrametric star tree. Therefore the NRP unifies many approaches from economics and biology to the problem of biodiversity conservation, and has the potential to become a fertile meeting ground for the exchange of new ideas between these two disciplines.

Finally, other special cases of the NRP have been the subject of research even outside conservation biology: for example, the Budgeted Maximum Coverage problem [**KMN99**] coincides with the $0 \xrightarrow{c_r} 0/1$ NRP applied to a star tree.

4.8. Related and future work

This chapter dealt essentially with taking into account the survival probabilities of species in conservation issues. There are a number of questions regarding these probabilities, however, that also deserve some attention.

First, how should these probabilities be evaluated? A quantitative assessment of the extinction risks of a species is precisely the aim of Population Viability Analysis (PVA) [**BM02**]. Another possible approach is to use existing classifications of the degree of threat faced by the species: for example, Redding and Mooers [**RM06**] and Pardi and Goldman [**PG07**] associated probabilities to each of the five major threat categories (CR, EN, VU, LR, LC) in the IUCN Red List [**IUC06**] (see also the legend of fig. 4.4.1).

Second, an important decision regarding the survival probabilities is about the time scale of conservation management. Recall that p_s represents the probability that taxon s is not extinct at a fixed future time t — for example 5 years or 100 years from now. Some of the results I have presented are sensitive to the choice of a particular t. Hartmann and Steel [**HS07**] show that the solution to the $a_s \xrightarrow{c_s} 1$ NAP can change quite radically as we increase a_s, corresponding to consideration of a nearer future t: as we increase the various a_s, the total branch length in the transformed tree $\mathcal{T}'$ (see sec. 4.4) becomes more and more concentrated in the terminal branches and therefore it is the relative length of these branches that will determine the optimal

taxa to conserve (i.e., the importance of the internal branches becomes negligible). Because of this sensitivity to t, it is important to choose an appropriate time scale.

Finally, another topic being investigated is the robustness of the results to uncertainties in the estimation of survival probabilities (Hartmann; personal communication).

Whereas I am not aware of any previous work on the distribution of the future PD, the optimisation problems discussed here have been investigated before. When Weitzman introduced the NAP [**Wei98**], his main objective was to create a framework in which to address a number of questions in conservation biology, for instance establishing that extreme conservation policies — involving for each species full conservation or no conservation at all — are the most advisable (unless the assumptions of the NAP are violated). In particular, Weitzman did not provide an exact algorithm to solve the NAP, as instead he was interested in a criterion to prioritize species for conservation. In his own suggestive words ([**Wei98**], p. 1294):

> He [Noah] does not want, however, to mess around with a complicated algorithm. Noah is a practical outdoors man. He needs robustness and rugged performance "in the field". As he stands at the door of the ark, Noah desires to use a simple priority ranking list from which he can check off one species at a time for boarding.

In fact, Weitzman used the NAP to provide a theoretical justification for a criterion for species prioritization. This criterion actually coincides with the one in (4.6.3) and leads to a rough approximation to an optimal solution [**Wei98**]. However, as I noted in section 2.2, the corresponding species ranking should not be used in the way suggested by the metaphor above: as illustrated in figure 2.2.1, there may be two (or more) species that come high in the ranking, but such that conserving one of them makes conserving the other(s) unnecessary. Ideally, measures of conservation value such as (4.6.3) should be updated each time a new species is conserved.

The algorithmic aspects of the NAP have been investigated by Hartmann and Steel [**HS06**], who showed that some of its special cases can actually be solved with a greedy algorithm: first, as illustrated in the example of figure 4.6.2, the fractional (i.e., admitting partial conservation) NAP with $a_s = 0$ and $b_s = 1$ for all taxa s, applied to an ultrametric tree, and second, the $1 - q_s \xrightarrow{c} 1 - \kappa q_s$ NAP, where for every taxon s the initial probability of extinction q_s can be reduced by a constant factor κ ($0 \leq \kappa \leq 1$) by paying a constant price c. The same authors [**HS07**] also independently showed the transformation idea of Proposition 4.4.1, but used it to prove the greedy tractability of the $a_s \xrightarrow{c} 1$ NAP. Note that this result also derives from the fact that the $a_s \xrightarrow{c} 1$ NAP is a subproblem of the greedy tractable $1 - q_s \xrightarrow{c} 1 - \kappa q_s$ NAP.

As I mentioned in section 4.1, maximising the expected future PD is just one of the possible objectives for an optimisation problem aimed at preserving future PD. An interesting subject for future research would be a problem where — unlike the

NAP — the aim is to minimise the probability that the future PD φ will fall below some given fixed critical value L_0. This is a philosophically different approach: rather than maximising the future PD in the average-case scenario, we wish to minimise the the risk of a worst-case scenario.

Since we can now approximate or even calculate exactly the distribution of φ, the objective function $\mathbb{P}[\varphi < L_0]$ is readily obtainable. In particular, if the distribution of φ is approximately normal (see Theorem 4.2.1), this coincides with

$$\Phi\left(\frac{L_0 - \mathbb{E}[\varphi]}{\sqrt{\mathrm{Var}[\varphi]}}\right),$$

where Φ denotes the cumulative distribution function of a standard normal variable $N(0,1)$. The main difference with the NAP seems to be that, not only must we try to maximise $\mathbb{E}[\varphi]$, but (assuming that we manage to ensure $\mathbb{E}[\varphi] > L_0$) we should also try to minimise $\mathrm{Var}[\varphi]$. Optimal solutions to the NAP may therefore be suboptimal with respect to $\mathbb{P}[\varphi < L_0]$.

CHAPTER 5

The generalised Noah's Ark problem and its solution

5.1. The beauty of curves: general effects of conservation on survival

A limitation of the optimisation problems discussed in Chapter 4 is that they model conservation policies as binary decisions: a taxon is either subject to conservation or it is not. In reality, however, a conservation policy is better described as a way to *distribute* the available resources among all the possible taxa.

Weitzman's NAP [**Wei98**], described in section 4.6, is a step in that direction: for each taxon i, a variable x_i is used to represent the amount of resources allocated towards the conservation of i — I will call this the *conservation expenditure* of i — and the aim is to find the best way to partition the budget B among all the different expenditures x_i. However, Weitzman also assumed that the probabilities of survival of taxa grow as linear (or, more precisely, affine) functions of their expenditures, and this has been shown in turn to imply that optimal conservation policies again consist of binary decisions, in other words about *which* taxa to conserve [**Wei98**] (see sec. 4.6 for more detail).

In this chapter, I propose an optimisation problem that relaxes the assumption of linearity in Weitzman's NAP, allowing arbitrarily general relationships between expenditures and survival probabilities: practically any function $p_i(x_i)$ can be adopted to describe how the probability of survival of i grows with the conservation expenditure x_i. In contrast with what happens for the NAP, for some taxa it will then become optimal to have expenditures that are neither zero nor equal to the maximum value allowed: although *some* spending on those taxa is desirable, any further investment is better spent into other taxa, in accordance with a law of diminishing returns. I call this problem the *generalised Noah's Ark problem* (GNAP) and describe it formally in section 5.2.

The main novel result of this chapter, described in section 5.3, is an algorithm that solves the GNAP. It is partly based on the dynamic programming algorithm for BMAXPD in the rooted case (sec. 3.4; [**PG07**]), with some new "geometrical" ideas inspired by a new kind of optimal substructure property. As usual, I accompany the description of this algorithm with an example of its functioning (sec. 5.4).

Because of the lack of worst-case polynomial complexity guarantees (discussed later), section 5.5 presents an empirical investigation of the computational efficiency of the algorithm for the GNAP, which shows that its running time and memory

requirements are practical on many realistic instances. As usual, the chapter closes with a discussion regarding related (and future) work (sec. 5.6).

5.2. The generalised Noah's Ark problem

Like the optimisation problems of Chapter 4, the GNAP assumes a generalised field of bullets model (sec. 4.2), where the probabilities of survival depend on the proposed solution to the problem. However, whereas before a solution consisted of a subset of taxa S (and the probability of survival p_i could assume one of two values depending on whether or not $i \in S$), now a solution is described by a vector of non-negative integer numbers $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ representing the conservation expenditures assigned to each taxon. The assumption that the expenditures be integer numbers is again due to the fact that it allows us to subdivide the problem into a finite number of subproblems with integer budgets. Also note that throughout this chapter, for simplicity, taxa will be identified by integer numbers $1, 2, \ldots, n$.

The probability of survival of each taxon i is defined by a function $p_i(x_i)$ of the conservation expenditure x_i. In practice this function is given in input as a sequence of probabilities $p_i(0), p_i(1), p_i(2), \ldots$ corresponding to each possible expenditure for taxon i; note that each taxon is associated with a different sequence. Given an integer budget B, I will assume that each of these sequences has length $B + 1$, i.e., that it goes up to define $p_i(B)$. Note that the continuous version of the GNAP — where the expenditures are not required to be integer — can be approximated within any desired precision by making the expenditure units sufficiently small.

The taxon expenditures $\mathbf{x}$ therefore fully determine a generalised field of bullets model. Assuming that the taxa correspond to the leaves of a phylogenetic tree $\mathcal{T}$ with branch lengths, the random variable φ quantifying the future PD is also determined. As in sections 4.4 and 4.6, let $\mathbb{E}[\varphi|\mathbf{x}]$ denote the expectation of this variable under the assumption that the taxon expenditures are given by $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.

Finally note that throughout this chapter, for consistency with the literature on the NAP, I assume a rooted definition for the future PD φ. I am now ready to introduce the central problem of this chapter.

GNAP:

Input: A rooted tree $\mathcal{T}_0$ with leaves $\{1, 2, \ldots, n\}$ and with non-negative lengths t_e associated with each branch e. An integer budget $B \geq 0$. For each leaf i, a sequence of probabilities $p_i(0), p_i(1), \ldots, p_i(B)$.

Output: Non-negative integer expenditures $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ associated with each taxon that

$$\text{maximise } \mathbb{E}[\varphi|\mathbf{x}],$$

$$\text{subject to } \sum_{i=1}^{n} x_i \leq B.$$

5.3. An algorithm for the GNAP

An algorithm to find optimal solutions to the GNAP can be obtained by using the main ideas of the dynamic programming algorithm for BMAXPD in the rooted case (sec. 3.4; [**PG07**]). Again, we subdivide the problem into a number of subproblems corresponding to every possible combination of a clade $\mathcal{T}$ of the input tree $\mathcal{T}_0$ and a budget $b \in \{0, 1, \ldots, B\}$; we deal with these subproblems in the same bottom-up order as described in section 3.4. I will again assume, without loss of generality, that $\mathcal{T}_0$ is bifurcating and rooted at the top of a root branch. In addition to that, I will make the realistic assumption that the survival probability functions are non-decreasing, i.e., that $p_i(0) \leq p_i(1) \leq \ldots \leq p_i(B)$. Note that it is straightforward to modify the input sequences (without changing the desired output) so that this assumption is met: just set $p_i(b) = \max_{c \in \{0,1,\ldots,b\}} p_i(c)$ for every $b \in \{0, 1, \ldots, B\}$.

The most important difference with the previous algorithm is that whereas before a single optimal solution was (implicitly) stored for each of the subproblems, now this is not sufficient. As explained in section 4.6, unfortunately the optimal substructure property is valid neither for the $a_s \xrightarrow{c_s} b_s$ NAP nor for the more general GNAP (it is easy to convert the instance in fig. 4.6.1 into an instance of the GNAP). Intuitively, for some clades such as $\mathcal{T}$ in fig. 4.6.1, a solution that ensures a high probability of survival for at least one taxon in $\mathcal{T}$ — a quantity which I denote with $\mathbb{P}[\mathcal{S}_\mathcal{T}|\mathbf{x}]$ (see Def. 5.3.2 below) — may be preferable to a solution that maximises the expected future PD in $\mathcal{T}$ — which I denote with $\mathbb{E}[\varphi_\mathcal{T}|\mathbf{x}]$. This is true especially when $\mathcal{T}$ lies below a long branch or path which has a high risk of being lost as a result of extinctions. This shows that optimal solutions to the GNAP may be composed of suboptimal solutions to its subproblems. So how can we solve the GNAP?

It turns out that if we judge candidate solutions in relation to the two criteria mentioned above — $\mathbb{E}[\varphi_\mathcal{T}|\mathbf{x}]$ and $\mathbb{P}[\mathcal{S}_\mathcal{T}|\mathbf{x}]$ — a proposition that is reminiscent of the optimal substructure property holds: solutions that have a "non-dominated" combination of these two scores can be derived from other solutions with "non-dominated" combinations (this is formalised in Proposition 5.3.8 below). Loosely speaking, a "non-dominated" combination of scores is one such that there is no feasible solution that has better scores under both criteria (but in reality this is not all that is required, as it will become clear in the definitions below).

The main difference between the algorithm that I present here and the one for BMAXPD is that now, for each subproblem, not just one solution needs to be stored, but several — as many as there are "non-dominated" score combinations. For each subproblem (associated with a clade $\mathcal{T}$ and a sub-budget b), these solutions will be stored in a set $L_\mathcal{T}(b)$ and each of these sets will be constructed from the sets already constructed, in a manner similar to the derivation of optimal solutions to the subproblems of BMAXPD from the optimal solutions to their own subproblems.

At the end of the algorithm, a number of solutions with "non-dominated" combinations of scores will have been derived for the global problem (the one relative to $\mathcal{T}_0$ and B). The optimal solution to the GNAP will be found among these.

In order to formalise these ideas, the following definitions introduce the necessary notation. They all assume that $\mathcal{T}$ is a clade of the input tree $\mathcal{T}_0$ and b is a budget in $\{0, 1, \ldots, B\}$.

DEFINITION 5.3.1. A *solution* for $\mathcal{T}$ is a vector of expenditures $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ such that, for every leaf i not in $\mathcal{T}$, $x_i = 0$. A *(feasible) solution* for $\mathcal{T}$ and b is a solution for $\mathcal{T}$ such that $\sum_i x_i \leq b$.

For example, in Figure 5.3.1, $\mathbf{x}_1 = (1, 1, 0, \ldots, 0)$ is a feasible solution for clade $\mathcal{L}$ and budget 2. Note that $\mathbf{x}_1$ is also a feasible solution for $\mathcal{T}$ and 2.

DEFINITION 5.3.2. Let $\mathbf{x}$ be a solution for $\mathcal{T}$. Let $\mathcal{S}_\mathcal{T}$ denote the event that at least one taxon in $\mathcal{T}$ survives. Therefore, $\mathbb{P}[\mathcal{S}_\mathcal{T}|\mathbf{x}]$ denotes the probability of $\mathcal{S}_\mathcal{T}$, resulting from spending $\mathbf{x}$.

In Figure 5.3.1, for example, $\mathbb{P}[\mathcal{S}_\mathcal{L}|\mathbf{x}_1] = 1-(1-p_1(1))(1-p_2(1)) = 1-0.5\times 0.7 = 0.65$. In general, it is easy to check that

$$\mathbb{P}[\mathcal{S}_\mathcal{T}|\mathbf{x}] = 1 - \prod_{i\in\mathcal{T}}(1 - p_i(x_i)),$$

where I use the informal notation $i \in \mathcal{T}$ to express that i is a taxon in $\mathcal{T}$.

DEFINITION 5.3.3. Let $\mathbf{x}$ be a solution for $\mathcal{T}$. $\varphi_\mathcal{T}$ denotes the future PD in $\mathcal{T}$ and $\mathbb{E}[\varphi_\mathcal{T}|\mathbf{x}]$ denotes its expectation, assuming the expenditures in $\mathbf{x}$.

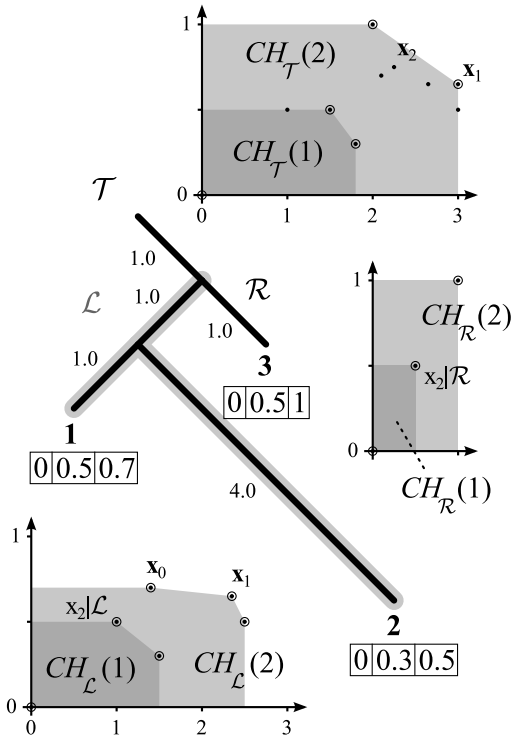
For example, in Figure 5.3.1, $\mathbb{E}[\varphi_\mathcal{L}|\mathbf{x}_1] = 1.0\times p_1(1)+4.0\times p_2(1)+1.0\times\mathbb{P}[\mathcal{S}_\mathcal{L}|\mathbf{x}_1] = 0.5 + 1.2 + 0.65 = 2.35$. Note that in general the value of $\mathbb{E}[\varphi_\mathcal{T}|\mathbf{x}]$ can be calculated following (4.2.1).

DEFINITION 5.3.4. Let $\mathbf{x}$ be a solution for $\mathcal{T}$; $\mathbf{x}$ is said to *lie* in the point $(\mathbb{E}[\varphi_\mathcal{T}|\mathbf{x}], \mathbb{P}[\mathcal{S}_\mathcal{T}|\mathbf{x}])$. When $\mathcal{T}$ can be inferred from the context, I will write this point as $(e_\mathbf{x}, p_\mathbf{x})$. Conversely, $(e_\mathbf{x}, p_\mathbf{x})$ is said to *correspond* to $\mathbf{x}$.

This definition introduces a very useful way to view solutions: they should be imagined as points on the plane whose coordinates correspond to the two main criteria to evaluate them. In fact this is precisely the way the algorithm actually stores the solutions, a fact that will be explained further in section 5.3.1. Figure 5.3.1 shows several examples of this: the plane at the bottom of the figure shows the points corresponding to all feasible solutions for $\mathcal{L}$ and 2; in particular $\mathbf{x}_1 = (1, 1, 0, \ldots 0)$ lies in $(e_{\mathbf{x}_1}, p_{\mathbf{x}_1}) = (2.35, 0.65)$, the point labelled with "$\mathbf{x}_1$" in the bottom graph in figure 5.3.1.

DEFINITION 5.3.5. Let $S = \{(e_1, p_1), (e_2, p_2), \ldots, (e_m, p_m)\}$ be all the points corresponding to the solutions for $\mathcal{T}$ and b. The *convex hull* for $\mathcal{T}$ and b, denoted $CH_\mathcal{T}(b)$, is the smallest convex polygon containing all the points in $S \cup$

FIGURE 5.3.1. A part of an instance of the GNAP and its solution. At the centre, clade $\mathcal{T}$, assumed to be a clade of a larger input tree $\mathcal{T}_0$. Branch lengths are indicated by the numbers to the side of the branches. The boxes below each taxon i specify the survival probabilities $p_i(0), p_i(1)$ and $p_i(2)$ resulting from spending 0, 1 and 2 on i, respectively, and the budget is $B = 2$. Clade $\mathcal{T}$ has two subclades: $\mathcal{L}$ (highlighted in gray) and $\mathcal{R}$ (consisting uniquely of a branch attaching to taxon 3). For each of these three clades, a graph shows the points corresponding to the feasible solutions that the algorithm derives (see Def. 5.3.4). The convex hulls for each of these clades and $b = 1, 2$ (Def. 5.3.5) are also shown (shaded polygons) and their proper vertices are indicated by circled points.



$\{(0,0), (\max_i e_i, 0), (0, \max_i p_i)\}$. The *proper vertices* of $CH_{\mathcal{T}}(b)$ are all the vertices of this polygon lying on its top-right boundary and that are also in S, i.e., such that there is a solution for $\mathcal{T}$ and b lying in them.

Figure 5.3.1 shows the convex hulls for several clades and budgets. For example, it is easy to see that the solutions for clade $\mathcal{L}$ and budget 2 are $(0,0,0,\ldots 0)$, $(1,0,0,\ldots 0)$, $(0,1,0,\ldots 0)$, $(2,0,0,\ldots 0)$, $(1,1,0,\ldots 0)$, $(0,2,0,\ldots 0)$ and that they correspond to the points $(0,0)$, $(1,0.5)$, $(1.5,0.3)$, $(1.4,0.7)$, $(2.35,0.65)$, $(2.5,0.5)$, which are all the points indicated in the graph at the bottom. Out of these, only the last three are proper vertices of $CH_{\mathcal{L}}(2)$, which is indicated by the circles around these three points; $(0,0)$, $(1,0.5)$ and $(1.5,0.3)$ are also circled, as they are proper vertices of $CH_{\mathcal{L}}(0)$ (the first) and of $CH_{\mathcal{L}}(1)$ (the second and the third).

A detailed account of convex hulls and their computational geometry is given in section 33.3 in [**CLRS01**]. Their usefulness in the context of the GNAP comes from

the fact that solutions lying in proper vertices of $CH_{\mathcal{T}}(b)$ are precisely what I meant before as solutions with a "non-dominated" combination of their scores $e_{\mathbf{x}}$ and $p_{\mathbf{x}}$. It is clear that, for these solutions, there is no other feasible solution that has better scores under both criteria $e_{\mathbf{x}}$ and $p_{\mathbf{x}}$; however the requirement of lying in a proper vertex of $CH_{\mathcal{T}}(b)$ is even stronger than this: for example $\mathbf{x}_2 = (1, 0, 1, 0 \ldots, 0)$, a solution for $\mathcal{T}$ and 2, lies in point $(2.25, 0.75)$, for which there is no other point corresponding to a feasible solution with better scores under both criteria (these points are all shown in the graph at the top of fig. 5.3.1). However, since it does not lie in a vertex of $CH_{\mathcal{T}}(2)$, this solution is not considered to have "non-dominated" scores. It is interesting to note that the notion of "non-domination" developed here is related (but not equivalent) to the notion of Pareto-efficient solutions in Economics [**Wik08**].

The importance of solutions lying in the proper vertices comes from the fact that (at least) a solution to the GNAP must itself lie in a proper vertex: in fact, any solution for $\mathcal{T}_0$ and B lying in the rightmost proper vertex of $CH_{\mathcal{T}_0}(B)$ (i.e., the one with the largest $e_{\mathbf{x}}$ coordinate) must be a solution to the GNAP (since it has maximum $\mathbb{E}[\varphi_{\mathcal{T}_0}|\mathbf{x}]$ among all solutions for $\mathcal{T}_0$ and B). For example, in figure 5.3.1, if we imagine that $\mathcal{T}$ coincides with the global tree, solution $\mathbf{x}_1 = (1, 1, 0)$ lies in the rightmost vertex of $CH_{\mathcal{T}}(2)$ and is an optimal solution for the GNAP applied to $\mathcal{T}$ and budget 2.

If we make sure that we derive, for each combination of a clade $\mathcal{T}$ and budget $b \leq B$, at least one solution per proper vertex of $CH_{\mathcal{T}}(b)$, then in the end we will have also derived an optimal solution to the GNAP. I will now show how this can be achieved.

DEFINITION 5.3.6. Let $\mathcal{T}$ be a clade with subclades $\mathcal{L}$ and $\mathcal{R}$. Let $\mathbf{x}$ be a solution for $\mathcal{T}$. The *restriction* of $\mathbf{x}$ on $\mathcal{L}$ (respectively, $\mathcal{R}$), denoted by $\mathbf{x}|\mathcal{L}$ (respectively, $\mathbf{x}|\mathcal{R}$), is the vector of expenditures obtained from $\mathbf{x}$ by setting to 0 all expenditures x_i for taxa i not in $\mathcal{L}$ (respectively, $\mathcal{R}$). Clearly $\mathbf{x}|\mathcal{L}$ is a solution for $\mathcal{L}$ (and similarly $\mathbf{x}|\mathcal{R}$ is a solution for $\mathcal{R}$). Now let $\mathbf{y}$ be a solution for $\mathcal{L}$, and $\mathbf{z}$ a solution for $\mathcal{R}$. The *combination* of $\mathbf{y}$ and $\mathbf{z}$ is simply the sum $\mathbf{y} + \mathbf{z}$ of these two vectors, and clearly is a solution for $\mathcal{T}$. Note that $\mathbf{x} = \mathbf{x}|\mathcal{L} + \mathbf{x}|\mathcal{R}$.

For example, if we consider again $\mathbf{x}_2 = (1, 0, 1, 0 \ldots, 0)$, we have that this solution for $\mathcal{T}$ can be obtained by combining $\mathbf{x}_2|\mathcal{L} = (1, 0, 0, 0, \ldots, 0)$ and $\mathbf{x}_2|\mathcal{R} = (0, 0, 1, 0, \ldots, 0)$. The points these solutions lie in are labelled in figure 5.3.1.

REMARK 5.3.7. Let $\mathcal{T}$ be a clade with subclades $\mathcal{L}$ and $\mathcal{R}$ and with a root branch a. Let $\mathbf{x}$ be a solution for $\mathcal{T}$. The points corresponding to $\mathbf{x}$, $\mathbf{x}|\mathcal{L}$ and $\mathbf{x}|\mathcal{R}$ satisfy the following relationships:

$$p_{\mathbf{x}} = 1 - (1 - p_{\mathbf{x}|\mathcal{L}})(1 - p_{\mathbf{x}|\mathcal{R}}), \tag{5.3.1}$$

$$e_{\mathbf{x}} = e_{\mathbf{x}|\mathcal{L}} + e_{\mathbf{x}|\mathcal{R}} + t_a p_{\mathbf{x}}. \tag{5.3.2}$$

For example, in Figure 5.3.1, since $\mathbf{x}_2|\mathcal{L}$ lies in $(1, 0.5)$ and $\mathbf{x}_2|\mathcal{R}$ lies in $(0.5, 0.5)$, then $\mathbf{x}_2$ lies in $(2.25, 0.75)$, as $p_{\mathbf{x}_2} = 1 - 0.5 \times 0.5 = 0.75$ and $e_{\mathbf{x}_2} = 1.0 + 0.5 + 1.0 \times 0.75 = 2.25$.

It turns out that the solutions lying in the proper vertices of a convex hull $CH_{\mathcal{T}}(b)$ can be obtained by combining (in the sense of Def. 5.3.6) the solutions lying in the proper vertices of the convex hulls for the subclades of $\mathcal{T}$ and budgets less or equal than b. This is made more precise by the following proposition.

PROPOSITION 5.3.8. *Let $\mathcal{T}$ be a clade with subclades $\mathcal{L}$ and $\mathcal{R}$, and $b \in \{0, 1, \ldots, B\}$. Every proper vertex of $CH_{\mathcal{T}}(b)$ corresponds to at least one solution $\mathbf{x}$ for $\mathcal{T}$ and b, such that $\mathbf{x}|\mathcal{L}$ and $\mathbf{x}|\mathcal{R}$ lie in proper vertices of $CH_{\mathcal{L}}(i)$ and $CH_{\mathcal{R}}(b-i)$, respectively, for some $i \in \{0, 1, \ldots, b\}$.*

In other words, at least one of the solutions lying in a proper vertex of $CH_{\mathcal{T}}(b)$ can be obtained by combining solutions lying in proper vertices from $CH_{\mathcal{L}}(i)$ and $CH_{\mathcal{R}}(b-i)$, for some $i \in \{0, 1, \ldots, b\}$. The proof of this result can be found in section 5.3.2. As an example of this, consider again the rightmost proper vertex of $CH_{\mathcal{T}}(2)$ in figure 5.3.1. Solution $\mathbf{x}_1 = (1, 1, 0, \ldots 0)$ lies here and its restrictions, $\mathbf{x}_1|\mathcal{L} = (1, 1, 0, \ldots, 0) = \mathbf{x}_1$ and $\mathbf{x}_1|\mathcal{R} = (0, 0, \ldots, 0)$ lie in a vertex of $CH_{\mathcal{L}}(2)$ and in a vertex of $CH_{\mathcal{R}}(0)$, respectively; the proposition is verified for $i = 2$.

The central idea of the algorithm is to store, for each clade $\mathcal{T}$ and budget $b \in \{0, 1, \ldots, B\}$, a set of solutions $L_{\mathcal{T}}(b)$ containing one solution per vertex of $CH_{\mathcal{T}}(b)$ (and lying in that vertex). The proposition above implies that this can be achieved by filling the various $L_{\mathcal{T}}(b)$ first for small clades and then for the larger clades they compose, in a bottom-up fashion: when $\mathcal{T}$ is composed of subclades $\mathcal{L}$ and $\mathcal{R}$, $L_{\mathcal{T}}(b)$ can be obtained by combining together (in the sense of Def. 5.3.6) all pairs of solutions from $L_{\mathcal{L}}(0)$ and $L_{\mathcal{R}}(b)$, all pairs from $L_{\mathcal{L}}(1)$ and $L_{\mathcal{R}}(b-1)$, and so on up to all pairs from $L_{\mathcal{L}}(b)$ and $L_{\mathcal{R}}(0)$. This generates a set of solutions, which, by Proposition 5.3.8, must contain at least one solution per vertex of $CH_{\mathcal{T}}(b)$ (see Proposition 5.3.10 and Corollary 5.3.11 for the correctness of this algorithm).

Back to the example in figure 5.3.1, when the time to fill $L_{\mathcal{T}}(2)$ comes, sets $L_{\mathcal{R}}(0)$, $L_{\mathcal{R}}(1)$, $L_{\mathcal{R}}(2)$ and $L_{\mathcal{L}}(0)$, $L_{\mathcal{L}}(1)$, $L_{\mathcal{L}}(2)$ will have already been filled with one solution per proper vertex of their respective convex hulls. A quick look at figure 5.3.1, reveals that $L_{\mathcal{R}}(0)$, $L_{\mathcal{R}}(1)$ and $L_{\mathcal{R}}(2)$ must contain one solution each (because $CH_{\mathcal{R}}(0)$, $CH_{\mathcal{R}}(1)$ and $CH_{\mathcal{R}}(2)$ have one proper vertex each), whereas $L_{\mathcal{L}}(0)$, $L_{\mathcal{L}}(1)$ and $L_{\mathcal{L}}(2)$ must contain 1, 2 and 3 solutions, respectively (because $CH_{\mathcal{R}}(0)$, $CH_{\mathcal{R}}(1)$ and $CH_{\mathcal{R}}(2)$ have 1, 2 and 3 proper vertices, respectively). Therefore combining solutions in the way described above must lead to adding $1 \times 3 + 1 \times 2 + 1 \times 1 = 6$ solutions to $L_{\mathcal{T}}(2)$. These solutions lie in the six points shown in the top-right portion of $CH_{\mathcal{T}}(2)$; two of them lie in the proper vertices of this convex hull (circled in fig. 5.3.1).

Algorithm 6 GNAPSOLVER($\mathcal{T}_0$,B, $\mathbf{p}_1, \ldots, \mathbf{p}_n$)

for every clade $\mathcal{T}$ of $\mathcal{T}_0$ (in a bottom-up order) **do**
  if $\mathcal{T}$ only consists of a terminal branch ending in taxon i **then**
    for $b = 0, \ldots, B$ **set** $L_{\mathcal{T}}(b) = \{(0, \ldots 0, b, 0, \ldots, 0)\}$,
      where the only non-zero component of $(0, \ldots 0, b, 0, \ldots, 0)$ is the i-th
  end if
  if $\mathcal{T}$ consists of an internal branch e and subclades $\mathcal{L}$ and $\mathcal{R}$ **then**
    for $b = 0, \ldots, B$
      $L_{\mathcal{T}}(b) = \emptyset$
      for $i = 0, \ldots, b$ **set** $L_{\mathcal{T}}(b) = L_{\mathcal{T}}(b) \cup (L_{\mathcal{L}}(i) \oplus L_{\mathcal{R}}(b-i))$
      CLEAN($L_{\mathcal{T}}(b)$)
    end for
  end if
end for
return an $\mathbf{x} \in L_{\mathcal{T}_0}(B)$ with maximum $e_{\mathbf{x}}$

Algorithm 7 CLEAN(S), where S is a set of solutions

Sort S by $e_{\mathbf{x}}$, i.e., let $S = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ with $e_{\mathbf{x}_1} \leq e_{\mathbf{x}_2} \leq \ldots \leq e_{\mathbf{x}_m}$.
Let t be such that $\mathbf{x}_t$ has maximum $p_{\mathbf{x}}$ among all $\mathbf{x} \in S$.
Let R be an empty queue.
for $i = t, \ldots, m$ **do**
  while $|R| \geq 2$ **and not** RIGHT-TURN (NEXT-TO-LAST(R), LAST(R), $\mathbf{x}_i$)
    POP LAST(R) from the back of R.
  PUSH $\mathbf{x}_i$ at the back of R.
end for
Reset $S = R$.

Since we are only interested in the solutions lying in the proper vertices of $CH_{\mathcal{T}}(b)$, many of the solutions that are added to $L_{\mathcal{T}}(b)$ in the way described above can be discarded. We wish to keep only one solution per proper vertex of $CH_{\mathcal{T}}(b)$. To this end, it is easy to devise a procedure that discards from $L_{\mathcal{T}}(b)$ all solutions that are somehow "dominated" by other solutions in $L_{\mathcal{T}}(b)$, and therefore do not lie on a proper vertex of $CH_{\mathcal{T}}(b)$. This procedure consists of an easy modification of Graham's scan [**Gra72, CLRS01**], an algorithm to find the vertices of any convex hull in the plane. The exact way this is done is described by the pseudocode in Algorithm 7; since its details are not important for the understanding of the algorithm presented here, I will not discuss this procedure further (except for defining the subroutines it uses; see next section). In the example just discussed, four of the six points added to $L_{\mathcal{T}}(2)$ do not lie in a proper vertex of $CH_{\mathcal{T}}(2)$ and are therefore deleted from $L_{\mathcal{T}}(2)$.

At the end of the algorithm's execution, each $L_{\mathcal{T}}(b)$ set will contain one solution per proper vertex of $CH_{\mathcal{T}}(b)$. In particular, the solution $\mathbf{x}$ in $L_{\mathcal{T}_0}(B)$ with maximum $e_{\mathbf{x}}$ lies in the rightmost proper vertex of $CH_{\mathcal{T}_0}(B)$ and is therefore a solution to the GNAP.

5.3.1. Pseudocode and actual implementation of the algorithm. Formally, the algorithm can be described by the pseudocode in Algorithm 6, where $X \oplus Y$ denotes the set of solutions obtained by combining every pair of solutions from X and Y, i.e., $X \oplus Y = \{\mathbf{x} + \mathbf{y} \,|\, \mathbf{x} \in X, \mathbf{y} \in Y\}$, and CLEAN($L_{\mathcal{T}}(b)$) is the procedure that eliminates from $L_{\mathcal{T}}(b)$ all solutions that are not deemed useful for the construction of an optimal solution to the global problem. In the definition of CLEAN given by Algorithm 7, this procedure only consists of the Graham's scan mentioned before; however I will show later (sec. 5.3.3) that a further simple cleaning step can be added, thus further reducing the final size of $L_{\mathcal{T}}(b)$.

Note the pseudocode in Algorithm 7 makes use of a number of elementary procedures that need some explanation: POP and PUSH simply consist of the addition and removal of one element at the end of the queue; LAST(R) and NEXT-TO-LAST(R) respectively return the element at the end of queue R and the one immediately preceding it; finally, RIGHT-TURN ($\mathbf{x}, \mathbf{y}, \mathbf{z}$) is a boolean function returning TRUE if and only if the segment from the point corresponding to $\mathbf{y}$ to the point corresponding to $\mathbf{z}$ represents a right-turn with respect to the segment from the point corresponding to $\mathbf{x}$ to the point corresponding to $\mathbf{y}$ (see [**CLRS01**], sec. 33.1 for details).

It is important to note that the algorithm does not need to store the solutions in $L_{\mathcal{T}}(b)$ as vectors of expenditures $\mathbf{x}$. A much better approach is to store their corresponding points $(e_{\mathbf{x}}, p_{\mathbf{x}})$ and pointers that permit reconstruction of the corresponding solutions. When two solutions represented as points are combined, the resulting point $(e_{\mathbf{x}}, p_{\mathbf{x}})$ can be calculated using equations (5.3.1) and (5.3.2) above; we also store pointers indicating the points from which $(e_{\mathbf{x}}, p_{\mathbf{x}})$ was obtained. As for the solutions for "trivial" clades $\mathcal{T}$ composed of only a terminal branch a leading to taxon i, in these cases $L_{\mathcal{T}}(b)$ only needs to store the point $(p_i(b) \cdot t_a,\, p_i(b))$ corresponding to the optimal solution for $\mathcal{T}$ and b.

It is clear that only storing $(e_{\mathbf{x}}, p_{\mathbf{x}})$ and the described pointers for each solution is entirely sufficient for the algorithm presented: the CLEAN procedure only needs the $(e_{\mathbf{x}}, p_{\mathbf{x}})$ coordinates in order to decide whether or not a solution lies in a proper vertex, and in the end the solution lying in the point with maximum $e_{\mathbf{x}}$ coordinate is reconstructed in a straightforward way using the stored pointers.

As an example of how this works in practice consider again figure 5.3.1. Clade $\mathcal{R}$ only consists of one branch leading to taxon 3, and therefore we set:

$$L_{\mathcal{R}}(0) = \{(p_3(0) \cdot 1.0,\, p_3(0))\} = \{(0,0)\},$$
$$L_{\mathcal{R}}(1) = \{(0.5, 0.5)\},$$
$$L_{\mathcal{R}}(2) = \{(1,1)\}.$$

As an example of how the sets relative to nontrivial clades are obtained, consider the derivation of $L_{\mathcal{T}}(2)$. Because $\mathcal{L}$ and $\mathcal{R}$ are subclades of $\mathcal{T}$, when the time to derive $L_{\mathcal{T}}(2)$ comes, the sets $L_{\mathcal{R}}(0)$, $L_{\mathcal{R}}(1)$, $L_{\mathcal{R}}(2)$ and $L_{\mathcal{L}}(0)$, $L_{\mathcal{L}}(1)$, $L_{\mathcal{L}}(2)$ will

have already been derived, the former in the way shown above and the latter so that:

$$L_{\mathcal{L}}(0) = \{(0,0)\},$$

$$L_{\mathcal{L}}(1) = \{(1,0.5),(1.5,0.3)\},$$

$$L_{\mathcal{L}}(2) = \{(1.4,0.7),(2.35,0.65),(2.5,0.5)\},$$

(note that I am here ignoring the pointers for the backtracking). Note that the points in the three sets above coincide with the proper vertices of $CH_{\mathcal{L}}(0)$, $CH_{\mathcal{L}}(1)$ and $CH_{\mathcal{L}}(2)$, shown in the graph at the bottom of figure 5.3.1.

Now apply the procedure described above in order to fill in $L_{\mathcal{T}}(2)$: combine the solution in $L_{\mathcal{R}}(0) = \{(0,0)\}$ with every solution in $L_{\mathcal{L}}(2)$, which gives the following set of points (by equations (5.3.1) and (5.3.2) above):

$$\{(2.1,0.7),(3,0.65),(3,0.5)\};$$

then combine the solution in $L_{\mathcal{R}}(1) = \{(0.5,0.5)\}$ with every solution in $L_{\mathcal{L}}(1)$, which gives:

$$\{(2.25,0.75),(2.65,0.65)\};$$

finally combine the solution in $L_{\mathcal{R}}(2) = \{(1,1)\}$ with the solution in $L_{\mathcal{L}}(0) = \{(0,0)\}$, giving:

$$\{(2,1)\}.$$

Note that the six points just derived are precisely the six points shown in the top-right portion of the graph at the top of figure 5.3.1. We then apply procedure CLEAN on these six points, which discards the four points internal to $CH_{\mathcal{T}}(2)$ and sets

$$L_{\mathcal{T}}(2) = \{(2,1),(3,0.65)\}.$$

5.3.2. Correctness of the algorithm. *Proof of Proposition 5.3.8.* First of all, note that a point $(e,p) \in CH_{\mathcal{T}}(b)$ is not a proper vertex of $CH_{\mathcal{T}}(b)$ if and only if there are two solutions for $\mathcal{T}$ and b, $\mathbf{y}$ and $\mathbf{z}$, not lying in (e,p) and there is an $h \in [0,1]$ such that $(e,p) \leq h \cdot (e_{\mathbf{y}}, p_{\mathbf{y}}) + (1-h) \cdot (e_{\mathbf{z}}, p_{\mathbf{z}})$. Note that when we write $(e,p) \leq (e',p')$, we mean the usual component-wise inequalities $e \leq e'$ *and* $p \leq p'$.

Now let (e,p) be a proper vertex of $CH_{\mathcal{T}}(b)$ and $\mathbf{x}$ be any solution for $\mathcal{T}$ and b lying in (e,p). Consider $\mathbf{x}|\mathcal{L}$ and $\mathbf{x}|\mathcal{R}$, the restrictions of $\mathbf{x}$ on $\mathcal{T}$'s two subclades $\mathcal{L}$ and $\mathcal{R}$. It is clear that $\mathbf{x}|\mathcal{L}$ is a solution for $\mathcal{L}$ and i, and $\mathbf{x}|\mathcal{R}$ is a solution for $\mathcal{R}$ and $b-i$, for some $i \in \{0,1,\ldots,b\}$.

Let us initially assume that $p_{\mathbf{x}|\mathcal{R}} \neq 1$. We show now that this implies that $\mathbf{x}|\mathcal{L}$ lies in a proper vertex of $CH_{\mathcal{L}}(i)$. Suppose the contrary. Then let $\mathbf{y}$ and $\mathbf{z}$ be solutions for $\mathcal{L}$ and i that do not lie in $(e_{\mathbf{x}|\mathcal{L}}, p_{\mathbf{x}|\mathcal{L}})$ and let $h \in [0,1]$ be such that $(e_{\mathbf{x}|\mathcal{L}}, p_{\mathbf{x}|\mathcal{L}}) \leq h \cdot (e_{\mathbf{y}}, p_{\mathbf{y}}) + (1-h) \cdot (e_{\mathbf{z}}, p_{\mathbf{z}})$. Then define $\mathbf{y}' = \mathbf{y} + \mathbf{x}|\mathcal{R}$ and $\mathbf{z}' = \mathbf{z} + \mathbf{x}|\mathcal{R}$ and note that these are solutions for $\mathcal{T}$ and b. It is easy to prove that the assumption $p_{\mathbf{x}|\mathcal{R}} \neq 1$ implies that $\mathbf{y}'$ and $\mathbf{z}'$ do not lie in (e,p) and that

$(e,p) \leq h \cdot (e_{\mathbf{y}'}, p_{\mathbf{y}'}) + (1-h) \cdot (e_{\mathbf{z}'}, p_{\mathbf{z}'})$.[1] But this would contradict the assumption that (e,p) is a proper vertex of $CH_{\mathcal{T}}(b)$ and therefore $\mathbf{x}|\mathcal{L}$ lies in a proper vertex of $CH_{\mathcal{L}}(i)$.

If instead $p_{\mathbf{x}|\mathcal{R}} = 1$, then $\mathbf{x}|\mathcal{L}$, among all possible solutions for $\mathcal{L}$ and i, must be optimal with respect to e: if there was a solution $\mathbf{y}$ for $\mathcal{L}$ and i with $e_{\mathbf{y}} > e_{\mathbf{x}|\mathcal{L}}$ then, if we define $\mathbf{y}' = \mathbf{y} + \mathbf{x}|\mathcal{R}$, we have that $\mathbf{x}$ and $\mathbf{y}'$ lie on $(e_{\mathbf{x}}, 1)$ and $(e_{\mathbf{y}'}, 1)$, respectively, with $e_{\mathbf{y}'} = e_{\mathbf{y}} + e_{\mathbf{x}|\mathcal{R}} + t_a > e_{\mathbf{x}|\mathcal{L}} + e_{\mathbf{x}|\mathcal{R}} + t_a = e_{\mathbf{x}}$. But this would contradict the assumption that $\mathbf{x}$ lies on a proper vertex of $CH_{\mathcal{T}}(b)$ and therefore $\mathbf{x}|\mathcal{L}$ must be optimal with respect to e, in other words lie on the "rightmost" face of $CH_{\mathcal{L}}(i)$. Now let $\mathbf{x}^*$ be a solution lying in the rightmost proper vertex of $CH_{\mathcal{L}}(i)$. It is easy to verify that, if we replace $\mathbf{x}|\mathcal{L}$ with $\mathbf{x}^*$ in $\mathbf{x}$, thus obtaining $\mathbf{x}^* + \mathbf{x}|\mathcal{R}$, this solution must still lie in (e,p).

