UCLouvain

epl

École polytechnique de Louvain

Reconstructing phylogenies: BMEP model implementation and research on the definition of the optimization set

Author: **Arthur DE WALQUE**
Supervisor: **Jean-Charles DELVENNE**
Readers: **Daniele CATANZARO, Geovani GRAPIGLIA**
Academic year 2022–2023
Master [120] in Mathematical Engineering

Contents

1 Summary

The subject of our work is the BME (Balanced Minimum Evolution), an estimation criterion for the reconstruction of phylogenetic trees, part of the distance-based methods. It was introduced in 2002 by Gascuel, based on the work of Pauplin (2000). The BMEP (the optimization of the BME) is NP-hard, and its optimal solution has been proven to be statistically consistent, which makes it a good model and justifies the numerous efforts made to decrease the computational cost and increase the number of taxa (molecular sequences, DNA, RNA or proteins, or any other data that can be estimated in evolutionary distance) that it can take into account.

Our introduction will focus on a historical presentation of phylogenetic analysis and its biological significance, as well as the various existing methods in order to situate our subject in the general landscape of phylogenetic systematics.

The first part will provide the mathematical foundations of optimization and combinatorial analysis on which the rest of the work will be based. Some notions of graph theory will also be presented.

The second part will try to renew the first proposal to reach the optimal solution, introduced by Pardi in 2009, based on the combinatorial properties of the BMEP.

The third part of the work is in line with another path to the optimal solution, taken by Catanzaro in 2012, on mathematical grounds by integer linear programming. This method needs a characterization of the set of path-length matrices. The necessary conditions identified so far are proven sufficient for a number of taxa between 3 and 11 but remain open to conjecture for more than 11 taxa. Our approach will consist of an enumeration of all possible matrices for increasing values of n, hoping to reduce the computational cost of our algorithm and result in an enumeration for n = 12. Our hope is to find a counterexample to the conjecture, a number of enumerated matrices greater than the expected number, which would be sufficient to prove that the necessary conditions identified so far are not sufficient for n greater or equal than 12.

2 Introduction

The first part of this introduction will be devoted to a few words about the history of phylogenetic analysis and some underlying biological concepts. A second part will deal, in a more practical way, with the different steps required to build a phylogenetic tree and will mention the different methods available. The idea is to give an account of the general landscape and the place occupied by the main subject of this thesis, the BMEP, by evoking the steps downstream and upstream of the use of this criterion.

Phylogenetic analysis is one of the main tools of evolutionary biology. Evolutionary theory predicts that common ancestry groups result from evolutionary processes termed speciation, this is a fundamental presupposition in biology. While Charles Darwin was not the first to state that some species were genealogically related to others, the publication of his book On The Origin of Species [1] in 1859 had a very strong impact and is considered to be the foundation of evolutionary biology. A key goal of evolutionary biology is to reconstruct the his-

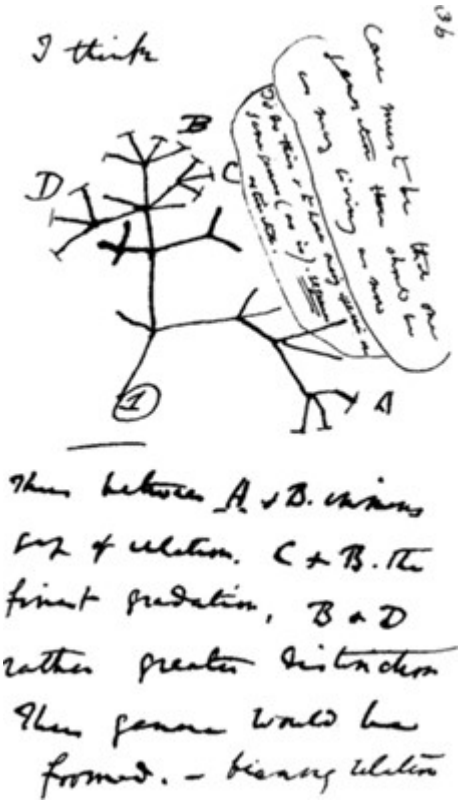


Figure 1: A detail from Darwin's Notebook B, (1837), showing his first sketch of a tree depicting phylogenetic relationships among species [2]

tory of speciation events and build phylogenetic trees. Phylogenetic trees (also called phylogenies) are graphical representations of the relationships between the characteristics studied, in other words, a phylogeny is a tree representing

the speciation history of a group of organisms. The leaves of the tree are typically existing species (or taxa), the root corresponds to their most common ancestor, and each branching of the tree indicates a speciation event.