So we have proved that one can always construct a solution $\mathbf{x}'$ for $\mathcal{T}$ and b from $\mathbf{x}$ ($\mathbf{x}' = \mathbf{x}$ in the case $p_{\mathbf{x}|\mathcal{R}} \neq 1$ and $\mathbf{x}' = \mathbf{x}^* + \mathbf{x}|\mathcal{R}$ in the case $p_{\mathbf{x}|\mathcal{R}} = 1$), still lying in (e,p), whose left restriction lies in a proper vertex of $CH_{\mathcal{L}}(i)$ and whose right restriction is the same as the one in $\mathbf{x}$. The arguments above can be equally applied to $\mathbf{x}'$ (by swapping every mention of $\mathcal{L}$ with $\mathcal{R}$ and vice versa), leading to an $\mathbf{x}''$ with both its restrictions $\mathbf{x}''|\mathcal{L}$ ($= \mathbf{x}'|\mathcal{L}$) and $\mathbf{x}''|\mathcal{R}$ lying in vertices of $CH_{\mathcal{L}}(i)$ and $CH_{\mathcal{R}}(b-i)$, respectively.

REMARK 5.3.9. The solutions stored in each set $L_{\mathcal{T}}(b)$ are necessarily feasible solutions for $\mathcal{T}$ and b.

PROOF. By induction on the number of taxa in $\mathcal{T}$. □

PROPOSITION 5.3.10. *The execution of the algorithm described above results in each proper vertex of $CH_{\mathcal{T}}(b)$ having at least one solution in $L_{\mathcal{T}}(b)$ lying in it (for any clade $\mathcal{T}$ and $b \in \{0, 1, \ldots, B\}$).*

PROOF. By induction on the number of taxa in $\mathcal{T}$. When $\mathcal{T}$ only contains one taxon, this is trivially true. When $\mathcal{T}$ is composed of two subclades $\mathcal{L}$ and $\mathcal{R}$, then, since $\mathcal{L}$ and $\mathcal{R}$ contain fewer taxa than $\mathcal{T}$, we can assume (inductive hypothesis) that the thesis holds for $\mathcal{L}$ and $\mathcal{R}$. Consider one proper vertex (e,p) of $CH_{\mathcal{T}}(b)$. We wish to prove that $L_{\mathcal{T}}(b)$ ends up containing a solution lying in (e,p).

---

[1] For simplicity, I have omitted the proofs of these statements. Here they are:
We know that $\mathbf{y}$ does not lie in $(e_{\mathbf{x}|\mathcal{L}}, p_{\mathbf{x}|\mathcal{L}})$. This means that either (1) $p_{\mathbf{y}} \neq p_{\mathbf{x}|\mathcal{L}}$ or, (2) $p_{\mathbf{y}} = p_{\mathbf{x}|\mathcal{L}}$ and $e_{\mathbf{y}} \neq e_{\mathbf{x}|\mathcal{L}}$. In case (1), $p_{\mathbf{y}'} = p_{\mathbf{x}|\mathcal{R}} + p_{\mathbf{y}}(1 - p_{\mathbf{x}|\mathcal{R}}) \neq p_{\mathbf{x}|\mathcal{R}} + p_{\mathbf{x}|\mathcal{L}}(1 - p_{\mathbf{x}|\mathcal{R}}) = p_{\mathbf{x}} = p$, where the inequality only holds because we assumed $p_{\mathbf{x}|\mathcal{R}} \neq 1$. In case (2), we have that $p_{\mathbf{y}'} = p_{\mathbf{x}}$ and so $e_{\mathbf{y}'} = e_{\mathbf{y}} + e_{\mathbf{x}|\mathcal{R}} + t_a p_{\mathbf{y}'} = e_{\mathbf{y}} + e_{\mathbf{x}|\mathcal{R}} + t_a p_{\mathbf{x}} \neq e_{\mathbf{x}|\mathcal{L}} + e_{\mathbf{x}|\mathcal{R}} + t_a p_{\mathbf{x}} = e_{\mathbf{x}} = e$. In both cases we have that $(e_{\mathbf{y}'}, p_{\mathbf{y}'}) \neq (e,p)$. We can prove in a similar way that $(e_{\mathbf{z}'}, p_{\mathbf{z}'}) \neq (e,p)$, i.e., neither $\mathbf{y}'$ nor $\mathbf{z}'$ lie in (e,p).
As for $(e,p) \leq h \cdot (e_{\mathbf{y}'}, p_{\mathbf{y}'}) + (1-h) \cdot (e_{\mathbf{z}'}, p_{\mathbf{z}'})$, we prove it one component at a time: $p = p_{\mathbf{x}|\mathcal{R}} + p_{\mathbf{x}|\mathcal{L}}(1 - p_{\mathbf{x}|\mathcal{R}}) \leq p_{\mathbf{x}|\mathcal{R}} + (hp_{\mathbf{y}} + (1-h)p_{\mathbf{z}})(1 - p_{\mathbf{x}|\mathcal{R}}) = h\left(p_{\mathbf{x}|\mathcal{R}} + p_{\mathbf{y}}(1 - p_{\mathbf{x}|\mathcal{R}})\right) + (1-h)\left(p_{\mathbf{x}|\mathcal{R}} + p_{\mathbf{z}}(1 - p_{\mathbf{x}|\mathcal{R}})\right) = hp_{\mathbf{y}'} + (1-h)p_{\mathbf{z}'}$; $e = e_{\mathbf{x}|\mathcal{L}} + e_{\mathbf{x}|\mathcal{R}} + t_a p \leq he_{\mathbf{y}} + (1-h)e_{\mathbf{z}} + e_{\mathbf{x}|\mathcal{R}} + t_a(hp_{\mathbf{y}'} + (1-h)p_{\mathbf{z}'}) = h(e_{\mathbf{y}} + e_{\mathbf{x}|\mathcal{R}} + t_a p_{\mathbf{y}'}) + (1-h)(e_{\mathbf{z}} + e_{\mathbf{x}|\mathcal{R}} + t_a p_{\mathbf{z}'}) = he_{\mathbf{y}'} + (1-h)e_{\mathbf{z}'}$.

By Proposition 5.3.8, there is at least one solution $\mathbf{x}$ for $\mathcal{T}$ and b, lying in (e,p), such that $\mathbf{x}|\mathcal{L}$ and $\mathbf{x}|\mathcal{R}$ lie in vertices of $CH_{\mathcal{L}}(i)$ and $CH_{\mathcal{R}}(b-i)$, respectively, for some $i \in \{0,1,\ldots,b\}$. By the inductive hypothesis, there must be two solutions, $\mathbf{y}$ and $\mathbf{z}$ in $L_{\mathcal{L}}(b)$ and $L_{\mathcal{R}}(b-i)$, respectively, that lie in these two vertices. When the update $L_{\mathcal{T}}(b) = L_{\mathcal{T}}(b) \cup (L_{\mathcal{L}}(i) \oplus L_{\mathcal{R}}(b-i))$ takes place, these two solutions are combined into $\mathbf{x}' = \mathbf{y} + \mathbf{z}$ which is stored in $L_{\mathcal{T}}(b)$. It is easy to see that $\mathbf{x}'$ must lie in (e,p). □

COROLLARY 5.3.11. *The algorithm returns an optimal solution for the given instance of the* GNAP.

PROOF. The algorithm returns the solution $\mathbf{x}$ in $L_{\mathcal{T}_0}(B)$ with maximum $e_{\mathbf{x}}$. Because $L_{\mathcal{T}_0}(B)$ only contains solutions for $\mathcal{T}_0$ and B (Remark 5.3.9) and at least one of these lies in the rightmost proper vertex of $CH_{\mathcal{T}_0}(B)$ (Proposition 5.3.10), $\mathbf{x}$ is a solution for $\mathcal{T}_0$ and B and lies in a point of $CH_{\mathcal{T}_0}(B)$ with maximal e coordinate. Since any solution $\mathbf{x}'$ for $\mathcal{T}_0$ and B must lie in $CH_{\mathcal{T}_0}(B)$, we have $e_{\mathbf{x}'} \leq e_{\mathbf{x}}$ and therefore $\mathbf{x}$ is optimal with respect to $\mathbb{E}[\varphi_{\mathcal{T}_0}|\mathbf{x}]$. □

Note that in Proposition 5.3.10, I state that *at least one* solution in $L_{\mathcal{T}}(b)$ must lie in any given proper vertex of $CH_{\mathcal{T}}(b)$. In fact a stronger proposition holds: there is a one-to-one correspondence between proper vertices of $CH_{\mathcal{T}}(b)$ and elements of $L_{\mathcal{T}}(b)$, such that each solution in $L_{\mathcal{T}}(b)$ lies in its corresponding vertex. This property is guaranteed by the actual functioning of the Graham's scan implemented in the CLEAN procedure, which leaves exactly one solution lying in each proper vertex of $CH_{\mathcal{T}}(b)$.

Since the correctness of the algorithm does not depend on this property, for simplicity I do not prove it here. However, it is important to realise that the efficiency of the algorithm depends on the size of the $L_{\mathcal{T}}(b)$ lists — the smaller, the better — and it is therefore important to discard as many solutions as possible from $L_{\mathcal{T}}(b)$. This consideration naturally leads us to the next subsection.

5.3.3. Further cleaning. For some solutions $\mathbf{x}$ with small $e_{\mathbf{x}}$ and large $p_{\mathbf{x}}$ that are retained because of their potential to ensure survival of the path above their clade, such path turns out to be too short to justify their retention. I will first show this with an example and then derive a new general criterion for discarding solutions from $L_{\mathcal{T}}(b)$.

Consider again figure 5.3.1 and assume that $\mathcal{T}$ coincides with the global tree; that is, that we are seeking a solution to the GNAP applied to $\mathcal{T}$. Because $CH_{\mathcal{L}}(2)$ has three proper vertices, after the Graham's scan $L_{\mathcal{L}}(2)$ contains three solutions lying in each of those vertices. However, I will now show that the solution lying in the leftmost vertex, $\mathbf{x}_0 = (2,0,\ldots,0)$, cannot be part of an optimal solution to the GNAP we wish to solve, and therefore its inclusion in $L_{\mathcal{T}}(2)$ is superfluous. This remark depends upon observing that the path from the root of $\mathcal{T}$ to the root of $\mathcal{L}$

(consisting only of one branch of length 1.0) is too short to justify inclusion of $\mathbf{x}_0$: the only advantage that $\mathbf{x}_0$ has over $\mathbf{x}_1 = (1, 1, 0, \dots 0)$ (whose corresponding point is also shown in fig. 5.3.1) is the greater survival probability that $\mathbf{x}_0$ confers to $\mathcal{L}$, which may turn out to be useful if it is important to ensure survival of long paths above $\mathcal{L}$; however no such long path above $\mathcal{L}$ exists. I now make this more precise.

Let $\mathbf{y}$ be a solution for $\mathcal{R}$ that may be combined with $\mathbf{x}_0$ or $\mathbf{x}_1$ to form a solution for $\mathcal{T}$. Then, if we call t the length of the path that lies above $\mathcal{L}$ and $\mathcal{R}$ (in this case 1.0), it is easy to show that

$$\mathbb{E}[\varphi_{\mathcal{T}}|\mathbf{x}_1 + \mathbf{y}] - \mathbb{E}[\varphi_{\mathcal{T}}|\mathbf{x}_0 + \mathbf{y}] = e_{\mathbf{x}_1} - e_{\mathbf{x}_0} + t(1 - p_{\mathbf{y}})(p_{\mathbf{x}_1} - p_{\mathbf{x}_0}).$$

If we note that $\mathbf{x}_0$ lies in $(1.4, 0.7)$ and $\mathbf{x}_1$ lies in $(2.35, 0.65)$, we have that

$$\mathbb{E}[\varphi_{\mathcal{T}}|\mathbf{x}_1 + \mathbf{y}] - \mathbb{E}[\varphi_{\mathcal{T}}|\mathbf{x}_0 + \mathbf{y}] = 0.95 - 0.05 \cdot t \cdot (1 - p_{\mathbf{y}}).$$

But then it is clear that, whatever the value of $p_{\mathbf{y}}$, and because $t = 1$, the right-hand side above will always be positive and therefore $\mathbf{x}_0$ cannot be part of an optimal solution to the GNAP applied to $\mathcal{T}$. Also note that inclusion of $\mathbf{x}_0$ in $L_{\mathcal{T}}(2)$ is instead justified for a sufficiently large t.

Now let us generalise the considerations above. I will show that if the path between the roots of the global tree $\mathcal{T}_0$ and clade $\mathcal{T}$ is short (or, more precisely, its "endangered portion", defined below), then only solutions with a large e coordinate need be kept in $L_{\mathcal{T}}(b)$.

Let $\mathbf{x}$ be a solution for clade $\mathcal{T}$. I wish to study the contribution of $\mathbf{x}$ towards the expected future PD in the entire input tree $\mathcal{T}_0$. Any solution that has $\mathbf{x}$ as its restriction on $\mathcal{T}$ can be expressed as $\mathbf{x}+\mathbf{y}$, where $\mathbf{y}$ is a solution for the entire input tree $\mathcal{T}_0$ that does not include any expenditure on $\mathcal{T}$, i.e., such that, for every taxon i in $\mathcal{T}$, $y_i = 0$. If we denote by $\rho\mathcal{T}$ the path from the root of $\mathcal{T}_0$ to the root of $\mathcal{T}$, and recall the notations $\mathcal{S}_X$ and $C(e)$ from the previous chapter (sec. 4.2), then the expected future PD given $\mathbf{x} + \mathbf{y}$ can be expressed as:

$$\mathbb{E}[\varphi_{\mathcal{T}_0}|\mathbf{x}+\mathbf{y}] = \mathbb{E}[\varphi_{\mathcal{T}}|\mathbf{x}] + \sum_{e\in\rho\mathcal{T}} t_e \cdot \mathbb{P}[\mathcal{S}_{C(e)}|\mathbf{x}+\mathbf{y}] + \sum_{e\in\mathcal{T}_0-\mathcal{T}-\rho\mathcal{T}} t_e \cdot \mathbb{P}[\mathcal{S}_{C(e)}|\mathbf{y}], \tag{5.3.3}$$

where I have used the independence of event $\mathcal{S}_{C(e)}$ from expenditures for taxa not in $C(e)$. Now define the *endangered portion of $\rho\mathcal{T}$ given* $\mathbf{y}$ as

$$d_{\mathcal{T}}(\mathbf{y}) = \sum_{e\in\rho\mathcal{T}} t_e \cdot \prod_{i\in C(e)-\mathcal{T}} (1 - p_i(y_i)).$$

It is then easy to check that the second term in the right-hand side of (5.3.3) can be expressed as

$$\mathbb{P}[\mathcal{S}_{\mathcal{T}}|\mathbf{x}] \cdot d_{\mathcal{T}}(\mathbf{y}) + \sum_{e\in\rho\mathcal{T}} t_e - d_{\mathcal{T}}(\mathbf{y}).$$

If all the terms that do not depend on $\mathbf{x}$ are grouped and denoted by $K_{\mathbf{y}}$, we can express the expected future PD as a linear function of the point $(e_{\mathbf{x}}, p_{\mathbf{x}})$ in which $\mathbf{x}$

lies:

$$\mathbb{E}[\varphi_{\mathcal{T}_0}|\mathbf{x}+\mathbf{y}] = e_{\mathbf{x}} + p_{\mathbf{x}} \cdot d_{\mathcal{T}}(\mathbf{y}) + K_{\mathbf{y}}. \tag{5.3.4}$$

This equation provides insight into the potential usefulness of a solution $\mathbf{x}$ for a clade $\mathcal{T}$ in constructing an optimal solution to the global problem. For example, it can be used to provide an alternative proof that the only solutions that can be part of an optimal solution to the GNAP are those lying on the top-right boundary of $CH_{\mathcal{T}}(b)$. Here, I use it to prove the following proposition:

PROPOSITION 5.3.12. *Let $\mathbf{x}_0$ and $\mathbf{x}_1$ be two solutions for clade $\mathcal{T}$ and budget b, respectively lying in point (e_0, p_0) and (e_1, p_1), with $e_0 < e_1$ and $p_0 > p_1$. If*

$$\frac{e_1 - e_0}{p_0 - p_1} > d_{\mathcal{T}}((0, 0, \ldots, 0)), \tag{5.3.5}$$

then $\mathbf{x}_0$ cannot be part of an optimal solution to the GNAP.

PROOF. Let $\mathbf{y}$ be a solution for the entire input tree $\mathcal{T}_0$ and budget $B - b$ that does not include any expenditure on $\mathcal{T}$, i.e., such that, for every taxon i in $\mathcal{T}$, $y_i = 0$. It is easy to check that the endangered portion of $\rho\mathcal{T}$ is bounded above by $d_{\mathcal{T}}((0, 0, \ldots, 0))$, that is,

$$d_{\mathcal{T}}(\mathbf{y}) \leq d_{\mathcal{T}}((0, 0, \ldots, 0)).$$

But then we have

$$\frac{e_1 - e_0}{p_0 - p_1} > d_{\mathcal{T}}(\mathbf{y}).$$

A little algebra shows that this is equivalent to

$$e_0 + p_0 \cdot d_{\mathcal{T}}(\mathbf{y}) < e_1 + p_1 \cdot d_{\mathcal{T}}(\mathbf{y}).$$

But then, because of equation (5.3.4),

$$\mathbb{E}[\varphi_{\mathcal{T}_0}|\mathbf{x}_0 + \mathbf{y}] < \mathbb{E}[\varphi_{\mathcal{T}_0}|\mathbf{x}_1 + \mathbf{y}],$$

which shows that $\mathbf{x}_0 + \mathbf{y}$ cannot be an optimal solution for $\mathcal{T}_0$ and B. □

We saw before that a modified Graham scan allows us to leave in $L_{\mathcal{T}}(b)$ exactly one solution lying in each proper vertex of $CH_{\mathcal{T}}(b)$. Proposition 5.3.12 suggests that more solutions can potentially be discarded from $L_{\mathcal{T}}(b)$. The following remark allows us to come up with a procedure to quickly find out the solutions to discard, if any.

REMARK 5.3.13. Order the solutions in $L_{\mathcal{T}}(b)$ on the basis of their e coordinate, so that $L_{\mathcal{T}}(b) = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{L-1})$, with $\mathbf{x}_i$ lying in (e_i, p_i) and $e_0 < e_1 < \ldots < e_{L-1}$ (and, consequently, $p_0 > p_1 > \ldots > p_{L-1}$). If any pair of solutions in $L_{\mathcal{T}}(b)$ satisfy the assumptions of Proposition 5.3.12, then so do $\mathbf{x}_0$ and $\mathbf{x}_1$.

PROOF. Equation (5.3.5) is basically requiring that the (negative) slope $(p_1 - p_0)/(e_1 - e_0)$ between the two points be larger than $-1/d_{\mathcal{T}}((0, 0, \ldots, 0))$. The remark follows from simply observing that because $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{L-1}$ lie in adjacent proper

vertices of $CH_{\mathcal{T}}(b)$, the slope between consecutive points $\mathbf{x}_i, \mathbf{x}_{i+1}$ is a decreasing function of i. □

The remark above simply means that we can scan the elements of $L_{\mathcal{T}}(b)$ in order from the one with the smallest e coordinate to the one with the largest (in fact the Graham's scan already returns solutions in this order), and assess whether the first two solutions satisfy the assumptions of Proposition 5.3.12; in the affirmative case, the first solution gets discarded from $L_{\mathcal{T}}(b)$ and the scan continues; in the negative case, the scan terminates. In other words we can add the following instruction at the end of the CLEAN procedure (Algorithm 7):

while $|S| \geq 2$ **and** $e_{S[1]} - e_{S[0]} \geq d_{\mathcal{T}}((0,0,\ldots,0)) \cdot (p_{S[0]} - p_{S[1]})$
do POP $S[0]$

Note that $d_{\mathcal{T}}((0,0,\ldots,0)) = \sum_{e \in \rho\mathcal{T}} t_e \cdot \prod_{i \in C(e)-\mathcal{T}} (1 - p_i(0))$ can be calculated in $O(n)$ time for each clade $\mathcal{T}$.

Returning to the example in figure 5.3.1, with the additional assumption that $\mathcal{T} = \mathcal{T}_0$, it is clear that solutions $\mathbf{x}_0$ and $\mathbf{x}_1$ in $L_{\mathcal{L}}(2)$ satisfy the assumptions of Proposition 5.3.12, and therefore $\mathbf{x}_0$ should be discarded:

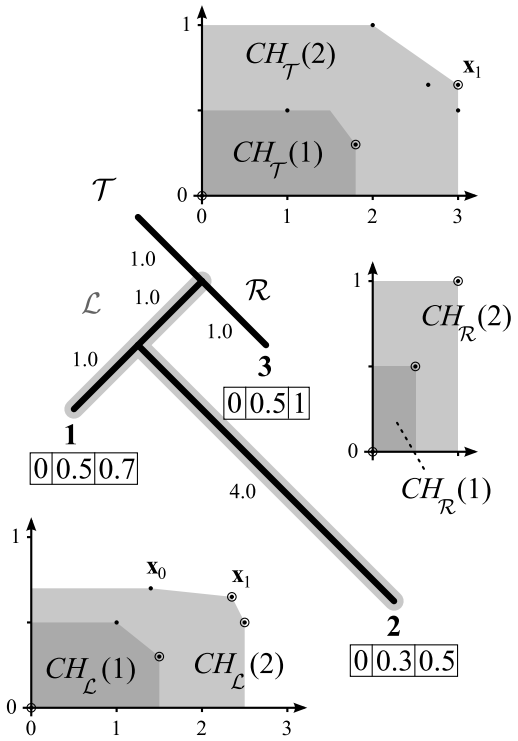
$$\frac{e_1 - e_0}{p_0 - p_1} = \frac{0.95}{0.05} = 19 > d_{\mathcal{L}}((0,0,0)) = 1.$$

This also shows that the path above the root of $\mathcal{L}$ should have at least length 19 in order for $\mathbf{x}_0$ to be kept in $L_{\mathcal{L}}(2)$.

As for the other solutions in this example, straightforward application of Proposition 5.3.12 shows that only *one* solution need be kept in $L_{\mathcal{L}}(1)$, and *two* in $L_{\mathcal{L}}(2)$ (see fig. 5.3.2). The reduced number of solutions in these sets implies that when we combine these sets with those for clade $\mathcal{R}$, a smaller number of solutions for clade $\mathcal{T}$ are generated: only two in $L_{\mathcal{T}}(1)$ and four in $L_{\mathcal{T}}(2)$. Note that, because of the new cleaning idea, sometimes $L_{\mathcal{T}}(b)$ fails to contain any solution lying in some vertex of $CH_{\mathcal{T}}(b)$ (this happens here for the top vertex of $CH_{\mathcal{T}}(1)$). This is not a problem as the solutions that are not derived would have been discarded anyway by application of Proposition 5.3.12.

Returning to the theory, I note that an additional criterion to discard solutions from $L_{\mathcal{T}}(b)$ may be devised by deriving a *lower* bound (instead of an upper bound) to the endangered portion $d_{\mathcal{T}}(\mathbf{y})$, leading to potentially discarding solutions $\mathbf{x}$ whose $p_{\mathbf{x}}$ is too small to guarantee sufficient protection of the path above their clade (this will only happen when such lower bound is large). I have not fully explored this possibility yet. Computational experiments show that the efficiency gains allowed by Proposition 5.3.12 are already substantial.

FIGURE 5.3.2. Effect of the new cleaning step on the execution of the algorithm.
Points correspond to the solutions that the algorithm generates. Circled points correspond to the solutions that are kept in the various $L_{\mathcal{T}}(b)$ after the new cleaning step.



5.4. Examples

A toy example illustrating the behaviour of the algorithm has already been presented in the previous section (see fig. 5.3.1).

Figure 5.4.1 describes an instance of the GNAP with relevance to the conservation of lemurs in Madagascar. It is the natural successor of the examples in figures 2.8.1, 3.7.1, 4.4.1 which were instances of MAXPD, BMAXPD and of the $a_i \xrightarrow{c_i} 1$ NAP, respectively. For each taxon i, we assume that its probability of extinction reduces as an exponential function of its conservation expenditure. That is, we have that

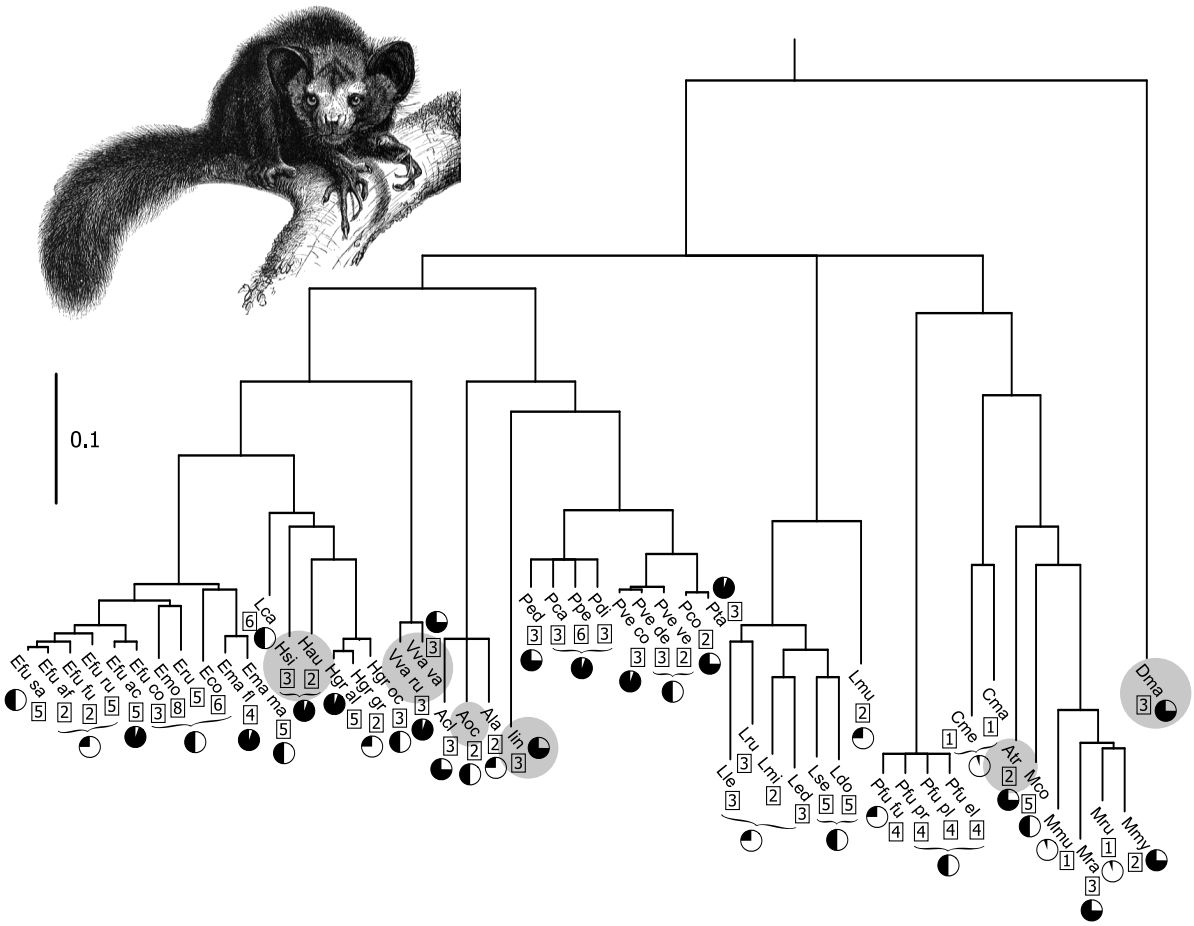
$$p_i(x) = 1 - (1 - a_i)e^{-\frac{x}{c_i}},$$

where the various a_i and c_i are depicted to the side of their taxa (in circles and boxes, respectively). All these functions are depicted in figure 5.4.2 as either coloured lines (for taxa Aoc, Atr, Dma, Hau, Hsi, Iin, Vva_ru, Vva_va) or grey dashed lines. The budget is assumed to be $B = 19$.

The optimal solution to this instance of the GNAP was calculated with a program I describe and experiment with in the next section. This solution consist in spending 6 on Dma, 4 on Iin, 3 on Atr, 2 on Vva_ru, and 1 on Aoc, Vva_va, Hau and Hsi (and

FIGURE 5.4.1. An instance of the GNAP relevant to the conservation of Madagascar lemurs.

The phylogenetic tree is the same as in fig. 2.8.1. The budget is $B = 19$. The survival probability of taxon i, assuming expenditure x on i, is defined by $p_i(x) = 1 - (1 - a_i)\exp(-x/c_i)$, where a_i and c_i are given as follows. For each taxon i, the probability of survival a_i (one of 5%, 25%, 50%, 75%, and 95%) if no conservation action is taken is represented by the white area in the adjacent circle. These probabilities are the same as in figure 4.4.1 (see the legend there for details). Inside the boxes I indicate the "costs" c_i of each taxon (the same as the ones in fig. 3.7.1), which here should be thought of as a measure of how expensive it is to reduce the taxon's risk of extinction. Spending c_i on i results in the same reduction (about 63%) of extinction risk regardless of the taxon. Highlighted in gray are the eight taxa towards which the optimal solution to this instance of the GNAP distributes expenditures (see also figure 5.4.2). See the Appendix for the species and subspecies corresponding to the taxon identifiers used here.

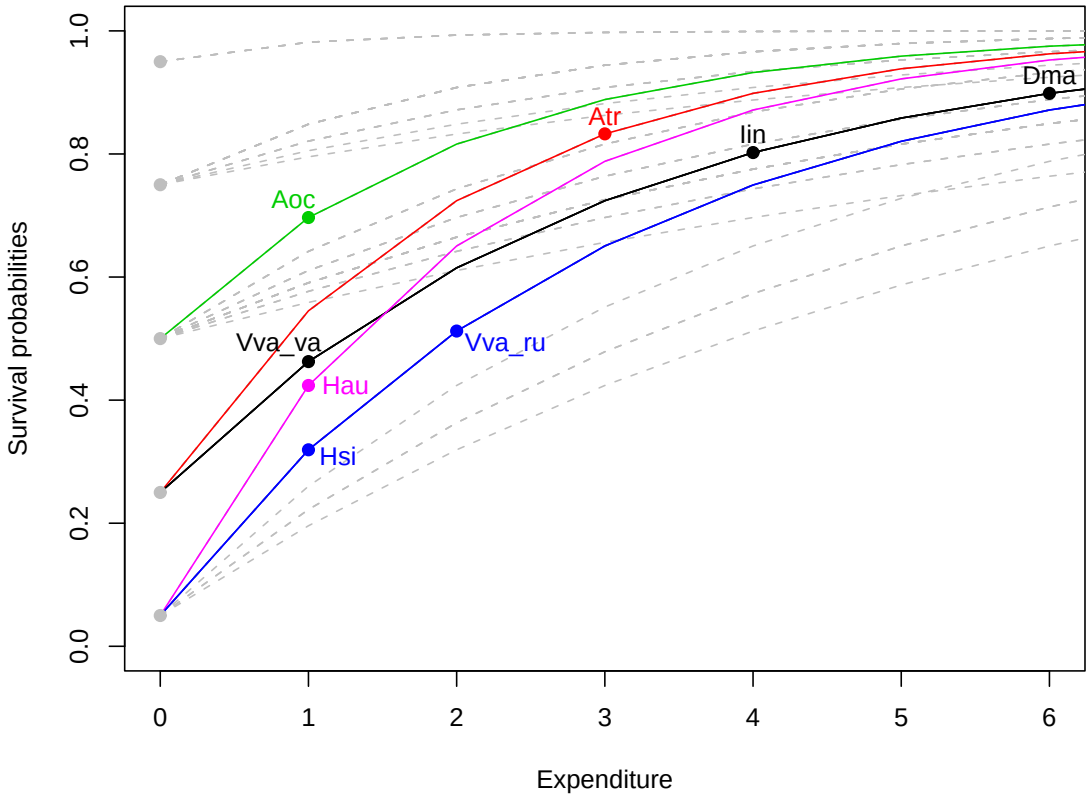


0 on the remaining taxa). The eight taxa with non-zero expenditure in this solution are highlighted in figure 5.4.1 and their survival probability curves are coloured in figure 5.4.2.

Figure 5.4.2 also shows the optimal amount of spending assigned to each taxon. In contrast with what happens for the versions of the NAP that I have presented in

FIGURE 5.4.2. Survival probability functions in the instance of figure 5.4.1.

For each taxon i in figure 5.4.1, its survival probability $p_i(x) = 1 - (1 - a_i)\exp(-x/c_i)$ is plotted. (Note that only integer expenditures are allowed by the GNAP.) For the taxa that do not get assigned any expenditure in the optimal solution to this instance, the plot consists of gray dashed lines. For the remaining eight taxa, coloured solid lines are used. For each taxon, a small filled circle indicates its optimal expenditure and resulting survival probability. For the taxa with non-zero optimal expenditure, identifiers are shown to the sides of these circles. Note that for many taxa (for example Dma, Iin and Vva_va) the survival probability functions coincide. This is why we do not observe 53 different curves.



earlier chapters, which favour "extreme" solutions with expenditures either equal to zero or to the maximum value allowed, here investment on a taxon stops as soon as a certain threshold is reached. This fact is due to the concave shape of the survival probability functions: had we used affine functions, we would have again observed extreme solutions (see section 4.6).

5.5. The algorithm's efficiency and experiments

The most problematic aspect in the analysis of the algorithm that I have presented here lies in the size of the $L_{\mathcal{T}}(b)$ sets. Although it is easy to see that these sets cannot contain more solutions than the number of feasible solutions for $\mathcal{T}$ and b, that is,

$$\left(\!\binom{|\mathcal{T}|}{b}\!\right) = \binom{|\mathcal{T}| + b - 1}{b},$$

where $|\mathcal{T}|$ denotes the the number of taxa in $\mathcal{T}$ and the ones in double parentheses are multiset coefficients, practice suggests that, typically, $|L_{\mathcal{T}}(b)|$ is much smaller than that. In the second part of this section, I will present experiments which show that in many realistic cases we can expect this number to be very small.

It is important to distinguish between the size that $L_{\mathcal{T}}(b)$ has *before* and *after* a number of its solutions have been discarded thanks to the Graham's scan described in Algorithm 7 and the additional idea of section 5.3.3. For the purpose of analysing the algorithm, it is sufficient to consider the second quantity. Let then L be an upper bound to the number of solutions that remain in $L_{\mathcal{T}}(b)$ *after* the "cleaning of dominated solutions" has taken place. We can imagine that L is some (unknown) function of the input parameters n (the number of taxa) and B (the budget).

It is then easy to provide an asymptotic analysis of our algorithm: it consists of the derivation of $O(Bn)$ $L_{\mathcal{T}}(b)$ sets; the construction of each $L_{\mathcal{T}}(b)$ set requires that we combine together $O(b)$ pairs of other $L_{\mathcal{T}'}(i)$ sets (the innermost for loop in Algorithm 6). Each combination can be done in $O(L^2)$ time, as at most L^2 pairs of solutions must be combined and each combination can be done in constant time (provided that solutions are represented as points); this results in $O(bL^2)$ points in $L_{\mathcal{T}}(b)$; since it is easy to check that CLEAN(S) runs in $O(|S| \log |S|)$ time, the construction and subsequent "cleaning" of $L_{\mathcal{T}}(b)$ requires in total $O(bL^2 + bL^2 \log bL^2) = O(bL^2 \log bL)$ time. Since this must be repeated $O(Bn)$ times, the total running time of the algorithm is $O(B^2 n L^2 \log BL)$.

The dominating factor for memory requirements is again the size of the stored $L_{\mathcal{T}}(b)$ sets. We already saw that the size of $L_{\mathcal{T}}(b)$ before CLEAN is called is $O(bL^2)$. Since at most one such "uncleaned" set must be stored at any given time, and the storage of all cleaned $L_{\mathcal{T}}(b)$ requires $O(BnL)$ space, the algorithm presented here has a space complexity of $O(BL^2 + BnL)$.

Since all the asymptotic bounds I presented depend on L, which seems to be hard to analyse theoretically, it remains to be seen whether the algorithm runs efficiently in practice. I have therefore implemented the algorithm with a program called `rats` (C++ sources and documentation available online at http://www.ebi.ac.uk/ goldman/rats/) and practice shows that its performance is acceptable for many randomly-generated instances of the GNAP. The rest of this section provides support for this claim.

TABLE 1. Average (±s.d.) time and memory requirements of `rats` over 2000 random instances of GNAP of size n.

n	CPU time (s)	memory (MB)	$\max\|L_{\mathcal{T}}(b)\|$	$\max\|L'_{\mathcal{T}}(b)\|$
200	0.47 ± 0.02	$2 \pm 0^*$	95.5 ± 27	3.38 ± 0.65
500	5.50 ± 0.05	$2 \pm 0^*$	123 ± 30	3.94 ± 0.66
1000	40.3 ± 0.33	335 ± 59	143 ± 33	4.32 ± 0.64
2000	314 ± 3.4	1394 ± 30	168 ± 36	4.74 ± 0.68
5000	4751 ± 37	9029 ± 12	200 ± 39	5.30 ± 0.70

*: these results are probably artefacts of the resource usage reporting system.

In the absence of a clear understanding of what constitutes a "typical" instance of the GNAP, I have chosen a deliberately simple simulation framework (in fact, the simplest I could think of). For each n in $\{200, 500, 1000, 2000, 5000\}$, 2000 instances of the GNAP were each constructed in the following way:

- a tree on n taxa was generated according to a Yule-Harding model [**Yul24, Har71, SS03**] with exponentially-distributed branch lengths; the program PDA [**MKvH06**] was used for this task;
- the budget was set to $B = 2n$;
- for each taxon $i \in \{1, 2, \ldots, n\}$, a random "cost" c_i was generated according to a Poisson distribution with expectation 10, and two random probabilities of survival were generated following a uniform distribution in the interval $[0, 1]$; call a_i and b_i the smaller and the larger of these, respectively; the probability of survival of taxon i, given expenditure $x \in \{0, 1, \ldots, B\}$ is then defined by:

$$p_i(x) = \begin{cases} a_i + x(b_i - a_i)/c_i & \text{if } x \leq c_i, \\ b_i & \text{if } x > c_i. \end{cases}$$

  Note that the choice of this particular form for the survival probability functions makes these instances particular cases of a discretised version of Weitzman's NAP (discussed in sec. 4.6).

The program `rats` was run with each of these instances in input, on machines with Intel Xeon L5420 / 2.50 GHz processors and 32 GB memory. A few parameters describing the used resources were recorded.

Table 1 shows, for each n, the average and standard deviation (over the 2000 instances) of four such parameters: (1) the CPU time employed to reach termination, (2) the amount of memory used, (3) the maximum number of points stored in any $L_{\mathcal{T}}(b)$, which I denote by $\max|L_{\mathcal{T}}(b)|$, and (4) the maximum number of points stored in any $L_{\mathcal{T}}(b)$ *after* the cleaning of dominated solutions has taken place, which I denote by $\max|L'_{\mathcal{T}}(b)|$. Note that $\max|L_{\mathcal{T}}(b)|$ and $\max|L'_{\mathcal{T}}(b)|$ refer to a *single* execution of `rats`, so that the numbers indicated in these columns are averages (±s.d.) over 2000 maxima.

The first and most important thing to note about these results is that the approach I described in this chapter does indeed work for instances of large size (thousands of taxa). A possible concern is that the sizes of the $L_{\mathcal{T}}(b)$ sets may grow too quickly, making the program inefficient even for moderate values of n. This does not happen for any of the instances I generated. Although this may still happen for instances constructed in a different way, these experiments show that for many realistic instances the algorithm runs rather efficiently. The main concern with the current implementation is the amount of memory used (9GB with 5000 taxa), something that could be important to address with future improvements of the code.

Second, if we treat L as a constant, because we have here $B = 2n = O(n)$, the analyses above predict that `rats` should run in $O(n^3 \log n)$ time and $O(n^2)$ space (in fact, it would be more precise to write $\Theta(n^3 \log n)$ time and $\Theta(n^2)$; see [**CLRS01**], sec. 3.1). The first two columns in Table 1 are largely consistent with these predictions, as each time n doubles, times get approximately multiplied by a factor of 8 and memory usages by a factor of 4 (ignoring the first two results in the second column, which are clearly artefactual).

Third, the growth of $\max|L_{\mathcal{T}}(b)|$ and $\max|L'_{\mathcal{T}}(b)|$ is surprisingly slow. This is very good news and it is what allows the algorithm to run efficiently. It seems that these numbers grow less than linearly with n, which may indicate that, for data generated in the way I described, on average L also grows less than linearly with n or that it is a constant. (The fact that these maxima increase with n may simply be due to the fact that they are maxima over a growing sample of numbers.)

Fourth, the standard deviations of the resource usages are extremely small. This is perhaps not surprising: the only elements of randomness in the execution of `rats` are the sizes of the $L'_{\mathcal{T}}(b)$ sets. If these sets always had the same size, then every execution of `rats` should require the same amount of time and memory (also because I ran `rats` on identical machines). Since in our experiments $|L'_{\mathcal{T}}(b)|$ is always of the order of the units, it is not surprising that resource usages vary so little.

It remains, however, to be established whether other realistic instances of the GNAP may cause the $L_{\mathcal{T}}(b)$ sets to have larger sizes than the ones observed in my experiments (and therefore cause much more variable behaviours of `rats`). The simulation framework I adopted here probably samples a narrow portion of the scenarios one may encounter in practice. Probably the most limiting factor in my simulations is represented by the survival probability functions, which are here assumed to grow linearly in the conservation expenditures. The GNAP allows much more general cost-benefit relationships than linear ones (as shown in sec. 5.4) and the algorithm may have a somewhat different behaviour when other relationships are assumed. However, it is still interesting to investigate the performance of our algorithm on instances assuming linear cost-benefit relationships, as these are discretised versions of the classical Noah's Ark problem by M.Weitzman [**Wei98**].

5.6. Related and future work

The choice of the name "generalised Noah's Ark problem" for the problem presented in this chapter was motivated by its use by Hartmann and Steel [**HS06**], who show cases where a greedy algorithm is guaranteed an optimal solution. In addition to the cases I have already discussed (sec. 4.8), they also show the greedy tractability of the scenario where the survival probabilities are given by:

$$p_i(x) = 1 - (1 - a_i)e^{-\frac{x}{c}}. \tag{5.6.1}$$

Note that this does not allow taxon-specific rates of convergence $1/c_i$, and so the instance of section 5.4 is not a particular case of this scenario.

What both the functions in (5.6.1) and the ones in section 5.4 are particular cases of is the class of functions $p_i(x)$ such that $\log(1 - p_i(x))$ is convex. This class is interesting, because apart from being reasonably realistic (see fig. 5.4.2), it also makes the continuous version of the GNAP solvable with standard techniques of convex optimisation. For this observation I am indebted to Tim Massingham. In brief, the objective function of the GNAP can be expressed as:

$$\mathbb{E}[\varphi|\mathbf{x}] = \sum_e t_e \cdot \left(1 - \prod_{i \in C(e)} (1 - p_i(x_i))\right).$$

If we assume that the various $\log(1 - p_i(x))$ are convex, then so is their sum:

$$\sum_{i \in C(e)} \log(1 - p_i(x)) = \log \prod_{i \in C(e)} (1 - p_i(x_i)).$$

But if the logarithm of a function is convex then also that function is convex. Therefore $\prod_{i \in C(e)}(1 - p_i(x_i))$ is convex and its opposite

$$1 - \prod_{i \in C(e)} (1 - p_i(x_i))$$

is concave. Since the linear combination of concave functions is concave, we conclude that the objective function of the GNAP is concave. The continuous version of the GNAP therefore consists of maximising a concave function over the convex set $\{\mathbf{x} :$ for every i, $x_i \geq 0$ and $\sum_i x_i \leq B\}$. For such optimisation problems, any local maximum of the objective function is a global maximum and it can be found with standard techniques. It will be interesting to investigate whether this observation has any implication on the applicability of greedy algorithms to the discrete version of the GNAP.

Finally, I note that an algorithm with some similarities to the one I present here has recently (and independently) been published [**HCMZ08**]. However, it deals with the $a_i \xrightarrow{c_i} b_i$ NAP and it produces an approximate solution within any desired factor from optimal. Unlike the algorithm presented here, theoretically-guaranteed polynomial bounds to its running time are provided.

Future work on the algorithm for the GNAP presented in this chapter will involve experiments on instances of the GNAP simulated with different frameworks from the one I have used here. In particular, the simulations I have used do not exploit the full power of the GNAP to express arbitrary relationships between conservation expenditure and survival probability. This is an important feature, as it allows incorporation of more realistic models for this relationship (see [**SMG$^+$03**] for a discussion of three such models). Experimenting with a wider range of (and more complex) survival probability functions would provide a better understanding of the limits of applicability of `rats`.

CHAPTER 6

Balanced minimum evolution as a criterion for tree reconstruction

6.1. BME: fast and accurate distance-based tree reconstruction

From this chapter onwards, I will deal with a more basic problem, the most central one in phylogenetics: how to infer the phylogenetic tree of a group of taxa, based on their characteristics. Among the many algorithms developed for this task, *distance methods* infer a tree based on a matrix of distances between each pair of taxa. From the user's perspective, their main characteristic is speed of execution; therefore they are used whenever this aspect is of critical importance (e.g., with many taxa, or when many trees have to be reconstructed). The distances between taxa can be estimated both from morphological and from molecular data, with various techniques which I will not discuss here. This chapter will instead introduce and review a recently proposed criterion for reconstructing a phylogenetic tree given an already estimated distance matrix: balanced minimum evolution (BME). Although some new results will be presented, this chapter mainly serves as an introduction to my work on the algorithmics of this criterion, which is the subject of the next chapter.

BME is a particular type of minimum evolution (ME) tree reconstruction. An overview of distance methods with particular emphasis on ME is given in section 6.2, which also introduces the formalisms used throughout this chapter. The distinctive aspect of BME is that it is based on a succinct formula $\Lambda_{\boldsymbol{\delta}}(T)$ for estimating the total branch length of a tree based on its topology T and the given distance matrix $\boldsymbol{\delta}$. This formula, the motivations behind its introduction and its capacity to discriminate between the correct and incorrect trees are the subject of section 6.3. BME aims to reconstruct the tree T that minimises $\Lambda_{\boldsymbol{\delta}}(T)$, the *balanced length* of T.

Recent interest in BME is due to both theoretical and practical reasons. Some of the most basic theoretical properties of the balanced length are shown in section 6.4 (for the first time, I believe). More importantly, BME sheds light on one of the most popular distance methods, neighbor-joining (NJ): recently, it was shown that NJ can be viewed as a greedy algorithm aiming to minimise the balanced length $\Lambda_{\boldsymbol{\delta}}(T)$. This result is reviewed in detail in section 6.5. Furthermore, section 6.6 proves that if the distances in $\boldsymbol{\delta}$ are within a simple "safety radius" of the correct distances, then the BME-optimal tree coincides with the correct one — the main novel result of this chapter.

On the practical side, it is interesting to note that the balanced length formula was initially proposed as a computationally fast approach to approximate a more traditional ME method. Later it turned out that using this formula not only speeds up computation (even with respect to NJ), but also leads to tree reconstructions that are among the most accurate for distance methods. These practical advantages are described in section 6.7.

Finally, section 6.8 discusses other known properties of BME and some related problems that still need to be solved.

6.2. Distance methods and minimum evolution

All distance methods for tree reconstruction have as input an $n \times n$ symmetric *distance matrix* $\boldsymbol{\delta} = (\delta_{ij})$, where $\{1, 2, \ldots, n\}$ are the taxa under consideration and δ_{ij} is the distance between taxa i and j. The only constraints on this matrix are that, for every $i, j \in \{1, 2, \ldots, n\}$, $\delta_{ij} = \delta_{ji}$ and that $\delta_{ii} = 0$.

These distances can be estimated with all kinds of methods, differing both in the nature of the data they use (DNA, protein sequences, morphological features etc.) and in the actual estimation techniques they use (see, e.g., [**Fel03**], Ch.'s 11, 13, 23). I will not describe these methods, as here I am uniquely concerned in the process of reconstructing a tree starting from a distance matrix $\boldsymbol{\delta}$. Nevertheless, it is important to note that the data used to estimate the distances is often seen as the end-product of a random process along the branches of an unknown tree $\mathcal{T}$ with branch lengths. This means that the distances in $\boldsymbol{\delta}$ should be treated as *estimates* of the distances between taxa on this unknown tree.

In order to make this more precise, recall that I will use calligraphic fonts such as $\mathcal{T}$ and $\mathcal{W}$ to indicate trees with lengths associated with their branches, and normal fonts such as T and W to indicate tree topologies (see Chapter 1). Given a tree with branch lengths, $\mathcal{T}$, and two of its taxa i and j, the distance between i and j on $\mathcal{T}$ is defined as the sum of the lengths of the branches in the path between i and j in $\mathcal{T}$. This is denoted by $\mathrm{d}_{ij}^{\mathcal{T}}$. The $n \times n$ matrix containing these distances is denoted by $\mathbf{d}^{\mathcal{T}}$. For example, figure 6.3.2 shows a tree $\mathcal{T}$ and its corresponding distances $\mathbf{d}^{\mathcal{T}}$.

We can therefore assume that $\boldsymbol{\delta}$ is an estimate of (and hopefully approximates) $\mathbf{d}^{\mathcal{T}}$, for some unknown tree $\mathcal{T}$. The central question that distance methods address then is: can we reconstruct $\mathcal{T}$ from $\boldsymbol{\delta}$? With a few exceptions, distance methods can be described by specifying two independent components: (1) a way to fit the branch lengths of any given topology T to the given distance matrix $\boldsymbol{\delta}$, and (2) a criterion to choose among the trees generated by step (1) (which have different topologies).

Component (1) is aimed at producing a tree $\mathcal{T}$ with the specified topology T so that the distances on $\mathcal{T}$, $\mathbf{d}^{\mathcal{T}}$, are as "close" as possible to those in the input matrix $\boldsymbol{\delta}$. Different methods differ by how they define "closeness" between distance matrices: for example, ordinary least squares (OLS) assigns branch lengths so that the sum $\sum_{ij}(\delta_{ij} - \mathrm{d}_{ij}^{\mathcal{T}})^2$ is minimised [**CSE67**], whereas weighted least squares (WLS) aims

to minimise a weighted version of that sum [**FM67**]. Note that most of the proposed methods do not constrain branch lengths to be positive.

As for component (2), some distance methods use the same criterion used to define closeness in step (1) and look for the tree topology that, once assigned branch lengths, produces the tree $\mathcal{T}$ with the "closest" $\mathbf{d}^{\mathcal{T}}$ to the input matrix $\boldsymbol{\delta}$. Minimum evolution (ME) uses a different criterion, the *length* of $\mathcal{T}$, defined here as the sum of all its branches' lengths [**RN92**]. (Note that, although other definitions have been proposed to deal with negative branch lengths [**KSZ71, SOWH96, GBD01**], this sum may include negative terms.) ME aims to reconstruct the topology that, once assigned branch lengths with step (1), produces the tree of minimum length [**KSZ71, RN92**].

Different ME methods thus differ essentially in component (1) — the way they fit branch lengths — and any of the existing methods for this task can be used. For example we may use OLS to assign branch lengths to any topology we consider and seek the topology that produces the shortest tree; we may call this method OLSME. In the same way, we may call WLSME the combination of WLS for (1) and ME for (2).

So what method does BME use for fitting the branch lengths in a topology? One possible answer is that it uses a "balanced" scheme derived from the formulae for OLS branch length estimation (sec. 6.3.1). Alternatively, one can think of BME as a method that skips step (1) altogether (sec. 6.3.2). This is explained in detail in the next section.

6.3. Balanced tree length estimation: Pauplin's formula

The central aspect of BME is that it is based on a formula for estimating the length of a tree without explicitly estimating the lengths of its branches. In the simplest case, when the tree is bifurcating, this formula has the following form [**Pau00**]:

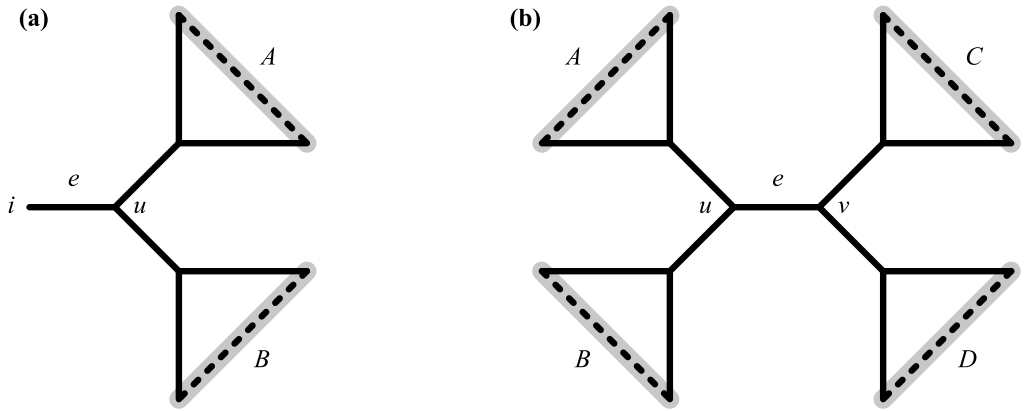
$$\Lambda_{\boldsymbol{\delta}}(T) = \sum_{i<j} \frac{1}{2^{b_{ij}-1}} \delta_{ij}, \tag{6.3.1}$$

where i and j are taxon indices in $\{1, 2, \ldots, n\}$ and where b_{ij} denotes the number of branches on the path from i to j in T. The $\boldsymbol{\delta}$ subscript will sometimes be dropped from the $\Lambda_{\boldsymbol{\delta}}(T)$ notation when not needed. Equation (6.3.1) is known as Pauplin's formula [**Pau00**].

The next two subsections present two logically different ways of deriving Pauplin's formula. Although these have been shown elsewhere [**Pau00, SS04**], I give a personal account of these derivations and the motivations behind them.

Subsection 6.3.1 will present the original way (6.3.1) was introduced: if we estimate branch lengths with a simple modification of the analytical formulae for the OLS branch lengths [**Vac89, RN93**], then the total tree length is equal to the value in (6.3.1). In fact, this formula was first proposed by Pauplin [**Pau00**] as a

FIGURE 6.3.1. Clades involved in the estimation of the length of branch e.



fast method to calculate an approximation to the OLS tree length, one that skips altogether the estimation of each individual branch length.

Subsection 6.3.2 shows that Pauplin's formula can also be obtained as the average of many "circular estimates" of the tree length. This can be generalised to trees containing multifurcations, leading to a generalisation of (6.3.1) [**SS04**]. In this framework, it is also easy to prove that the tree that minimises Pauplin's formula coincides with the correct tree $\mathcal{T}$, provided that $\boldsymbol{\delta}$ is sufficiently close to $\mathbf{d}^{\mathcal{T}}$ [**DG04, DG05**]. This result — a stronger version of which is proven in section 6.6 — can be seen as the main theoretical justification for BME tree reconstruction.

6.3.1. Derivation from the formulae for OLS branch lengths. Vach [**Vac89**] and Rzhetzky and Nei [**RN93**] showed that OLS branch length estimation — i.e., the problem of finding the assignment to the branches of a tree $\mathcal{T}$ with a fixed topology so that $\sum_{ij}(\delta_{ij} - \mathrm{d}_{ij}^{\mathcal{T}})^2$ is minimised — can be done using relatively simple analytical formulae. These are based on a definition of average distance between clades: in this chapter, let a *clade* in a tree topology T be defined as any set of taxa lying on one side of a branch in T; then δ_{XY}^u, where X and Y are clades, denotes the expected distance between a random taxon chosen uniformly from X and one chosen uniformly from Y, that is

$$\delta_{XY}^u \quad = \quad \frac{1}{|X||Y|} \sum_{\substack{i \in X \\ j \in Y}} \delta_{ij}.$$

Using this definition, the OLS estimate of the length of a branch e in a bifurcating topology can be expressed as

$$\hat{t}_e = \frac{1}{2}(\delta_{A\{i\}}^u + \delta_{B\{i\}}^u - \delta_{AB}^u), \tag{6.3.2}$$

if e is terminal and connects taxon i to clades A and B (see fig. 6.3.1(a)) or as

$$\hat{t}_e = \frac{1}{2}\left[\alpha(\delta_{AC}^u + \delta_{BD}^u) + (1-\alpha)(\delta_{AD}^u + \delta_{BC}^u) - (\delta_{AB}^u + \delta_{CD}^u)\right], \tag{6.3.3}$$

if e is an internal branch and connects clades A and B on one side to clades C and D on the other side (see fig. 6.3.1(b)); α is a factor between 0 and 1 suitably defined on the basis of the relative sizes of A, B, C and D (not shown here), and it equals $\frac{1}{2}$ when $|A| = |B|$ or $|C| = |D|$.

Estimating the length of a branch using (6.3.2) and (6.3.3) requires $O(n^2)$ time, and therefore the calculation of the total length of the tree can be done in $O(n^3)$ time[1] [**RN93**]. Pauplin's idea was that a slight modification of these formulae — giving approximately (but not exactly) the same branch length estimates — allows the sum of all the branch length estimates to be calculated using the simple formula given in (6.3.1) [**Pau00**]. This brings down the time for estimating the total tree length from $O(n^3)$ to $O(n^2)$. Only later was it found that not only is this approach potentially faster than OLS tree length estimation, but also it allows more accurate reconstructions of the tree topology (see sec. 6.7). I will now show the modifications proposed by Pauplin, and the way they lead to (6.3.1).

The main difference between the estimates in (6.3.2) and (6.3.3) and those proposed by Pauplin is that the latter use a different, "balanced", definition of average distance between clades: in a bifurcating tree, a clade X can be naturally decomposed into two other clades X_1 and X_2 so that $X = X_1 \cup X_2$; the balanced idea consists of making sure that the average distance of X from any other clade Y is equal to the mean between the average distances of X_1 and X_2 from Y. In other words the *balanced average distance* δ_{XY} (note that this is written without superscript, to distinguish it from δ^u_{XY}) is now defined so that this holds:

$$\delta_{XY} = \frac{1}{2}(\delta_{X_1 Y} + \delta_{X_2 Y}). \tag{6.3.4}$$

This can be achieved by defining δ_{XY} again as the expected distance between two taxa chosen randomly from X and Y; however, whereas for δ^u_{XY} the taxa are drawn uniformly (hence the use of the superscript), here each taxon is chosen from its clade in the following way: imagine a random walk that starts from the *root of the clade* (i.e., the root of the restriction of the input tree on the clade; see Def. 2.5.1) and then goes towards the leaves so that, each time a new node is entered, the walk continues along a uniformly chosen branch among the ones attached to that node (excluding the branch used to enter the node). Clearly this walk will end up in a leaf of the tree, and the corresponding taxon is the chosen one. For bifurcating trees we then have:

$$\delta_{XY} = \sum_{\substack{i \in X \\ j \in Y}} \frac{1}{2^{b_{iX}}} \frac{1}{2^{b_{jY}}} \delta_{ij} = \sum_{\substack{i \in X \\ j \in Y}} \frac{1}{2^{b_{ij} - b_{XY}}} \delta_{ij}, \tag{6.3.5}$$

[1] This assumes a naive algorithm for the calculation of (6.3.2) and (6.3.3). Following Rzhetzky and Nei [**RN93**], other authors proposed faster algorithms that allow calculation of $\hat{t}_e$ for all branches in $O(n^2)$ time [**Gas97b, BW98, MP08**].

where b_{iX}, b_{jY} and b_{XY} denote the number of branches between the root of clade X and taxon i, between the root of Y and j and between the roots of X and Y, respectively. Intuitively, whereas δ^u_{XY} is an unweighted average of the various δ_{ij} for $i \in X$ and $j \in Y$, in δ_{XY} the distances between taxa that are separated by few branches get large weights, whereas those between more topologically distant taxa get smaller weights. Giving different weights to different distances has a statistical rationale: as I have already observed, the various δ_{ij} can be seen as estimates of the correct distances $d^{\mathcal{T}}_{ij}$ on the correct tree; however we can expect that some of these estimates will be more precise than others and therefore we should give higher weights to the estimates that we consider more precise. The implicit assumption in the balanced average distances above is that the precision of the distances δ_{ij} decreases as we increase the number of branches b_{ij} between i and j.

If we use δ_{XY} instead of δ^u_{XY} in (6.3.2) and (6.3.3) and if we set $\alpha = 1/2$, in line with the principle of balancedness, then the formulae for estimating branch lengths in a bifurcating topology become

$$\hat{t}_e = \frac{1}{2}(\delta_{A\{i\}} + \delta_{B\{i\}} - \delta_{AB}), \tag{6.3.6}$$

and

$$\hat{t}_e = \frac{1}{4}(\delta_{AC} + \delta_{BD} + \delta_{AD} + \delta_{BC}) - \frac{1}{2}(\delta_{AB} + \delta_{CD}). \tag{6.3.7}$$

The observation that motivates the introduction of these new estimates is that adding them all together precisely leads to Pauplin's formula:

$$\sum_e \hat{t}_e = \sum_{i<j} \frac{1}{2^{b_{ij}-1}} \delta_{ij}.$$

This can be shown by noting that, for each pair of taxa $i < j$, the term $\frac{1}{2^{b_{ij}-1}}\delta_{ij}$ appears in the sum above b_{ij} times with a positive sign (once for each $\hat{t}_e$ with e on the path between i and j) and $b_{ij} - 1$ times with a negative sign (once for each $\hat{t}_e$ with e incident – but not belonging – to the path between i and j) (see [**Pau00**] for more detail). Note that normally the branch length estimates obtained with (6.3.6) and (6.3.7) are not equal to the OLS estimates, and therefore also the total tree length is different.

There is however an important property that the balanced branch length estimates given by (6.3.6) and (6.3.7) share with the OLS estimates: if the distances in $\boldsymbol{\delta} = (\delta_{ij})$ coincide with those on a tree $\mathcal{T}$, then the estimates $\hat{t}_e$ for the branches of $\mathcal{T}$ also coincide with their correct values t_e — this is an important requirement as, typically, the distances $\boldsymbol{\delta}$ approximate those for the correct tree. In order to see this, consider again the right-hand sides of (6.3.6) and (6.3.7): if we substitute A with any taxon from A, B with any taxon from B, C with any taxon from C and D with any taxon from D, then it is easy to see that these formulae give the correct value for t_e. The right hand-sides of (6.3.6) and (6.3.7) — and similarly those of (6.3.2)

and (6.3.3) — can then be seen as obtained by averaging over a large number of formulae, each giving the correct length of e.

Another important observation regarding the balanced branch length estimates — one that was noted only some time after their introduction — is that they coincide with the WLS branch length estimates assuming weights proportional to $2^{-b_{ij}}$. In other words, if we are given a bifurcating topology T and we seek the assignment of branch lengths giving the tree $\mathcal{T}$ that minimises

$$\sum_{ij} 2^{-b_{ij}} (\delta_{ij} - \mathrm{d}_{ij}^{\mathcal{T}})^2, \tag{6.3.8}$$

then the solution to this problem is precisely given by the the formulae in (6.3.6) and (6.3.7) [**DG04, MP08**].

There are statistical reasons that may justify the use of exponential weights in (6.3.8). Discussing them here would be outside the scope of this introduction to BME, but the interested reader can consult Felsenstein's textbook ([**Fel03**], pp. 153–154) for the general statistical meaning of weights in WLS and [**DG04**] for a discussion of the exponential weights in particular (see also [**NSS85, NJ89, Bul91, BSH00**]).

6.3.2. Balanced tree length as an average of circular estimates. The fundamental question that needs to be addressed by every minimum evolution method is: given a distance matrix $\boldsymbol{\delta}$ that approximates that of a tree of known topology T, how can we estimate the length of this tree?

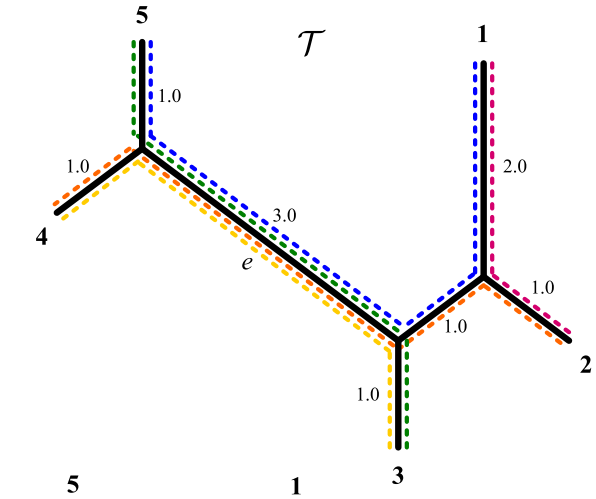
A very simple way to answer this question is to sum together the distances between taxa that are consecutive when traversing T in a clockwise manner and then taking half of the sum thus obtained. Consider the example in figure 6.3.2: the clockwise traversal of T leads to visiting the taxa in the order $o = (1, 2, 3, 4, 5)$, and therefore to the following length estimate:

$$\Lambda_{\boldsymbol{\delta}}(o) = \frac{1}{2}(\delta_{12} + \delta_{23} + \delta_{34} + \delta_{45} + \delta_{51}).$$

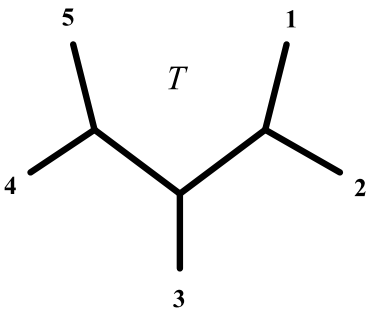
The reason why this works is that if $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$ for some tree $\mathcal{T}$ with topology T (as assumed in fig. 6.3.2), then δ_{12} equals the sum of the lengths of the branches in the path between taxa 1 and 2, δ_{23} equals the sum of the lengths of the branches in the path between 2 and 3, and so on, down to δ_{51}, which equals the sum of the lengths of the branches in the path between 5 and 1. It is easy to see that each branch belongs to exactly two of these paths and thus its length is counted twice in the sum above. Therefore, taking half of the sum precisely gives the length of $\mathcal{T}$, and this should still be approximately true if $\boldsymbol{\delta}$ only approximates $\mathbf{d}^{\mathcal{T}}$.

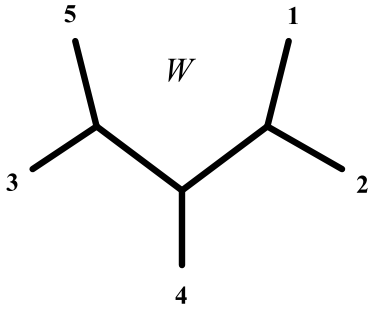
It is important to note that the formula above depends on the particular way one chooses to draw the tree topology. If in figure 6.3.2 we swap taxa 4 and 5 in T, the topology remains the same but the taxon order obtained with a clockwise traversal is now $o' = (1, 2, 3, 5, 4)$, leading to a different tree length estimator $\Lambda_{\boldsymbol{\delta}}(o') = (\delta_{12} + \delta_{23} + \delta_{35} + \delta_{54} + \delta_{41})/2$.

FIGURE 6.3.2. A tree $\mathcal{T}$ and its corresponding distance matrix $\mathbf{d}^{\mathcal{T}}$. Two alternative topologies T and W, their circular orderings and resulting tree length estimates assuming $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$.



$$\mathbf{d}^{\mathcal{T}} = \boldsymbol{\delta} = \begin{bmatrix} 0 & 3 & 4 & 7 & 7 \\ 3 & 0 & 3 & 6 & 6 \\ 4 & 3 & 0 & 5 & 5 \\ 7 & 6 & 5 & 0 & 2 \\ 7 & 6 & 5 & 2 & 0 \end{bmatrix}$$





o	$\Lambda_\delta(o)$
(1,2,3,4,5)	10
(1,2,3,5,4)	10
(1,2,4,5,3)	10
(1,2,5,4,3)	10
(1,3,4,5,2)	10
(1,3,5,4,2)	10
(1,4,5,3,2)	10
(1,5,4,3,2)	10

o	$\Lambda_\delta(o)$
(1,2,4,3,5)	13
(1,2,4,5,3)	10
(1,2,3,5,4)	10
(1,2,5,3,4)	13
(1,4,3,5,2)	13
(1,4,5,3,2)	10
(1,3,5,4,2)	10
(1,5,3,4,2)	13

To generalise the arguments above, consider all the possible permutations of taxa that can be obtained by drawing T on the plane and then traversing the tree in a clockwise manner; these are also known as the *circular orderings* for T (see **[SS03]** for a formal definition). Any circular ordering $o = (o(1), o(2), \ldots, o(n))$ gives rise to a *circular estimate* for the tree length:

$$\Lambda_{\boldsymbol{\delta}}(o) = \frac{1}{2} \sum_{i=1}^{n} \delta_{o(i),o(i+1)},$$

where I adopt the convention that $o(n+1) = o(1)$. Note that the circular estimate above is defined more generally for any permutation o of the taxa indexing δ. As

already observed for the case in figure 6.3.2, the following proposition (first proved by Semple and Steel [**SS04**]) holds:

PROPOSITION 6.3.1. *If $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$ for some tree $\mathcal{T}$ and if o is a circular ordering for $\mathcal{T}$, then $\Lambda_{\boldsymbol{\delta}}(o)$ is equal to the length of $\mathcal{T}$.*

PROOF. For each branch e in $\mathcal{T}$, at least one of the two clades on its sides — call them $C(e)$ and $D(e)$ — must be of the form $\{o(i), o(i+1), \ldots, o(j-1), o(j)\}$, with $1 \leq i \leq j \leq n$. Consider all the n paths in $\mathcal{T}$ between consecutive taxa in o (including the path between $o(n)$ and $o(1)$): two of these paths link $C(e)$ to $D(e)$ — the one between $o(i-1)$ and $o(i)$ and the one between $o(j)$ and $o(j+1)$ — whereas all the other paths are internal to one of the two clades. Therefore branch e belongs to exactly two of these paths and its length is counted twice in the sum $\sum_i \delta_{o(i),o(i+1)}$: once in $\delta_{o(i-1),o(i)}$ and once in $\delta_{o(j),o(j+1)}$. Since this is true for every branch in $\mathcal{T}$, taking half of this sum therefore gives the length of $\mathcal{T}$. □

Figure 6.3.2 shows all the eight circular orderings for the correct topology and for an alternative topology. For each of these orderings, the value of the corresponding $\Lambda_{\boldsymbol{\delta}}(o)$ is also shown. As just proved, the $\Lambda_{\boldsymbol{\delta}}(o)$ for the circular orderings of the correct topology T all give the correct tree length. On the other hand, some of the $\Lambda_{\boldsymbol{\delta}}(o)$ for the alternative topology W overestimate the tree length. To understand why this happens, consider the first circular ordering listed for W: $o'' = (1, 2, 4, 3, 5)$. The circular estimate for this ordering is $\Lambda_{\boldsymbol{\delta}}(o'') = (\delta_{12} + \delta_{24} + \delta_{43} + \delta_{35} + \delta_{51})/2$ $(= 13.0)$. Because $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$, this corresponds to summing the lengths of the paths in $\mathcal{T}$ between taxa that are consecutive in o''; in figure 6.3.2 these paths are indicated by the coloured dotted lines to the sides of the branches of $\mathcal{T}$. By observing these paths, it is clear that branch e belongs to *four* of them and therefore its length is counted *twice* in $\Lambda_{\boldsymbol{\delta}}(o'')$, instead of just once: in fact $\Lambda_{\boldsymbol{\delta}}(o'')$ is greater than the correct tree length by precisely the length of e (3.0). In general, we have that: [**SS04**]

PROPOSITION 6.3.2. *If $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$ for some tree $\mathcal{T}$ and if the permutation o of the taxa in $\mathcal{T}$ is not a circular ordering for $\mathcal{T}$, then $\Lambda_{\boldsymbol{\delta}}(o)$ is greater than the length of $\mathcal{T}$ by at least the length of the shortest interior branch in $\mathcal{T}$.*

PROOF. Using similar arguments to those in the proof of Proposition 6.3.1, it is not difficult to show that if o is not a circular ordering for $\mathcal{T}$, then some of $\mathcal{T}$'s internal branches belong to four or more of the paths between consecutive taxa in o (in any case an even number). Therefore the length of these branches is counted more than once in $\Lambda_{\boldsymbol{\delta}}(o)$. □

This observation is very relevant to a question that is more general than the one I opened this section with: given a distance matrix $\boldsymbol{\delta}$ that approximates that of a tree $\mathcal{T}$ of *unknown* topology T^*, how can we identify T^*? Assume that T^* is bifurcating and, for the moment, that $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$ (as in fig. 6.3.2). Then the only topology whose circular

orderings give all the correct estimate $\Lambda_{\boldsymbol{\delta}}(o)$ for the tree length is the correct topology T^*. Any other topology will have some circular orderings that are not circular orderings for T^*, and therefore will give an overestimate of the correct tree length (this will be proved in detail in the proof of Theorem 6.3.3). Even without knowing the correct tree length, in principle it is then possible to identify the correct topology: it is the one whose circular orderings consistently give the smallest estimates $\Lambda_{\boldsymbol{\delta}}(o)$ for the tree length. If $\boldsymbol{\delta}$ is sufficiently close to $\mathbf{d}^T$, for reasons of continuity, this should still hold.

However, calculating all the circular estimates $\Lambda_{\boldsymbol{\delta}}(o)$ is computationally infeasible. An alternative would be to calculate, for any given topology T, the *average* of the $\Lambda_{\boldsymbol{\delta}}(o)$ values across all the circular orderings of T. It turns out that it is possible to do this efficiently, without the need for calculating separately a large number of $\Lambda_{\boldsymbol{\delta}}(o)$. I will now define a distribution over all the circular orderings for T, so that the expectation of $\Lambda_{\boldsymbol{\delta}}(o)$ is well-defined. Imagine drawing T on the plane in the following random way: draw an internal node u and then draw around it all its incident branches in T so that their clockwise order around u is chosen uniformly at random among all possible orders; then iterate this procedure on all the internal nodes at the other extremes of the branches just drawn. At the end of this procedure, the entire tree has been drawn and a clockwise traversal of all the leaves identifies a circular ordering for T. This random procedure defines a distribution over all the circular orderings for T. It is possible to show that this distribution is simply uniform, but I will not need this observation in what follows.

Let us now see how to efficiently compute the expectation of $\Lambda_{\boldsymbol{\delta}}(o)$ where o is a circular ordering for T constructed randomly with the procedure described above. First, note that

$$\mathbb{E}[\Lambda_{\boldsymbol{\delta}}(o)] = \sum_{o \in O(T)} \mathbb{P}[o] \cdot \Lambda_{\boldsymbol{\delta}}(o) = \sum_{o \in O(T)} \mathbb{P}[o] \cdot \frac{1}{2} \sum_{(i,j)} \delta_{ij} I[j \text{ follows } i \text{ in } o],$$

where $O(T)$ denotes the set of all circular orderings for T and $I[j \text{ follows } i \text{ in } o]$ equals 1 or 0 depending on whether j follows i in o or not, respectively. Note that "j follows i in o" whenever there is $k \in \{1, 2, \ldots, n\}$ such that $o(k) = i$ and $o(k+1) = j$ (remember that I adopt the convention that $o(n+1) = o(1)$). Inverting the order of the summations, one obtains

$$\mathbb{E}[\Lambda_{\boldsymbol{\delta}}(o)] = \frac{1}{2} \sum_{(i,j)} \delta_{ij} \sum_{o \in O(T)} \mathbb{P}[o] \cdot I[j \text{ follows } i \text{ in } o] \; = \; \frac{1}{2} \sum_{(i,j)} \delta_{ij} \mathbb{P}[o \text{ such that } j \text{ follows } i \text{ in } o].$$

Let us now look more closely at the probability that o is such that j follows i in it. Let $(v_1 = i, v_2, \ldots, v_{b_{ij}}, v_{b_{ij}+1} = j)$ be the sequence of nodes in T in the path between i and j. The condition that j follows i in o can be restated by imposing that, in the random drawing of T that gives rise to o, for every internal node v_i ($i \in \{2, \ldots, b_{ij}\}$), v_{i+1} immediately follows v_{i-1} in a clockwise visit of all the nodes that are linked to v_i. Considering the random way the drawing is done, the probability that this

condition is satisfied for a given v_i is $1/(\deg(v_i)-1)$, where $\deg(v)$ denotes the *degree* of v, i.e., the number of branches that are attached to it. Therefore,

$$\mathbb{P}[o \text{ such that } j \text{ follows } i \text{ in } o] = \prod_{v \in P_{ij}(T)} \frac{1}{\deg(v)-1},$$

where $P_{ij}(T) = \{v_2, \ldots, v_{b_{ij}}\}$ is the set of nodes internal to the path between i and j in T. Finally,

$$\mathbb{E}[\Lambda_{\boldsymbol{\delta}}(o)] = \frac{1}{2}\sum_{(i,j)} \delta_{ij} \prod_{v \in P_{ij}(T)} \frac{1}{\deg(v)-1} = \sum_{i<j} \delta_{ij} \prod_{v \in P_{ij}(T)} \frac{1}{\deg(v)-1}.$$

Note that the above formula is a generalisation of Pauplin's formula: it reduces to it when T is bifurcating, as in this case $\prod_{v \in P_{ij}(T)} (\deg(v)-1)^{-1} = 2^{1-b_{ij}}$. It is what I call the *balanced length of* T and it is the guiding criterion for balanced minimum evolution tree reconstruction:

$$\Lambda_{\boldsymbol{\delta}}(T) = \sum_{i<j} \delta_{ij} \prod_{v \in P_{ij}(T)} \frac{1}{\deg(v)-1}. \tag{6.3.9}$$

This formula was first derived (in a slightly different way) by Semple and Steel [**SS04**]. It has several advantages compared to the circular estimates introduced above. One is that it does not depend on a particular way of drawing the topology T — as I just showed, it coincides with the mean of the circular estimates over all possible ways of drawing T. Another advantage is that whereas a single circular estimate does not necessarily discriminate between the correct and an incorrect topology, $\Lambda_{\boldsymbol{\delta}}(T)$ does have this property:

THEOREM 6.3.3. *If $\boldsymbol{\delta}$ is sufficiently close to $\mathbf{d}^{\mathcal{T}}$ for some tree $\mathcal{T}$ with positive branch lengths and with a bifurcating topology T^*, then $\Lambda_{\boldsymbol{\delta}}(W) > \Lambda_{\boldsymbol{\delta}}(T^*)$ for every "incorrect" topology $W \neq T^*$ on the same set of taxa.*[2]

This was first proved by Desper and Gascuel in 2004 [**DG04**], who gave the main ideas for a simpler proof in 2005 [**DG05**]. Below I give my version of this simpler proof.

PROOF. Let us first consider the case $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$. I will first show that $\Lambda_{\mathbf{d}^{\mathcal{T}}}(W) > \Lambda_{\mathbf{d}^{\mathcal{T}}}(T^*)$.

Since T^* is fully resolved and $W \neq T^*$, there must be in T^* at least one branch e that separates two clades of T^*, $C(e)$ and $D(e)$, which are not clades in W. Since these are clades of T^*, every circular ordering o for T^* must be such that one of these clades, say $C(e)$, can be expressed as $C(e) = \{o(i), o(i+1), \ldots, o(j-1), o(j)\}$, with $1 \leq i \leq j \leq n$. On the other hand, since $C(e)$ and $D(e)$ are not clades of W, it is possible to draw W so that these two sets of taxa are interleaved. That is, it is

[2] I require the correct topology T^* to be bifurcating because otherwise it is not difficult to see that if $W \neq T^*$ is obtained by resolving some of the multifurcations in T^* then we would have $\Lambda_{\boldsymbol{\delta}}(W) = \Lambda_{\boldsymbol{\delta}}(T^*)$. An alternative approach would be to drop the requirement that T^* be bifurcating and define "incorrect topologies" as those that cannot be obtained by refining multifurcations in T^*.

possible to find a circular ordering o' for W such that neither $C(e)$ nor $D(e)$ can be expressed as $\{o'(i), o'(i+1), \ldots, o'(j-1), o'(j)\}$. Therefore o' cannot be a circular ordering of T^*.

This proves that there is at least one circular ordering o' for W that is not a circular ordering for the correct topology T^*. Applying Proposition 6.3.2 with $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$, and noting that branch lengths in $\mathcal{T}$ are positive, we have that $\Lambda_{\mathbf{d}^{\mathcal{T}}}(o') > L(\mathcal{T})$, where $L(\mathcal{T})$ denotes the length of $\mathcal{T}$. Since $\Lambda_{\mathbf{d}^{\mathcal{T}}}(W)$ equals the mean of many $\Lambda_{\mathbf{d}^{\mathcal{T}}}(o)$ terms — each of which is either equal to $L(\mathcal{T})$ (Prop. 6.3.1), or greater than $L(\mathcal{T})$ (Prop. 6.3.2) — and since for at least one of these $\Lambda_{\mathbf{d}^{\mathcal{T}}}(o) > L(\mathcal{T})$, we have that $\Lambda_{\mathbf{d}^{\mathcal{T}}}(W) > L(\mathcal{T})$.

Now consider $\Lambda_{\mathbf{d}^{\mathcal{T}}}(T^*)$. Since $\Lambda_{\mathbf{d}^{\mathcal{T}}}(T^*) = \mathbb{E}[\Lambda_{\mathbf{d}^{\mathcal{T}}}(o)]$ where o is a random circular ordering for T^*, and since for any of these circular orderings $\Lambda_{\mathbf{d}^{\mathcal{T}}}(o) = L(\mathcal{T})$ (Prop. 6.3.1), we have that $\Lambda_{\mathbf{d}^{\mathcal{T}}}(T^*) = L(\mathcal{T})$, which concludes the proof of $\Lambda_{\mathbf{d}^{\mathcal{T}}}(W) > \Lambda_{\mathbf{d}^{\mathcal{T}}}(T^*)$.

Finally, note that $\Lambda_{\boldsymbol{\delta}}(T^*)$ and $\Lambda_{\boldsymbol{\delta}}(W)$ are continuous functions in $\boldsymbol{\delta}$. Because of that and since $\Lambda_{\boldsymbol{\delta}}(W) > \Lambda_{\boldsymbol{\delta}}(T^*)$ for $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$, the same must still hold in a neighbourhood of $\mathbf{d}^{\mathcal{T}}$ (i.e., for $\boldsymbol{\delta}$ "sufficiently close" to $\mathbf{d}^{\mathcal{T}}$). This is true for each incorrect topology W, each with its own neighbourhood of $\mathbf{d}^{\mathcal{T}}$ in which $\Lambda_{\boldsymbol{\delta}}(W) > \Lambda_{\boldsymbol{\delta}}(T^*)$. More strongly, $\Lambda_{\boldsymbol{\delta}}(T^*)$ is smaller than all the values for $\Lambda_{\boldsymbol{\delta}}(W)$ in the intersection of all these neighbourhoods, which (since these neighborhoods are finite in number) is itself a neighbourhood of $\mathbf{d}^{\mathcal{T}}$. □

The topology with the smallest balanced length among all possible topologies is called the *BME topology*. Theorem 6.3.3 shows that the BME topology coincides with the correct topology, provided that $\boldsymbol{\delta}$ is sufficiently close to the correct distance matrix $\mathbf{d}^{\mathcal{T}}$. As the amount of data used for estimating $\boldsymbol{\delta}$ gets larger and larger, if the evolutionary model used in the estimation is correct, the probability that $\boldsymbol{\delta}$ falls into the neighborhood of $\mathbf{d}^{\mathcal{T}}$ where the BME topology coincides with the correct topology tends to 1. In other, more technical words, this can be stated by saying that BME is a *consistent* criterion for tree reconstruction (first proved by Desper and Gascuel [**DG04**]).

Theorem 6.3.3 can be made more precise, as it leaves unanswered the following question: how close to $\mathbf{d}^{\mathcal{T}}$ does $\boldsymbol{\delta}$ need to be in order for the BME topology to be correct? This question will be addressed in section 6.6, where I give the first derivation of a "safety radius" within which the correctness of BME is guaranteed.

6.4. Some mathematical facts on BME

In this section I will show a number of results that are both interesting in themselves and useful in the development of the theory around BME. To the best of my knowledge, this is the first time they are published.

The first result (Prop. 6.4.1 below) shows that if we view a tree T in terms of a collection of clades that compose it, the balanced length formula $\Lambda_{\boldsymbol{\delta}}(T)$ can be re-expressed in terms of the balanced length of these clades and of the average distances between the clades. The notion of average distance between clades was introduced in section 6.3.1 where, however, I was only dealing with bifurcating trees. Recall that the average distance δ_{XY} between clades X and Y is defined as the expected distance between two taxa chosen randomly from X and Y according to random walks starting from the roots of these clades. For bifurcating trees δ_{XY} is simply given by the formula in (6.3.5). For a general tree T, possibly containing multifurcations, it is easy to see that the following holds:

$$\delta_{XY} = \sum_{\substack{i\in X\\ j\in Y}} \delta_{ij} \prod_{v\in P_{ij}-P_{XY}} \frac{1}{\deg(v)-1},$$

where P_{ij} is as before (I drop the parenthesis from $P_{ij}(T)$ for simplicity) and P_{XY} denotes the set of nodes internal to the path between the root of X and the root of Y.

In the following proposition, and in what follows, I drop the subscript $\boldsymbol{\delta}$ from $\Lambda_{\boldsymbol{\delta}}(T)$ when the distance matrix can be inferred from the context. Also, if X is a clade of T, I will denote by $\Lambda(X)$ the balanced length formula restricted to X, i.e., the same as in (6.3.9), but with i and j varying in X instead of the entire set of taxa $\{1,2,\ldots,n\}$. Note that, if $|X|=1$, then $\Lambda(X)=0$.

Proposition 6.4.1. *Suppose the taxa in a topology T can be partitioned into $X_1, X_2, \ldots, X_m$, where each X_i is a clade of T. Then*

$$\Lambda(T) = \sum_{h=1}^{m} \Lambda(X_h) + \sum_{1\le h<k\le m} \delta_{X_hX_k} \prod_{v\in P_{hk}} \frac{1}{\deg(v)-1},$$

where P_{hk} is an abbreviation for $P_{X_hX_k}$.

As an example, consider again the tree in figure 6.3.1(b): its taxa can be partitioned into clades A, B, C and D and so this proposition establishes that in this case:

$$\Lambda(T) = \Lambda(A)+\Lambda(B)+\Lambda(C)+\Lambda(D)+\frac{1}{2}\delta_{AB}+\frac{1}{2}\delta_{CD}+\frac{1}{4}\delta_{AC}+\frac{1}{4}\delta_{AD}+\frac{1}{4}\delta_{BC}+\frac{1}{4}\delta_{BD}.$$

In general, Proposition 6.4.1 asserts that the balanced length of a tree equals the sum of the balanced lengths of its constituent clades plus a sum with the same form as the balanced length formula (6.3.9):

$$\Lambda(T) = \sum_{i<j} \delta_{ij} \prod_{v\in P_{ij}} \frac{1}{\deg(v)-1},$$

except that the sum is now across clades instead of taxa: instead of having distances between taxa δ_{ij}, we now have average distances between clades $\delta_{X_hX_k}$; and the paths between taxa P_{ij} are now replaced by paths P_{hk} between clades.