Phylogenetic trees have been constructed for years using morphological, anatomical and/or physiological features to compare living organisms and study their relationship, based on the similarity (or analogy) of their characters.

It was not until the end of the 19th century, following the work of Gregor Mendel [3] (who formalized the laws of heredity) and August Weissman [4] (who initiated the distinction between somatic cells and germline cells, the latter being the only ones to transmit genetic identity to one's descendants and are therefore responsible for heredity), that genetics was explicitly mentioned in the debate on the evolution of life. Several schools of thought have since sought to model the evolution of life-based on genetic data [5].

In 1950, the entomologist Willi Hennig [6] postulated that systematics should express evolutionary knowledge: instead of trying to measure global similarity, the reconstruction of the relationship between species can only be done by specifying the ancestral and derived states of homologous characters.
In 1953, James Watson and Francis Crick [7] discovered the double helix structure of DNA, and the rapid sequencing of DNA segments was made possible in the mid-1980s by the development of PCR [8] (Polymerase Chain Reaction). The discovery of DNA as a medium of heredity and the development of molecular biology constituted a methodological revolution in phylogenetic analysis. One of the most important aspects of the molecular revolution for phylogenetics was the sheer amount of new data available to systematists. Molecular data also provided evidence where other data were scarce, allowing us to compare very different organisms or to distinguish very close organisms from a morphological point of view. Moreover, this new type of data promoted the mathematization of phylogenetics.

Historically, Emile Zuckerkandl and Linus Pauling [9] in 1965, then Walter Fitch and Emanuel Margoliash [10] appear classically as the pioneers of molecular phylogenies based on gene sequence data. Zuckerkandl and Pauling introduced and used the hypothesis of a molecular clock. The basic idea is the following: given some estimated rate of molecular evolution, the relative ages of branches and nodes may be estimated given some set of molecular data mapped onto a phylogenetic topology. Fitch and Margoliash funded strong similarities between molecular and morphological phylogenies of vertebrates and made it possible to understand that copies of a gene can be sisters by speciation (a series of mutations followed by speciation), but also by duplication within the same genome, and this is a factor to be taken in account when interpreting phylogenetic trees. These conceptual tools are still in use today.

So, the basic assumption of phylogenetic analysis is that all living things descend from a common ancestor. In the phylogenetic system, two taxa are related if they share a common ancestry. A monophyletic group (or clade) consists of an ancestor and all of its descendants. Each node on a tree is the origin of a monophyletic group. The construction and interpretation of a phylogeny should

consider the following biological concept: the biological meaning of a common ancestor is homology. Evolution is about homology, e.g., the similarity due to common ancestry. Thus, homology is a result of divergent evolution. There must be a good understanding of the functions of the genes of the proteins before starting the phylogenetic tree reconstruction.

In fact, it can sometimes be difficult to distinguish between homology (similarity inherited directly from an ancestor) and homoplasy (similarity may have occurred independently, does not emerge from a common ancestor, but results of convergent and parallel evolution), especially at a molecular level. It is important to emphasize the difference between similar and homologous genetic sequences, as one does not imply the other. The notion of similarity will be used in particular by the distance-based methods discussed in more detail later in this work.

Moreover, phylogenetic analyses that consider genes as taxa (gene trees) do not match the phylogenies of organisms or species (species trees). Homologous genes can be present in different species, but in different ways, notably by a duplication mechanism. Duplication refers to the evolutionary process by which new strands of DNA are created by copying an initial strand. If duplication is involved, the genes are said to be paralogous, and the gene tree differs from the species tree. To reconstruct a species tree, it is necessary to have orthologous genes: two homologous genes are orthologous if their most recent common ancestor has not undergone a gene duplication. The ortholog region is the most suitable molecular marker for underlying speciation events [11, 12, 13].

Phylogenetic systematists reconstruct the evolutionary relationship among organisms, species, and taxa using homologies that are hypothesized to indicate unique genealogic or at least evolutionary relationships. A phylogenetic tree is a mathematical structure used to model the evolutionary history of a group of organisms (or taxa). Phylogenies are hypotheses. They cannot be observed; they can only be inferred because they reflect past evolutionary events.