PROOF. Decompose the sum for $\Lambda(T)$ between the terms where i and j are internal to a clade and those where i and j are in different clades:

$$\Lambda(T) = \sum_{h=1}^{m} \sum_{\substack{i,j \in X_h \\ i<j}} \delta_{ij} \prod_{v \in P_{ij}} \frac{1}{\deg(v)-1} + \sum_{1 \leq h < k \leq m} \sum_{\substack{i \in X_h \\ j \in X_k}} \delta_{ij} \prod_{v \in P_{ij}} \frac{1}{\deg(v)-1}.$$

The first part of this expression is simply the sum of the clades' balanced lengths, and the second part can be simplified by decomposing P_{ij} into P_{hk} and $P_{ij} - P_{hk}$:

$$\begin{aligned}
\Lambda(T) &= \sum_h \Lambda(X_h) + \sum_{1 \leq h < k \leq m} \sum_{\substack{i \in X_h \\ j \in X_k}} \left(\delta_{ij} \prod_{v \in P_{hk}} \frac{1}{\deg(v)-1} \prod_{v \in P_{ij}-P_{hk}} \frac{1}{\deg(v)-1} \right) \\
&= \sum_h \Lambda(X_h) + \sum_{1 \leq h < k \leq m} \left(\prod_{v \in P_{hk}} \frac{1}{\deg(v)-1} \right) \sum_{\substack{i \in X_h \\ j \in X_k}} \delta_{ij} \prod_{v \in P_{ij}-P_{hk}} \frac{1}{\deg(v)-1} \\
&= \sum_h \Lambda(X_h) + \sum_{1 \leq h < k \leq m} \left(\prod_{v \in P_{hk}} \frac{1}{\deg(v)-1} \right) \delta_{X_h X_k}.
\end{aligned}$$

□

Another way of viewing Proposition 6.4.1 is to imagine that each clade X_i is collapsed into a single taxon, thus giving rise to a new topology T', and that the distance matrix is replaced with $\boldsymbol{\delta}'$, the matrix of the average distances between clades (i.e., $\delta'_{hk} = \delta_{X_h X_k}$). Proposition 6.4.1 is then equivalent to stating

$$\Lambda_{\boldsymbol{\delta}}(T) = \sum_h \Lambda_{\boldsymbol{\delta}}(X_h) + \Lambda_{\boldsymbol{\delta}'}(T').$$

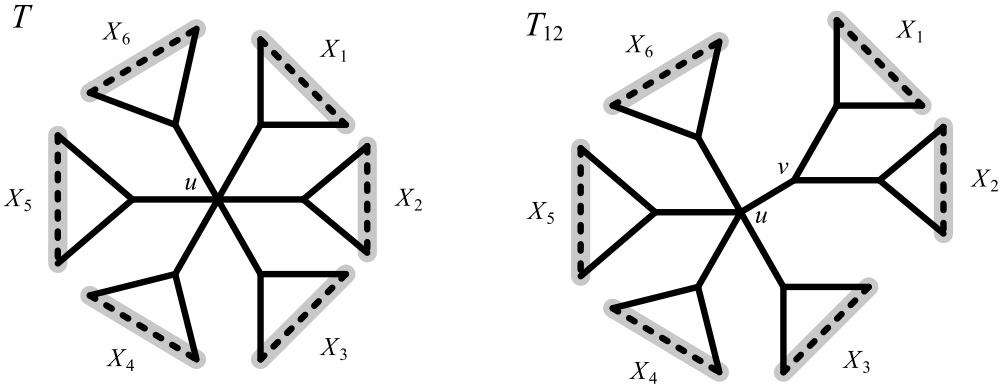
In order to prove the next result, I need to introduce a new concept: given a multifurcating topology T, the topology T' is an *agglomeration* of T if T' can be obtained from T by partially resolving one of its multifurcations in the following way: let u be the node in T at the centre of a multifurcation and let $X_1, X_2, \ldots, X_m$ be the clades connected to u; the agglomeration of X_h and X_k in T ($1 \leq h \neq k \leq m$) is obtained by deleting the two branches that connect u to X_h and X_k and replacing them with a new node v connected to u, X_h and X_k. For example, in figure 6.4.1, T_{12} is an agglomeration of T.

Given the above definition, we have:

PROPOSITION 6.4.2. *Let T be multifurcating. Then $\Lambda(T)$ equals the average of the $\Lambda(T')$ values across all T' that are agglomerations of T.*

PROOF. Assume for now that T has only one multifurcation at node u, which is connected to m clades $X_1, X_2, \ldots, X_m$. Let T_{hk} denote the agglomeration of X_h and

FIGURE 6.4.1. An agglomeration T_{12} of a multifurcating topology T.



X_k in T. By applying Proposition 6.4.1, we have that

$$\Lambda(T_{hk}) = \sum_i \Lambda(X_i) \tag{6.4.1}$$

$$+\frac{1}{2}\,\delta_{X_hX_k} + \sum_{\substack{i<j \\ i,j\notin\{h,k\}}} \frac{1}{m-2}\,\delta_{X_iX_j} + \sum_{i\notin\{h,k\}} \frac{1}{2(m-2)}\,\delta_{X_iX_h} + \sum_{i\notin\{h,k\}} \frac{1}{2(m-2)}\,\delta_{X_iX_k}.$$

The sum of all these values is equal to

$$\sum_{h<k} \Lambda(T_{hk}) = \binom{m}{2} \sum_i \Lambda(X_i) + \sum_{i<j} w_{ij}\delta_{X_iX_j},$$

where w_{ij} is the sum of all the coefficients that $\delta_{X_iX_j}$ is multiplied for, as h and k vary in (6.4.1). Note that when $h = i$ and $k = j$, the coefficient of $\delta_{X_iX_j}$ is $\frac{1}{2}$; when neither h nor k belongs to $\{i, j\}$, the coefficient is $1/(m-2)$; when exactly one of h and k belongs to $\{i, j\}$, the coefficient is $1/2(m-2)$. Therefore, multiplying these coefficients by the number of times they appear in $\sum_{h<k} \Lambda(T_{hk})$, we have that

$$w_{ij} \;=\; 1 \cdot \frac{1}{2} \;+\; \binom{m-2}{2} \cdot \frac{1}{m-2} \;+\; 2(m-2) \cdot \frac{1}{2(m-2)} \;=\; \frac{m}{2}.$$

Then the average of $\Lambda(T_{hk})$ across all agglomerations T_{hk} is given by:

$$\frac{1}{\binom{m}{2}} \sum_{h<k} \Lambda(T_{hk}) = \sum_i \Lambda(X_i) + \sum_{i<j} \frac{m}{2\binom{m}{2}}\,\delta_{X_iX_j} = \sum_i \Lambda(X_i) + \sum_{i<j} \frac{1}{m-1}\,\delta_{X_iX_j}.$$

By applying Proposition 6.4.1, this is precisely equal to $\Lambda(T)$.

If T has multiple multifurcations, let $\overline{\Lambda}_u$, where u is the node at the centre of a multifurcation, be the average balanced length across all agglomerations of two clades connected to u. It is easy to see that the average of $\Lambda(T')$, across all agglomerations T', is equal to a weighted average of the $\overline{\Lambda}_u$ values. Because of the arguments above, each $\overline{\Lambda}_u$ value equals $\Lambda(T)$ and therefore any of their weighted averages is itself equal to $\Lambda(T)$. □

A stronger result than that in Proposition 6.4.2 would be to prove that the balanced length estimate for any branch in T is the average of the balanced length estimates for that branch in the various agglomerations of T. However I have not given formulae for balanced branch length estimation in the general multifurcating case, and Proposition 6.4.2 is all I need for the results that follow.

Recall that T is a *BME topology* if it has the minimum value of $\Lambda(T)$ among all possible topologies. Proposition 6.4.2 allows us to establish that, when searching for BME topologies, it is sufficient to consider bifurcating trees:

COROLLARY 6.4.3. *At least one BME topology is bifurcating.*

PROOF. Let T be a BME topology. If T is not bifurcating, Proposition 6.4.2 implies that at least one of the agglomerations T' of T must be such that $\Lambda(T') \leq \Lambda(T)$ (because $\Lambda(T)$ is the average of the various $\Lambda(T')$). This argument can be repeated until a fully resolved (i.e., bifurcating) topology T^* is obtained, with $\Lambda(T^*) \leq \Lambda(T)$. Therefore, also T^* is a BME topology. □

Because of this result, we can restrict our search for a BME topology to bifurcating trees. Thus, the general formula for $\Lambda(T)$ introduced in (6.3.9) is ultimately superfluous: Pauplin's formula in (6.3.1) is all we need for the purpose of BME.

It would also be interesting to investigate whether the properties I have shown in this section for the balanced length can be extended, more in general, to WLS methods, since the balanced estimates can be seen as a particular type of WLS estimates (sec. 6.3.1).

6.5. Relation between BME and the neighbor-joining algorithm

The concept of agglomeration (introduced in the previous section and illustrated in fig. 6.4.1) is at the basis of several distance methods for tree reconstruction: UPGMA [**SM58**], WPGMA [**SS73**], ADDTREE [**ST77**], NJ [**SN87**], BIONJ [**Gas97a**], UNJ [**Gas97b**], MVR [**Gas00a**], Weighbor [**BSH00**]. These are all *agglomerative* algorithms: starting from a star tree connecting all the taxa under consideration, they iteratively agglomerate two suitably chosen clades connected to the central multifurcation until the whole tree is fully resolved (i.e., bifurcating).

NJ is arguably the best known and most used among agglomerative algorithms (possibly even among all distance methods), as it is considered to achieve a good tradeoff between reconstruction accuracy and speed of execution. It was introduced by Saitou and Nei in 1987 [**SN87**] and a more efficient formulation was later given by Studier and Keppler [**SK88**]. The two formulations are *equivalent* [**Gas94**], in the sense that they lead to the same sequence of agglomerations and therefore to the same tree. I will now describe the original version of NJ [**SN87**], and show that a recently proven connection between NJ and BME [**DG05, GS06**] can be easily derived by using the framework developed in this chapter.

NJ initialises T as the star topology where each taxon is directly connected to a central multifurcation. At any point during the execution of NJ, let $X_1, X_2, \ldots, X_m$ be the clades attached to this multifurcation. Initially, each of these clades contains precisely one taxon, that is, we can assume $X_i = \{i\}$, for every $i \in \{1, 2, \ldots, m\}$. Also, NJ stores a matrix D of distances between each pair of clades in $X_1, X_2, \ldots, X_m$. Initially, it is set so that $D(\{i\}, \{j\}) = \delta_{ij}$, for every $i, j \in \{1, 2, \ldots, n\}$.

Selection step: Modify T by aggregating the two clades X_h and X_k ($h, k \in \{1, 2, \ldots, m\}$) that minimise

$$Q_D(X_h, X_k) = \frac{1}{2} D(X_h, X_k) + \frac{1}{m-2} \sum_{\substack{i<j \\ i,j \notin \{h,k\}}} D(X_i, X_j) + \frac{1}{2(m-2)} \sum_{i \notin \{h,k\}} (D(X_i, X_h) + D(X_i, X_k)).$$

(If several pairs of clades minimise this criterion, choose one of these at random.)

Reduction step: replace X_h and X_k with the new clade $X_h \cup X_k$ and, for every $i \in \{1, 2, \ldots, m\} - \{h, k\}$, define

$$D(X_i, X_h \cup X_k) = \frac{1}{2}(D(X_i, X_h) + D(X_i, X_k)). \tag{6.5.1}$$

These two steps are repeated until T is bifurcating, i.e., they are performed exactly $n-3$ times. The topology T thus constructed is the result of NJ. (In reality NJ also assigns lengths to the branches, but here I am only concerned in the topology that is obtained.)

The theoretical justifications given by Saitou and Nei [**SN87**] for NJ were minimal, but its early introduction and its good performance in practice (both with simulated and real data [**SN87, SI89, KF94**]) ensured its vast popularity. However, the question remained of how to explain such good performance. In particular it was not clear why the $Q_D(X_h, X_k)$ criterion should be used in order to select the clades to agglomerate.

The rationale for $Q_D(X_h, X_k)$ originally provided [**SN87**] was the following: for each possible agglomeration of T, consider the tree obtained from this agglomeration by removing the clades $X_1, X_2, \ldots, X_m$ and estimate the lengths of the remaining $m+1$ branches by using OLS on the current distance matrix D; then the sum of all these branch lengths is precisely given by $Q_D(X_h, X_k)$. By choosing the agglomeration with the smallest $Q_D(X_h, X_k)$ we are therefore implicitly following an approach inspired by OLSME: we use OLS to estimate (some) branch lengths and ME to choose among alternative agglomerations. However, NJ should not be simply seen as a way to obtain a good tree with respect to OLSME: in fact, it has been shown that producing better (shorter) trees with respect to this criterion often results in a *decrease* – not an increase – of reconstruction accuracy [**Gas00b, DG02**]. This means that the reasons for the good performance of NJ lie beyond the merits of OLSME.

Until recently, the nature of the $Q_D(X_h, X_k)$ formula and the question of whether NJ does indeed aim to optimise some criterion have been the subject of much specialistic research (see the introduction in [**Bry05**]), but progress had been limited; as recently as 2003, in his popular textbook Felsenstein wrote ([**Fel03**], p. 170):

> Thus neighbor-joining has some relation to unweighted least squares and some to minimum evolution, without being definable as an approximate algorithm for either.

What researchers failed to realise was that in fact NJ is a heuristic algorithm aiming to optimise a criterion, and that this criterion is BME. This was first proven by Desper and Gascuel [**DG05**] some time after they first became interested in BME [**DG02**]. I now explain in more detail what this means and why it holds; see also [**GS06**] for a good review.

The first thing to note is that the distances computed by NJ coincide with the balanced average distances, that is

$$D(X_i, X_j) = \delta_{X_i X_j},$$

for all $i, j \in \{1, 2, \ldots, m\}$. This is evident if we note that the equations defining $D(X_i, X_j)$ — i.e., $D(\{i\}, \{j\}) = \delta_{ij}$ and (6.5.1) — are also satisfied by $\delta_{X_i X_j}$ (this was already noted in (6.3.4)).

Second, thanks to the observation above, we can rewrite the selection criterion in this way:

$$Q_D(X_h, X_k) = \frac{1}{2}\,\delta_{X_h X_k} + \frac{1}{m-2} \sum_{\substack{i<j \\ i,j \notin \{h,k\}}} \delta_{X_i X_j} + \frac{1}{2(m-2)} \sum_{i \notin \{h,k\}} (\delta_{X_i X_h} + \delta_{X_i X_k}).$$

Because of Proposition 6.4.1, and as observed in (6.4.1), we have that

$$Q_D(X_h, X_k) = \Lambda_{\boldsymbol{\delta}}(T_{hk}) - \sum_{i=1}^{m} \Lambda_{\boldsymbol{\delta}}(X_i),$$

where T_{hk} is the tree that is obtained from the current tree T by agglomerating X_h and X_k (fig. 6.4.1). This means that the clades X_h and X_k that minimise $Q_D(X_h, X_k)$ are the same that minimise $\Lambda_{\boldsymbol{\delta}}(T_{hk})$, as $\sum_i \Lambda_{\boldsymbol{\delta}}(X_i)$ is a constant independent of the choice of X_h and X_k. I have therefore proved the following:

PROPOSITION 6.5.1. *Imagine an algorithm that, starting from the star topology, iteratively produces the agglomeration of the current topology with the smallest balanced length, until no further agglomeration is possible. This algorithm is equivalent to NJ.*

In other words, NJ can be seen as a greedy agglomerative algorithm aiming to produce a tree topology T with low $\Lambda_{\boldsymbol{\delta}}(T)$. It does not find an optimal tree with respect to BME, and in fact better greedy-like – but not agglomerative – algorithms are possible and are the subject of section 6.7. As I will show, trees with a lower

$\Lambda_{\boldsymbol{\delta}}(T)$ than the tree constructed by NJ, T_{NJ}, are in general more accurate than T_{NJ}, a fact that allows us to interpret the good performance of NJ as derived from that of the BME criterion.

6.6. Safety radius for the BME criterion

In section 6.3.2, I have proved that BME is a consistent criterion for tree reconstruction, that is, if $\boldsymbol{\delta}$ is sufficiently close to the correct distance matrix $\mathbf{d}^{\mathcal{T}}$, then the BME topology coincides with the correct topology [**DG04, DG05**]. The same property was proved for NJ [**SN87, SK88**], and subsequently made more precise by Atteson [**Att99**], who showed how close $\boldsymbol{\delta}$ needs to be to $\mathbf{d}^{\mathcal{T}}$ in order for NJ (and other methods) to return the correct topology: calling ϵ the length of the shortest branch in the correct (bifurcating) tree $\mathcal{T}$, Atteson proved that, if all the distances in $\boldsymbol{\delta}$ are within $\epsilon/2$ from the distances in $\mathbf{d}^{\mathcal{T}}$, then the topology reconstructed by NJ is the correct one, that of $\mathcal{T}$. In Atteson's terminology, NJ has l_∞ radius $\frac{1}{2}$, which is an optimal result in the sense that this is the largest radius that any distance methods can have [**Att99**].

Like for many other distance methods, it is then interesting to determine the l_∞ radius of the BME criterion. In 2005, Desper and Gascuel wrote ([**DG05**], p. 29):

> Although we have no reason to believe BME has a small safety radius, the exact value of its radius has yet to be determined.

Given that BME can be seen as the criterion that NJ implicitly tries to optimise (sec. 6.5), it is reasonable to expect that its l_∞ radius is also optimal, i.e., equals $\frac{1}{2}$, a proposition that has been recently conjectured by Bordewich et al. [**BGHM09**]. This section proves that this is indeed true. This novel result is formalised in the following theorem.

THEOREM 6.6.1. *Assume that $\boldsymbol{\delta}$ is such that, for some bifurcating tree $\mathcal{T}$ and for every $i, j \in \{1, 2, \ldots, n\}$,*

$$|\delta_{ij} - d_{ij}^{\mathcal{T}}| < \frac{\epsilon}{2},$$

where $\epsilon > 0$ is the length of the shortest branch in $\mathcal{T}$. Then, the topology of $\mathcal{T}$ is the unique BME topology.

PROOF. I start by recalling a basic property of the balanced length formula that will be useful in this proof: let $\mathcal{T}'$ be a tree with topology T', and W' another distinct topology on the same set of taxa. Then, denoting by $L(\mathcal{T}')$ the length of $\mathcal{T}'$:

$$\Lambda_{\mathbf{d}^{\mathcal{T}'}}(T') = L(\mathcal{T}') \leq \Lambda_{\mathbf{d}^{\mathcal{T}'}}(W'). \tag{6.6.1}$$

That this holds should be obvious from the considerations in section 6.3.2, where I proved that $\Lambda_{\boldsymbol{\delta}}(T)$ is an average of many circular estimates $\Lambda_{\boldsymbol{\delta}}(o)$. Since for $\Lambda_{\mathbf{d}^{\mathcal{T}'}}(T')$ each of the circular estimates is equal to $L(\mathcal{T}')$ and for $\Lambda_{\mathbf{d}^{\mathcal{T}'}}(W')$ each of them is greater or equal to $L(\mathcal{T}')$, the relationship in (6.6.1) is verified.

Now let T be the (bifurcating) topology of $\mathcal{T}$ and let W be an arbitrary bifurcating topology distinct from T. Let b_{ij} and b'_{ij} be the number of branches in the paths between i and j in T and W, respectively. Since $T \neq W$, it is clear that there must be pairs of taxa for which $b_{ij} \neq b'_{ij}$. I wish to prove that $\Lambda_{\boldsymbol{\delta}}(W) > \Lambda_{\boldsymbol{\delta}}(T)$, as this implies that T it the unique (bifurcating) BME topology.

Because T and W are both bifurcating, we can use Pauplin's formula:

$$\Lambda_{\boldsymbol{\delta}}(W) - \Lambda_{\boldsymbol{\delta}}(T) = \sum_{i<j} \left(2^{1-b'_{ij}} - 2^{1-b_{ij}}\right) \delta_{ij}. \tag{6.6.2}$$

Now note that

$$d^{\mathcal{T}}_{ij} - \frac{\epsilon}{2} < \delta_{ij} < d^{\mathcal{T}}_{ij} + \frac{\epsilon}{2}.$$

This allows us to obtain a lower bound for (6.6.2) by replacing δ_{ij} with $d^{\mathcal{T}}_{ij} - \frac{\epsilon}{2}$ or with $d^{\mathcal{T}}_{ij} + \frac{\epsilon}{2}$, depending on whether the coefficient of δ_{ij} is positive or negative, respectively:

$$\Lambda_{\boldsymbol{\delta}}(W) - \Lambda_{\boldsymbol{\delta}}(T) > \sum_{i<j} \left(2^{1-b'_{ij}} - 2^{1-b_{ij}}\right) \left(d^{\mathcal{T}}_{ij} - \frac{\epsilon}{2} \operatorname{sgn}\left(2^{1-b'_{ij}} - 2^{1-b_{ij}}\right)\right),$$

where $\operatorname{sgn}(x)$ denotes the sign function of x, which equals 1, 0 or -1, depending on whether x is greater, equal or less than 0, respectively. Therefore,

$$\begin{aligned} \Lambda_{\boldsymbol{\delta}}(W) - \Lambda_{\boldsymbol{\delta}}(T) &> \sum_{i<j} \left(2^{1-b'_{ij}} - 2^{1-b_{ij}}\right) d^{\mathcal{T}}_{ij} - \frac{\epsilon}{2} \sum_{i<j} \left|2^{1-b'_{ij}} - 2^{1-b_{ij}}\right| \\ &= \Lambda_{\mathbf{d}^{\mathcal{T}}}(W) - \Lambda_{\mathbf{d}^{\mathcal{T}}}(T) - \frac{\epsilon}{2} \sum_{i<j} \frac{1}{2^{b'_{ij}-1}} \left|1 - 2^{b'_{ij}-b_{ij}}\right|. \end{aligned} \tag{6.6.3}$$

I now use a formula whose derivation will be the aim of the rest of this section (Corollary 6.6.3), namely that the following holds:

$$\sum_{i<j} \frac{1}{2^{b'_{ij}-1}} \left(b_{ij} - b'_{ij} - \frac{1}{2}\left|2^{b'_{ij}-b_{ij}} - 1\right|\right) \geq 0. \tag{6.6.4}$$

This allows us to calculate an upper bound for the last term in (6.6.3):

$$\begin{aligned} \frac{\epsilon}{2} \sum_{i<j} \frac{1}{2^{b'_{ij}-1}} \left|1 - 2^{b'_{ij}-b_{ij}}\right| &\leq \epsilon \sum_{i<j} \frac{1}{2^{b'_{ij}-1}} (b_{ij} - b'_{ij}) \\ &= \epsilon \left(\sum_{i<j} \frac{1}{2^{b'_{ij}-1}} b_{ij} \; - \; \Lambda_{\mathbf{b}'}(W)\right). \end{aligned} \tag{6.6.5}$$

Note that $\mathbf{b}' = (b'_{ij})$ is the distance matrix on a tree with topology W where all branches have length 1. Applying (6.6.1), we have that $\Lambda_{\mathbf{b}'}(W)$ must equal the length of this tree, which is $2n - 3$.

Furthermore, define σ^e_{ij} as 1 or 0 depending on whether e belongs to the path between i and j in T or not, respectively. Then we can make some useful derivations:

$$\begin{aligned}\epsilon\left(\sum_{i<j}\frac{1}{2^{b'_{ij}-1}}b_{ij} \;-\; \Lambda_{\mathbf{b}'}(W)\right) &= \epsilon\left(\sum_{i<j}\frac{1}{2^{b'_{ij}-1}}\sum_{e\in T}\sigma^e_{ij} \;-\; (2n-3)\right)\\ &= \epsilon\sum_{e\in T}\left(\sum_{i<j}\frac{1}{2^{b'_{ij}-1}}\sigma^e_{ij} \;-\; 1\right). \qquad (6.6.6)\end{aligned}$$

Now note that $\boldsymbol{\sigma}^e = (\sigma^e_{ij})$ coincides with the distance matrix on the right-hand tree in figure 6.6.1. Therefore, applying the inequality part in (6.6.1), $\Lambda_{\boldsymbol{\sigma}^e}(W) = \sum_{i<j} 2^{1-b'_{ij}}\sigma^e_{ij}$ is greater or equal than the length of that tree (1). Therefore, each of the summands in the outer sum in (6.6.6) is non-negative. If we multiply each of these summands by the length t_e of its corresponding branch in $\mathcal{T}$, instead of ϵ, then, since $t_e \geq \epsilon$:

$$\begin{aligned}\epsilon\sum_{e\in T}\left(\sum_{i<j}\frac{1}{2^{b'_{ij}-1}}\sigma^e_{ij} \;-\; 1\right) &\leq \sum_{e\in T} t_e\left(\sum_{i<j}\frac{1}{2^{b'_{ij}-1}}\sigma^e_{ij} \;-\; 1\right)\\ &= \sum_{i<j}\frac{1}{2^{b'_{ij}-1}}\sum_{e\in T} t_e\sigma^e_{ij} \;-\; \sum_{e\in T} t_e\\ &= \sum_{i<j}\frac{1}{2^{b'_{ij}-1}}d^{\mathcal{T}}_{ij} - L(\mathcal{T})\\ &= \Lambda_{\mathbf{d}^{\mathcal{T}}}(W) - \Lambda_{\mathbf{d}^{\mathcal{T}}}(T).\end{aligned}$$

Putting together all we have derived from (6.6.5) onwards, we have that

$$\frac{\epsilon}{2}\sum_{i<j}\frac{1}{2^{b'_{ij}-1}}\left|1-2^{b'_{ij}-b_{ij}}\right| \leq \Lambda_{\mathbf{d}^{\mathcal{T}}}(W) - \Lambda_{\mathbf{d}^{\mathcal{T}}}(T).$$

But then, going back to (6.6.3):

$$\Lambda_{\boldsymbol{\delta}}(W) - \Lambda_{\boldsymbol{\delta}}(T) > \Lambda_{\mathbf{d}^{\mathcal{T}}}(W) - \Lambda_{\mathbf{d}^{\mathcal{T}}}(T) - \frac{\epsilon}{2}\sum_{i<j}\frac{1}{2^{b'_{ij}-1}}\left|1-2^{b'_{ij}-b_{ij}}\right| \geq 0,$$

FIGURE 6.6.1. Tree referred to in the proof of Theorem 6.6.1. Let A and B be the two clades separated by e in T. $\mathcal{T}'$ consists of a branch of length 1 whose two extremes are connected to all taxa in A and B via branches of length 0. Note that the distance matrix on this tree is such that $\mathbf{d}^{\mathcal{T}'} = \boldsymbol{\sigma}^e$.

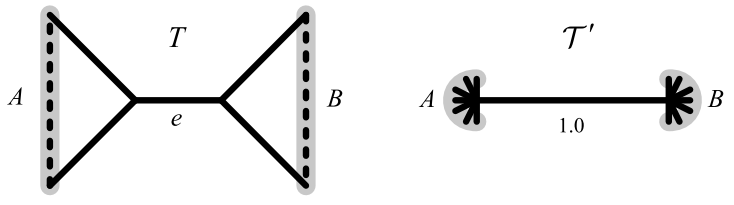
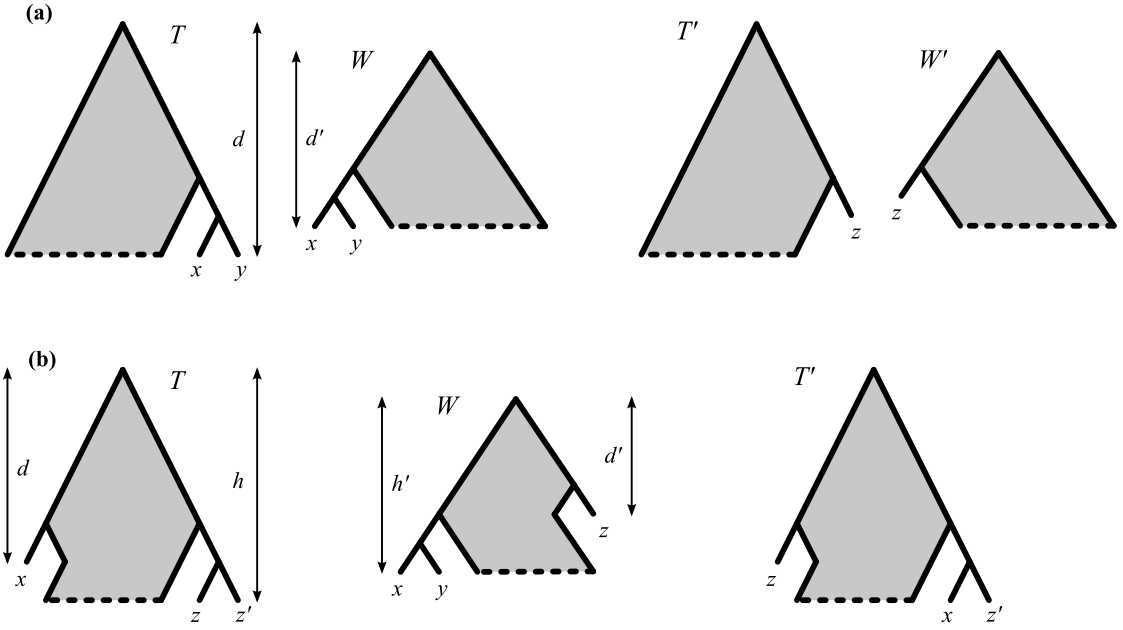


FIGURE 6.6.2. Tree topologies in the proof of Lemma 6.6.2.



which implies $\Lambda_{\boldsymbol{\delta}}(W) > \Lambda_{\boldsymbol{\delta}}(T)$. This proves that T is the unique bifurcating BME topology.

As for multifurcating topologies, if one of these had balanced length equal to the optimum value then, by applying Proposition 6.4.2, all its resolutions should have the same optimal balanced length. But this contradicts the fact that there is a unique bifurcating BME topology. Therefore T is the unique BME topology — not only among bifurcating ones. □

In order to prove the inequality in (6.6.4), I will first prove a simpler proposition, from which (6.6.4) will easily follow. Define the *depth* of a taxon i in a rooted tree T as the number of branches in the path, in T, from the root to i.

LEMMA 6.6.2. *Let T and W be bifurcating rooted tree topologies over the same set of taxa $\{1, 2, \ldots, n\}$. Let d_i and d'_i denote the depths of $i \in \{1, 2, \ldots, n\}$ in T and W, respectively. Then,*

$$\sum_{i=1}^{n} \frac{1}{2^{d'_i}} \left(d_i - d'_i - \frac{1}{2} \left| 2^{d'_i - d_i} - 1 \right| \right) \geq 0.$$

PROOF. By induction on the number of taxa n. If $n = 1$ or 2, then T and W must be the same topology and therefore the sum above is equal to 0. The case $n \geq 3$ is considerably more complex. Denote by $f(T, W)$ the sum in the statement. Also, call a clade of two taxa a *cherry*. I will consider two cases: either T and W have a common cherry, or they do not.

If they do have a common cherry, say $\{x, y\}$, then define T' and W' as the trees that are obtained from T and W, respectively, by removing the cherry $\{x, y\}$ and replacing it with a new taxon z (see fig. 6.6.2(a)). Since T' and W' have $n-1$ taxa, we can apply the induction hypothesis and have that $f(T', W') \geq 0$. The difference $f(T, W) - f(T', W')$ is easy to calculate, as the two sums only differ for the terms corresponding to taxa x, y and z. Calling d and d' the depths of x (and therefore y) in T and W, respectively, and noting that therefore the depths of z in T' and W' equal $d-1$ and $d'-1$ (see fig. 6.6.2(a)), we have that:

$$\begin{aligned} f(T,W) - f(T',W') &= 2\frac{1}{2^{d'}}\left(d - d' - \frac{1}{2}\left|2^{d'-d} - 1\right|\right) - \frac{1}{2^{d'-1}}\left(d - d' - \frac{1}{2}\left|2^{d'-d} - 1\right|\right) \\ &= 0. \end{aligned}$$

Therefore $f(T, W) = f(T', W') \geq 0$.

If T and W do not have a common cherry, let $\{x, y\}$ be a cherry with maximum depth in W. I will show that if we swap taxa in T so as to form $\{x, y\}$ in the resulting tree T'', this tree is such that $f(T, W) \geq f(T'', W)$. Since T'' and W do have a common cherry, for the arguments above, we have that $f(T'', W)$ – and therefore $f(T, W)$ – is non-negative.

Let $\{z, z'\}$ be a cherry with maximum depth in T. Because T and W do not have common cherries, at least one of z and z' must not be in $\{x, y\}$. Without loss of generality, assume $z \notin \{x, y\}$. Swap x and z in T and call the resulting tree T' (see fig. 6.6.2(b)). Let d and h be the depths in T of x and z, respectively, and note that this implies that, in T', x and z have depths h and d, respectively. Also, let d' and h' be the depths in W of z and x, respectively.

I now show that $f(T, W) \geq f(T', W)$. The two sums only differ for the terms corresponding to x and z:

$$\begin{aligned} f(T,W) - f(T',W) &= \frac{1}{2^{h'}}\left(d - h' - \frac{1}{2}\left|2^{h'-d} - 1\right|\right) - \frac{1}{2^{h'}}\left(h - h' - \frac{1}{2}\left|2^{h'-h} - 1\right|\right) \\ &+ \frac{1}{2^{d'}}\left(h - d' - \frac{1}{2}\left|2^{d'-h} - 1\right|\right) - \frac{1}{2^{d'}}\left(d - d' - \frac{1}{2}\left|2^{d'-d} - 1\right|\right) \end{aligned}$$

$$= \left(2^{-d'} - 2^{-h'}\right)(h-d) + \frac{1}{2}\left(\left|2^{-h} - 2^{-h'}\right| - \left|2^{-d} - 2^{-h'}\right| + \left|2^{-d} - 2^{-d'}\right| - \left|2^{-h} - 2^{-d'}\right|\right).$$

Because $h \geq d$ and $h' \geq d'$ (recall that h and h' are the maximum depths in T and W), it is not difficult to prove that the expression above is non-negative: first of all, note that if $h = d$ then $f(T, W) - f(T', W) = 0$. I then assume $h > d$. There are now six possible cases to consider:

(1) $h > d \geq h' \geq d'$. Then,

$$f(T,W) - f(T',W) = (2^{-d'} - 2^{-h'})(h-d) \geq 0.$$

(2) $h' \geq d' \geq h > d$. Then,

$$f(T,W) - f(T',W) = (2^{-d'} - 2^{-h'})(h-d) \geq 0.$$

(3) $h \geq h' \geq d' \geq d$ with $h > d$. Then,

$$\begin{aligned} f(T,W) - f(T',W) &= (2^{-d'} - 2^{-h'})(h-d) + 2^{-h'} - 2^{-d'} \\ &= (2^{-d'} - 2^{-h'})(h-d-1) \geq 0. \end{aligned}$$

(4) $h' \geq h > d \geq d'$. Then,

$$\begin{aligned} f(T,W) - f(T',W) &= (2^{-d'} - 2^{-h'})(h-d) + 2^{-h} - 2^{-d} \\ &\geq (2^{-d'} - 2^{-h'})(h-d) + 2^{-h'} - 2^{-d'} \\ &= (2^{-d'} - 2^{-h'})(h-d-1) \geq 0. \end{aligned}$$

(5) $h \geq h' \geq d \geq d'$ with $h > d$. Then,

$$\begin{aligned} f(T,W) - f(T',W) &= (2^{-d'} - 2^{-h'})(h-d) + 2^{-h'} - 2^{-d} \\ &\geq (2^{-d'} - 2^{-h'})(h-d) + 2^{-h'} - 2^{-d'} \\ &= (2^{-d'} - 2^{-h'})(h-d-1) \geq 0. \end{aligned}$$

(6) $h' \geq h \geq d' \geq d$ with $h > d$. Then,

$$\begin{aligned} f(T,W) - f(T',W) &= (2^{-d'} - 2^{-h'})(h-d) + 2^{-h} - 2^{-d'} \\ &\geq (2^{-d'} - 2^{-h'})(h-d) + 2^{-h'} - 2^{-d'} \\ &= (2^{-d'} - 2^{-h'})(h-d-1) \geq 0. \end{aligned}$$

This completes the proof that $f(T,W) \geq f(T',W)$. Now T' has $\{x, z'\}$ as a cherry. If $z' = y$, then define $T'' = T'$; otherwise let T'' be obtained by swapping z' with y in T'. In any case, by the same arguments as above, $f(T',W) \geq f(T'',W)$.

Since T'' has a cherry in common with T, we have that $f(T'',W) \geq 0$. But then,

$$f(T,W) \geq f(T',W) \geq f(T'',W) \geq 0.$$

□

I am now ready to prove the inequality in (6.6.4):

COROLLARY 6.6.3. *Let T and W be (unrooted) bifurcating tree topologies over the same set of taxa $\{1, 2, \ldots, n\}$. Let b_{ij} and b'_{ij} denote the number of branches in the paths between i and j in T and W, respectively. Then,*

$$\sum_{i<j} \frac{1}{2^{b'_{ij}-1}} \left(b_{ij} - b'_{ij} - \frac{1}{2} \left| 2^{b'_{ij}-b_{ij}} - 1 \right| \right) \geq 0.$$

PROOF. The sum above can be re-expressed in the following way:

$$\frac{1}{2} \sum_{j=1}^{n} \sum_{\substack{i=1 \\ i \neq j}}^{n} \frac{1}{2^{b'_{ij}-1}} \left(b_{ij} - b'_{ij} - \frac{1}{2} \left| 2^{b'_{ij}-b_{ij}} - 1 \right| \right).$$

Now define T_j and W_j as the rooted topologies that are obtained from T and W by rooting in taxon j and removing the branch at the root. Then,

$$\sum_{\substack{i=1\\ i\neq j}}^{n} \frac{1}{2^{b'_{ij}-1}} \left(b_{ij} - b'_{ij} - \frac{1}{2} \left| 2^{b'_{ij}-b_{ij}} - 1 \right| \right) = \sum_{\substack{i=1\\ i\neq j}}^{n} \frac{1}{2^{d'_i}} \left(d_i - d'_i - \frac{1}{2} \left| 2^{d'_i-d_i} - 1 \right| \right),$$

where $d_i = b_{ij} - 1$ and $d'_i = b'_{ij} - 1$ are the depths of i in T_j and W_j, respectively. Because of Lemma 6.6.2, this sum is non-negative, and therefore the whole sum in the statement — which is a sum of sums with this form — is also non-negative. □

This concludes the proof that BME has l_∞ radius $\frac{1}{2}$.

6.7. Heuristics for BME

Although this chapter has focused on the theoretical foundations of BME, arguably the strongest case in favour of the use of BME for distance-based tree reconstruction is provided by experiments that show its effectiveness (on simulated data) [**DG02, DG04, VvH05**]. Since BME is a *criterion* guiding the reconstruction of a phylogenetic tree — not an algorithm — I will next describe some of the algorithms that use in practice BME and I will then briefly review the experiments on their performance, a subject that will be explored further in the next chapter.

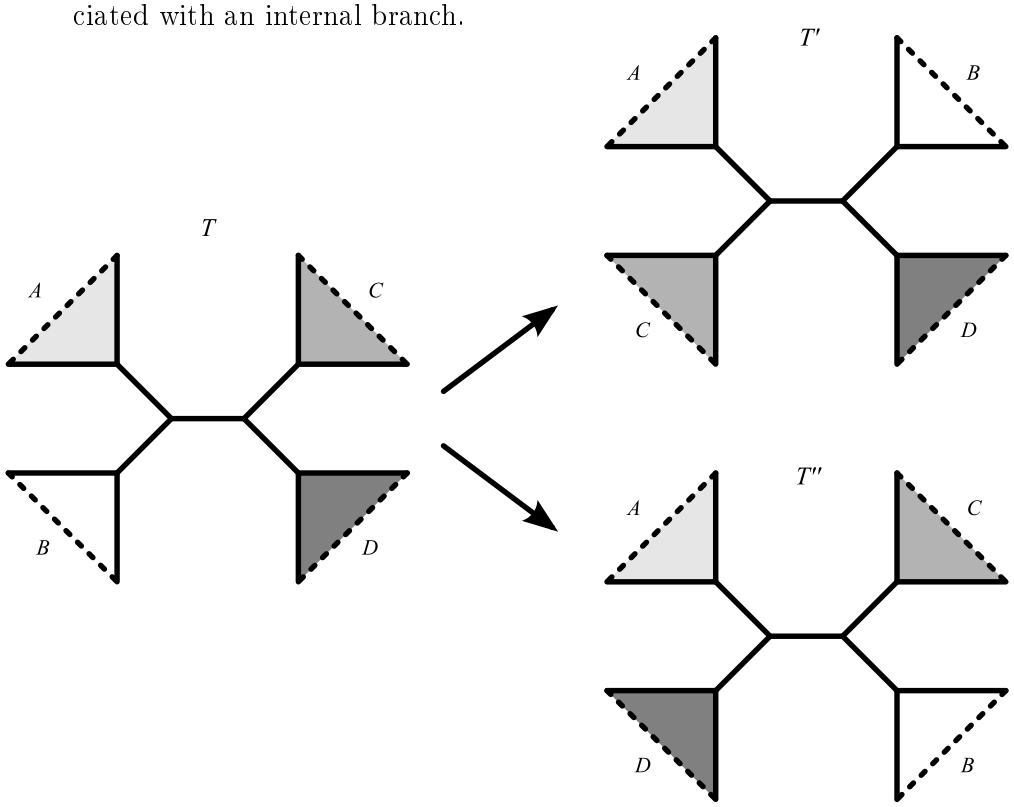
As we have already seen (sec. 6.5), one algorithm (implicitly) using BME is NJ. Since it constructs trees that are not necessarily optimal with respect to BME, NJ can be described as a heuristic for BME. It is natural to ask whether better heuristics than NJ can be devised for BME, a question that has been addressed with success in the program FastME by Desper and Gascuel [**DG02**].

The most effective heuristic implemented in FastME is called BNNI, which is based on *nearest-neighbor interchanges* (NNIs, [**Fel03**], pp. 41-44). An NNI applied to a tree topology T consists of swapping two clades attached to the same internal branch of T, but not to the same node. Figure 6.7.1 illustrates the two possible NNIs that can be done around a given internal branch. It is easy to see that there are exactly $2(n-3)$ different topologies that can be obtained by applying an NNI to any bifurcating topology (two per internal branch). The BNNI algorithm simply consists of a BME-guided steepest-descent search in the space of bifurcating topologies, where NNIs are the only allowed moves. It can be described in the following way:

- Construct an initial bifurcating topology T using any fast distance method.
- Iterate the following:
  - Apply every possible NNI to T and define T' as the rearrangement of T thus obtained with the smallest balanced length $\Lambda(T')$.
  - If $\Lambda(T') \geq \Lambda(T)$ then stop and return T; otherwise redefine T as T'.

Note that since the objective of BNNI is to optimise BME, the initial tree will typically be constructed with a method inspired by this criterion. FastME implements new methods for this step [**DG02**] — much faster than NJ — which in effect are

FIGURE 6.7.1. The two possible nearest-neighbor interchanges associated with an internal branch.



greedy algorithms based on a *sequential addition* strategy ([**Fel03**], pp. 47-48). Since it has been shown that the trees reconstructed by BNNI are largely independent of the initial tree [**DG02, VvH05**], it is clear that such fast methods for the initialisation step should be preferred. As for the stopping criterion, BNNI returns a local minimum for $\Lambda(T)$; clearly this local minimum is not guaranteed to coincide with a global minimum and therefore BNNI does not, in general, reconstruct BME-optimal topologies.

BNNI is computationally very efficient: even if we include the construction of the initial tree (via sequential addition), its running time is generally considerably shorter than that of NJ. This efficiency is due to a very simple way to express the change in $\Lambda(T)$ resulting from one NNI: with reference to the upper NNI in figure 6.7.1 — the one from T to T' — a simple consequence of Proposition 6.4.1 (first derived by Desper and Gascuel [**DG02**]) is that

$$\Lambda(T') - \Lambda(T) = \frac{1}{4}(\delta_{AC} + \delta_{BD} - \delta_{AB} - \delta_{CD}). \tag{6.7.1}$$

This formula can be applied repeatedly to find the T' with the largest decrease in $\Lambda(T')$ and also to realise when a local minimum has been reached. If during the execution of BNNI we maintain a matrix Δ^T containing all the δ_{XY} between every pair of non-overlapping clades in T (an idea that I will use again in the next chapter),

then the calculation of (6.7.1) can be done in constant time. Looking in more detail at the computational complexity of BNNI, since there are $2(n-3)$ possible NNIs, each iteration will take $O(n)$ time, plus the time needed to update Δ^T. This turns out to be $O(n \cdot \mathrm{diam}(T))$ [**DG02**], where $\mathrm{diam}(T)$ is the diameter of T, i.e., the maximum number of branches between any two leaves in T. Since Δ^T for the initial tree can be constructed in $O(n^2)$ [**DG02**], BNNI runs in $O(n^2 + p \cdot n \cdot \mathrm{diam}(T))$, where p, the number of NNIs that are performed in order to reach a local minimum, is typically much smaller than n [**DG02**]. If one compares $O(n^2 + p \cdot n \cdot \mathrm{diam}(T))$ to the complexity of other "fast" methods, such as NJ, BIONJ [**Gas97a**], and Weighbor [**BSH00**] — which all run in $O(n^3)$ time — BNNI's computational efficiency becomes fully apparent.

But the most interesting observation about BNNI is another one: not only it is extremely fast, but it is also remarkably accurate in reconstructing trees, significantly more accurate than the other commonly used distance methods. A number of studies [**DG02, DG04, VvH05**] showed this by comparing the trees reconstructed by various methods from simulated distance matrices to the ("correct") trees used to generate the simulated data. The difference between the reconstructed and the correct trees can be quantified using the Robinson and Foulds (RF) distance [**RF81**], a measure that will also be useful in the next chapter and that is defined as follows:

DEFINITION 6.7.1. Each branch e in a topology T induces a *bipartition* $A_e|B_e$ where A_e are all the taxa on one side of e and B_e are all the taxa on the other side. T therefore induces a collection $S(T)$ of bipartitions corresponding to its branches. The *Robinson and Foulds distance* between two topologies T and T' (over the same set of taxa), denoted $d_{RF}(T, T')$, is defined as the number of bipartitions induced by one tree but not the other, i.e.,

$$d_{RF}(T, T') = |S(T) - S(T')| + |S(T') - S(T)|.$$

Note that if T and T' are bifurcating trees, then the two set differences above have the same size and $d_{RF}(T, T')$ is an even number between 0 (when the topologies are identical) and $2(n-3)$ (when the only bipartitions that T and T' have in common are those induced by their terminal branches).

The simulation studies by [**DG02, DG04, VvH05**] showed that the trees reconstructed by BNNI are, on average, substantially closer (as measured by d_{RF}) to the correct trees than the ones reconstructed by other commonly used distance methods: NJ, BIONJ, Weighbor and the PAUP* [**Swo98**] heuristic for WLS (where w_{ij} is assumed to be proportional to δ_{ij}^{-2}). This was shown under a number of simulation settings, which had in common the fact of using randomly generated trees to evolve random sequence data, from which the distance matrices were then estimated. The studies differed in several aspects: the method to generate the random trees, the number of sequences in the trees, their length, the model of molecular evolution used to produce the sequences and the method to estimate pairwise distances. In all cases,

BNNI was more accurate than the other distance methods tested — with the possible exception of the FITCH program (another heuristic for WLS) from the PHYLIP package [**Fel89**], which however was too slow to be tested extensively [**DG02**].

If we measure the *topological accuracy* of a method by the average RF distance between the correct trees and those reconstructed by that method, the comparison between methods can be made quantitatively precise. For example, for BNNI and NJ, Desper and Gascuel [**DG02**] showed that BNNI was 3-7% more accurate than NJ for their 24-taxon datasets and 5-20% more accurate than NJ for their 96-taxon datasets (sequence length 500bp); furthermore BNNI was on average 10% more accurate than NJ for the 100-taxon datasets in [**DG02**] (sequence length 600bp) and 6-50% more accurate than NJ for the 1000-to-5000-taxon datasets in [**VvH05**] (sequence length 1000bp). Although the varied simulation settings of the different studies clearly have strong effects on these numbers, the superior accuracy of BNNI over the other methods tested has been confirmed across all studies.

Besides BNNI, another heuristic algorithm for BME has been recently proposed [**BGHM09**], based on the well known *subtree pruning and regrafting* (SPR) rearrangements ([**Fel03**], pp. 41-44). These are a generalisation of NNIs: whereas an NNI can be described as the pruning of a clade X and its regrafting on a branch adjacent to that to which X was originally connected, an SPR relaxes the adjacency requirement, thus allowing X to be regrafted on any other branch of the tree. The proposed heuristic is called BSPR and is exactly the same as BNNI except that at each step the SPR rearrangements of the current tree, instead of just its NNI rearrangements, are taken into account. Using the fact that we can order all the possible SPRs so that each SPR can be obtained by composing an NNI with a previous SPR, it is easy to see that we can calculate $\Lambda(T')$ for every SPR rearrangement of the current tree T in $O(n^2)$.

Although some theoretical properties of BSPR have been investigated (consistency and safety radius) [**BGHM09**], its algorithmic aspects have not been explored in detail: for example it may be advisable to combine BSPR and BNNI so that SPRs are only taken into account when a local minimum with respect to NNIs is reached. Furthermore, although we can expect BSPR to be somewhat more topologically accurate than BNNI, the extent of its gains in accuracy has yet to be investigated.

6.8. Related and future work

In this chapter I have tried to give an overview of the main facts about BME and its relationship with other distance methods, such as NJ. Recently, many other interesting results have been obtained. This section will summarise these results, describe some open directions for future research and give pointers to the relevant literature.

In section 6.3.1, I explained that Pauplin's formula can be obtained as the sum of the balanced branch length estimates given by formulae (6.3.6) and (6.3.7) and

that these estimates are optimal in a WLS sense: they minimise $\sum_{ij} w_{ij}(\delta_{ij} - \mathrm{d}^{\mathcal{T}}_{ij})^2$, when w_{ij} is assumed to be proportional to $2^{-b_{ij}}$ [**DG04, MP08**]. In this framework, the branch length estimates for BME are analogous to those for OLS — (6.3.2) and (6.3.3) — which assume constant weights w_{ij}. It is then natural to ask: are there general, efficient formulae for the optimal branch lengths under a WLS model with arbitrary weights? Mihaescu and Pachter [**MP08**] have shown such formulae for a class of standard "tree-multiplicative" weights, which admit OLS and BME as particular cases. For this class, it is possible to calculate all branch length estimates (and therefore the tree length) in $O(n^2)$.

Pairwise weights w_{ij} are said to be *tree-multiplicative* with respect to topology T when they can be obtained as the product of branch-associated weights, that is:

$$w_{ij} = \prod_{e \in P_{ij}} w_e,$$

for some positive weights w_e associated with each branch e of T. (Here, P_{ij} denotes the set of branches in the path between i and j in T.) OLS and the balanced length estimation are particular types of tree-multiplicative WLS in the following sense: assuming $w_e = 1$ for every branch e in T, the resulting tree-multiplicative weights are constant and therefore WLS is equivalent to OLS. As for the balanced scheme, the pairwise weights $2^{-b_{ij}}$ are obtained by setting $w_e = \frac{1}{2}$ for every branch e in T. Obviously, other reasonable tree-multiplicative weights are possible, and potentially may lead to more accurate branch length estimates. The use of other tree-multiplicative WLS models for tree reconstruction, for example in combination with ME, is an open direction for future work and it would be interesting to investigate both the effectiveness in practice and the theoretical properties of these new distance methods.

In sections 6.5 and 6.7, I have described the NJ, BNNI and BSPR heuristic algorithms for BME. An important theoretical question about them is that of their consistency: if $\boldsymbol{\delta}$ coincides with the correct distance matrix, will they return the correct tree topology? And what if $\boldsymbol{\delta}$ is only approximately equal to the correct matrix?

Whereas the consistency of NJ [**SN87, SK88**] and BSPR [**BGHM09**] have been established, that of BNNI has only been conjectured [**DG04, BGHM09**] and is an interesting open problem. (Note that asserting that local search algorithms such as BNNI or BSPR are consistent means that, assuming $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$, they must be able to reconstruct the correct tree whatever the initial tree from which they start their search.) For BSPR, similarly to NJ [**Att99**], not only consistency but also a l_∞ safety radius of at least $\frac{1}{3}$ has been established [**BGHM09**] (a result that I suspect can be improved).

In section 6.6, I have shown, for the first time, that the l_∞ safety radius of BME equals its best possible value, $\frac{1}{2}$. The safety radiuses of other criteria can

also be investigated: for example, Gascuel and Guillemot (personal communication) have shown that the safety radius of OLSME (ME using OLS for branch length estimation) tends to 0 as the number of taxa tends to infinity. This discrepancy between the safety radiuses of BME and OLSME may explain (at least in part) the greater accuracy of BME over OLSME shown in simulation studies [**DG02, VvH05**]. In the near future, I will be working on a manuscript (together with Gascuel and Guillemot) presenting these results on the safety radiuses of these criteria [**PGG09**].

Returning to NJ, another paper by Mihaescu, Pachter and collaborators [**MLP09**] described general conditions on the input distance matrix $\boldsymbol{\delta}$ that guarantee that NJ reconstructs the correct tree. These conditions admit Atteson's requirement of a maximum gap between input and correct distances [**Att99**] as a special case. It would be interesting to investigate whether these general conditions (or even more general ones) also guarantee the correctness of the BME topology and whether they have anything to suggest regarding the consistency of BNNI.

Another recent paper with relevance to the relationship between NJ and BME is that by Eickmeyer et al. [**EHPY08**], who take a geometric approach enabling them to investigate questions such as how often NJ can be expected to construct a BME-optimal tree.

Prior to the discovery of the connection between BME and NJ, a number of works appeared aiming to clarify NJ's selection criterion $Q_D(X_h, X_k)$ given in section 6.5. One of the most interesting is that of Bryant [**Bry05**] (also reviewing previous similar work), who proved that any criterion satisfying three desirable properties he describes is equivalent to NJ's selection criterion. These properties are (1) the consistency of the resulting agglomerative algorithm, (2) the independence of the criterion from the order with which the taxa are given and (3) its linear dependence upon the distances.

CHAPTER 7

A branch and bound approach for BME-optimal trees

7.1. An exact algorithm for BME: is "the best the enemy of the good"?

Chapter 6 introduced the balanced minimum evolution (BME) criterion for reconstructing phylogenetic trees: it sets the objective to find the tree topology T that minimises $\Lambda_{\boldsymbol{\delta}}(T)$, the *balanced length* of T (see sec. 6.3). In section 6.7, I have described some efficient algorithms for reconstructing trees that are good with respect to BME, although not necessarily optimal. In fact, achieving optimality seems a very difficult task: it is very likely that no polynomial algorithm producing BME-optimal trees can be devised, although no proof that this problem is NP-hard has appeared in the literature as yet.

Despite the likely computational hardness of this problem, it would still be useful to have an "exact" algorithm for BME, that is, an algorithm able to construct BME-optimal trees, at least for moderately-sized distance matrices. "Branch and bound" is a popular technique to attack difficult problems without losing the ability to find their optimal solutions (see, e.g., [**PS98**], ch. 18). This chapter presents a branch and bound algorithm for BME and some experiments that I carried out with a program implementing it.

As their name implies, branch and bound algorithms are specified by two main components: the branching and the bounding. The first component, the *branching*, is a strategy that allows us to subdivide the space of all possible solutions to the problem into two or more subspaces. This subdivision is applied recursively to each of these subspaces until either we can find an optimal solution for the subspace (typically when the subspace only contains one solution) or we can show that the subspace cannot contain an optimal solution to the global problem. The latter case is what makes use of the *bounding*: this consists of a method to evaluate a best-case-scenario solution belonging to the subspace under consideration; if, even in the best-case-scenario, this solution cannot be as good as the optimal solution for another subspace, the exploration of the current subspace is abandoned, as it will not lead to finding any optimal solution. For sake of generality, this description is necessarily very abstract; section 7.2 will explain what this means in practice, in the context of phylogenetics and BME in particular. The details of the algorithm — the bounding it uses and its practical implementation — are presented in the two subsequent sections (sec. 7.3 and sec 7.4, respectively).

The branch and bound algorithm I introduce here is, in general, inefficient for large distance matrices, although simple considerations show that if the input distances are sufficiently close to the correct distances, then the algorithm is guaranteed polynomial running time (see sec. 7.3.6). Although it is often impractical for typical problem instances, an exact algorithm may be used to assess how good the existing heuristics are and how much margin of improvement might be achieved by developing better heuristics. At present, I see this as the main use of the algorithm I develop here: section 7.5 shows and discusses my first experiments in this direction. To put it in more general terms, the question I try to answer is whether it is worthwhile in this case to pursue the optimal solution, possibly at the expense of efficiency, or, to use Voltaire's words, whether or not "the best is the enemy of the good" [**Vol72**].

Throughout this chapter, notation ($\boldsymbol{\delta}$, δ_{XY}, $\Lambda_{\boldsymbol{\delta}}(T)$) and terminology (e.g., *clades*) will be as in Chapter 6.

7.2. A branch and bound algorithm for BME

Branch and bound algorithms have a long history in phylogenetics. They have been applied to several criteria for tree reconstruction: primarily maximum parsimony [**HP82, PH87, PBTK00, AN06**], but also least squares constrained to ultrametric trees [**CLP80**]; recently progress has also been made towards application to maximum likelihood [**HH03**]. An entire chapter of Felsenstein's textbook ([**Fel03**], ch. 5) is dedicated to this topic.

The algorithm I present here — which I call BBBME for branch and bound for balanced minimum evolution — is based on the same branching idea as Hendy and Penny's original algorithm for maximum parsimony [**HP82**]. In fact, I am indebted to them for the early ideas in the development of this algorithm (and to B. Holland, who made a crucial observation for the bounding part of the algorithm). Hendy and Penny's branching strategy is best described by imagining a large tree which has bifurcating topologies as its own nodes. In order to express the fact that this is a tree connecting other trees, I will call this the *metatree* (in the branch and bound literature this is usually known as the *search tree*). Furthermore, I will refer to the branches and the nodes of the metatree as its *edges* and *vertices*, respectively, in order to avoid confusion with the branches and nodes of its constituent trees. Figure 7.2.1 shows an example of the top portion of the metatree, to help understanding of the following description. At the root of the metatree lies the only bifurcating topology T_0 connecting the first three taxa. From the root T_0, three other topologies branch off: they are exactly those that can be obtained by adding the fourth taxon onto any of T_0's branches. From these topologies other topologies branch off, recursively, so that each vertex in the metatree is associated with a tree topology over a subset of the taxa: in general, the vertex corresponding to a topology T over h of the n available taxa (with $h < n$) will branch off into $2h - 3$ other topologies, obtained by attaching a taxon not yet in T onto each of T's branches in turn. Note that saying

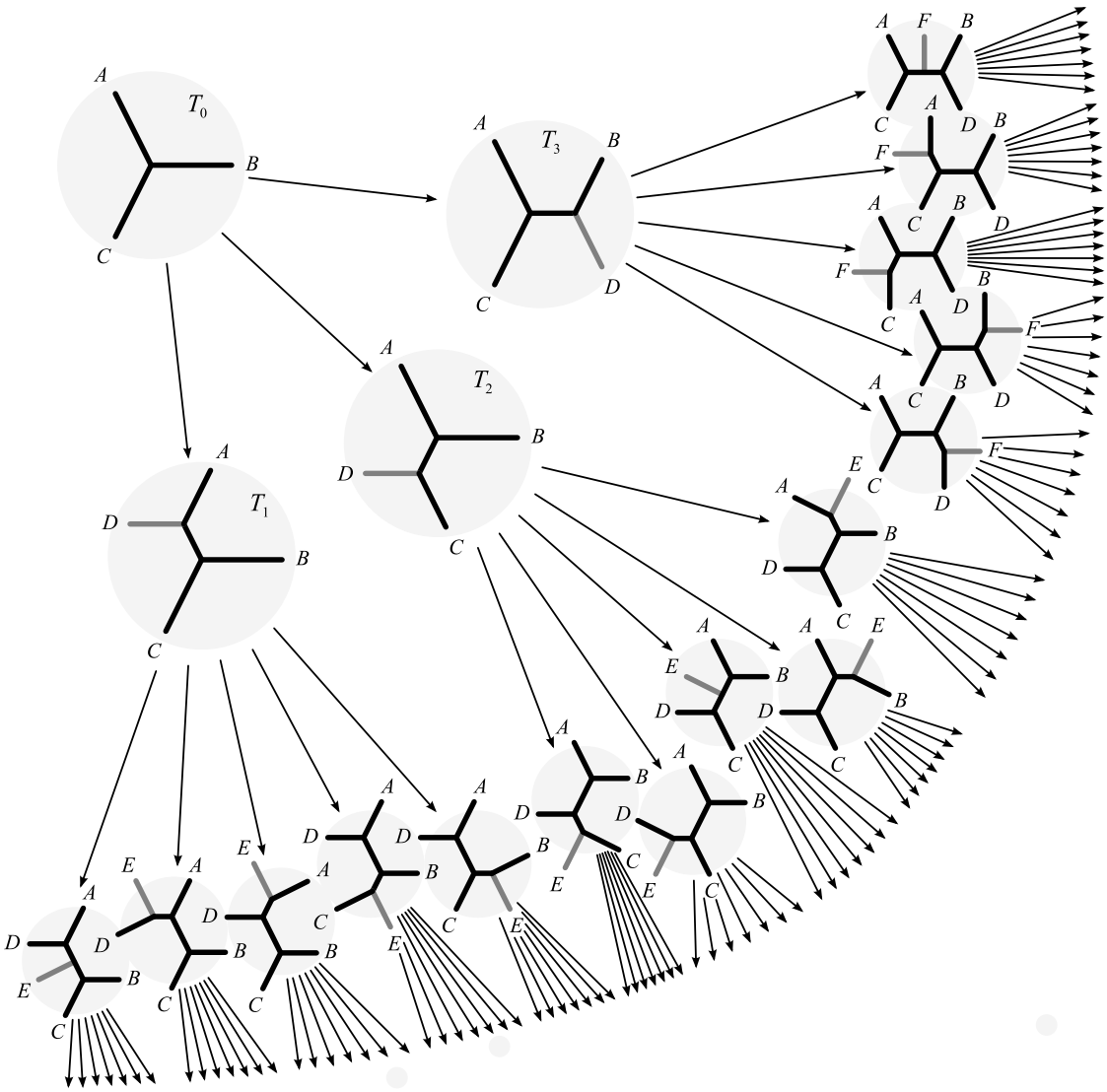
that a taxon k gets attached onto a branch e means that a terminal branch with k at one of its extremes is attached (via its other extreme) to a newly introduced node that splits e into two branches. Only bifurcating topologies can be obtained through this process, as new taxa are always attached to the *interior* of an existing branch — never directly to nodes. The only topologies not having descendants are those containing all n taxa, as obviously there are no more taxa to add. Not only do the leaves of the metatree correspond to bifurcating topologies over all the available taxa, but also, conversely, every bifurcating topology over these taxa corresponds to exactly one of the leaves in the metatree: this is because each path from the root to a leaf in the metatree corresponds to a sequence of choices specifying where the next taxon should be added in the current tree, and for each bifurcating topology there is one, and exactly one, such sequence of choices. Therefore the leaves of the metatree specify all possible solutions to the tree reconstruction problem we are trying to solve. Note that, because of the considerations in the previous chapter (Corollary 6.4.3), limiting our search to bifurcating topologies is not restrictive — and the same holds for maximum parsimony and any other popular phylogenetic criterion.

BBBME, and most branch and bound algorithms for maximum parsimony, consist of a search applied to the metatree described above. In theory, the search may be executed in any top-down order ensuring that a vertex is visited only after its parent vertex has been visited. In practice, at least in the case of BBBME, a depth-first search ([**CLRS01**], sec. 22.3) is preferred, as this ensures that only the data relative to a limited number of vertices needs to be stored. As I will explain, visiting a vertex corresponding to T — which should be viewed as a partial solution to our tree reconstruction problem — involves assessing the potential of this partial solution. If a partial solution is deemed unpromising, then the portion of the metatree below it is left unexplored. This is what allows branch and bound algorithms to be applicable to problems of moderate size, even when the metatree is in theory so large that no current or future computer would be able to explore it fully in a reasonable time (see fig. 7.2.1 for a discussion on the size of the metatree).

This concludes the description of the branching strategy for BBBME, although two "ordering issues" will be addressed at the end of this section: first, the order with which the taxa are added to the partial topologies in the metatree and, second, the order with which the vertices that branch off an internal vertex of the metatree are visited. Also note that in the previous section I described the branching step as a subdivision of a subspace of solutions into smaller subspaces. This is true also for the description I gave here: if we view each vertex in the metatree as associated with the subspace of solutions lying at the leaves that can be reached from this vertex (i.e., *below* it), the fact that a vertex T branches off into several other *child vertices* precisely corresponds to a division of the subspace below T into the subspaces below its child vertices. Furthermore this division is precisely a partition, i.e., it is such that the new subspaces are not overlapping (and their union is equal to the original

FIGURE 7.2.1. Metatree for the branch and bound algorithms based on the branching strategy corresponding to sequential addition of taxa [**HP82**].

For each topology, the newly added branch is coloured in gray. Note that only the portion at the top three levels of the metatree is illustrated here. The paths from the root to the leaves of the metatree are composed of $n-3$ edges and they split off at each vertex they cross in ever-increasing numbers (a vertex corresponding to a tree with h taxa branches off into $2h-3$ other vertices). It is easy to see that the metatree has $(2n-5)!! = 3 \cdot 5 \cdot \ldots \cdot (2n-7) \cdot (2n-5)$ such paths, leading to an equal number of leaves. This number grows super-exponentially with n. For example, when a dataset comprises just 20 taxa, the metatree has 221,643,095,476,699,771,875 leaves [**Sch70, CSE67, Fel78**].



subspace). This is a desirable property for any branch and bound algorithm, as we wish to avoid exploring any portion of the solution space more than once.

As for the bounding step, each time a new vertex is encountered — corresponding to a topology T — the algorithm assesses whether or not it is worthwhile to continue the search into the subspace of solutions that can be generated by extending T; that is, whether or not any of the topologies T^+ at the leaves of the subtree rooted in T may be an optimal solution to our problem. This assessment is achieved by calculating an "optimistic" bound to the score of T^+; if the optimistic bound is not as good as the score of the best solution previously found, then clearly no optimal solution can lie in the subtree rooted in T and therefore we can ignore this subtree and continue our search into other, more promising, parts of the metatree.

In the case of BME, which aims to *minimise* the balanced length, we use a *lower* bound to the balanced length of any T^+ at a leaf of the subtree rooted in T. In the next section, I will show a number of possible bounds which have the following generic form:

$$\Lambda_{\boldsymbol{\delta}}(T^+) \geq \beta(T, \boldsymbol{\delta}).$$

When this bound is calculated, if the algorithm has previously encountered some complete topologies — i.e., containing the entire set of taxa in input — let Λ^* be the minimum balanced length among all these topologies. If this quantity is smaller than the lower bound for T^+, that is, if $\Lambda^* < \beta(T, \boldsymbol{\delta})$, then clearly every complete topology T^+ below T will have a greater balanced length than that of a complete topology previously encountered and therefore cannot be optimal. As a consequence, the traversal of the subtree rooted in T is not carried out and the search continues as if this subtree had already been completely examined. In branch and bound terminology, we say that this subtree is *pruned* from the metatree.

An example of a very simple bound is provided by the algorithms for parsimony based on the branching strategy described above. In this context, the score of a topology T is not given by $\Lambda_{\boldsymbol{\delta}}(T)$, but by the minimum number of substitution events required along the branches of T in order to generate the sequences in input, and again the aim is to minimise the score of T. It is easy to see (e.g., [**Fel03**], Ch. 2) that the scores of the trees cannot decrease as we go down a path from the root to a leaf of the metatree. Therefore the score of the current partial tree T is itself a lower bound to the scores of any tree T^+ below T.

Note that the execution of a branch and bound algorithm should be preceded by that of a heuristic that constructs an initial solution to the problem at hand, so that Λ^* can be set to a value against which the bound can immediately make comparisons. Clearly, the better the heuristic, the more effective the pruning will be from the start. (Of course, an alternative approach would be to initialise Λ^* with ∞, but this will result in longer running times.) Whenever the branch and bound algorithm reaches a complete topology T at a leaf of the metatree — i.e., whenever the bound does not result in any pruning for none of the vertices on the path from

Algorithm 8 SIMPLE BBBME ($\boldsymbol{\delta}$)

Execute a heuristic that constructs an initial topology T^*.
Set $\Lambda^* = \Lambda(T^*)$ and $\Theta^* = \{T^*\}$.
Choose the first three taxa $i, j, k \in \{1, 2, \ldots, n\}$.
Construct T_0, the topology connecting i, j and k.
PROCESS $(T_0, \Lambda^*, \Theta^*, \boldsymbol{\delta})$.
return Θ^*

the root to that leaf — the currently minimum balanced length Λ^* is compared to the balanced length of T and possibly updated.

Although there are still many aspects of BBBME that need to be described — not least the bound it uses — its general structure should already be clear. The pseudocode in Algorithms 8 and 9 describes BBBME in its general lines. A number of formalisms that will be useful throughout the rest of this chapter are introduced here: the forms $i \in T$ and $i \notin T$ respectively express that i is a leaf of T and that $i \in \{1, 2, \ldots, n\}$ is not a leaf of T; also, assuming that e is a branch in T, the form $A \overset{e}{-} B$ indicates that A and B are the two clades at the sides of e in T; finally, assuming $A \overset{e}{-} B$ in T, the expression $T \underset{B}{\overset{A}{\vdash}} k$ denotes the topology that is obtained from T by attaching taxon k onto e and will be referred to as an *extension* of T. In section 7.4, I will give a more detailed description of the implementation of BBBME, including some of the information that can be stored in order to speed up its execution. Note that, as in the previous chapter, I will usually drop the subscript $\boldsymbol{\delta}$ from $\Lambda_{\boldsymbol{\delta}}(T)$ from now on — we can safely assume that $\boldsymbol{\delta}$ is always the distance matrix given in input. Finally, note that the algorithm returns the set Θ^* of *all* the BME-optimal topologies.

In this section, the main points left to discuss about BBBME are the steps where a choice has to be made either regarding the order with which the various extensions $T \underset{B_i}{\overset{A_i}{\vdash}} k$ of T (where i varies in $\{1, 2, \ldots, m\}$ and k is fixed) are processed — I will call this the *branching order* — and regarding the order with which the taxa are added to the topologies — the *taxon order*. Note that the latter also includes the choice of the first three taxa to include in T_0. In Algorithms 8 and 9, these are precisely the steps that begin with "choose".

It is clear that the branching order has no effect on the structure of the metatree, but only on the order with which the search is carried out inside the metatree. This may still have important consequences: different orders will mean that good solutions may be encountered earlier or later in the search. Clearly, the earlier good and optimal solutions tend to be encountered, the lower the score Λ^* of the best solution found tends to be at any point during the execution of BBBME. The lower this score, the more likely the condition $\beta(T, \boldsymbol{\delta}) > \Lambda^*$ is to be met and therefore the more likely it is for prunings to occur. Therefore a branching order that tends to give

Algorithm 9 PROCESS $(T, \Lambda^*, \Theta^*, \boldsymbol{\delta})$

if T contains all the taxa in $\{1, 2, \ldots, n\}$ **then**
  if $\Lambda(T) < \Lambda^*$ **then**
    $\Theta^* = \{T\}$
    $\Lambda^* = \Lambda(T)$
  end if
  if $\Lambda(T) = \Lambda^*$ **then** $\Theta^* = \Theta \cup \{T\}$
else
  Calculate the lower bound $\beta(T, \boldsymbol{\delta})$ to any T^+ below T.
  if $\beta(T, \boldsymbol{\delta}) > \Lambda^*$ **then return**
  Choose the next taxon to add, $k \notin T$.
  Choose an order $e_1, e_2, \ldots, e_m$ for the branches in T.
  Let clades A_i and B_i be defined for all $i \in \{1, 2, \ldots, m\}$ by $A_i \overset{e_i}{-} B_i$.
  for $i = 1, \ldots, m$ **do** PROCESS $(T \underset{B_i}{\overset{A_i}{\vdash}} k, \Lambda^*, \Theta^*, \boldsymbol{\delta})$
end else

good and optimal solutions early on may reduce the portion of the metatree that is actually traversed.

Intuitively, a way to try and encounter good solutions early on is to deal with the most promising partial solutions first: for example, the branching order could give precedence to the extensions $T'_i = T \underset{B_i}{\overset{A_i}{\vdash}} k$ with lower balanced length. As I will show in section 7.4, if we store the right information, each $\Lambda(T'_i)$ can be calculated from $\Lambda(T)$ in constant time; therefore we can efficiently sort the various extensions T'_i on the basis of their balanced length and deal with them starting from the shortest and ending with the longest. Alternatively, we could measure how "promising" an extension T'_i is by its lower bound $\beta(T'_i, \boldsymbol{\delta})$ and deal with the extensions in increasing order of $\beta(T'_i, \boldsymbol{\delta})$. However, for the bounds that I consider (sec. 7.3), this approach is equivalent to the one based on the balanced length of T'_i, and therefore I will not discuss this possibility further.

Although the computational overhead of sorting the extensions of a given T according to their balanced length is not very significant, a "lazier" alternative would be to deal with them in no particular order. If the heuristic used to construct the initial solution is sufficiently good, we can expect the lazy branching order to be more efficient, as often Λ^* will have already been set to its optimal value by the initial heuristic and there will be no need to improve it. Currently, my implementation of BBBME allows the user to choose between these two alternatives (the lazy and the sorted branching order).

The choice of an effective taxon order is, in my opinion, more important. Whereas the branching order only determines the order with which the metatree is explored,

the taxon order has an effect on which partial trees T appear at the vertices of the metatree. Ideally, we would like to construct partial topologies with large lower bounds $\beta(T, \boldsymbol{\delta})$ early on — that is, high up in the metatree — as these topologies are the most likely to have $\beta(T, \boldsymbol{\delta}) > \Lambda^*$ and therefore the most susceptible to prunings. Clearly, the earlier that prunings occur when going down the paths from the root of the metatree, the larger the pruned portions of the metatree will be, and the smaller will be the part of the metatree that is actually explored.

Therefore, we should choose the first three taxa, and the ones to add afterwards, so that the produced topologies T have large lower bounds $\beta(T, \boldsymbol{\delta})$. At first sight, this objective is in contrast to that for the branching order, where we want to produce promising partial topologies early on. However, note that the two choices are about very different things: when producing an extension $T \overset{A}{\underset{B}{\vdash}} k$ of the current tree T, first we decide *which* taxon k to add next and then *where* — between which clades A and B — to add it (or, more precisely, in which order it should be attached to the various branches). Because of the considerations above, good taxon and branching orders deal with the most "problematic" taxa early on and insert them in the most "promising" positions first. There is no real contradiction in doing so.

Dealing with the most problematic choices first (i.e. where to insert taxa that do not seem to fit well in the current tree) allows us to recognise early on solutions that are not worth pursuing.

An important point to note is that the taxon order does not need to be the same along each path from the root to a leaf of the metatree: for example, in figure 7.2.1, whereas for T_1 and T_2 the next taxon added is E, for T_3 taxon F is preferred — which means that adding E is postponed to other descendant vertices of T_3. This point allows us to make an important distinction between taxon ordering strategies [**PBTK00**]: a *static ordering* of the taxa is one where the order is chosen once and never changed throughout the rest of the execution of BBBME; a *dynamic ordering* is one where the choice of what taxon to add next is made at each vertex of the metatree. Clearly, whereas static orders are the same along each root-leaf path, dynamic orderings may, and usually will, produce orders that vary among different paths (as in fig. 7.2.1).

Many taxon ordering strategies — both static and dynamic — are possible, and their design depends more on the intuition of the designer than on any generally accepted principles. Because the choice of a taxon order should be guided towards producing topologies T with large $\beta(T, \boldsymbol{\delta})$ at high levels of the metatree, perhaps the most natural taxon order is given by the following directives:

- choose the first three taxa so that the topology T_0 connecting them maximises

$$\beta(T_0, \boldsymbol{\delta}), \tag{7.2.1}$$

- given a bifurcating topology T over a subset of $\{1, 2, \ldots, n\}$, with branches $e_1, e_2, \ldots, e_m$ and such that $A_i \overset{e_i}{-} B_i$ for every $i \in \{1, 2, \ldots, m\}$, choose the next taxon to add, $k \notin T$, so as to maximise

$$\min_{i \in \{1,2,\ldots,m\}} \beta(T \overset{A_i}{\underset{B_i}{\vdash}} k, \boldsymbol{\delta}). \tag{7.2.2}$$

Less formally, the "maximin" criterion in (7.2.2) can be explained like this: since we want many child vertices T' of the current vertex T to have a large lower bound, we choose the taxon k that *maximises the minimum lower bound* among the various child vertices T'. This ensures that *all* the child vertices of T make progress towards having a bound greater than Λ^* and therefore towards pruning. While implementing BBBME, I also tried an approach that took the *average* of the various bounds in (7.2.2), instead of the minimum. This clearly resulted in fewer prunings and therefore was quickly abandoned.

If we assume that the criterion in (7.2.2) is applied at each internal vertex T of the metatree, the taxon ordering strategy specified above is clearly dynamic. However, any dynamic ordering can easily be turned into a static one, with a technique that I will call *freezing*. It consists of exploiting criteria normally used for dynamic ordering to construct a static order that is then used for the rest of the execution of the branch and bound algorithm. Initially, only the first three positions in the order are defined — for example using (7.2.1). A new taxon is added to the rest of the order whenever the search reaches a new level of the metatree. Imagine we have a partial topology with, say, h taxa and we are considering the addition of one more taxon: we then look at the next position in the static order, the $(h+1)$th; if this has not yet been defined, a criterion such as that in (7.2.2) is used, and the resulting taxon k is appended in position $h + 1$ at the end of the order; from then on, whenever the algorithm deals with a partial topology on h taxa, taxon k will be the one to be added next. Thus the chosen dynamic criterion is used only once per level of the metatree, each time establishing the next taxon in the static order. When the search reaches one of the leaves of the metatree (typically very soon in the execution of the algorithm), the static order is then completely determined.

One of the advantages of using a static order instead of a dynamic one is that it reduces the amount of processing at each vertex T of the metatree. Whatever criterion we choose, dynamic ordering requires that we consider the various candidate taxa for addition and this may involve a non-negligible amount of computation: for example, in order to choose k with (7.2.2), we need to evaluate $(2h-3)(n-h)$ lower bounds (where h is the number of taxa in T). For static ordering, once the order has been established, these computations are avoided altogether.

Note that these considerations do not necessarily mean that static ordering leads to faster running times: the use of a dynamic order will tend to produce partial solutions with large bounds at higher levels in the metatree, which means that,

compared to a static ordering using the same bound, dynamic ordering will result in more pruning. However, sometimes static ordering allows us to calculate better (i.e., larger) lower bounds than dynamic ordering: the best bounds $\beta(T, \boldsymbol{\delta})$ that I will present (sec. 7.3) depend on the precise order with which the taxa not in T will be added to it. Since a dynamic order does not allow us to know this order in advance, these bounds can only be combined with a static order. The possibility of using better bounds implies that, overall, static orderings may lead to more prunings than dynamic orderings. Since the relative efficiency at pruning of the dynamic and static approaches may depend on the data in input, both of them are currently implemented in the BBBME program.

Finally note that the taxon ordering strategies based on (7.2.1) and (7.2.2) require calculation of the lower bounds $\beta(T, \boldsymbol{\delta})$ for a large number of topologies. If the calculation of these bounds is inefficient, it may be convenient to use the balanced length $\Lambda_{\boldsymbol{\delta}}(T)$ instead of $\beta(T, \boldsymbol{\delta})$, thus giving precedence to adding those taxa that lead to the construction of long trees, rather than trees with large lower bounds. This approach is formally identical to the one described above, and is obtained by replacing every appearance of $\beta(T, \boldsymbol{\delta})$ with $\Lambda_{\boldsymbol{\delta}}(T)$ in (7.2.1) and (7.2.2). Although the bounds that I present in the next section are very efficient to calculate, the alternative approach is also available for testing in the current implementation of BBBME.

7.3. Bounds for BME

A very convenient property of the parsimony scores of phylogenetic trees — which was key in the successful application of branch and bound to this criterion — is that, as we add taxa to partial trees (those at the internal vertices of the metatree) the parsimony score cannot decrease. Unfortunately, the score used by BME does not have this property. Let us then study the change in balanced length produced by adding a taxon k onto a branch of T.

First, straightforward application of Proposition 6.4.1 shows that

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) = \Lambda(A) + \Lambda(B) + \frac{1}{2}\,\delta_{AB} + \frac{1}{2}\,\delta_{A\{k\}} + \frac{1}{2}\,\delta_{B\{k\}},$$

and

$$\Lambda(T) = \Lambda(A) + \Lambda(B) + \delta_{AB}.$$

Therefore, the change in balanced length produced by adding taxon k onto the branch between clades A and B is given by

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) = \frac{1}{2}(\delta_{A\{k\}} + \delta_{B\{k\}} - \delta_{AB}). \tag{7.3.1}$$

As we will see, this is a crucial result that will be very useful to speed up the practical calculation of the balanced length of any extension of a partial tree T. For now, it is easy to see that the quantity in (7.3.1) can be negative — as it is easy to

construct distance matrices $\boldsymbol{\delta}$ in which δ_{AB} is very large and both $\delta_{A\{k\}}$ and $\delta_{B\{k\}}$ very small — and therefore the balanced length of a topology can actually decrease when adding a new taxon.

Note that the right-hand side in (7.3.1) is identical to the formula in (6.3.6) for the balanced estimate of the length of the branch leading to k in $T \overset{A}{\underset{B}{\vdash}} k$. If we recall that $\Lambda(T)$ can be defined as the sum of the balanced length estimates for all the branches in T (as explained in sec. 6.3.1), it is easy to see why this happens: using the formulae (6.3.6) and (6.3.7) for balanced length estimation, it is not difficult to find out (but not shown here) that the addition of k onto e, where $A \overset{e}{-} B$, does not alter the sum of the branch length estimates in A and B and that it divides e into two branches whose length estimates sum to the length estimate for e. Therefore, the only change in the balanced length of T resulting from the addition of k comes from the length of the newly added terminal branch leading to k, which is precisely given by (7.3.1).

For the purpose of devising a lower bound for BBBME, the most important consequence of (7.3.1) is that the change in balanced length can also be seen as a weighted average of many terms with a very simple form. This is expressed by the following proposition.

PROPOSITION 7.3.1. *Let T be a bifurcating tree topology over a subset of $\{1, 2, \ldots, n\}$ and $k \notin T$. Also, let A and B be two clades separated by exactly one branch in T. Then,*

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) = \sum_{\substack{i \in A \\ j \in B}} \frac{1}{2^{b_{iA}+b_{jB}}} \lambda_k^{ij}, \tag{7.3.2}$$

where b_{iX} denotes the depth of taxon i in clade X, that is, the number of branches between i and the root of X in T, and the λ_k^{ij} terms are defined as

$$\lambda_k^{ij} = \frac{1}{2}(\delta_{ik} + \delta_{jk} - \delta_{ij}). \tag{7.3.3}$$

PROOF. First of all note that, for any clade X,

$$\sum_{i \in X} \frac{1}{2^{b_{iX}}} = 1, \tag{7.3.4}$$

which is easy to see once $1/2^{b_{iX}}$ is seen as the probability of reaching taxon i when descending from the root of clade X, according to the very simple random walk described for the definition of the balanced average distance between clades (sec. 6.3.1). Other consequences of this definition (see (6.3.5)), and of (7.3.4), are the following equalities:

$$\delta_{A\{k\}} = \sum_{i \in A} \frac{1}{2^{b_{iA}}} \delta_{ik} = \sum_{i \in A} \frac{1}{2^{b_{iA}}} \delta_{ik} \sum_{j \in B} \frac{1}{2^{b_{jB}}} = \sum_{\substack{i \in A \\ j \in B}} \frac{1}{2^{b_{iA}+b_{jB}}} \delta_{ik},$$

$$\delta_{B\{k\}} = \sum_{j\in B} \frac{1}{2^{b_{jB}}}\,\delta_{jk} = \sum_{j\in B} \frac{1}{2^{b_{jB}}}\,\delta_{jk} \sum_{i\in A} \frac{1}{2^{b_{iA}}} = \sum_{\substack{i\in A\\ j\in B}} \frac{1}{2^{b_{iA}+b_{jB}}}\,\delta_{jk},$$

$$\delta_{AB} = \sum_{\substack{i\in A\\ j\in B}} \frac{1}{2^{b_{iA}}}\,\frac{1}{2^{b_{jB}}}\,\delta_{ij} = \sum_{\substack{i\in A\\ j\in B}} \frac{1}{2^{b_{iA}+b_{jB}}}\,\delta_{ij}.$$

If we use these formulae for $\delta_{A\{k\}}$, $\delta_{B\{k\}}$ and δ_{AB} in (7.3.1), then we obtain

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) = \frac{1}{2} \sum_{\substack{i\in A\\ j\in B}} \frac{1}{2^{b_{iA}+b_{jB}}}\,(\delta_{ik} + \delta_{jk} - \delta_{ij}) = \sum_{\substack{i\in A\\ j\in B}} \frac{1}{2^{b_{iA}+b_{jB}}}\,\lambda_k^{ij}.$$

□

Note also that, because of equation (7.3.4), the factors that multiply the λ_k^{ij} terms in (7.3.2) sum to 1:

$$\left(\sum_{i\in A} \frac{1}{2^{b_{iA}}}\right)\left(\sum_{j\in B} \frac{1}{2^{b_{jB}}}\right) = \sum_{\substack{i\in A\\ j\in B}} \frac{1}{2^{b_{iA}+b_{jB}}} = 1,$$

which means that we can see the expression in (7.3.2) as a weighted average of the various λ_k^{ij} terms, for $i \in A$ and $j \in B$.

The importance of (7.3.2) lies in the fact that it gives us a solid basis to devise bounds for BME: since it expresses the change in $\Lambda(T)$ produced by the addition of *one* taxon, if we can bound its value then probably we can also bound the change in $\Lambda(T)$ produced by the addition of *any* number of taxa. This very simple idea is at the basis of all the bounds I will present in the rest of this section.

Recall that our aim is to derive a lower bound to $\Lambda(T^+)$ for any bifurcating topology T^+ over $\{1, 2, \ldots n\}$ that can be obtained by iteratively extending T with all the taxa $k \notin T$ (i.e., for any T^+ lying at a leaf below T in the metatree described in sec. 7.2). I will start by describing two simple bounds that I initially devised for BBBME (sections 7.3.1 and 7.3.2). Although subsequently I found better bounds (sections 7.3.3 and 7.3.5), the simple bounds are useful to introduce some of the ideas that we need for the rest of this section. Note that good lower bounds are those that are close to the real minimum among all $\Lambda(T^+)$, for all possible iterated extensions T^+, as they are the most likely to result in prunings. A good bound is also said to be *tight*.

7.3.1. "α bound" for static orders. Since the right-hand side in (7.3.2) is a weighted average of many λ_k^{ij} terms, a very simple lower bound to its value is given by the minimum among these terms (which is independent of the attachment point specified by A and B), that is:

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) \geq \min_{i,j\in T:\, i\neq j} \lambda_k^{ij}. \tag{7.3.5}$$

Note that the $\min_{i,j} \lambda_k^{ij}$ in (7.3.5) does not depend on the topology of T, but only on the set of taxa in T. If the taxa are added in a known order, for example determined through static ordering early on during the execution of BBBME, then the set of taxa in T is totally determined by k and therefore $\min_{i,j} \lambda_k^{ij}$ can be calculated as soon as the order is available.

Assume, without loss of generality, that the taxa are numbered so that the initial tree T_0 is composed of taxa $1, 2$ and 3 and that the subsequent taxa are added in the order $4, 5, \ldots, n$. Then, when we proceed to add taxon k ($4 \leq k \leq n$) onto the partial topology T, the taxa in T must be $\{1, 2, \ldots, k-1\}$ and we can rewrite the right-hand side of (7.3.5) as

$$\alpha_k = \min_{i,j:\, 1 \leq i < j < k} \lambda_k^{ij}, \tag{7.3.6}$$

where we impose $i < j$, instead of $i \neq j$, because $\lambda_k^{ij} = \lambda_k^{ji}$.

Since the addition of taxon k will cause a change in balanced length of at least α_k, it is very simple now to express a lower bound to the balanced length of any complete extension T^+ of T:

$$\Lambda(T^+) \geq \Lambda(T) + \sum_{k \notin T} \alpha_k. \tag{7.3.7}$$

Note that α_k can be calculated as soon as the position of taxon k is determined in the static order; α_k is then stored and used throughout the execution of BBBME.

As for the amount of computation involved in calculating the bound in (7.3.7), assume that T is obtained by adding taxon h to another partial topology T'. It is easy to see that the sum $\sum_{k \notin T} \alpha_k$ can be obtained in constant time by simply subtracting α_h from the corresponding sum for T' (which has been calculated when visiting T'). Since also $\Lambda(T)$ can be calculated in $O(1)$ time from $\Lambda(T')$ (once some necessary information for T' is calculated and stored; see sec. 7.4), the bound in (7.3.7) can be calculated very efficiently in constant time.