Tumor evolution differs from "classical" speciation events (hypermutability, chromosome or microsatellite instability). Single cells in a tumor evolve differently from organisms within a population. Although the idea of adapting methods for reconstructing species evolution to the study of tumors has proven powerful, the analogy has limits. Several computational models that arise in the study of tumor phylogenetics have been developed to account for this specific case. The topology of the phylogeny is rooted (the root is the taxa relative to a healthy cell). Distance-based methods can incorporate in the distance matrix any type of data for which a dissimilarity is available and are, therefore, particularly suitable for dealing with this problem [14, 15].

Speciation events are not the only processes by which evolution can occur. Consideration of these processes alone is a simplification of the real evolutionary process, and the reconstruction of phylogenetic trees based on speciation events alone is a model based on this simplifying assumption. For example, this may explain some of the discrepant results between the species tree and the gene tree. The graphical representation of these evolutionary events can be done in

the form of a phylogenetic network. This is beyond the scope of the present work, but it is worth noting. A rooted phylogenetic network is a generalization of a phylogenetic tree. It is a rooted acyclic digraph that is used to represent complex evolution, where reticulation events such as horizontal gene transfer (e.g., evolutionary event in which genetic material moves between organisms other than by the transmission of DNA from parent to offspring), recombination, etc., play a role. They have been used to model evolutionary relationships in population genetics, plant biology (hybridization network), and genome evolution [16].

A second part of this introduction will deal, in a more practical way, with the different steps required to build a phylogenetic tree and will mention the different methods available. The idea is to give an account of the general landscape and the place occupied by the main subject of this thesis, the BMEP, by evoking the steps downstream and upstream of the use of this criterion [17, 18, 19].

The methods of analysis based on molecular sequences used in phylogeny were initially developed for the study of evolutionary relationships between organisms, genes or protein sequences. They are currently used in several fields and have many applications such as the study of biodiversity [20], epidemiology [21, 22], identification of gene functions [23], and in microbiology [24]. To illustrate with an unfortunately still current episode, the SARS-CoV-2 pandemic has demonstrated the crucial importance of these phylogenetic analyses for public health, and a guideline published by the World Health Organization details the necessary steps and technical aspects of genomic sequencing and analysis of SARS-CoV-2 [25].

Selection of genetic marker and assembling a data set
The first step of a phylogenetic analysis is the selection of appropriate genetic markers, i.e., identification of homologous sequences (gene with the same ancestral sequence). This require a good biological understanding of the function of protein or gene and the process in which the phylogenetic analysis is carried out. The assembly of a dataset in order to make a comparison between the sequences is a crucial step. In order to be easily accessible to researchers and to facilitate information processing, the nucleotide sequences obtained by biologists are stored in databases, accompanied by annotations allowing targeted research, for example DDBJ (DNA Data Bank of Japan) or GenBank [26, 27].

Multiple sequence alignment
Once the sequences have been identified and the dataset assembled, they must be aligned in order to be able to compare them and determine the homology between each of the nucleotides (DNA/RNA) or amino acids (proteins). The aim is to ensure an optimal correspondence of the sites between them. The basic premise of a multiple sequence alignment is that, for each column in the alignment, every residue from every sequence is homologous; that is, has evolved from the same position in a common ancestral sequence without insertion or deletion (called indels). This is important, because the models of molecular evolution (see further) are based on nucleotide substitutions models (AND/ARN) or protein and codon substitution models (and not on insertions or deletions).

The multiple alignment step is crucial because it determines the comparative basis on which the phylogenetic tree is built. Molecular phylogenetic analysis relies heavily on the accuracy of the sequence alignment. A molecular phylogeny is only as good as the alignment it's based on. Poor alignment introduces phylogenetic bias. At best, misaligned sequence has no useful phylogenetic information; at worst, it might have convincing misinformation [28].

Many softwares are available to solve the problem of multiple sequence alignment, such as MAFFT [29] (Multiple Alignment using Fast Fourier Transform), MUSCLE [30], ClustalW [31] and others [32].