7.3.2. "α' bound" for dynamic orders. The α bound above relies on the assumption that, for every taxon k still to add, it is possible to determine the set of taxa that will have been added before k. Unfortunately, if the taxon order is determined dynamically, this is not possible. A way to solve this issue is to assume that *any* taxa may have been added before k: if we go back to the bound in (7.3.5), instead of taking the minimum over all pairs of distinct taxa in T, we can take the minimum over all pairs of distinct taxa from $\{1, 2, \ldots, n\} - \{k\}$. Because this minimum is over a larger set, it must be less or equal than the previous minimum, and therefore it will still be a lower bound to the change in $\Lambda(T)$. Therefore,

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) \geq \min_{\substack{1 \leq i < j \leq n, \\ i,j \neq k}} \lambda_k^{ij}.$$

If now we define

$$\alpha'_k = \min_{\substack{1\leq i<j\leq n,\\ i,j\neq k}} \lambda_k^{ij}, \tag{7.3.8}$$

then, as in (7.3.7), we have that

$$\Lambda(T^+) \geq \Lambda(T) + \sum_{k\notin T} \alpha'_k. \tag{7.3.9}$$

Clearly, since $\alpha'_k \leq \alpha_k$, this bound is not as tight as the α bound for static orders in (7.3.7) and will result in fewer prunings. Although it is mostly intended for dynamic orders, it can also be used while a static order is being constructed, for example to choose the first three taxa in the order.

Note that the α'_k values can be calculated right at the start of the execution of BBBME (as they only depend on $\boldsymbol{\delta}$). Because of the same considerations as for (7.3.7), the bound in (7.3.9) can be calculated very efficiently in constant time.

7.3.3. "γ bound" for static orders. If we look back at (7.3.2), it is not hard to see that taking the minimum among the λ_k^{ij} terms does not provide a very tight lower bound to the right-hand side: there are many other terms in this weighted average that contribute to its value; taking them into account has the potential to make the bound significantly tighter.

Note that the weights in the right-hand side of (7.3.2) depend on the topologies of A and B, the clades at the sides of the newly added taxon k. However, a lower bound to (7.3.2) should be independent of A and B: when evaluating the bound for a T not including k, neither the topology, nor the position to which k will be added are known (however we do know that the topology will have been obtained by repeated extension of T). Because, as in section 7.3.1, we here assume that taxa are numbered $1, 2, \ldots, n$ in accordance with a static order, we can assume that k is added to an unknown topology over $\{1, 2, \ldots, k-1\}$. By hypothesizing an "extreme" shape for this unknown topology, we can then derive an extreme form for the right-hand side in (7.3.2). This is the main idea leading to the bound presented in this section. The following definition will be central to express this bound.

Definition 7.3.2. Denote by $\lambda_k^{(h)}$ the h-th smallest value among the λ_k^{ij} such that $1 \leq i < j < k$. This means that $(\lambda_k^{(1)}, \lambda_k^{(2)}, \ldots, \lambda_k^{\binom{k-1}{2}})$ is obtained by sorting the vector $(\lambda_k^{ij})_{1\leq i<j<k}$ in ascending order.

Note that the definition above, although apparently straightforward, conceals an important detail in case of ties: for example, if $k = 4$ and $(\lambda_k^{12}, \lambda_k^{13}, \lambda_k^{23}) = (1, 2, 1)$, the second smallest among these values, $\lambda_k^{(2)}$ is equal to 1: multiple occurrences of the same value are counted separately.

I will start by deriving bounds to (7.3.2) for small values of k and then I will generalise them to any value of k. If $k = 4$, the topology to which 4 is added is T_0, connecting taxa 1, 2 and 3 to a central node. Depending on the branch onto which

4 is attached, the one leading to 1, to 2 or to 3, the right-hand side of (7.3.2) will have one of the following forms:

$$\frac{1}{2}\lambda_4^{12} + \frac{1}{2}\lambda_4^{13},$$
$$\frac{1}{2}\lambda_4^{12} + \frac{1}{2}\lambda_4^{23},$$
$$\frac{1}{2}\lambda_4^{13} + \frac{1}{2}\lambda_4^{23}.$$

Recall that $\lambda_4^{(1)}$ denotes the smallest value in $(\lambda_4^{12}, \lambda_4^{13}, \lambda_4^{23})$, and $\lambda_4^{(2)}$ the second smallest among these values (Def. 7.3.2); then the minimum among the three expressions above is simply given by:

$$\frac{1}{2}\lambda_4^{(1)} + \frac{1}{2}\lambda_4^{(2)}, \tag{7.3.10}$$

which is therefore an optimally tight bound on the balanced length change caused by adding 4, and is independent of where 4 actually attaches.

If $k = 5$, the right-hand side of (7.3.2) will have either the form

$$\frac{1}{4}\lambda_5^{i_1 j_1} + \frac{1}{4}\lambda_5^{i_1 j_2} + \frac{1}{4}\lambda_5^{i_2 j_1} + \frac{1}{4}\lambda_5^{i_2 j_2},$$

or

$$\frac{1}{2}\lambda_5^{ij_1} + \frac{1}{4}\lambda_5^{ij_2} + \frac{1}{4}\lambda_5^{ij_3}$$

depending on whether 5 gets attached to the internal branch or one of the terminal branches. Using again Definition 7.3.2, it is easy to see that a lower bound to the expressions above is given by

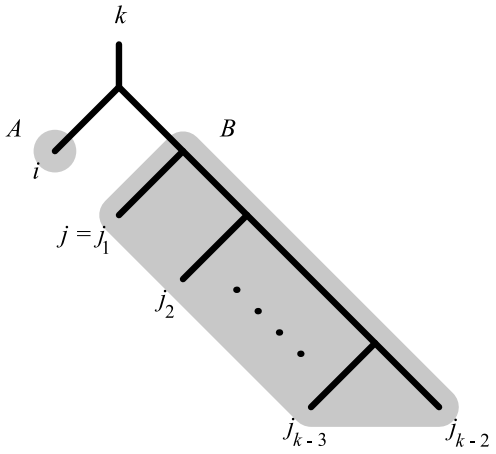
$$\frac{1}{2}\lambda_5^{(1)} + \frac{1}{4}\lambda_5^{(2)} + \frac{1}{4}\lambda_5^{(3)}. \tag{7.3.11}$$

The case $k = 6$ is also instructive, but showing the details here would be tedious; by enumerating all possible forms for the right-hand side of (7.3.2) one can show that a lower bound to them is:

$$\frac{1}{2}\lambda_5^{(1)} + \frac{1}{4}\lambda_5^{(2)} + \frac{1}{8}\lambda_5^{(3)} + \frac{1}{8}\lambda_5^{(4)}. \tag{7.3.12}$$

In general, a lower bound to the right-hand side of (7.3.2) is obtained by hypothesizing that the topology to which k is added is such that *the smallest* λ_k^{ij} *are multiplied by the largest possible weights*. Let A and B denote once again the clades at the sides of the branch to which k is attached. The largest possible weight that the minimum λ_k^{ij} term, that is $\lambda_k^{(1)}$, can have is $\frac{1}{2}$, which is obtained when one taxon between i and j, say, i is at depth 0 in, say, A and j is at depth 1 in the other clade B, as shown in figure 7.3.1 (note that j cannot be at depth 0, unless $k = 3$). Assuming that $\lambda_k^{(1)}$ gets a weight of $\frac{1}{2}$, the largest possible weight that $\lambda_k^{(2)}$ can have is then $\frac{1}{4}$: this can only happen if $\lambda_k^{(2)} = \lambda_k^{ij_2}$, where i is the only taxon in clade A and j_2 is a taxon at depth 2 in B. If we continue reasoning like this, we find that, for every $h \in \{1, 2, \ldots, k-3\}$, the largest weight that $\lambda_k^{(h)}$ can have — assuming all

FIGURE 7.3.1. Topology justifying Proposition 7.3.3.
The dots indicate that the terminal branches leading to $j_3, j_4, \ldots, j_{k-4}$ are omitted.



smaller $\lambda_k^{(i)}$ have been assigned weights in the way above — is $1/2^h$, which can only happen if $\lambda_k^{(h)} = \lambda_k^{ij_h}$ and the topology to which k is added is that shown in figure 7.3.1.

The greedy-like procedure described above leads to the scenario that gives the lowest possible value for the right-hand side of (7.3.2): it is the scenario where k is added to the topology in figure 7.3.1 (and in the position shown), where $\lambda_k^{ij_h} = \lambda_k^{(h)}$ for every $h \in \{1, 2, \ldots, k-2\}$. The change in balanced length for this scenario provides a lower bound to the change in balanced length for any possible scenario; it is a generalisation of (7.3.10), (7.3.11) and (7.3.12) for any value of $k \geq 3$. This is formalised by the following proposition.

PROPOSITION 7.3.3. *Let T be a bifurcating tree topology over $\{1, 2, \ldots, k-1\}$ with $k \in \{3, 4, \ldots, n\}$. Also, let A and B be two clades separated by exactly one branch in T. Then,*

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) \geq \sum_{i=1}^{k-3} \frac{1}{2^i}\,\lambda_k^{(i)} + \frac{1}{2^{k-3}}\,\lambda_k^{(k-2)}. \tag{7.3.13}$$

The reasons why this holds should now be intuitive, but a formal proof for the statement above will be given in the next subsection.

Now define γ_k as the right-hand side of (7.3.13), that is,

$$\gamma_k = \sum_{i=1}^{k-3} \frac{1}{2^i}\,\lambda_k^{(i)} + \frac{1}{2^{k-3}}\,\lambda_k^{(k-2)}. \tag{7.3.14}$$

Clearly, γ_k is tighter than α_k (defined in sec. 7.3.1) as a lower bound to the balanced length change caused by adding k, simply because $\gamma_k \geq \lambda_k^{(1)} = \alpha_k$. As a consequence, the following is a tighter lower bound than the α bound specified by (7.3.6) and

(7.3.7):

$$\Lambda(T^+) \geq \Lambda(T) + \sum_{k \notin T} \gamma_k. \tag{7.3.15}$$

Computationally, the overhead for the calculation of γ_k instead of α_k is minimal, as each γ_k is calculated only once as soon as the position of k is determined in the static order; γ_k is then stored and used throughout the execution of BBBME.

Again, $\sum_{k \notin T} \gamma_k$ can be obtained by simply subtracting a γ_h value from the corresponding sum for the parent vertex of T in the metatree. Since this means that also the bound in (7.3.15) can be calculated in constant time, the γ bound presented here should be preferred to the less tight α bound (which was only described for introductory purposes).

7.3.4. Correctness of the γ bound. In this subsection, intuitive concepts such as weightings and weighted averages are given precise meanings. This allows us to prove Proposition 7.3.3, although in a rather convoluted way.

Definition 7.3.4. A *weighting* is a vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_m)$ of non-negative real numbers summing to 1. Given a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_t)$ with at least as many components as $\boldsymbol{w}$ (i.e., $m \leq t$), the *weighted average of $\boldsymbol{x}$ given $\boldsymbol{w}$* is defined by:

$$\boldsymbol{w} \cdot \boldsymbol{x} = \sum_{i=1}^{m} w_i x_i.$$

Weightings are just a way to represent distributions over a finite set of elements. I here introduce a notion of dominance between weightings that corresponds to that of "first-order stochastic dominance" between distributions [**SS94**].

Definition 7.3.5. Let $\boldsymbol{w}$ and $\boldsymbol{v}$ be two vectors of lengths m and n, respectively. We say that $\boldsymbol{w}$ *dominates* $\boldsymbol{v}$ if, for every $k \leq \min\{m, n\}$,

$$\sum_{i=1}^{k} w_i \leq \sum_{i=1}^{k} v_i.$$

The definition above will normally be applied to weightings. For example, it is easy to check that $\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$ dominates $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$. An intuitive way of describing the dominance relation above is to say that $\boldsymbol{w}$ dominates $\boldsymbol{v}$ when $\boldsymbol{w}$ is more "to the right" than $\boldsymbol{v}$. The rationale for the definition above comes from the following observation (whose proof will be given further below in this section).

Proposition 7.3.6. *The weighting*

$$\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{k-4}}, \frac{1}{2^{k-3}}, \frac{1}{2^{k-3}}\right)$$

is dominated by every weighting of the form

$$\left(\frac{1}{2^{b_{iA} + b_{jB}}}\right)_{i \in A,\, j \in B} \tag{7.3.16}$$

(for whatever order its components appear in), where A and B are clades that partition the taxa in $\{1, 2, \ldots, k-1\}$.

The above proposition, together with the following lemma, implies Proposition 7.3.3.

LEMMA 7.3.7. *Let $\boldsymbol{w}$ and $\boldsymbol{v}$ be two weightings such that $\boldsymbol{w}$ dominates $\boldsymbol{v}$. Then, for every non-decreasing vector $\boldsymbol{x}$ with at least as many elements as $\boldsymbol{w}$ and $\boldsymbol{v}$,*

$$\boldsymbol{v} \cdot \boldsymbol{x} \leq \boldsymbol{w} \cdot \boldsymbol{x}.$$

PROOF. Assume, without loss of generality, that vectors $\boldsymbol{w}$, $\boldsymbol{v}$ and $\boldsymbol{x}$ all have the same length n: in fact, if this is not the case, it suffices to append at the end of $\boldsymbol{w}$ and $\boldsymbol{v}$ a number of zero components until their length equals that of $\boldsymbol{x}$; it is easy to see that this transformation does not change the values of $\boldsymbol{w} \cdot \boldsymbol{x}$ and $\boldsymbol{v} \cdot \boldsymbol{x}$.

Now define, for all $k \in \{1, 2, \ldots, n\}$, $\boldsymbol{d}_k$ as a vector of length n with the first $k-1$ components equal to 0 and the remaining ones equal to $x_k - x_{k-1}$ (with $x_0 := 0$), that is:

$$\begin{aligned}
\boldsymbol{d}_1 &= x_1(1, 1, \ldots, 1), \\
\boldsymbol{d}_2 &= (x_2 - x_1)(0, 1, , \ldots, 1), \\
\boldsymbol{d}_3 &= (x_3 - x_2)(0, 0, 1, , \ldots, 1),
\end{aligned}$$

and so on. Then it is easy to see that

$$\boldsymbol{x} = \boldsymbol{d}_1 + \boldsymbol{d}_2 + \ldots + \boldsymbol{d}_n. \tag{7.3.17}$$

I now prove that, for every $k \in \{1, 2, \ldots, n\}$,

$$\boldsymbol{v} \cdot \boldsymbol{d}_k \leq \boldsymbol{w} \cdot \boldsymbol{d}_k. \tag{7.3.18}$$

First, this clearly holds for $k = 1$, as $\boldsymbol{v} \cdot \boldsymbol{d}_1 = x_1 = \boldsymbol{w} \cdot \boldsymbol{d}_1$. Then, for every $k \in \{2, 3, \ldots, n\}$,

$$\begin{aligned}
\boldsymbol{v} \cdot \boldsymbol{d}_k &= (x_k - x_{k-1}) \sum_{i=k}^{n} v_i \\
&= (x_k - x_{k-1})(1 - \sum_{i=1}^{k-1} v_i) \\
&\leq (x_k - x_{k-1})(1 - \sum_{i=1}^{k-1} w_i) \\
&= (x_k - x_{k-1}) \sum_{i=k}^{n} w_i \;=\; \boldsymbol{w} \cdot \boldsymbol{d}_k.
\end{aligned}$$

Therefore, because of (7.3.17) and (7.3.18), we have that

$$\boldsymbol{v} \cdot \boldsymbol{x} = \boldsymbol{v} \cdot \boldsymbol{d}_1 + \boldsymbol{v} \cdot \boldsymbol{d}_2 + \ldots + \boldsymbol{v} \cdot \boldsymbol{d}_n \leq \boldsymbol{w} \cdot \boldsymbol{d}_1 + \boldsymbol{w} \cdot \boldsymbol{d}_2 + \ldots + \boldsymbol{w} \cdot \boldsymbol{d}_n = \boldsymbol{w} \cdot \boldsymbol{x}.$$

□

I now show that Proposition 7.3.6 and Lemma 7.3.7 above imply Proposition 7.3.3. Let T, A, B and $\lambda_k^{(i)}$ be as in the statement of Proposition 7.3.3. By re-expressing the right-hand side of (7.3.2), we obtain:

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) = \sum_{\substack{i\in A\\ j\in B}} \frac{1}{2^{b_{iA}+b_{jB}}}\,\lambda_k^{ij} = \boldsymbol{w}\cdot(\lambda_k^{(1)}, \lambda_k^{(2)}, \ldots, \lambda_k^{\binom{k-1}{2}}), \tag{7.3.19}$$

where $\boldsymbol{w}$ a weighting obtained by rearranging $\left(\frac{1}{2^{b_{iA}+b_{jB}}}\right)_{i\in A,\, j\in B}$ in some way. Because of Proposition 7.3.6, we have that $\boldsymbol{w}$ dominates $\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{k-4}}, \frac{1}{2^{k-3}}, \frac{1}{2^{k-3}}\right)$. Therefore, because of Lemma 7.3.7, we have that

$$\boldsymbol{w}\cdot(\lambda_k^{(1)}, \lambda_k^{(2)}, \ldots, \lambda_k^{\binom{k-1}{2}}) \geq \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{k-4}}, \frac{1}{2^{k-3}}, \frac{1}{2^{k-3}}\right)\cdot(\lambda_k^{(1)}, \lambda_k^{(2)}, \ldots, \lambda_k^{\binom{k-1}{2}}). \tag{7.3.20}$$

But (7.3.19) and (7.3.20) together imply that

$$\Lambda\left(T \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T) \geq \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{k-4}}, \frac{1}{2^{k-3}}, \frac{1}{2^{k-3}}\right)\cdot(\lambda_k^{(1)}, \lambda_k^{(2)}, \ldots, \lambda_k^{\binom{k-1}{2}}),$$

which is just a restatement of (7.3.13), the inequality I set out to prove.

The rest of this section is devoted to the proof of Proposition 7.3.6. I start by stating a number of basic facts about the properties of the dominance relation defined above. The proof of the first two is trivial and therefore not included here.

LEMMA 7.3.8. *The dominance relation of Definition 7.3.5 is a partial order over the set of all weightings. That is, if $\boldsymbol{w}$ and $\boldsymbol{v}$ are two weightings, then: $\boldsymbol{w}$ dominates $\boldsymbol{w}$ (reflexivity); if $\boldsymbol{w}$ dominates $\boldsymbol{v}$ and $\boldsymbol{v}$ dominates $\boldsymbol{w}$, then $\boldsymbol{w}$ and $\boldsymbol{v}$ are the same weighting, up to inclusion of trailing zeros (antisymmetry); if $\boldsymbol{w}$ dominates $\boldsymbol{v}$ and $\boldsymbol{v}$ dominates $\boldsymbol{u}$, then $\boldsymbol{w}$ dominates $\boldsymbol{u}$ (transitivity).*

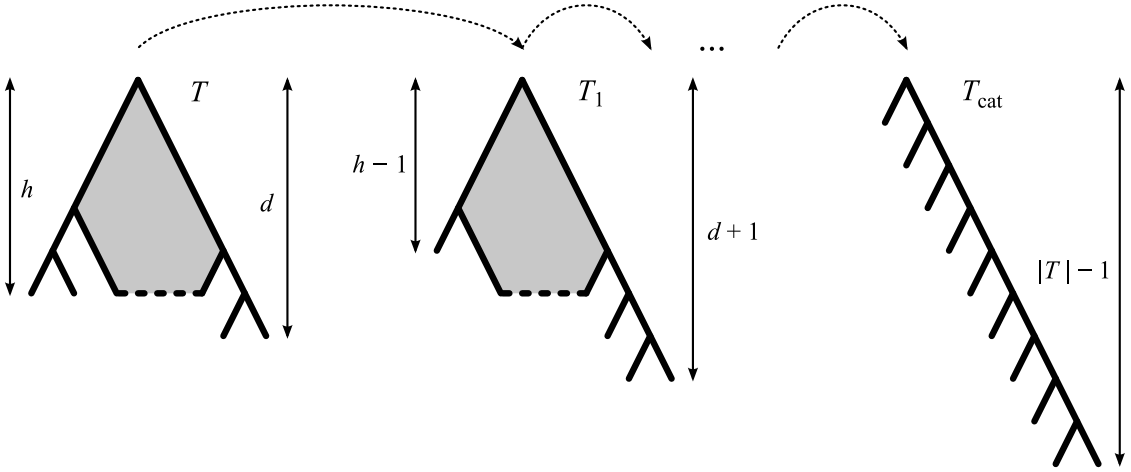
LEMMA 7.3.9. *Let $\boldsymbol{w}$ and $\boldsymbol{v}$ be identical weightings, except for a number of components, say, the i_1-th, the i_2-th, down to the i_k-th (with $i_1 < i_2 < \ldots < i_k$). If $(w_{i_1}, w_{i_2}, \ldots, w_{i_k})$ dominates $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$, then $\boldsymbol{w}$ dominates $\boldsymbol{v}$.*

Note that two weightings may differ on components where one of the two weightings is undefined; in this case we assume that the undefined weight equals 0. For example, we noted above that $\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$ dominates $\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$. Because of Lemma 7.3.9, this can be proved by simply remarking that the two weightings differ at the first and fourth component and clearly $\left(\frac{1}{4}, \frac{1}{4}\right)$ dominates $\left(\frac{1}{2}, 0\right)$.

LEMMA 7.3.10. *Given a weighting $\boldsymbol{w}$, let $\boldsymbol{w}^s$ denote the vector obtained by sorting $\boldsymbol{w}$ in descending order. Then, $\boldsymbol{w}$ dominates $\boldsymbol{w}^s$.*

PROOF. Note that $\boldsymbol{w}$ can be transformed into $\boldsymbol{w}^s$ via a number of "swaps" between consecutive components (a procedure also known as *bubblesort*; see [**CLRS01**], Problem 2-2). Each swap consists of transforming a weighting $(\ldots, x, y, \ldots)$ into the

FIGURE 7.3.2. Illustration of the proof of Proposition 7.3.11.



new weighting $(\ldots, y, x, \ldots)$, with $x \leq y$, and where all components denoted by the ellipses are left unchanged. Because of Lemma 7.3.9, the old weighting dominates the new one. By repeatedly applying the transitive property of dominance we conclude that $\boldsymbol{w}$ dominates $\boldsymbol{w}^s$. □

LEMMA 7.3.11. *Let T be a rooted topology (therefore, a clade). The weighting*

$$\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|T|-2}}, \frac{1}{2^{|T|-1}}, \frac{1}{2^{|T|-1}}\right)$$

is dominated by every weighting of the form

$$\left(\frac{1}{2^{b_{iT}}}\right)_{i \in T},$$

irrespective of the components' order.

PROOF. Note that T being a clade, all the notation for clades carries over to T. Recall that b_{iT} is also called the *depth* of i in T. Also, call a clade of two taxa a *cherry*.

Imagine repeatedly applying the following operation to T: unless T has a caterpillar shape (see T_{cat} in fig. 7.3.2), remove one of the taxa from a cherry at depth h and insert it into a different cherry of maximal depth d, with $h \leq d$. It is clear that the above procedure terminates when T has been transformed into a topology T_{cat} with a caterpillar shape.

Now let $\boldsymbol{w}_0 = \left(\frac{1}{2^{b_{iT}}}\right)_{i \in T}$. Irrespective of its components' order, this weighting dominates

$$\boldsymbol{w}_0^s = \left(\ldots, \frac{1}{2^h}, \frac{1}{2^h}, \ldots, \frac{1}{2^d}, \frac{1}{2^d}\right),$$

which is a consequence of Lemma 7.3.10. But, in turn, this weighting dominates the following weighting corresponding to T_1:

$$\boldsymbol{w}_1 = \left(\ldots, \frac{1}{2^{h-1}}, \frac{1}{2^d}, \ldots, \frac{1}{2^{d+1}}, \frac{1}{2^{d+1}}\right),$$

which is easy to verify thanks to Lemma 7.3.9 (the components denoted by the ellipses are left unchanged from $\boldsymbol{w}_0^s$). Finally, again because of Lemma 7.3.10, $\boldsymbol{w}_1$ dominates $\boldsymbol{w}_1^s$. Because of the transitive property of dominance, then $\boldsymbol{w}_0$ dominates $\boldsymbol{w}_1^s$, where, I recall, $\boldsymbol{w}_1^s$ is the vector obtained by sorting $\left(\frac{1}{2^{b_{iT_1}}}\right)_{i\in T_1}$ in descending order.

Proceeding like this, every time we move a new taxon to a cherry of maximum depth, the weighting corresponding to the new topology, once sorted in descending order, is dominated by the weighting corresponding to the old topology. In the end, applying the transitive property multiple times, we have that the sorted weighting corresponding to T_{cat}, that is

$$\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|T|-2}}, \frac{1}{2^{|T|-1}}, \frac{1}{2^{|T|-1}}\right),$$

is dominated by $\boldsymbol{w}_0$, which is precisely what I set out to prove. □

LEMMA 7.3.12. *Let $\boldsymbol{w}$ and $\boldsymbol{v}$ be two weightings of lengths m and n, respectively, and let $\boldsymbol{w}\otimes\boldsymbol{v}$, their Kronecker product, be the vector containing the products between every pair of components from $\boldsymbol{w}$ and $\boldsymbol{v}$, that is,*

$$\boldsymbol{w}\otimes\boldsymbol{v} = (w_1v_1, w_1v_2, \ldots, w_1v_n, w_2v_1, w_2v_2, \ldots, w_2v_n, \ldots, w_mv_1, w_mv_2, \ldots, w_mv_n).$$

Then $\boldsymbol{w}\otimes\boldsymbol{v}$ is a weighting. Furthermore, if $\boldsymbol{w}$ dominates another weighting $\boldsymbol{w}'$, then $\boldsymbol{w}\otimes\boldsymbol{v}$ dominates $\boldsymbol{w}'\otimes\boldsymbol{v}$. Similarly, if $\boldsymbol{v}$ dominates another weighting $\boldsymbol{v}'$, then $\boldsymbol{w}\otimes\boldsymbol{v}$ dominates $\boldsymbol{w}\otimes\boldsymbol{v}'$.

PROOF. That $\boldsymbol{w}\otimes\boldsymbol{v}$ is a weighting is trivial.

The fact that $\boldsymbol{w}$ dominates $\boldsymbol{w}'$ implies that $(w_1v_i, w_2v_i, \ldots, w_mv_i)$ dominates $(w_1'v_i, w_2'v_i, \ldots, w_m'v_i)$, for every $i \in \{1, 2, \ldots, n\}$. Therefore, by Lemma 7.3.9,

$$\boldsymbol{w}\otimes\boldsymbol{v} = (w_1v_1, w_1v_2, \ldots, w_1v_n, w_2v_1, w_2v_2, \ldots, w_2v_n, \ldots, w_mv_1, w_mv_2, \ldots, w_mv_n)$$

dominates

$$(w_1'v_1, w_1v_2, \ldots, w_1v_n, w_2'v_1, w_2v_2, \ldots, w_2v_n, \ldots, w_m'v_1, w_mv_2, \ldots, w_mv_n),$$

which in turn dominates

$$(w_1'v_1, w_1'v_2, \ldots, w_1v_n, w_2'v_1, w_2'v_2, \ldots, w_2v_n, \ldots, w_m'v_1, w_m'v_2, \ldots, w_mv_n),$$

which, in turn, by repeated application of Lemma 7.3.9, dominates

$$(w_1'v_1, w_1'v_2, \ldots, w_1'v_n, w_2'v_1, w_2'v_2, \ldots, w_2'v_n, \ldots, w_m'v_1, w_m'v_2, \ldots, w_m'v_n) = \boldsymbol{w}'\otimes\boldsymbol{v}.$$

The proof that $\boldsymbol{w}\otimes\boldsymbol{v}$ dominates $\boldsymbol{w}\otimes\boldsymbol{v}'$ is similar. □

DEFINITION 7.3.13. Let $\boldsymbol{w}$ be a weighting of length m. We say that $\boldsymbol{v}$ is a *subvector* of $\boldsymbol{w}$ if $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) = (w_{i_1}, w_{i_2}, \ldots, w_{i_n})$, with $1 \leq i_1 < i_2 < \ldots < i_n \leq m$. A collection of subvectors of $\boldsymbol{w}$ form a *partition* of $\boldsymbol{w}$ if their concatenation is a rearrangement of $\boldsymbol{w}$.

LEMMA 7.3.14. *Let $\boldsymbol{w}$, $\boldsymbol{v}$, $\boldsymbol{w}'$ and $\boldsymbol{v}'$ be weightings such that $\boldsymbol{w}$ dominates $\boldsymbol{w}'$ and $\boldsymbol{v}$ dominates $\boldsymbol{v}'$. If $\boldsymbol{w}$ is non-increasing then $(\boldsymbol{w} \otimes \boldsymbol{v})^s$ dominates $(\boldsymbol{w}' \otimes \boldsymbol{v})^s$. If $\boldsymbol{v}$ is non-increasing then $(\boldsymbol{w} \otimes \boldsymbol{v})^s$ dominates $(\boldsymbol{w} \otimes \boldsymbol{v}')^s$.*

PROOF. The proof is very similar to that of Lemma 7.3.12. If $\boldsymbol{w}$ is non-increasing, then the various $(w_1 v_i, w_2 v_i, \ldots, w_m v_i)$ are subvectors of $(\boldsymbol{w} \otimes \boldsymbol{v})^s$ (and form a partition of this weighting). Note that this was also true in the proof of Lemma 7.3.12. Similarly to that proof, we can then substitute in $(\boldsymbol{w} \otimes \boldsymbol{v})^s$ each subvector $(w_1 v_i, w_2 v_i, \ldots, w_m v_i)$ with $(w'_1 v_i, w'_2 v_i, \ldots, w'_m v_i)$ and the resulting vector, $\boldsymbol{u}$, is dominated by $(\boldsymbol{w} \otimes \boldsymbol{v})^s$ (by repeated application of Lemma 7.3.9). Since $\boldsymbol{u}$ dominates $\boldsymbol{u}^s = (\boldsymbol{w}' \otimes \boldsymbol{v})^s$, I conclude that $(\boldsymbol{w} \otimes \boldsymbol{v})^s$ dominates $(\boldsymbol{w}' \otimes \boldsymbol{v})^s$.

The proof that if $\boldsymbol{v}$ is non-increasing then $(\boldsymbol{w} \otimes \boldsymbol{v})^s$ dominates $(\boldsymbol{w} \otimes \boldsymbol{v}')^s$ is similar. □

I am now ready to prove Proposition 7.3.6, thus concluding the proof of Proposition 7.3.3. Let A and B be two clades that partition the taxa in $\{1, 2, \ldots, k-1\}$. I wish to prove that

$$\boldsymbol{w}_0 = \left(\frac{1}{2^{b_{iA} + b_{jB}}} \right)_{i \in A,\, j \in B},$$

irrespective of the order in which its components appear, dominates

$$\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{k-4}}, \frac{1}{2^{k-3}}, \frac{1}{2^{k-3}} \right).$$

First, for any clade X, define

$$\boldsymbol{w}_X = \left(\frac{1}{2^{b_{iA}}} \right)_{i \in X}.$$

Then, it is easy to see that

$$\boldsymbol{w}_0^s = (\boldsymbol{w}_A \otimes \boldsymbol{w}_B)^s = (\boldsymbol{w}_A^s \otimes \boldsymbol{w}_B^s)^s.$$

Now define A_1 (and B_1) as any clade with a caterpillar topology, containing all the taxa in A (and B) (see fig. 7.3.3). Note that

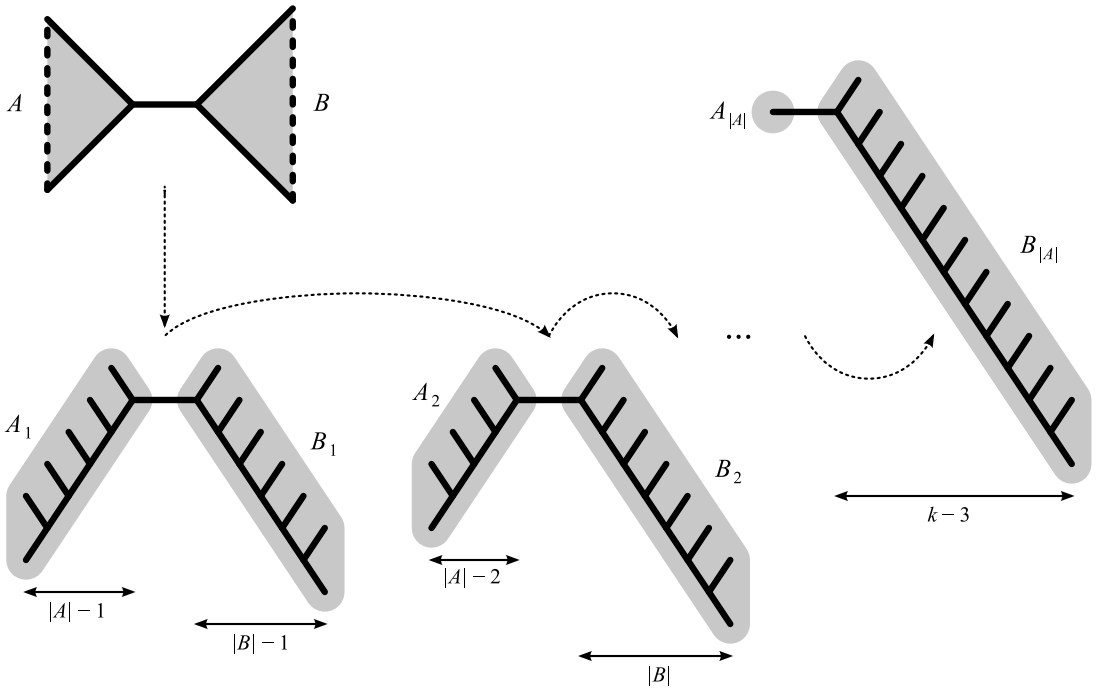
$$\boldsymbol{w}_{A_1}^s = \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|A|-2}}, \frac{1}{2^{|A|-1}}, \frac{1}{2^{|A|-1}} \right)$$

and

$$\boldsymbol{w}_{B_1}^s = \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|B|-2}}, \frac{1}{2^{|B|-1}}, \frac{1}{2^{|B|-1}} \right).$$

Because of Lemma 7.3.11, $\boldsymbol{w}_A^s$ dominates $\boldsymbol{w}_{A_1}^s$ and $\boldsymbol{w}_B^s$ dominates $\boldsymbol{w}_{B_1}^s$. Then, because of Lemma 7.3.14, $\boldsymbol{w}_0^s = (\boldsymbol{w}_A^s \otimes \boldsymbol{w}_B^s)^s$ dominates $(\boldsymbol{w}_{A_1}^s \otimes \boldsymbol{w}_{B_1}^s)^s$.

FIGURE 7.3.3. Illustration of the proof of Proposition 7.3.6.



Now assume, without loss of generality, that $|A| \leq |B|$. Imagine moving one of the taxa in A_1 into B_1, so that B_1 has still a caterpillar topology. I call the resulting clades A_2 and B_2. If we iterate this operation, we obtain a sequence of pairs of clades $(A_1, B_1), (A_2, B_2), \ldots, (A_{|A|}, B_{|A|})$ that terminates when one clade is composed of one taxon and the other of $k-2$ taxa (see fig. 7.3.3).

Now define, for every $h \in \{1, 2, \ldots, |A|\}$,

$$\boldsymbol{w}_h = \left(\frac{1}{2^{b_{iA_h} + b_{jB_h}}} \right)_{i \in A_h, j \in B_h}.$$

Because $\boldsymbol{w}_0$ dominates $\boldsymbol{w}_0^s$, which in turn dominates $(\boldsymbol{w}_{A_1}^s \otimes \boldsymbol{w}_{B_1}^s)^s = \boldsymbol{w}_1^s$, we have that $\boldsymbol{w}_0$ dominates $\boldsymbol{w}_1^s$. I will now prove that $\boldsymbol{w}_1^s$ dominates $\boldsymbol{w}_2^s$. The proof is valid for every pair of consecutive weightings in the sequence $\boldsymbol{w}_1^s, \boldsymbol{w}_2^s, \ldots, \boldsymbol{w}_{|A|}^s$. Therefore, because of transitivity, we have that $\boldsymbol{w}_0$ dominates

$$\boldsymbol{w}_{|A|}^s = \left(\frac{1}{2^{b_{jB_{|A|}}}} \right)_{j \in B_{|A|}}^s = \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{k-4}}, \frac{1}{2^{k-3}}, \frac{1}{2^{k-3}} \right),$$

which is what I set out to prove.

It remains to be proved that $\boldsymbol{w}_1^s$ dominates $\boldsymbol{w}_2^s$. Since $\boldsymbol{w}_{A_1}^s$ can be partitioned into subvectors $(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|A|-2}})$ and $(\frac{1}{2^{|A|-1}}, \frac{1}{2^{|A|-1}})$ and, on the other hand, $\boldsymbol{w}_{B_1}^s$ can be partitioned into subvectors $(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|B|-1}})$ and $(\frac{1}{2^{|B|-1}})$, then the following four subvectors of $\boldsymbol{w}_1^s = (\boldsymbol{w}_{A_1}^s \otimes \boldsymbol{w}_{B_1}^s)^s$ form a partition of $\boldsymbol{w}_1^s$:

$$\boldsymbol{a} = \left(\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|A|-2}} \right) \otimes \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|B|-1}} \right) \right)^s,$$

$$\boldsymbol{b} = \left(\left(\frac{1}{2^{|A|-1}}, \frac{1}{2^{|A|-1}}\right) \otimes \left(\frac{1}{2^{|B|-1}}\right)\right)^s = \left(\frac{1}{2^{|A|+|B|-2}}, \frac{1}{2^{|A|+|B|-2}}\right),$$

$$\boldsymbol{c} = \left(\left(\frac{1}{2}, \frac{1}{4}, \cdots, \frac{1}{2^{|A|-2}}\right) \otimes \left(\frac{1}{2^{|B|-1}}\right)\right)^s = \left(\frac{1}{2^{|B|}}, \frac{1}{2^{|B|+1}}, , \cdots, \frac{1}{2^{|A|+|B|-3}}\right),$$

$$\boldsymbol{d} = \left(\left(\frac{1}{2^{|A|-1}}, \frac{1}{2^{|A|-1}}\right) \otimes \left(\frac{1}{2}, \frac{1}{4}, \cdots, \frac{1}{2^{|B|-1}}\right)\right)^s$$

$$= \left(\frac{1}{2^{|A|}}, \frac{1}{2^{|A|}}, \frac{1}{2^{|A|+1}}, \frac{1}{2^{|A|+1}}, \cdots, \frac{1}{2^{|A|+|B|-2}}, \frac{1}{2^{|A|+|B|-2}}\right).$$

If we merge together $\boldsymbol{c}$ and $\boldsymbol{d}$, we obtain another subvector of $\boldsymbol{w}_1^s$:

$$\boldsymbol{x} = \left(\frac{1}{2^{|A|}}, \frac{1}{2^{|A|}}, \frac{1}{2^{|A|+1}}, \frac{1}{2^{|A|+1}}, \cdots, \frac{1}{2^{|B|-1}}, \frac{1}{2^{|B|-1}}, \frac{1}{2^{|B|}}, \frac{1}{2^{|B|}}, \frac{1}{2^{|B|}},\right.$$

$$\left.\frac{1}{2^{|B|+1}}, \frac{1}{2^{|B|+1}}, \frac{1}{2^{|B|+1}}, \cdots, \frac{1}{2^{|A|+|B|-3}}, \frac{1}{2^{|A|+|B|-3}}, \frac{1}{2^{|A|+|B|-3}}, \frac{1}{2^{|A|+|B|-2}}, \frac{1}{2^{|A|+|B|-2}}\right).$$

Now imagine substituting, in $\boldsymbol{w}_1^s$, the above subvector with the following one:

$$\boldsymbol{y} = \left(\frac{1}{2^{|A|-1}}, 0, \frac{1}{2^{|A|}}, 0, \ldots, \frac{1}{2^{|B|-2}}, 0, \frac{1}{2^{|B|-1}}, \frac{1}{2^{|B|+1}}, \frac{1}{2^{|B|+1}},\right.$$

$$\left.\frac{1}{2^{|B|}}, \frac{1}{2^{|B|+2}}, \frac{1}{2^{|B|+2}}, \cdots, \frac{1}{2^{|A|+|B|-4}}, \frac{1}{2^{|A|+|B|-2}}, \frac{1}{2^{|A|+|B|-2}}, \frac{1}{2^{|A|+|B|-3}}, 0\right).$$

(note that there is a one-to-one correspondence between the representation of $\boldsymbol{x}$ above and the one for $\boldsymbol{y}$). The weighting obtained with this substitution, which I call $\boldsymbol{u}_2$, has two properties:

(1) $\boldsymbol{w}_1^s$ dominates $\boldsymbol{u}_2$, as $\boldsymbol{x}$ dominates $\boldsymbol{y}$ (apply Lemma 7.3.9);
(2) if the zeros it contains are ignored, $\boldsymbol{u}_2$ is a rearrangement of $\boldsymbol{w}_2$.

The second property can be proved as follows. Since $\boldsymbol{w}_{A_2}^s$ can be partitioned into subvectors $(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|A|-2}})$ and $(\frac{1}{2^{|A|-2}})$ and $\boldsymbol{w}_{B_2}^s$ can be partitioned into subvectors $(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|B|-1}})$ and $(\frac{1}{2^{|B|}}, \frac{1}{2^{|B|}})$, then the following four subvectors of $\boldsymbol{w}_2^s = (\boldsymbol{w}_{A_2}^s \otimes \boldsymbol{w}_{B_2}^s)^s$ form a partition of $\boldsymbol{w}_2^s$:

$$\boldsymbol{a}' = \left(\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|A|-2}}\right) \otimes \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|B|-1}}\right)\right)^s,$$

$$\boldsymbol{b}' = \left(\left(\frac{1}{2^{|A|-2}}\right) \otimes \left(\frac{1}{2^{|B|}}, \frac{1}{2^{|B|}}\right)\right)^s = \left(\frac{1}{2^{|A|+|B|-2}}, \frac{1}{2^{|A|+|B|-2}}\right),$$

$$\boldsymbol{c}' = \left(\left(\frac{1}{2^{|A|-2}}\right) \otimes \left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|B|-1}}\right)\right)^s = \left(\frac{1}{2^{|A|-1}}, \frac{1}{2^{|A|}}, , \ldots, \frac{1}{2^{|A|+|B|-3}}\right),$$

$$\boldsymbol{d}' = \left(\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^{|A|-2}}\right) \otimes \left(\frac{1}{2^{|B|}}, \frac{1}{2^{|B|}}\right)\right)^s$$

$$= \left(\frac{1}{2^{|B|+1}}, \frac{1}{2^{|B|+1}}, \frac{1}{2^{|B|+2}}, \frac{1}{2^{|B|+2}}, \ldots, \frac{1}{2^{|A|+|B|-2}}, \frac{1}{2^{|A|+|B|-2}}\right).$$

Now recall that $\boldsymbol{u}_2$, on the other hand, is partitioned in subvectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{y}$ and observe that $\boldsymbol{a} = \boldsymbol{a}'$ and $\boldsymbol{b} = \boldsymbol{b}'$. It suffices to observe that $\boldsymbol{y}$ can be obtained by

merging and rearranging $\boldsymbol{c}'$ and $\boldsymbol{d}'$ (and inserting a few zeros) to conclude that $\boldsymbol{u}_2$ is a rearrangement of $\boldsymbol{w}_2$ (with a few zeros inserted).

Because $\boldsymbol{w}_1^s$ dominates $\boldsymbol{u}_2$ (point (1) above), $\boldsymbol{u}_2$ dominates $\boldsymbol{u}_2^s$ (Lemma 7.3.10) and $\boldsymbol{u}_2^s$ dominates $\boldsymbol{w}_2^s$ (because, except for a few trailing zeros, they are identical), then $\boldsymbol{w}_1^s$ dominates $\boldsymbol{w}_2^s$.

7.3.5. "γ' bound" for dynamic orders. If the taxon order is determined dynamically, when we add taxon k not only do we not know *which* taxa will have been added before, but also we do not know *how many* taxa will be present in the topology to which k is added. In section 7.3.2, we solved the former issue by assuming that *any* taxa from $\{1, 2, \ldots, n\} - \{k\}$ may have been added before k, so we replaced α_k, the minimum λ_k^{ij} for $i, j \in \{1, 2, \ldots, k-1\}$, with α'_k, the minimum λ_k^{ij} for $i, j \in \{1, 2, \ldots, n\} - \{k\}$. Similarly, the following definition replaces Definition 7.3.2:

Definition 7.3.15. Denote by $\lambda_k^{(h)'}$ the h-th smallest value among the λ_k^{ij} such that $1 \leq i < j \leq n$ and $k \notin \{i, j\}$. This means that $(\lambda_k^{(1)'}, \lambda_k^{(2)'}, \ldots, \lambda_k^{\binom{n-1}{2}'})$ is obtained by sorting the vector $(\lambda_k^{ij})_{1 \leq i < j \leq n,\, i,j \neq k}$ in ascending order.

One may be tempted to obtain a bound by simply replacing $\lambda_k^{(i)}$ with $\lambda_k^{(i)'}$ in (7.3.13); however, that equation presupposes that the topology to which k is added exactly contains $k-1$ taxa, an assumption that we cannot make for dynamic orders.

Note however that the bounds we are discussing are evaluated when visiting a particular topology T. Then, for every $k \notin T$, we want to predict the effect that adding k has on the balanced length of an unknown topology that will have been obtained by repeated extension of T. Although we may not know how many taxa there will be in this unknown topology, we *can* however assume that it will contain at least the same number of taxa as T. Thanks to this assumption, a bound analogous to that in (7.3.13) can be devised; it is given in the following proposition, where I have chosen to call T' the unknown topology to which k is added so as to make clear the distinction with T.

Proposition 7.3.16. *Let T' be a bifurcating tree topology with at least h taxa ($h \geq 2$), but not containing k. Also, let A and B be two clades separated by exactly one branch in T'. Then,*

$$\Lambda\left(T' \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T') \geq \sum_{i=1}^{h-2} \frac{1}{2^i}\, \lambda_k^{(i)'} + \frac{1}{2^{h-2}}\, \lambda_k^{(h-1)'}. \tag{7.3.21}$$

Before showing a proof for this proposition, define $\gamma'_k(h)$ as the right-hand side in (7.3.21), that is,

$$\gamma'_k(h) = \sum_{i=1}^{h-2} \frac{1}{2^i}\, \lambda_k^{(i)'} + \frac{1}{2^{h-2}}\, \lambda_k^{(h-1)'}.$$

PROOF. Suppose that T' is over a set X of x taxa, with $h \leq x \leq n-1$. As for Proposition 7.3.3, where $X = \{1, 2, \ldots, k-1\}$, let $\lambda_k^{(i)}$ denote the i-th smallest value among the λ_k^{ij} such that $i, j \in X$ and $i < j$. Then, the following is equivalent to Proposition 7.3.3:

$$\Lambda\left(T' \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T') \geq \sum_{i=1}^{x-2} \frac{1}{2^i}\,\lambda_k^{(i)} + \frac{1}{2^{x-2}}\,\lambda_k^{(x-1)}.$$

Because $X \subseteq \{1, 2, \ldots, n\} - \{k\}$, then clearly $\lambda_k^{(i)} \geq \lambda_k^{(i)'}$. Therefore, if we replace each $\lambda_k^{(i)}$ with $\lambda_k^{(i)'}$ in the right-hand side of the equation above, we still have a lower bound to the change in balanced length and this lower bound is precisely equal to $\gamma'_k(x)$.

Since $h \leq x \leq n-1$, and since

$$\gamma'_k(h) \leq \gamma'_k(h+1) \leq \ldots \leq \gamma'_k(n-1),$$

then,

$$\Lambda\left(T' \overset{A}{\underset{B}{\vdash}} k\right) - \Lambda(T') \geq \gamma'_k(h).$$

□

Clearly, $\gamma'_k(h)$, for any $h \geq 2$, is tighter than α'_k (defined in sec. 7.3.2) as a lower bound to the balanced length change caused by adding k, simply because $\gamma'_k(h) \geq \lambda_k^{(1)'} = \alpha'_k$.

Once again, a lower bound to the balanced length change caused by adding one taxon can be used to derive a bound for when multiple taxa are added: if we denote the number of taxa in T by $|T|$, we have

$$\Lambda(T^+) \geq \Lambda(T) + \sum_{k \notin T} \gamma'_k(|T|). \tag{7.3.22}$$

Since $\gamma'_k(|T|) \geq \alpha'_k$, the γ' bound in (7.3.22) is tighter than the α' bound specified by (7.3.8) and (7.3.9). On the other hand, the γ bound specified by (7.3.14) and (7.3.15) is tighter than the γ' bound: as usual, assume that the taxa are numbered with $1, 2, \ldots, n$ in accordance with the static order; if $\lambda_k^{(i)}$ is defined as in Definition 7.3.2, then $\lambda_k^{(i)} \geq \lambda_k^{(i)'}$ and therefore

$$\begin{aligned} \gamma_k &= \sum_{i=1}^{k-3} \frac{1}{2^i}\,\lambda_k^{(i)} + \frac{1}{2^{k-3}}\,\lambda_k^{(k-2)} \\ &\geq \sum_{i=1}^{k-3} \frac{1}{2^i}\,\lambda_k^{(i)'} + \frac{1}{2^{k-3}}\,\lambda_k^{(k-2)'} = \gamma'_k(k-1). \end{aligned}$$

If T is a topology over $\{1, 2, \ldots, h\}$, then $|T| = h$ and

$$\Lambda(T) + \sum_{k=h+1}^{n} \gamma_k \geq \Lambda(T) + \sum_{k=h+1}^{n} \gamma'_k(k-1) \geq \Lambda(T) + \sum_{k=h+1}^{n} \gamma'_k(h),$$

which shows that the γ' bound is less tight than the γ bound.

Finally, note that although the various $\gamma'_k(x)$ values, for $x \in \{3, 4, \dots n-1\}$ and $k \in \{1, 2, \dots, n\}$, can be calculated right at the start of the execution of BBBME, the sum $\sum_{k \notin T} \gamma'_k(|T|)$ needs to be calculated *de novo* each time we visit a new topology T.

7.3.6. The α and the γ bounds are effective when the distances fit a tree. I will now show that the α bound (see (7.3.6) and (7.3.7)) and its improvement for static orders, the γ bound ((7.3.14) and (7.3.15)), share some desirable properties that ensure that, if the input distances are sufficiently close to the correct distances, then BBBME runs efficiently. In the rest of this section, the following notations will often be used.

DEFINITION 7.3.17. Let T and T^* be two tree topologies, such that the taxa in T form a subset of the taxa in T^*. The form $T \leq T^*$ indicates that T^* can be obtained by iteratively extending T with the taxa that are in T^* but not in T. Formally, this means that we require that there is a sequence of topologies $T_0, T_1, \dots, T_m$, with $m \geq 0$, such that $T_0 = T$, $T_m = T^*$ and, for every $i \in \{1, 2, \dots, m\}$, T_i coincides with $T_{i-1} \overset{A}{\underset{B}{\vdash}} k$ for some A and B clades of T_{i-1} and some taxon k not in T_{i-1}. Also, I will write $T \not\leq T^*$ to indicate that T^* cannot be obtained by iteratively extending T in the way described above.

Note that if T is at a vertex of the metatree and T^* is at a leaf of the metatree, then $T \leq T^*$ if and only if T is on the path between the root of the metatree and T^*.

A first interesting remark is that if the distances coincide with those on the correct tree and the partial topology T is correct for the taxa it contains, then both the α and the γ bound for T are optimally tight, i.e., they precisely give the balanced length of the best T^+ below T.

PROPOSITION 7.3.18. *Let $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$ for some tree $\mathcal{T}$ over $\{1, 2, \dots, n\}$ having a bifurcating topology T^* and let T be a topology over $\{1, 2, \dots, h\}$ such that $T \leq T^*$. Then the α and the γ bounds applied to T are equal to the balanced length of T^*:*

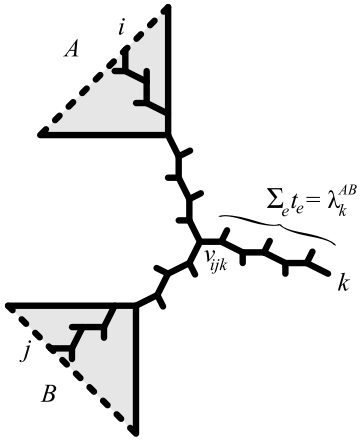
$$\Lambda(T) + \sum_{k \notin T} \alpha_k = \Lambda(T) + \sum_{k \notin T} \gamma_k = \Lambda(T^*). \tag{7.3.23}$$

PROOF. Since $T \leq T^*$, there must be a sequence of topologies $T_h = T, T_{h+1}, \dots, T_n = T^*$ such that, for every $k \in \{h+1, h+2, \dots, n\}$, T_k is over taxa $\{1, 2, \dots, k\}$ and is obtained by adding taxon k onto one of T_{k-1}'s branches. I will now prove that, for every $k \in \{h+1, h+2, \dots, n\}$,

$$\Lambda(T_k) - \Lambda(T_{k-1}) = \alpha_k = \gamma_k,$$

which implies (7.3.23).

FIGURE 7.3.4. Explanatory illustration for the proof of Proposition 7.3.18. The figure shows the three paths in T^* from i, j and k to the central node v_{ijk}. A and B are clades of T_k, but not necessarily of T^*. They are however subtrees of T^*.



First, because we assume $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$, it is easy to see that

$$\lambda_k^{ij} = \frac{1}{2}(\delta_{ik} + \delta_{jk} - \delta_{ij}) = \frac{1}{2}(d_{ik}^{\mathcal{T}} + d_{jk}^{\mathcal{T}} - d_{ij}^{\mathcal{T}}).$$

Taxa i, j and k are connected in $\mathcal{T}$ through three paths that lead to a central node v_{ijk}. Looking at the expression above, it is clear that λ_k^{ij} equals the sum of the lengths of the branches in the path in $\mathcal{T}$ between v_{ijk} and k. See figure 7.3.4 for an illustration of this fact.

Now let $T_k = T_{k-1} \overset{A}{\underset{B}{\vdash}} k$, for some clades A and B separated by one branch in T_{k-1}. Then, as expressed in equation (7.3.2), $\Lambda(T_k) - \Lambda(T_{k-1})$ equals a weighted average of many λ_k^{ij} terms with $i \in A$ and $j \in B$. As just observed above, each of these terms is equal to the length of the path in $\mathcal{T}$ between v_{ijk} and k; also, it is easy to see that v_{ijk} is independent of the particular choice of $i \in A$ and $j \in B$, which means that each of the λ_k^{ij} terms in the average is equal to the same quantity, which I call λ_k^{AB} (see figure 7.3.4). Therefore,

$$\Lambda(T_k) - \Lambda(T_{k-1}) = \lambda_k^{AB}. \tag{7.3.24}$$

Now consider the various λ_k^{ij} for $1 \leq i < j < k$. Since $A \cup B = \{1, 2, \ldots, k-1\}$, i and j are such that either they both belong to one clade between A and B, in which case it is easy to see that

$$\lambda_k^{ij} \geq \lambda_k^{AB}, \tag{7.3.25}$$

or one of them belongs to A and the other to B, in which case, as in figure 7.3.4,

$$\lambda_k^{ij} = \lambda_k^{AB}. \tag{7.3.26}$$

From (7.3.25) and (7.3.26) follows that

$$\alpha_k = \min\{\lambda_k^{ij} : 1 \leq i < j < k\} = \lambda_k^{AB}. \tag{7.3.27}$$

Furthermore, since there are at least $|A| \cdot |B| \geq k-2$ pairs of taxa i and j such that $\lambda_k^{ij} = \lambda_k^{AB}$, then $\lambda_k^{(1)} = \lambda_k^{(2)} = \ldots = \lambda_k^{(k-2)} = \lambda_k^{AB}$. Therefore,

$$\gamma_k = \sum_{i=1}^{k-3} \frac{1}{2^i} \lambda_k^{(i)} + \frac{1}{2^{k-3}} \lambda_k^{(k-2)} = \lambda_k^{AB}. \tag{7.3.28}$$

By putting together (7.3.24), (7.3.27) and (7.3.28), we finally have:

$$\Lambda(T_k) - \Lambda(T_{k-1}) = \alpha_k = \gamma_k.$$

□

The result above regarding the case that T is correct for the taxa it contains ($T \leq T^*$) is naturally followed by the following result, which deals with the alternative case where T is incorrect ($T \not\leq T^*$).

PROPOSITION 7.3.19. *Let $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$ for some tree $\mathcal{T}$ over $\{1, 2, \ldots, n\}$ with positive branch lengths and with a bifurcating topology T^*. Let T be a topology over $\{1, 2, \ldots, h\}$ such that $T \not\leq T^*$. Then the α and the γ bounds applied to T are strictly greater than the balanced length of T^*:*

$$\Lambda(T) + \sum_{k \notin T} \alpha_k = \Lambda(T) + \sum_{k \notin T} \gamma_k > \Lambda(T^*). \tag{7.3.29}$$

PROOF. Imagine eliminating taxon n and its terminal branch e from $\mathcal{T}$; if we replace the two branches e_1 and e_2 that were attached to e with a new branch, and assign to this new branch a length equal to the sum of the lengths of e_1 and e_2, then we obtain a new tree over $\{1, 2, \ldots, n-1\}$, which I call $\mathcal{T}_{n-1}$. If we continue eliminating one taxon at a time in this way, after $n-h$ steps we obtain a tree $\mathcal{T}_h$ over $\{1, 2, \ldots, h\}$, whose topology T_h is such that $T_h \leq T^*$ (as T^* can be obtained from T_h by reverting the procedure described above). Furthermore, because of the way $\mathcal{T}_h$ is constructed, the distances between pairs of taxa i and j in $\mathcal{T}_h$ still coincide with the δ_{ij} input. Therefore, we can see $\mathcal{T}_h$ as the "correct" tree and T as an alternative, incorrect topology over the same set of taxa. Because of the consistency of BME (Theorem 6.3.3), then we must have

$$\Lambda(T) > \Lambda(T_h). \tag{7.3.30}$$

Because $T_h \leq T^*$, we can then apply Proposition 7.3.18 to T_h and obtain

$$\Lambda(T_h) + \sum_{k \notin T} \alpha_k = \Lambda(T_h) + \sum_{k \notin T} \gamma_k = \Lambda(T^*). \tag{7.3.31}$$

By combining (7.3.30) and (7.3.31), the conclusion in (7.3.29) follows. □

The two propositions above give an important insight into the behaviour of BBBME. Imagine its execution with distances $\boldsymbol{\delta} = \mathbf{d}^{\mathcal{T}}$ that perfectly fit a tree. First of all, the chosen heuristic to construct the initial tree should be able to obtain the correct topology T^*, and therefore it should set Λ^* to its correct value $\Lambda(T^*)$. (The

ability to reconstruct the correct tree with perfect data is a minimal requirement that any tree reconstruction method should satisfy.) Then the proper branch and bound phase follows and the metatree described in section 7.2 is explored. When BBBME visits an internal vertex T of the metatree, if this vertex is on the path to T^*, i.e., $T \leq T^*$, then the α or the γ bound applied to T precisely give $\Lambda(T^*) = \Lambda^*$ (Prop. 7.3.18), which means that the search is iterated on all extensions of T. If instead T is not on the "correct" path to T^*, i.e., $T \not\leq T^*$, then, because of Proposition 7.3.19, the bound applied to T is strictly greater than Λ^*, and therefore the corresponding portion of the metatree is pruned. This means that the only the vertices that are visited are the ones on the path to T^*, and their direct descendants; thus BBBME will run very efficiently. This is not too surprising, if we consider that the optimal solution has already been found by the initial heuristic. However, it is a nice property that holds in the limit as the distances become more and more "tree-like". The following proposition formalises this in a slightly stronger way.

PROPOSITION 7.3.20. *Let $\mathcal{T}$ be a tree over $\{1, 2, \ldots, n\}$ with positive branch lengths and with a bifurcating topology T^*. If $\boldsymbol{\delta}$ is sufficiently close to $\mathbf{d}^{\mathcal{T}}$ then BBBME only visits the vertices of the metatree that are on the path from the root to T^* and their direct descendants.*

PROOF. First, note that reasonable heuristics for BME can be expected to construct the correct topology T^* for distances $\boldsymbol{\delta}$ in a neighbourhood of $\mathbf{d}^{\mathcal{T}}$. (Results on the size of this neighbourhood are available for heuristics such as NJ [**Att99**] and BSPR [**BGHM09**].)

For every topology T that BBBME encounters such that $T \not\leq T^*$, the relationships proved in Proposition 7.3.19,

$$\Lambda(T) + \sum_{k \notin T} \alpha_k > \Lambda(T^*)$$

and

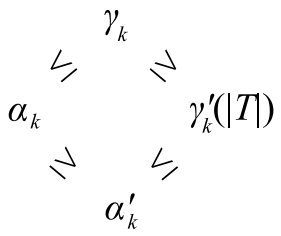
$$\Lambda(T) + \sum_{k \notin T} \gamma_k > \Lambda(T^*),$$

still hold for distances $\boldsymbol{\delta}$ in a neighbourhood of $\mathbf{d}^{\mathcal{T}}$, as all the quantities involved are continuous functions of $\boldsymbol{\delta}$. Therefore every incorrect partial T is pruned by BBBME.

These observations, together with the considerations above about the behaviour of BBBME with perfect data, complete this proof. □

Since BBBME runs in a time proportional to the number of visited topologies, the proposition above implies that BBBME will be efficient for distances that fit well onto a tree. Although most times the input distances in $\boldsymbol{\delta}$ will not be close enough to $\mathbf{d}^{\mathcal{T}}$ to ensure the performance guarantees of Proposition 7.3.20, we can nevertheless expect that the running times of BBBME grow somehow continuously as $\boldsymbol{\delta}$ gets further and further from $\mathbf{d}^{\mathcal{T}}$; it is certainly good news to know that in the limit BBBME runs in polynomial time.

FIGURE 7.3.5. Relative tightness of the bounds used by BBBME.



Proposition 7.3.20 still holds in the case where $\mathcal{T}$ has branches with zero length. In this case, there are multiple optimal bifurcating trees T^*, but BBBME will only visit the topologies T such that $T \leq T^*$ for some optimal T^*(and their direct descendants). Then, BBBME will run in a time proportional to the number of optimal solutions (which however grows exponentially in the number of internal branches with zero length).

7.3.7. Better bounds? All the bounds described in the last few subsections use the same idea: first find a lower bound to the change in balanced length produced by adding one taxon — α_k, α'_k, γ_k or $\gamma'_k(|T|)$ — and then sum up all the bounds for all taxa k not yet present in the current partial topology T. This gives a bound of the form:

$$\Lambda(T^+) \geq \Lambda(T) + \sum_{k \notin T} f_k(T).$$

The relative tightness of these bounds is determined by the relative tightness of the taxon-dependent bounds $f_k(T)$. A summary of the relationships among these bounds is given by the diagram in figure 7.3.5, which shows that the γ bound is better than both the α and the γ' bound, which in turn are both better than the α' bound.

An important observation is that — as implied by the notation I chose above — the bound $f_k(T)$ to the change associated to k can be potentially a function not only of k but also of the topology T being evaluated. Whereas α_k, α'_k and γ_k do not depend on T, in section 7.3.5 I did take into account one aspect of T: the number of taxa it contains. However, none of the bounds discussed uses the actual topology specified by T.

It is likely that better bounds than the ones I have presented here can be obtained by using more information about T. For example, if i and j are separated by b_{ij} branches in the current tree T, then the weight for λ_k^{ij} in the expression (7.3.2) for the change in balanced length will not be greater than $1/2^{b_{ij}-1}$. In the current best bound, the γ bound, if $\lambda_k^{ij} = \lambda_k^{(h)}$ (with $h \leq k-3$) then this term gets a weight of $1/2^h$, irrespective of whether or not this weight can in fact be achieved. This suggests that there is scope for better, tighter, bounds than the ones I have presented here. Probably, however, better bounds would involve more computation.

7.4. Computational aspects

The descriptions I gave in the previous two sections are enough for a first implementation of BBBME. However, there are some ideas that allow a considerable speed-up. This section will describe the most important among these ideas.

I already mentioned that there is no need to recalculate the balanced length for each new topology we encounter: it is much better to derive it from the balanced length of previously visited topologies. To see this, consider again equation (7.3.1):

$$\Lambda(T') = \Lambda(T) + \frac{1}{2}(\delta_{A\{k\}} + \delta_{B\{k\}} - \delta_{AB}),$$

where T' has been obtained from T by adding k between clades A and B. Since T' is visited after T, we can expect $\Lambda(T)$ to be available; if we ensure that the three δ terms in the right-hand side are also available (discussed below), then $\Lambda(T')$ can be calculated in constant time.

One possible approach would be to calculate $\delta_{A\{k\}}, \delta_{B\{k\}}$ and δ_{AB} each time we construct a new topology T'. However, it is much more efficient to calculate the terms above for all possible choices of A and B all together, when the current topology T is visited, as this avoids repeating many common operations. This is an idea originally by Desper and Gascuel, who used it for the heuristics implemented in FastME [**DG02**].

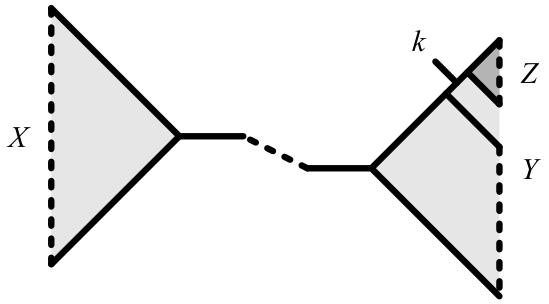
When we visit the partial topology T, there are two pieces of information that we would like to have in order to evaluate efficiently the balanced length of all extensions T' of T:

(1) For each clade X in T, the balanced average distance $\delta_{X\{k\}}$ between that clade and the taxon k that we are going to add next.
(2) For each pair of clades X and Y separated by a single branch in T, the balanced average distance δ_{XY} between those clades, which is useful when a taxon is inserted precisely between those clades.

The first interesting remark is that all the quantities in (1) can be calculated very efficiently in $O(|T|)$ time (where again $|T|$ denotes the number of taxa in T). This is achieved with two traversals of T (first bottom-up and then top-down), using the recursion $\delta_{X\{k\}} = (\delta_{X_1\{k\}} + \delta_{X_2\{k\}})/2$, where $X = X_1 \cup X_2$ (see Appendix 3 in [**DG02**] for details).

On the other hand, the *de novo* calculation of the δ_{XY} quantities in (2) would be inefficient. It is again more convenient to derive these quantities from the corresponding ones for the tree from which T was obtained. In fact, in order to make this derivation possible, it is necessary to store more than just the distances between clades separated by a single branch. To see this, let X and Y be two non-overlapping clades, i.e., such that $X \cap Y = \emptyset$. Suppose that a taxon k is added onto some branch of Y so that Y ceases to be a clade (see fig. 7.4.1). In the new tree topology, instead of Y we have the clade $Y' = Y \cup \{k\}$, and its average balanced distance from X is

FIGURE 7.4.1. Illustration useful for the derivation of $\delta_{XY'}$, with $Y' = Y \cup \{k\}$, from δ_{XY}.



given by:

$$\delta_{XY'} = \delta_{XY} + \frac{1}{2^{b_{kY'}}}(\delta_{X\{k\}} - \delta_{XZ}) \tag{7.4.1}$$

(see sec. 3.1 in [**DG02**] for the derivation), where Z is the subclade of Y that lies below the insertion point of k (see fig. 7.4.1). It is then clear that, even in the case that X and Y are separated by a single branch, in order to derive $\delta_{XY'}$ we need the average distance δ_{XZ} between clades that are separated by more than one branch.

As a consequence of this observation, BBBME stores a matrix Δ^T containing all balanced average distances δ_{XY} between all pairs of non-overlapping clades in T. As a new taxon k is added to T, this matrix is updated by using equation (7.4.1) and the $\delta_{X\{k\}}$ previously obtained as described above. As for the computational effort required to update Δ^T, note that for each clade X not containing k in the new tree T', the number of clades Y' for which $\delta_{XY'}$ needs to be calculated is precisely equal to the number of branches in the path from the root of X to k. Therefore there are $O(|T|\operatorname{diam}(T))$ pairs of clades X and Y' for which $\delta_{XY'}$ will be calculated. Since the calculation of a single $\delta_{XY'}$ requires constant time, the update of Δ^T then requires $O(|T|\operatorname{diam}(T))$ time.

The considerations I made in this section lead to a more detailed description of BBBME, the one in the pseudocode of Algorithms 10 and 11, where the new symbol $\boldsymbol{\delta}_k$ denotes a vector containing all balanced average distances $\delta_{X\{k\}}$ for all clades X in the current topology. Note that the processing of a topology is now carried out by the subroutine ATTACH (k, e, Λ); once this subroutine is called, the current topology T is modified by attaching the taxon specified by k onto the branch specified by e and all the necessary computations involved in visiting T are carried out. The third argument of ATTACH (k, e, Λ), Λ, specifies the balanced length $\Lambda(T)$ that will result from updating T; it is passed to the subroutine because it is already available before T is actually constructed: in the calling instance of ATTACH, the various Λ_e values are potentially used to choose a particular branching order.

There are still many details that are omitted from this description of BBBME: in particular the calculation of the bound and the choice of the taxa to add. However

Algorithm 10 BBBME ($\boldsymbol{\delta}$)

Execute a heuristic that constructs an initial topology T^*.
Choose the first three taxa $i, j, k \in \{1, 2, \ldots, n\}$.
Set the following as global variables:
    $\boldsymbol{\delta}$,
    $\Lambda^* = \Lambda(T^*)$,
    $\Theta^* = \{T^*\}$,
    $T = \{e\}$, the topology consisting of just one branch e connecting i and j,
    $\boldsymbol{\delta}_k = (\delta_{ik}, \delta_{jk})$,
    $\Delta^T = (\delta_{ij})$.
ATTACH $(k, e, \frac{1}{2}(\delta_{ik} + \delta_{jk} + \delta_{ij}))$
return Θ^*

Algorithm 11 ATTACH (k, e, Λ)

Calculate the lower bound β using Λ.
if $\beta > \Lambda^*$ **then return**
Update T by attaching k onto e.
if T contains all the taxa in $\{1, 2, \ldots, n\}$ **then**
    if $\Lambda < \Lambda^*$ **then** $\Theta^* = \{T\}$ **else** $\Theta^* = \Theta \cup \{T\}$
    $\Lambda^* = \Lambda$
else
    Update Δ^T (using $\boldsymbol{\delta}_k$).
    Choose the next taxon to add, $k' \notin T$.
    Calculate $\boldsymbol{\delta}_{k'}$ containing all $\delta_{X\{k'\}}$ values.
    for every branch e in T **do**
        Let A and B be defined by $A \overset{e}{-} B$.
        $\Lambda_e = \Lambda + \frac{1}{2}(\delta_{A\{k'\}} + \delta_{B\{k'\}} - \delta_{AB})$
    end for
    Choose an order $e_1, e_2, \ldots, e_m$ for the branches in T.
    for $i = 1, \ldots, m$ **do** ATTACH (k, e_i, Λ_{e_i})
    Revert all changes to Δ^T.
end else
Remove k and its terminal branch from T.

these steps depend on the chosen bound and on the chosen taxon ordering strategy, which have been extensively discussed in sections 7.3 and 7.2, respectively. Other details, such as the practical data structures used for T and Δ^T are too trivial to be discussed here.

The running time of BBBME strictly depends on the number of topologies visited, in other terms the number of times the main routine of the algorithm, ATTACH(k, e, Λ), is called. Assuming that the chosen bound can be calculated in constant time, for topologies whose bound leads to a pruning and for those over the entire set of taxa $\{1, 2, \ldots, n\}$, the visit of T is done in constant time. For all other topologies T, the calculations in the **else** block must be carried out: as discussed above, the update of Δ^T requires $O(|T| \operatorname{diam}(T))$ time; assuming for now that a static order is used, the choice of the next taxon to add requires constant time; the calculation of $\boldsymbol{\delta}_{k'}$ can be done in $O(|T|)$ time; the subsequent **for** loop consists of $O(|T|)$ cases each requiring constant time; finally, if we sort the e_i in increasing order of Λ_{e_i}, setting the branching order can be done in $O(|T| \log |T|)$ time ([**CLRS01**], Part II). Therefore, with a static order the visit of each non-pruned partial topology does not require much more than linear time in n. If dynamic ordering is used, the choice of the next taxon to add requires that we repeat the calculation of $\boldsymbol{\delta}_{k'}$ and the subsequent **for** loop once for each of the $n - |T|$ candidates for k', therefore leading to a quadratic number of operations.

The time needed to process one topology should then be multiplied by the number of topologies visited by BBBME. As we already saw there are conditions under which this number grows polynomially in n (sec. 7.3.6); however, these conditions are fairly strict and we can expect that they will rarely be met in practice. In general, the number of topologies visited can be expected to grow super-exponentially in n.

As for memory requirements, one aspect of BBBME that is only mentioned in ATTACH(k, e, Λ) is that we need to be able to revert the changes to Δ^T, so that, when we backtrack along the branches of the metatree, we always use the correct version of Δ^T. One possible approach is to have a local copy of Δ^T for each call to ATTACH(k, e, Λ). My implementation of BBBME chooses the alternative approach of storing only the $O(|T| \operatorname{diam}(T))$ changes that have been made to Δ^T and uses them for subsequent reversals. Asymptotically, the two approaches are equivalent: since at any moment during the execution of BBBME there are $O(n)$ active calls to ATTACH(k, e, Λ) and each of them uses up to $O(|T|^2)$ space in memory, BBBME uses $O(n^3)$ space.

7.5. Experiments

As will be further discussed in the next section, the fact that it may be possible to devise better bounds than the ones currently used by BBBME (sec. 7.3.7) means that it is perhaps premature to execute large-scale experiments with BBBME. Nevertheless, the current version of BBBME is already efficient enough to allow us to start investigating the question I raised in the introduction to this chapter: the margins of improvement that may be available to the current heuristics for BME. Since BBBME returns all the optimal trees with respect to BME, comparing its ability to recover the correct tree to that of BNNI (the heuristic implemented in FastME [**DG02**] and

described in section 6.7) can give us an idea of how much can be achieved by devising better and better heuristics. Although the experiments presented here represent just a preliminary investigation in this direction, their results will guide us in designing future, larger-scale experiments.

Because the size of the space explored by BBBME grows super-exponentially in the number of taxa (see fig. 7.2.1), clearly there are limits to the applicability of BBBME. However, compared to an exhaustive search among all possible trees, BBBME allows us to substantially push forwards the limit of what can be solved exactly. At present, BBBME seems to be able to deal with distance matrices with up to 20–30 taxa. (Because of the remarks in sec. 7.3.6, the applicability of BBBME is strongly dependent on how well the input distances fit on a tree.)

Quite conveniently, one of the two datasets that were originally used to test the performance of the heuristics implemented in FastME (see sec. 6.7) consists of distance matrices over 24 taxa [**DG02**], which can be easily dealt with BBBME. Other readily available datasets are too large for the current version of BBBME (96 taxa in [**DG02**], 100 taxa in [**DG04**], 1000–5000 taxa in [**VvH05**]). In this section, I present experiments on the performance of BBBME and of the heuristics for BME on this dataset (which is currently available at http://www.lirmm.fr/~guindon/simul/).

The 24-taxon dataset in [**DG02**] was generated in a way that I summarise as follows (full details in [**DG02**]): first, 2000 24-taxon trees with branch lengths violating the molecular clock (i.e., not ultrametric) were randomly generated using standard techniques producing realistic trees; then the branch lengths of these trees were multiplied by three different constants leading to three different sets of 2000 trees each, representative of "slow", "moderate" and "fast" evolutionary rates (average maximum pairwise distance of about 0.2, 0.4 and 1.0 substitutions per site, respectively). Each of these 6000 trees was then used to generate 24 random DNA sequences of 500 sites each according to a Kimura two-parameter model (K2P) [**Kim80**] with a transition/transversion ratio of 2.0. From the resulting 6000 alignments, 6000 distance matrices were then estimated using DNADIST from the PHYLIP package [**Fel89**] (assuming K2P with the known transition/transversion ratio). These distance matrices are naturally subdivided into three datasets of 2000 matrices each, which I call the *Slow*, the *Moderate* and the *Fast* datasets, differing in the evolutionary rates in the tree used to generate the matrix.

On these datasets, I tested the performance of a few BME-related methods: first of all, NJ [**SN87**] and one of its proposed improvements BIONJ [**Gas97a**]; then also a very efficient heuristic for BME [**DG02**] that I call Sadd (as it is based on a sequential addition strategy; in [**DG02**] this is called BME from the principle it tries to optimise, but I will not use this term to avoid confusion); then three versions of BNNI, which I call $\mathrm{BNNI}_{\mathrm{NJ}}$, $\mathrm{BNNI}_{\mathrm{BIONJ}}$ and $\mathrm{BNNI}_{\mathrm{Sadd}}$, differing in the method used to construct the initial tree (NJ, BIONJ and Sadd, respectively); and finally

TABLE 1. Topological accuracy and frequency of optimality of various algorithms for BME on three 24-taxon datasets.

	Slow			*Moderate*			*Fast*		
	$\overline{d_{RF}}(T,\hat{T})$		$\hat{T}=T^*$	$\overline{d_{RF}}(T,\hat{T})$		$\hat{T}=T^*$	$\overline{d_{RF}}(T,\hat{T})$		$\hat{T}=T^*$
NJ	4.650	0%	60.95%	3.605	0%	60.95%	3.461	0%	57.15%
BIONJ	4.647	-0.1%	44.55%	3.526	-2.2%	48.70%	3.333	-3.7%	43.10%
Sadd	4.975	7.0%	35.95%	4.045	12.2%	35.45%	4.112	18.8%	33.00%
BNNI_{NJ}	4.480	-3.7%	97.85%	3.468	-3.8%	98.10%	3.216	-7.1%	98.05%
$\text{BNNI}_{\text{BIONJ}}$	4.475	-3.8%	98.00%	3.459	-4.1%	97.85%	3.216	-7.1%	98.20%
$\text{BNNI}_{\text{Sadd}}$	4.499	-3.3%	97.65%	3.464	-3.9%	97.80%	3.223	-6.9%	97.40%
BBBME	4.493	-3.4%	100%	3.462	-4.0%	100%	3.229	-6.7%	100%

BBBME itself, whose performance can be viewed as the limit to which heuristics for BME tend as their ability to reconstruct BME-optimal trees improves.

Because for each distance matrix in the 24-taxon datasets described above we know the "correct" tree T that generated it, the performance of these methods can be measured by how different the tree they return is from T. Recall the definition of Robinson and Foulds (RF) distance between two trees (Def. 6.7.1). The topological accuracy of a method will be measured by the average RF distance $\overline{d_{RF}}(T,\hat{T})$ between the correct tree T and the tree $\hat{T}$ reconstructed by that method.

Table 1 shows the topological accuracies of the methods tested (corresponding to the rows of the table) across the *Slow*, the *Moderate* and the *Fast* datasets (corresponding to the three groupings of columns). The accuracy of a method X is shown both in terms of its absolute value $d_X = \overline{d_{RF}}(T,\hat{T}_X)$ and its value relative to that of NJ (i.e., $(d_X - d_{\text{NJ}})/d_{\text{NJ}}$). For each of the datasets, a third column (labelled $\hat{T}=T^*$) shows the frequency with which the method returns an optimal topology with respect to BME. Note that this frequency can be calculated thanks to BBBME, which returns the BME-optimal topology (or all of them, in the case that there are multiple optima).

Before discussing the results for BBBME, I wish to note that the results regarding the topological accuracies of NJ, BIONJ, Sadd and BNNI over these three datasets were already shown and discussed by Desper and Gascuel ([**DG02**], Table 1). (The precise figures I obtained here are very similar, but do not coincide with the original ones, possibly because of the indeterminate behaviour of the algorithms in case of ties and of recent changes in the code of FastME.) Note that BNNI outperforms NJ, BIONJ and Sadd and that its accuracy is largely independent of the method used to construct its initial tree. As discussed in section 6.7, the high relative topological accuracy of BNNI is a result that extends to most other commonly used distance methods.

As for the frequency with which the methods return an optimal topology (under $\hat{T} = T^*$), it is interesting to note that, although it is less accurate, NJ achieves optimality more often than BIONJ; this is probably because BIONJ does not choose agglomerations on the basis of balanced length. Sadd on the other hand, although directly inspired by BME, is the worst of the three methods, both in terms of its topological accuracy and of its optimality frequency; this reflects its substantially lower running times. Perhaps the most interesting result, however, is that BNNI returns an optimal topology in the overwhelming majority of the cases: about 98% of the times, across all initialisation methods and all three datasets.

Moving to the results for BBBME, it is then not very surprising that its topological accuracy largely coincides with that of BNNI: after all, in 98% of the cases BBBME reconstructs the same tree as BNNI. Note that often there are multiple optimal topologies returned by BBBME (about 11% of the cases in *Slow*, 2.1% in *Moderate*, 0.3% in *Fast*). In these cases, the distance $d_{RF}(T, \hat{T})$ is an average across all $\hat{T}$ that are optimal and therefore $\overline{d_{RF}}(T, \hat{T})$, for BBBME, is an average of averages. Table 1 shows that the topological accuracy of BBBME on the tested datasets is in general not better than that of BNNI: in *Slow*, BBBME is better than $\text{BNNI}_{\text{Sadd}}$ but not better than BNNI_{NJ} and $\text{BNNI}_{\text{BIONJ}}$; in *Moderate*, BBBME is the second best method after $\text{BNNI}_{\text{BIONJ}}$; in *Fast*, BBBME is beaten by all versions of BNNI. In all cases $\text{BNNI}_{\text{BIONJ}}$ seems to be the most accurate among the tested methods. However, the $\overline{d_{RF}}(T, \hat{T})$ have standard errors ranging between 0.055 and 0.065, which means that all the differences in performance between BBBME and the various types of BNNI are not statistically significant.

Also, whether BBBME performs better or worse than BNNI is largely determined by the relative performance of these two methods on the 2% cases where BNNI does not return an optimal tree. (There is also a very small difference in performance on the remaining 98% of the cases, which is due to the fact that when there are multiple optima, only one of these is returned by BNNI.) Table 2 shows a comparison of BBBME and $\text{BNNI}_{\text{BIONJ}}$ (seemingly the best among the tested versions of BNNI) in the 2% cases where the latter does not achieve optimality (i.e., for 40, 43 and 36 matrices in *Slow*, *Moderate* and *Fast*, respectively). The first three rows show in how many of these cases the BME optimal tree(s) returned by BBBME is (are) better, equally good or worse than the tree reconstructed by $\text{BNNI}_{\text{BIONJ}}$, as established by looking at the RF distances from the correct tree of the trees returned by BBBME and $\text{BNNI}_{\text{BIONJ}}$. The subsequent two rows show the topological accuracies of these two methods, again restricted to the 2% nonoptimal cases; note that these are much higher than those calculated over the entire dataset, which is probably due to the fact that these distance matrices represent somewhat "harder" cases. Moreover, Table 2 confirms that the difference in performance between BBBME and $\text{BNNI}_{\text{BIONJ}}$ observed in Table 1 is largely explained by the difference on these harder cases.

TABLE 2. Relative performance of BBBME and $\text{BNNI}_{\text{BIONJ}}$ on the the cases (about 2% of the total) for which BNNI does not reconstruct an optimal tree.

	Slow	*Moderate*	*Fast*
BBBME wins	7	16	11
Draw	20	8	7
BNNI wins	13	19	18
$\overline{d_{RF}}(T, T^*)$	7.6	5.5	5.7
$\overline{d_{RF}}(T, \hat{T}_{\text{BNNI}})$	7.0	5.5	5.1

Although $\text{BNNI}_{\text{BIONJ}}$ tends to beat BBBME in each of the three datasets, the difference in performance is not significant: testing whether the probabilities of a BBBME win and of a $\text{BNNI}_{\text{BIONJ}}$ win are different does not support a superiority of $\text{BNNI}_{\text{BIONJ}}$, neither when the three sets of results are taken separately ($p = 0.26, 0.74, 0.26$) nor jointly ($p = 0.10$). Similar tables to Table 2 can be produced by comparing BBBME with the other versions of BNNI, but the results in these cases are even less significant.

In conclusion, on these datasets BBBME does not show any improvement over BNNI (as measured by RF distance), which suggests that, for matrices of this size, further optimising BME may not yield any advantage. On the contrary, there appears to be some (weak) evidence that some versions of BNNI may return trees that are superior to BME-optimal trees, possibly because they complement BME with other criteria for tree reconstruction (used to choose the starting point of the BNNI search). This claim is currently vaguely hinted at by three sets of about 40 matrices each (the ones for which BNNI does not return an optimal tree). Clearly, this is a very small sample size, and future experiments should include many more matrices where BNNI does not return an optimal tree.

In order to do this, a simple approach would be to generate a much larger number of 24-taxon matrices, so that the number of "interesting" cases, even staying at about 2% of the total, will grow in proportion. Alternatively we could try to increase the proportion of such interesting cases: this could be done by using trees with higher evolutionary rates (i.e., longer branches) or with a larger number n of taxa. In the latter case, as we increase n above the current value of 24, it is likely that the rapid growth of the space of all possible topologies will cause a decrease in the frequency with which BNNI gives an optimal tree. As a consequence, the discrepancy in topological accuracy between BNNI and BBBME should become more and more pronounced. It is possible that, on larger matrices, further optimising BME has important effects and practical consequences.

Furthermore, comparing the performance of the heuristics for BME with that of BBBME on larger distance matrices also makes the experiments more realistic and

relevant in practice, as distance methods are mostly used with datasets with many taxa (because of their computational efficiency).

7.5.1. Experiments on real data? The experiments above indicate that if we want to understand whether there is any advantage to be gained from improving the current heuristics for BME, we need to push on our investigation to a larger scale. However, the quantity of data to use is not the only issue for concern; equally important is the *nature* of the distance matrices we use: all the experiments above use simulated data; in these simulations, (1) alignment problems were not accounted for, (2) the model of sequence evolution used to generate the data (K2P) was very simple and (3) the distance matrices were estimated assuming knowledge of that model. This is an ideal scenario; in reality, sequence evolution is an unknown, complex process and all proposed models used to estimate the distances are likely to be too simple — something that is often referred to as *model misspecification* — leading to imprecise (and often biased) distance estimates. As a consequence, "real" distance matrices estimated from real biological sequences can be expected to be further away from the correct distances in $\mathbf{d}^{\mathcal{T}}$ than matrices obtained by simulating sequences of similar lengths. Model misspecification may also cause distance estimates to be consistently biased: Susko, Roger and colleagues [**SIR04, WSSR08**] showed that, in many cases, failing to model certain aspects of sequence evolution results in systematic underestimation of long distances (and that there are also scenarios — probably less common — where long distances tend to be consistently overestimated).

Since all this indicates that real distance matrices are probably somewhat qualitatively different from simulated ones, it is natural to question the practical relevance of experiments on simulated data. As an alternative to simulations, distance matrices can be obtained from real biological sequences. The obvious problem with this approach is that we do not know the correct trees that generated the data and therefore measuring topological accuracies as we did above is not possible. However, there are alternative ways to evaluate the trees returned by the tested methods: for example, instead of using the RF distances from the correct tree, we can use the *likelihood* of the returned trees. In brief, given some sequence data and a model of sequence evolution, the likelihood of a tree $\mathcal{T}$ is the probability of the data, calculated assuming $\mathcal{T}$ and the given evolutionary model (see Ch. 4 in [**Yan06**] for an excellent introduction to likelihood and its applications). This criterion is a popular and effective way to evaluate how realistic a tree is, in light of the data. Using likelihood we will then be able to assess the relative goodness of the trees obtained with the different methods and to produce comparative tables similar in spirit to the top three rows in Table 2.

Furthermore, the difference between simulated and "real" distance matrices will probably also have consequences on the behaviour of the algorithms tested. My expectation is that BNNI will return optimal trees less frequently on real data than

on simulated data and therefore the differences in performance between BNNI and BBBME will be more marked. To support this claim, note that simulated distances in $\boldsymbol{\delta}$ have the nice property of converging, as the generated sequences become longer and longer, to the correct distances in $\mathbf{d}^{\mathcal{T}}$ (this is an important requirement of all methods for estimating $\boldsymbol{\delta}$). When $\boldsymbol{\delta}$ coincides with $\mathbf{d}^{\mathcal{T}}$, it has been conjectured that BNNI always returns the correct topology [**DG04, BGHM09**] as it has not been possible to find any counterexample to this conjecture. In other words, it appears that, with sufficiently long simulated sequences, the function $\Lambda(T)$ only has one local minimum with respect to NNIs. This regularity property of the shape of $\Lambda(T)$ is probably affected by the use of real data, for which there is no guarantee that the distances $\boldsymbol{\delta}$ converge towards $\mathbf{d}^{\mathcal{T}}$ for any tree $\mathcal{T}$. For real data, I expect $\Lambda(T)$ to have a less regular shape, with more numerous local minima, and therefore presenting more problems for the heuristics trying to optimise it. On such shapes for $\Lambda(T)$, the margin of improvement for BNNI may be substantially larger.

7.6. Discussion and Future Work

By summarising the input data (e.g., sequence, numerical, binary data) into a matrix of pairwise distances between taxa, distance methods inevitably face a loss of information; this usually leads to less accurate tree reconstructions than those of "character-based" methods that fully use the data in input. On the other hand, because of the simplicity of their input, heuristic distance methods are the fastest available and therefore are used whenever speed is an overwhelmingly important requirement (e.g., with a large number of taxa, or when a large number of trees must be generated). These considerations cast a doubt upon the usefulness of exact, but slow, distance methods such as the branch and bound algorithm presented in this chapter: if the time needed to run BBBME is available, why not use character-based methods for tree reconstruction?

Moreover, it is a well-known principle in operations research that, all other things being equal, it is usually better to find approximate solutions to good optimisation criteria than to find exact solutions to not-so-good criteria ([**MF00**], Ch. 1). In phylogenetics, likelihood-based criteria (such as posterior probability or likelihood itself) should be considered superior to those sacrificing information for speed (such as parsimony or distance-based criteria). Therefore, if time is not a limiting factor, distance methods are probably not the best choice for tree reconstruction.

What is then the use of BBBME? In this chapter I have demonstrated that this branch and bound algorithm can be used to investigate the limits of the criterion it optimises, BME, and to benchmark the algorithms used in practice for BME. In fact, the usefulness of an exact algorithm goes beyond its direct applicability in the practice of tree reconstruction. For example, it is possible that other, improved algorithms for BME may be devised thanks to insights provided by BBBME, which may be used to test conjectures about optimal solutions to this problem.

As an example of the potential of BBBME, consider modifying the algorithm so that prunings are performed whenever the current partial solution T is deemed to be unable to improve the current optimal solution beyond a specified ratio — say, of at least 5%. This can be done by replacing the current condition for pruning, $\beta(T, \boldsymbol{\delta}) > \Lambda^*$, with:

$$\beta(T, \boldsymbol{\delta}) > 0.95 \cdot \Lambda^*.$$

Although the resulting algorithm may fail to find an optimal solution, we can be assured that no solution will be more than 5% better than the solution returned. Clearly, since many more prunings will now occur, speed of execution will be substantially improved. For any desired maximum ratio between the returned and the optimal solution, we then have an approximation algorithm for BME that may potentially run efficiently on many realistic instances of the problem. I have not explored this interesting possibility.

There are in fact many directions for future work that clearly still need to be pursued; in particular, it is clear that the experiments in the last section need to be extended to a larger scale. However, I think that a higher priority should be given to improving the algorithm first: as discussed in section 7.3.7, it may be possible to devise tighter lower bounds than those currently used by BBBME. Better bounds have the potential to substantially increase the number of prunings and therefore reduce running times and extend the applicability of BBBME to larger distance matrices. Before proceeding to further experiments, I would like to be satisfied with the bounds used by BBBME, either by discovering better ones or by realising that large improvements are not possible — something that, because of time constraints, I have not given sufficient thought to.

Once BBBME has reached a stable point, the limits of its applicability will determine the scale of the experiments to run, in particular the size of the distance matrices. Compared to those of the last section, future experiments will involve a larger number of distance matrices, and matrices over more taxa; as anticipated in section 7.5.1, I also plan to use real, non-simulated data (or alternatively more realistic simulated data). Finally, it would be interesting to compare the running times of BBBME to those of the other implemented branch and bound algorithms for tree reconstruction (which all focus on maximum parsimony).

Conclusion

This thesis has presented a number of novel results, some of which form the basis of separate publications.

The main novel result of Chapter 2 is the correctness of the greedy algorithm to select a fixed number of species with maximum PD, which I first proved in a short paper appeared on *PLoS Genetics* [**PG05**].

Chapter 3 presents a number of dynamic programming algorithms for the budgeted version of the problem in Chapter 2: two are for the rooted definition of PD (sections 3.4 and 3.5) and one is for its unrooted version (sec. 3.6). Chronologically, the algorithms in sections 3.4 and 3.6 were the first I obtained and form the basis of a paper published in *Systematic Biology* [**PG07**]. The algorithm in section 3.5 is instead published on the *IEEE/ACM Transactions on Computational Biology and Bioinformatics* [**MPKvH09**] (a paper of which I am joint first author), together with another algorithm applicable to phylogenetic networks (see sec. 3.8).

In Chapter 4, the study of the distribution of the future PD (the PD of the surviving species after a number of random extinction events) was joint work with Faller and Steel, and led to a publication on the *Journal of Theoretical Biology* [**FPS08**]. The other main result of this chapter, the solution of the $a_s \stackrel{c_s}{\rightarrow} 1$ NAP (sec. 4.4), was also published in the *Systematic Biology* paper [**PG07**].

The main result of Chapter 5, the algorithm for the generalised Noah's Ark problem, is the central topic of a manuscript in preparation, which I hope to submit soon. In the future, I plan to experiment further with the implementation I have developed, over a wider range of instances of the problem.

The relevance of my work on the hierarchy of PD-optimisation problems of Chapters 2 – 5 very much depends on the impact that PD, as a measure of diversity in a tree, will have on real-world applications. This is already discussed at length in sections 2.2 – 2.4. In the case of comparative genomics, PD is a good predictor of statistical power in testing various evolutionary hypotheses (e.g., low substitution rate), although the correlation with power is far from being perfect — which suggests that better measures may be devised (see sec. 2.11 for a first step in that direction). In conservation biology, PD is clearly a better measure of biodiversity than simply counting species, which not only ignores degrees of relatedness, but is also very sensitive to the adopted definition of species. Although, again, better measures may exist (sec. 2.11), it seems to me that the relevance of PD to bioconservation is well-established and that is why, in Chapters 4 and 5, I chose to focus solely on

this application. PD may also have currently unsuspected applications outside genomics or bioconservation, as trees are a natural way to represent relationships (not necessarily evolutionary) in many different fields (sec. 2.4).

As for my work on BME, my results are currently unpublished. Chapter 6, which reviews of and introduces basic facts about BME, contains one novel result of some relevance: the fact that BME has a safety radius of $\frac{1}{2}$. I recently agreed to include this finding, together with related results by O. Gascuel and S. Guillemot (briefly discussed in sec. 6.8), in a manuscript I will be working on after completion of my Ph.D. [**PGG09**].

Finally, the branch and bound algorithm for BME of Chapter 7 needs some more work, as discussed in sections 7.3.7 and 7.6. My preliminary experiments with the implemented program show practically no improvement (in terms of reconstruction accuracy) over the available heuristics. Whether this will be confirmed or refuted by further experiments is still open. In both scenarios, I hope to be able to publish my findings, as they are the result of considerable work and should be of interest to many people.

Besides the results listed above, there are other findings to which I have contributed. Since these results have only been obtained in collaboration with other researchers, and since I judged that the topics would not naturally fit in the context of this thesis, I have decided not to include them in this thesis. A brief account of these results is provided in sections 1.4 and 1.5.

Appendix: species abbreviations in figures 2.8.1, 3.7.1, 4.4.1 and 5.4.1

Dma: *Daubentonia madagascariensis*, Mmy *Microcebus myoxinus*, Mru *Microcebus rufus*, Mra *Microcebus ravelobensis*, Mmu *Microcebus murinus*, Mco *Mirza coquereli*, Atr *Allocebus trichotis*, Cma *Cheirogaleus major*, Cme *Cheirogaleus medius*, Pfu el *Phaner furcifer electromontis*, Pfu pl *Phaner furcifer pallescens*, Pfu pr *Phaner furcifer parienti*, Pfu fu *Phaner furcifer furcifer*, Lmu *Lepilemur mustelinus*, Ldo *Lepilemur dorsalis*, Lse *Lepilemur septentrionalis*, Led *Lepilemur edwardsi*, Lmi *Lepilemur microdon*, Lru *Lepilemur ruficaudatus*, Lle *Lepilemur leucopus*, Pta *Propithecus tattersalli*, Pco *Propithecus coquereli*, Pve ve *Propithecus verreauxi verreauxi*, Pve de *Propithecus verreauxi deckeni*, Pve co *Propithecus verreauxi coronatus*, Pdi *Propithecus diadema*, Ppe *Propithecus perrieri*, Pca *Propithecus candidus*, Ped *Propithecus edwardsi*, Iin *Indri indri*, Ala *Avahi laniger*, Aoc *Avahi occidentalis*, Acl *Avahi cleesei*, Vva va *Varecia variegata variegata*, Vva ru *Varecia variegata rubra*, Hgr al *Hapalemur griseus alaotrensis*, Hgr oc *Hapalemur griseus occidentalis*, Hgr gr *Hapalemur griseus griseus*, Hau *Hapalemur aureus*, Hsi *Hapalemur simus*, Lca *Lemur catta*, Ema ma *Eulemur macaco macaco*, Ema fl *Eulemur macaco flavifrons*, Eco *Eulemur coronatus*, Eru *Eulemur rubriventer*, Emo *Eulemur mongoz*, Efu co *Eulemur fulvus collaris*, Efu ac *Eulemur fulvus albocollaris*, Efu ru *Eulemur fulvus rufus*, Efu fu *Eulemur fulvus fulvus*, Efu af *Eulemur fulvus albifrons*, Efu sa *Eulemur fulvus sanfordi*.

Bibliography

[ABCH+04] L. Addario-Berry, B. Chor, M. Hallett, J. Lagergren, A. Panconesi, and T. Wareham. Ancestral Maximum Likelihood of Evolutionary Trees is Hard. *Journal of Bioinformatics and Computational Biology*, 2:257–272, 2004.

[ACH+04] J.L. Arthur, J.D. Camm, R.G. Haight, C.A. Montgomery, and S. Polasky. Weighing conservation objectives: maximum expected coverage versus endabgered species protection. *Ecological Applications*, 14:1936–1945, 2004.

[ACL89] S.F. Altschul, R.J. Carroll, and D.J. Lipman. Weights for data related by a tree. *Journal of Molecular Biology*, 207:647–53, 1989.

[ACPR09] N. Alon, B. Chor, F. Pardi, and A. Rapoport. Approximate maximum parsimony and ancestral maximum likelihood. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2009. to appear.

[ACPS98] A. Ando, J. Camm, S. Polasky, and A. Solow. Species distributions, land values, and efficient conservation. *Science*, 279:2126–2128, 1998.

[AFR+06] N. Andriaholinirina, J.L. Fausser, C. Roos, D. Zinner, U. Thalmann, C. Rabarivola, I. Ravoarimanana, J.U. Ganzhorn, B. Meier, R. Hilgartner, L. Walter, A. Zaramody, C. Langer, T. Hahn, E. Zimmermann, U. Radespiel, M. Craul, J. Tomiuk, I. Tattersall, and Y. Rumpler. Molecular phylogeny and taxonomic revision of the sportive lemurs (Lepilemur, Primates). *BMC Evolutionary Biology*, 6:17, 2006.

[AL90] S.F. Altschul and D.J. Lipman. Equal animals. *Nature*, 348:493–494, 1990.

[AN06] E. Althaus and R. Naujoks. Computing steiner minimum trees in Hamming metric. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 172–181. ACM Press, 2006.

[Att99] K. Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25:251–278, 1999.

[Bar02] G.M. Barker. Phylogenetic diversity: a quantitative framework for measurement of priority and achievement in biodiversity conservation. *Biological Journal of the Linnean Society*, 76:165–194, 2002.

[BBC07] BBC. Protection for 'weirdest' species. BBC Online News, http://news.bbc.co.uk/1/hi/sci/tech/6263331.stm, 2007. Accessed in January 2008.

[BGHM09] M. Bordewich, O. Gascuel, K. Huber, and V. Moulton. Consistency of topological moves based on the balanced minimum evolution principle of phylogenetic inference. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6:110–117, 2009.

[BGJ03] A. Balmford, R.E. Green, and M. Jenkins. Measuring the changing state of nature. *Trends in Ecology & Evolution*, 18:326–330, 2003.

[BM02] S.R. Beissinger and D.R. McCullough. *Population Viability Analysis*. University Of Chicago Press, 2002.

[BMdF+06] T.M. Brooks, R.A. Mittermeier, G.A.B. da Fonseca, J. Gerlach, M. Hoffmann, J.F. Lamoreux, C.G. Mittermeier, J.D. Pilgrim, and A.S.L. Rodrigues. Global biodiversity conservation priorities. *Science*, 313:58–61, 2006.

[BMO+03] D. Boffelli, J. McAuliffe, D. Ovcharenko, K.D. Lewis, I. Ovcharenko, L. Pachter, and E.M. Rubin. Phylogenetic shadowing of primate sequences to find functional regions of the human genome. *Science*, 299:1391–1394, 2003.

[BPM+04] G. Bejerano, M. Pheasant, I. Makunin, S. Stephen, W.J. Kent, J.S. Mattick, and D. Haussler. Ultraconserved elements in the human genome. *Science*, 304:1321–1325, 2004.

[Bry05] D. Bryant. On the uniqueness of the selection criterion in neighbor-joining. *Journal of Classification*, 22:3–15, 2005.

[BS08] M. Bordewich and C. Semple. Nature reserve selection problem: a tight approximation algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5:275–280, 2008.

[BSD+07] E. Birney, J.A. Stamatoyannopoulos, A. Dutta, R. Guigó, T.R. Gingeras, E.H. Margulies, Z. Weng, M. Snyder, E.T. Dermitzakis, J.A. Stamatoyannopoulos, et al. Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. *Nature*, 447:799–816, 2007.

[BSH00] W.J. Bruno, N.D. Socci, and A.L. Halpern. Weighted neighbor joining: a likelihood-based approach to distance-based phylogeny reconstruction. *Molecular Biology and Evolution*, 17:189–197, 2000.

[BSKH05] G. Bejerano, A.C. Siepel, W.J. Kent, and D. Haussler. Computational screening of conserved genomic DNA in search of functional noncoding elements. *Nature Methods*, 2:535–545, 2005.

[BSS07] M. Bordewich, C. Semple, and A. Spillner. Optimizing phylogenetic diversity across two trees. Technical Report NI07068-PLG, Isaac Newton Institute, Cambridge, UK, 2007.

[Bul91] M. Bulmer. Use of the method of generalized least squares in reconstructing phylogenies from sequence data. *Molecular Biology and Evolution*, 8:868, 1991.

[BW98] D. Bryant and P. Waddell. Rapid evaluation of least-squares and minimum-evolution criteria on phylogenetic trees. *Molecular Biology and Evolution*, 15:1346–1359, 1998.

[CAD06] R.H. Crozier, P. Agapow, and L.J. Dunnett. Conceptual issues in phylogeny and conservation: a reply to Faith and Baker. *Evolutionary Bioinformatics Online*, 2:197–199, 2006.

[CB89] K.D. Cocks and I.A. Baird. Using mathematical programming to address the multiple reserve selection problem: an example from the Eyre Peninsula, South Australia. *Biological Conservation*, 49:113–130, 1989.

[CBP+03] G.M. Cooper, M. Brudno, NISC Comparative Sequencing Program, E.D. Green, S. Batzoglou, and A. Sidow. Quantitative estimates of sequence divergence for comparative analyses of mammalian genomes. *Genome Research*, 13:813, 2003.

[CLP80] J.L. Chandon, J. Lemaire, and J. Pouget. Construction de l'ultramétrique la plus proche d'une dissimilarité au sens des moindres carrés. *Recherche Operationnelle*, 14:157–170, 1980.

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[CM01] M. Cabeza and A. Moilanen. Design of reserve networks and the persistence of biodiversity. *Trends in Ecology & Evolution*, 16:242–248, 2001.

[Con07] Drosophila 12 Genomes Consortium. Evolution of genes and genomes on the Drosophila phylogeny. *Nature*, 450:203–218, 2007.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM Press, 1971.

[CPSC96] J.D. Camm, S. Polasky, A. Solow, and B. Csuti. A note on optimal algorithms for reserve site selection. *Biological Conservation*, 78:353–355, 1996.

[CPW+97] B. Csuti, S. Polasky, P.H. Williams, R.L. Pressey, J.D. Camm, M. Kershaw, A.R. Kiester, B. Downs, R. Hamilton, M. Huso, et al. A comparison of reserve selection algorithms using data on terrestrial vertebrates in Oregon. *Biological Conservation*, 80:83–97, 1997.

[Cro92] R.H. Crozier. Genetic diversity and the agony of choice. *Biological Conservation*, 61:11–15, 1992.

[Cro97] R.H. Crozier. Preserving the information content of species: genetic diversity, phylogeny, and conservation worth. *Annual Review of Ecology and Systematics*, 28:243–268, 1997.

[CS65] J.H. Camin and R.R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, 19:311–326, 1965.

[CSD96] R.L. Church, D.M. Stoms, and F.W. Davis. Reserve selection as a maximal covering location problem. *Biological Conservation*, 76:105–112, 1996.

[CSE67] L.L. Cavalli-Sforza and A.W.F. Edwards. Phylogenetic analysis: models and estimation procedures. *Evolution*, 21:550–570, 1967.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[DG02] R. Desper and O. Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of Computational Biology*, 9:687–705, 2002.

[DG04] R. Desper and O. Gascuel. Theoretical foundation of the balanced minimum evolution method of phylogenetic inference and its relationship to weighted least-squares tree fitting. *Molecular Biology and Evolution*, 21:587–598, 2004.

[DG05] R. Desper and O. Gascuel. The minimum evolution distance-based approach to phylogenetic inference. In O. Gascuel, editor, *Mathematics of Evolution & Phylogeny*, pages 1–32. Oxford University Press, 2005.

[Dia75] J.M. Diamond. The island dilemma: lessons of modern biogeographic studies for the design of natural reserves. *Biological Conservation*, 7:129–146, 1975.

[DRRW97] A.P. Dobson, J.P. Rodriguez, W.M. Roberts, and D.S. Wilcove. Geographic distribution of endangered species in the United States. *Science*, 275:550–553, 1997.

[ECS63] A.W.F. Edwards and L.L. Cavalli-Sforza. The reconstruction of evolution. *Heredity*, 18:104–105, 1963.

[ECS64] A.W.F. Edwards and L.L. Cavalli-Sforza. Reconstruction of evolutionary trees. In V.H. Heywood and J. McNeill, editors, *Phenetic and Phylogenetic Classification*, pages 67–76. Systematics Association Publ. No. 6, London, 1964.

[ED66] R.V. Eck and M.O. Dayhoff. *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, Silver Spring, Maryland, 1966.

[Edd05] S.R. Eddy. A model of the statistical power of comparative genome sequence analysis. *PLoS Biology*, 3:e10, 2005.

[EHPY08] K. Eickmeyer, P. Huggins, L. Pachter, and R. Yoshida. On the optimality of the neighbor-joining algorithm. *Algorithms for Molecular Biology*, 3:5, 2008.

[ELL01] B.S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Arnold, 4th edition, 2001.

[ESBB98] M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences USA*, 95:14863–14868, 1998.

[Fai92] D.P. Faith. Conservation evaluation and phylogenetic diversity. *Biological Conservation*, 61:1–10, 1992.

[FB06] D.P. Faith and A.M. Baker. Phylogenetic diversity (PD) and biodiversity conservation: some bioinformatics challenges. *Evolutionary Bioinformatics Online*, 2:70–77, 2006.

[Fel78] J. Felsenstein. The number of evolutionary trees. *Systematic Zoology*, 27:27–33, 1978.

[Fel89] J. Felsenstein. PHYLIP — phylogeny inference package (version 3.2). *Cladistics*, 5:164–166, 1989.

[Fel03] J. Felsenstein. *Inferring phylogenies*. Sinauer, 2003.

[FG82] L.R. Foulds and R.L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[FGG+04] E.A. Feingold, P.J. Good, M.S. Guyer, S. Kamholz, L. Liefer, K. Wetterstrand, F.S. Collins, T.R. Gingeras, D. Kampa, E.A. Sekinger, et al. The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science*, 306:636–640, 2004.

[FGR+07] F. Forest, R. Grenyer, M. Rouget, T.J. Davies, R.M. Cowling, D.P. Faith, A. Balmford, J.C. Manning, S. Proches, M. van der Bank, et al. Preserving the evolutionary potential of floras in biodiversity hotspots. *Nature*, 445:757–60, 2007.

[Fit71] W.M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971.

[FM67] W.M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279–284, 1967.

[FPS08] B. Faller, F. Pardi, and M. Steel. Distribution of phylogenetic diversity under random extinction. *Journal of Theoretical Biology*, 251:286–296, 2008.

[Gas94] O. Gascuel. A note on Sattath and Tversky's, Saitou and Nei's, and Studier and Keppler's algorithms for inferring phylogenies from evolutionary distances. *Molecular Biology and Evolution*, 11:961–3, 1994.

[Gas97a] O. Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution*, 14:685–695, 1997.

[Gas97b] O. Gascuel. Concerning the NJ algorithm and its unweighted version, UNJ. In F.S. Roberts A. Rzhetsky B. Mirkin, F.R. McMorris, editor, *Mathematical Hierarchies and Biology*, pages 149–170. American Mathematical Society, 1997.

[Gas00a] O. Gascuel. Data model and classification by trees: the minimum variance reduction (MVR) method. *Journal of Classification*, 17:67–99, 2000.

[Gas00b] O. Gascuel. On the optimization principle in phylogenetic analysis and the minimum-evolution criterion. *Molecular Biology and Evolution*, 17:401–405, 2000.

[GBD01] O. Gascuel, D. Bryant, and F. Denis. Strengths and limitations of the minimum evolution principle. *Systematic Biology*, 50:621–627, 2001.

[Geu07] K. Geuten. Experimental design criteria in phylogenetics: where to add taxa. *Systematic Biology*, 56:609–622, 2007.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, NY, 1979.

[Gol98] N. Goldman. Phylogenetic information and experimental design in molecular systematics. *Proceedings of the Royal Society B: Biological Sciences*, 265:1779–1786, 1998.

[Gra72] R.L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[GS06] O. Gascuel and M. Steel. Neighbor-joining revealed. *Molecular Biology and Evolution*, 23:1997–2000, 2006.

[GWM+04] R.A. Gibbs, G.M. Weinstock, M.L. Metzker, D.M. Muzny, E.J. Sodergren, S. Scherer, G. Scott, D. Steffen, K.C. Worley, P.E. Burch, et al. Genome sequence of the Brown Norway rat yields insights into mammalian evolution. *Nature*, 428:493–521, 2004.

[GYZ02] G. Gutin, A. Yeo, and A. Zverovich. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics*, 117:81–86, 2002.

[Har71] E.F. Harding. The probabilities of rooted tree-shapes generated by random bifurcation. *Advances in Applied Probability*, 3:44–77, 1971.

[HB06] D.H. Huson and D. Bryant. Application of phylogenetic networks in evolutionary studies. *Molecular Biology and Evolution*, 23:254–267, 2006.

[HCMZ08] G. Hickey, P. Carmi, A. Maheshwari, and N. Zeh. NAPX: a polynomial time approximation scheme for the Noah's Ark problem. In *Lecture Notes in Computer Science. Algorithms in Bioinformatics.*, volume 5251, pages 76–86. Springer, 2008.

[HH03] M.D. Hendy and B.R. Holland. Upper bounds on maximum likelihood for phylogenetic trees. *Bioinformatics*, 19:66–72, 2003.

[HKS08] C.J. Haake, A. Kashiwada, and F.E. Su. The Shapley value of phylogenetic trees. *Journal of Mathematical Biology*, 56:479–497, 2008.

[Hol01] B. Holland. *Evolutionary analyses of large data sets: trees and beyond.* PhD thesis, 2001.

[HP82] M.D. Hendy and D. Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, 59:277–290, 1982.

[HRW92] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem.* North Holland, 1992.

[HS06] K. Hartmann and M. Steel. Maximizing phylogenetic diversity in biodiversity conservation: greedy solutions to the Noah's Ark problem. *Systematic Biology*, 55:644–651, 2006.

[HS07] K. Hartmann and M. Steel. Phylogenetic diversity: from combinatorics to ecology. In O. Gascuel and M. Steel, editors, *Reconstructing Evolution: New Mathematical and Computational Advances.* Oxford University Press, 2007.

[HU80] A.J. Higgs and M.B. Usher. Should nature reserves be large or small? *Nature*, 285:568–569, 1980.

[HVD+98] P.C. Howard, P. Viskanic, T.R.B. Davenport, F.W. Kigenyi, M. Baltzer, C.J. Dickinson, J.S. Lwanga, R.A. Matthews, and A. Balmford. Complementarity and the use of indicator groups for reserve selection in Uganda. *Nature*, 394:472–475, 1998.

[IMM04] N.J.B. Isaac, J. Mallet, and G.M. Mace. Taxonomic inflation: its influence on macroecology and conservation. *Trends in Ecology & Evolution*, 19:464–469, 2004.

[ITC+07] N.J.B. Isaac, S.T. Turvey, B. Collen, C. Waterman, and J.E.M. Baillie. Mammals on the EDGE: conservation priorities based on threat and phylogeny. *PLoS ONE*, 2:e296, 2007.

[IUC06] IUCN. 2006 IUCN Red List of Threatened Species. http://www.iucnredlist.org, 2006. Accessed on 30 December 2006.

[JC69] T.H. Jukes and C.R. Cantor. Evolution of protein molecules. In H.N. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, 1969.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum press, 1972.

[KF94] M.K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11:459–468, 1994.

[Kim80] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16:111–120, 1980.

[KLS91] B.H. Korte, L. Lovász, and R. Schrader. *Greedoids (Algorithms and Combinatorics)*. Springer, 1991.

[KMN99] S. Khuller, A. Moss, and J. Naor. The Budgeted maximum coverage problem. *Information Processing Letters*, 70:39–45, 1999.

[KPE+03] M. Kellis, N. Patterson, M. Endrizzi, B. Birren, and E.S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423:241–254, 2003.

[KSZ71] K.K. Kidd and L.A. Sgaramella-Zonta. Phylogenetic analysis: concepts and methods. *American Journal of Human Genetics*, 23:235–252, 1971.

[LTWM+05] K. Lindblad-Toh, C.M. Wade, T.S. Mikkelsen, E.K. Karlsson, D.B. Jaffe, M. Kamal, M. Clamp, J.L. Chang, E.J. Kulbokas III, M.C. Zody, et al. Genome sequence, comparative analysis and haplotype structure of the domestic dog. *Nature*, 438:803–819, 2005.

[Mar07] E. Marris. Conservation priorities: what to let go. *Nature*, 450:152–155, 2007.

[May90] R.M. May. Taxonomy as destiny. *Nature*, 347:129–130, 1990.

[MBP+03] E.H. Margulies, M. Blanchette, NISC Comparative Sequencing Program, D. Haussler, and E.D. Green. Identification and characterization of multi-species conserved sequences. *Genome Research*, 13:2507–2518, 2003.

[MCA+07] E.H. Margulies, G.M. Cooper, G. Asimenos, D.J. Thomas, C.N. Dewey, A. Siepel, E. Birney, D. Keefe, A.S. Schwartz, M. Hou, et al. Analyses of deep mammalian sequence alignments and constraint predictions for 1% of the human genome. *Genome Research*, 17:760–774, 2007.

[MF98] C. Moritz and D.P. Faith. Comparative phylogeography and the identification of genetically divergent areas for conservation. *Molecular Ecology*, 7:419–429, 1998.

[MF00] Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2000.

[MG00] T. Massingham and N. Goldman. EDIBLE: experimental design and information calculations in phylogenetics. *Bioinformatics*, 16:294–295, 2000.

[MG05] T. Massingham and N. Goldman. Detecting Amino Acid Sites Under Positive Selection and Purifying Selection. *Genetics*, 169:1753–1762, 2005.

[MGP03] G.M. Mace, J.L. Gittleman, and A. Purvis. Preserving the tree of life. *Science*, 300:1707–1709, 2003.

[MHE+05] T.S. Mikkelsen, L.W. Hillier, E.E. Eichler, M.C. Zody, D.B. Jaffe, S.P. Yang, W. Enard, I. Hellmann, K. Lindblad-Toh, T.K. Altheide, et al. Initial sequence of the chimpanzee genome and comparison with the human genome. *Nature*, 437:69–87, 2005.

[MJP05] J.D. McAuliffe, M.I. Jordan, and L. Pachter. Subtree power analysis and species selection for comparative genomics. *Proceedings of the National Academy of Sciences USA*, 102:7900–7905, 2005.

[MKvH06] B.Q. Minh, S. Klaere, and A. von Haeseler. Phylogenetic diversity within seconds. *Systematic Biology*, 55:769–773, 2006.

[MKvH08] B.Q. Minh, S. Klaere, and A. von Haeseler. Phylogenetic diversity on split networks. Technical Report NI07090-PLG, Isaac Newton Institute, Cambridge, UK, 2008.

[MLP09] R. Mihaescu, D. Levy, and L. Pachter. Why neighbor-joining works. *Algorithmica*, 54:1–24, 2009.

[MNP88] C.R. Margules, A.O. Nicholls, and R.L. Pressey. Selecting networks of reserves to maximise biological diversity. *Biological Conservation*, 43:63–76, 1988.

[MP00] C.R. Margules and R.L. Pressey. Systematic conservation planning. *Nature*, 405:243–253, 2000.

[MP08] R. Mihaescu and L. Pachter. Combinatorics of least squares trees. *Proceedings of the National Academy of Sciences USA*, 105:13206–13211, 2008.

[MPKvH09] B.Q. Minh, F. Pardi, S. Klaere, and A. von Haeseler. Budgeted phylogenetic diversity maximisation on networks. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 6:22–29, 2009.

[MS57] C.D. Michener and R.R. Sokal. A quantitative approach to a problem in classification. *Evolution*, 11:130–162, 1957.

[MSS07] V. Moulton, C. Semple, and M. Steel. Optimizing phylogenetic diversity under constraints. *Journal of Theoretical Biology*, 246:186–194, 2007.

[MVM+05] E.H. Margulies, J.P. Vinson, W. Miller, D.B. Jaffe, K. Lindblad-Toh, J.L. Chang, E.D. Green, E.S. Lander, J.C. Mullikin, and M. Clamp. An initial strategy for the systematic identification of functional elements in the human genome by low-redundancy comparative sequencing. *Proceedings of the National Academy of Sciences USA*, 102:4795–4800, 2005.

[New07] L.A. Newberg. Effective species count and motif efficiency: the value of comparative genomics in characterizing conserved sequence positions. Technical Report 07-09, Rensselaer Polytechnic Institute, Department of Computer Science, Troy, NY, 2007.

[NJ89] M. Nei and L. Jin. Variances of the average numbers of nucleotide substitutions within and between populations. *Molecular Biology and Evolution*, 6:290–300, 1989.

[NL04] L.A. Newberg and C.E. Lawrence. Mammalian genomes ease location of human DNA functional segments but not their description. *Statistical Applications in Genetics and Molecular Biology*, 3:Article 23, 2004.

[NM97] S. Nee and R.M. May. Extinction and the loss of evolutionary history. *Science*, 278:692–694, 1997.

[NP02] K. Nehring and C. Puppe. A theory of diversity. *Econometrica*, 70:1155–1198, 2002.

[NP03] K. Nehring and C. Puppe. Diversity and dissimilarity in lines and hierarchies. *Mathematical Social Sciences*, 45:167–183, 2003.

[NP04] K. Nehring and C. Puppe. Modelling phylogenetic diversity. *Resource and Energy Economics*, 26:205–235, 2004.

[NSS85] M. Nei, J.C. Stephens, and N. Saitou. Methods for computing the standard errors of branching points in an evolutionary tree and their application to molecular data from humans and apes. *Molecular Biology and Evolution*, 2:66–85, 1985.

[NW70] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the ammo acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[OB03] H. Önal and R.A. Briers. Selection of a minimum-boundary reserve network using integer programming. *Proceedings of the Royal Society B: Biological Sciences*, 270:1487–1491, 2003.

[Öna04] H. Önal. First-best, second-best, and heuristic solutions in conservation reserve site selection. *Biological Conservation*, 115:55–62, 2004.

[Orw45] G. Orwell. *Animal Farm*. Penguin, 1945.

[Pau00] Y. Pauplin. Direct calculation of a tree length using a distance matrix. *Journal of Molecular Evolution*, 51:41–47, 2000.

[PBTK00] P.W. Purdom, Jr, P.G. Bradford, K. Tamura, and S. Kumar. Single column discrepancy and dynamic max-mini optimizations for quickly finding the most parsimonious evolutionary trees. *Bioinformatics*, 16:140–151, 2000.

[PCS+00] S. Polasky, J.D. Camm, A.R. Solow, B. Csuti, D. White, and R. Ding. Choosing reserve networks with incomplete species information. *Biological Conservation*, 94:1–10, 2000.

[PCVM01] S. Polasky, B. Csuti, C.A. Vossler, and S.M. Meyers. A comparison of taxonomic distinctness versus richness as criteria for setting conservation priorities for North American birds. *Biological Conservation*, 97:99–105, 2001.

[PFM02] J. Pastorini, M.R.J. Forstner, and R.D. Martin. Phylogenetic relationships among Lemuridae (Primates): evidence from mtDNA. *Journal of Human Evolution*, 43:463–478, 2002.

[PG02a] O.L. Petchey and K.J. Gaston. Extinction and the loss of functional diversity. *Proceedings of the Royal Society B: Biological Sciences*, 269:1721–1727, 2002.

[PG02b] O.L. Petchey and K.J. Gaston. Functional diversity (FD), species richness and community composition. *Ecology Letters*, 5:402–411, 2002.

[PG05] F. Pardi and N. Goldman. Species choice for comparative genomics: being greedy works. *PLoS Genetics*, 1:e71, 2005.

[PG07] F. Pardi and N. Goldman. Resource-aware taxon selection for maximizing phylogenetic diversity. *Systematic Biology*, 56:431–444, 2007.

[PGG09] F. Pardi, S. Guillemot, and O. Gascuel. Robustness of phylogenetic reconstruction based on minimum evolution. 2009. Manuscript in preparation.

[PH87] D. Penny and M.D. Hendy. Turbo Tree: a fast algorithm for minimal trees. *Bioinformatics*, 3:183–187, 1987.

[PHM+93] R.L. Pressey, C.J. Humphries, C.R. Margules, R.I. Vane-Wright, and P.H. Williams. Beyond opportunism: key principles for systematic reserve selection. *Trends in Ecology & Evolution*, 8:124–128, 1993.

[PME+01] J. Pastorini, R.D. Martin, P. Ehresmann, E. Zimmermann, and M.R.J. Forstner. Molecular phylogeny of the lemur family Cheirogaleidae (Primates) based on mitochondrial DNA sequences. *Molecular Phylogenetics and Evolution*, 19:45–56, 2001.

[POD05] S. Pavoine, S. Ollier, and A.B. Dufour. Is the originality of a species measurable? *Ecology Letters*, 8:579–586, 2005.

[PS98] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[RAB+04] A.S.L. Rodrigues, S.J. Andelman, M.I. Bakarr, L. Boitani, T.M. Brooks, R.M. Cowling, L.D.C. Fishpool, G.A.B. da Fonseca, K.J. Gaston, M. Hoffmann, et al. Effectiveness of the global protected area network in representing species diversity. *Nature*, 428:640–643, 2004.

[Rau92] D.M. Raup. *Extinction: Bad Genes Or Bad Luck?* W.W. Norton & Company, 1992.

[RF81] D.F. Robinson and L.R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.

[RG02] A.S.L. Rodrigues and K.J. Gaston. Maximising phylogenetic diversity in the selection of networks of conservation areas. *Biological Conservation*, 105:103–111, 2002.

[RM06] D.W. Redding and A.Ø. Mooers. Incorporating evolutionary measures into conservation prioritization. *Conservation Biology*, 20:1670–1678, 2006.

[RMSG+03] S.B. Reist-Marti, H. Simianer, J. Gibson, O. Hanotte, and J.E.O. Rege. Weitzman's approach and conservation of breed diversity: an application to African cattle breeds. *Conservation Biology*, 17:1299–1311, 2003.

[RN92] A. Rzhetsky and M. Nei. A simple method for estimating and testing minimum-evolution trees. *Molecular Biology and Evolution*, 9:945, 1992.

[RN93] A. Rzhetsky and M. Nei. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10:1073–1095, 1993.

[RSZ04] C. Roos, J. Schmitz, and H. Zischler. Primate jumping genes elucidate strepsirrhine phylogeny. *Proceedings of the National Academy of Sciences USA*, 101:10650–10654, 2004.

[RZ06] G. Robins and A. Zelikovsky. Tighter Bounds for Graph Steiner Tree Approximation. *SIAM Journal on Discrete Mathematics*, 19:122–134, 2006.

[SBB+03] L.D. Stein, Z. Bao, D. Blasiar, T. Blumenthal, M.R. Brent, et al. The genome sequence of Caenorhabditis briggsae: a platform for comparative genomics. *PLoS Biology*, 1:e45, 2003.

[Sch70] E. Schröder. Vier combinatorische Probleme. *Zeitschrift für Mathematik und Physik*, 15:361–376, 1870.

[SI89] N. Saitou and T. Imanishi. Relative efficiencies of the Fitch-Margoliash, maximum-parsimony, maximum-likelihood, minimum-evolution, and neighbor-joining methods of phylogenetic tree construction in obtaining the correct tree. *Molecular Biology and Evolution*, 6:514–525, 1989.

[Sid02] A. Sidow. Sequence first. Ask questions later. *Cell*, 111:13–16, 2002.

[SIR04] E. Susko, Y. Inagaki, and A.J. Roger. On inconsistency of the neighbor-joining, least squares, and minimum evolution estimation when substitution processes are incorrectly modeled. *Molecular Biology and Evolution*, 21:1629–1642, 2004.

[SK88] J.A. Studier and K.J. Keppler. A note on the neighbor-joining algorithm of Saitou and Nei. *Molecular Biology and Evolution*, 5:729, 1988.

[SLK+07] A. Stark, M.F. Lin, P. Kheradpour, J.S. Pedersen, L. Parts, J.W. Carlson, M.A. Crosby, M.D. Rasmussen, S. Roy, A.N. Deoras, et al. Discovery of functional elements in 12 Drosophila genomes using evolutionary signatures. *Nature*, 450:219–232, 2007.

[SM58] R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bullettin*, 38:1409–1438, 1958.

[SMG+03] H. Simianer, S.B. Marti, J. Gibson, O. Hanotte, and J.E.O. Rege. An approach to the optimal allocation of conservation funds to minimize loss of genetic diversity between livestock breeds. *Ecological Economics*, 45:377–392, 2003.

[SMM07] M. Steel, A. Mimoto, and A.Ø. Mooers. Hedging our bets: the expected contribution of species to future phylogenetic diversity. *Evolutionary Bioinformatics*, 3:237–244, 2007.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[Sne57] P.H. Sneath. The application of computers to taxonomy. *Journal of General Microbiology*, 17:201–26, 1957.

[SNM08] A. Spillner, B.T. Nguyen, and V. Moulton. Computing phylogenetic diversity for split systems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5:235–244, 2008.

[SOWH96] D.L. Swofford, G.J. Olsen, P.J. Waddell, and D.M. Hillis. Phylogenetic inference. In D. Hillis, C. Moritz, and B. Mable, editors, *Molecular Systematics*, pages 407–514. Sinauer, 1996.

[SS63] R.R. Sokal and P.H.A. Sneath. *Principles of Numerical Taxonomy*. W.H. Freeman, 1963.

[SS73] P.H.A. Sneath and R.R. Sokal. *Numerical taxonomy: the principles and practice of numerical classification*. W.H. Freeman, 1973.

[SS94] M. Shaked and J.G. Shanthikumar. *Stochastic Orders and their Applications*. Academic Press, 1994.

[SS03] C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.

[SS04] C. Semple and M. Steel. Cyclic permutations and evolutionary trees. *Advances in Applied Mathematics*, 32:669–680, 2004.

[ST77] S. Sattath and A. Tversky. Additive similarity trees. *Psychometrika*, 42:319–345, 1977.

[Ste05] M. Steel. Phylogenetic diversity and the greedy algorithm. *Systematic Biology*, 54:527–529, 2005.

[Swo98] D.L. Swofford. PAUP* — phylogenetic analysis using parsimony (* and other methods). Version 4.0. Sinauer, 1998.

[TTB+03] J.W. Thomas, J.W. Touchman, R.W. Blakesley, G.G. Bouffard, S.M. Beckstrom-Sternberg, E.H. Margulies, M. Blanchette, A.C. Siepel, P.J. Thomas, J.C. McDowell, et al. Comparative analyses of multi-species sequences from targeted genomic regions. *Nature*, 424:788–793, 2003.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[TWF+08] M.L. Tress, J.J. Wesselink, A. Frankish, G. Lopez, N. Goldman, A. Loytynoja, T. Massingham, F. Pardi, S. Whelan, J. Harrow, and A. Valencia. Determination and validation of principal gene products. *Bioinformatics*, 24:11–17, 2008.

[Und94] L.G. Underhill. Optimal and suboptimal reserve selection algorithms. *Biological Conservation*, 70:85–87, 1994.

[Vac89] W. Vach. Least squares approximation of additive trees. In O. Opitz, editor, *Conceptual and numerical analysis of data*, pages 230–238. Springer-Verlag, Berlin, 1989.

[vdHvdBvI05] C.M. van der Heide, J.C.J.M. van den Bergh, and E.C. van Ierland. Extending Weitzman's economic ranking of biodiversity protection: combining ecological and genetic considerations. *Ecological Economics*, 55:218–223, 2005.

[VG98] D.P. Vazquez and J.L. Gittleman. Biodiversity conservation: does phylogeny matter. *Current Biology*, 8:R379–81, 1998.

[Vol72] Voltaire. *La Bégueule, conte moral*. 1772.

[VvH05] S. Vinh and A. von Haeseler. Shortest triplet clustering: reconstructing large phylogenies using representative sets. *BMC Bioinformatics*, 6:92, 2005.

[VWHW91] R.I. Vane-Wright, C.J. Humphries, and P.H. Williams. What to protect?—Systematics and the agony of choice. *Biological Conservation*, 55:235–254, 1991.

[Wei92] M.L. Weitzman. On diversity. *The Quarterly Journal of Economics*, 107:363–405, 1992.

[Wei98] M.L. Weitzman. The Noah's Ark problem. *Econometrica*, 66:1279–1298, 1998.

[Wik08] Wikipedia. Pareto efficiency. http://en.wikipedia.org/wiki/Pareto-efficiency, 2008. Accessed in December 2008.

[WL95] L. Witting and V. Loeschcke. The optimization of biodiversity conservation. *Biological Conservation*, 71:205–207, 1995.

[WLTB+02] R.H. Waterston, K. Lindblad-Toh, E. Birney, J. Rogers, J.F. Abril, P. Agarwal, R. Agarwala, R. Ainscough, M. Alexandersson, P. An, et al. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420:520–562, 2002.

[WSSR08] H.C. Wang, E. Susko, M. Spencer, and A.J. Roger. Topological estimation biases with covarion evolution. *Journal of Molecular Evolution*, 66:50–60, 2008.

[WTL00] L. Witting, J. Tomiuk, and V. Loeschcke. Modelling the optimal conservation of interacting species. *Ecological Modelling*, 125:123–143, 2000.

[Yan06] Z. Yang. *Computational Molecular Evolution*. Oxford University Press, 2006.

[Yod97] A.D. Yoder. Back to the future: a synthesis of strepsirrhine systematics. *Evolutionary Anthropology Issues News and Reviews*, 6:11–22, 1997.

[YRG+00] A.D. Yoder, R.M. Rasoloarison, S.M. Goodman, J.A. Irwin, S. Atsalis, M.J. Ravosa, and J.U. Ganzhorn. Remarkable species diversity in Malagasy mouse lemurs (primates, Microcebus). *Proceedings of the National Academy of Sciences USA*, 97:11325–11330, 2000.

[Yul24] G.U. Yule. A mathematical theory of evolution, based on the conclusions of Dr. J.C. Willis, FRS. *Philosophical Transactions of the Royal Society of London. Series B*, 213:21–87, 1924.

[YWN05] Z. Yang, W.S.W. Wong, and R. Nielsen. Bayes Empirical Bayes Inference of Amino Acid Sites Under Positive Selection. *Molecular Biology and Evolution*, 22:1107–1118, 2005.

Index