It is still necessary to select a part of the alignment, areas of low similarity are ignored, as well as gaps, insertion and deletion areas, which most alignment algorithms have trouble handling properly. The goal is to keep only the sites from which we could extract phylogenetic information.

Solving the multiple alignment problem is NP-hard [33], and some alignment-free sequence comparison methods have been explored and proven useful for dealing with large molecular databases, being computationally less expensive. These methods are suitable for whole genome comparison, for the study of sequence rearrangement, or horizontal gene transfer. However, the alignment is still an essential step in many aspects of today's biology, like reconstruction of ancestral DNA sequences [34].

Determination of substitution model
The estimation of dissimilarities or evolutionary distances between sequences is the next step. The dissimilarity measure is based on molecular data extracted from the species. Indeed, the real evolutionary difference can only be estimated: a simple counting of the difference between sequences (or observed distance, i.e., the number of substitutions on the number of sites) underestimates the numerous mutations that have occurred during evolution. Many substitution models have been proposed to estimate the real difference between sequences, based on their present state (which is often the only information available). It is important to note that similarity (or homology) between two sequences is not identical to homology, but the more similar the sequences are, the more likely they are to be homologous. The substitution models of evolution constitute the basis of the evolutionary analysis of genetic data at the molecular level. These are probabilistic models that must be chosen according to the specific problem studied. Each of them carries its own assumptions.

There are three types of molecular sequences (nucleotides, amino acids and codons sequences). Therefore, there are three corresponding types of substitution models. All these substitution models are Markov chain models.

The basic principle of some classical models of nucleotide substitution are presented here. DNA is made up of four different nucleotides: adenine, thymine cytosine, and guanine (ATCG). These come in two different shapes; adenine and guanine are purine rings, cytosine and thymine are pyrimidine rings. Transitions are the exchange of nucleotide of the same shape (AG or CT). Transversions are the exchange of nucleotides of different shapes. The Jukes Cantor (JC) model

[35] is one of the first models proposed in 1969. It is a single parameter model. It assumes that the four bases have equal frequencies and that there is the same substitution rate for the four nucleotides. The Kimura's 2 Parameter Model [36] (K2P) proposed in 1960 allows transitions and transversions to occur at different rates. The HKY85 model (Hasegawa-Kishino-Yano, 1985) and the F84 model (Felsenstein 1984) accept arbitrary nucleotide frequencies, to name a few. All common phylogenetic methods are based on certain assumptions that are not biologically realistic (substitutions are independent of each other, the substitution rate is constant over time and between lineages, among others). Therefore, it is common to correct the data by a gamma distribution that approximates the rate heterogeneity over sites. [37, 12]

The substitution models based on amino acid or codon sequences are conceptually the same, except that the matrices are larger.

Tree building
There are two general categories of methods for calculating phylogenetic trees: the distance-based methods and the character-based methods. We will focus on the distance-based methods, of which the BMEP is a part.

Distance-based methods
The distance-based methods are based on the concept of evolutionary distance between any two sequences or between pair of taxa. These distances are estimated distances and are given a priori, they measure the dissimilarity between pairwise molecular data. The sequence information is transformed into a pairwise distance matrix.

A distance-based method has two components: the evolutionary distance matrix typically derived from a substitution model, which is the input of a tree-building method, using an algorithm that construct a tree from the distance matrix.

The BME criterion, based on the work of Pauplin in 2002 [38], was developed to overcome the inconsistencies of the MEP criterion pointed out by Rzhetsky and Nei in 1993 [39], namely that if the distance matrix D is a biased estimate of the real evolutionary distances, then negative values are found among the edge weights of the optimal solution, which has no biological basis [40, 41].

At present, there is no exact solution algorithm able to handle a large number of taxa, but several algorithms based on heuristic approaches are described in the literature and widely used. The NJ method (Neighbor Joining), introduced by Saitou and Nei in 1987 [42], is a greedy algorithm that build trees by iterative agglomeration of taxa. The criterion being minimized by NJ is the BME, as shown by Gascuel and Steel in 2006 [43]. FastME 2.0 [44], introduced by Desper Gascuel in 2002 [45], is a notable tool, fast and BME-based, that proposes several sub-methods of heuristic and topological moves (Nearest Neighborhood Interchange-NNI + Subtree Pruning and Regraphing-SPR neighborhoods).

The main advantage of approximative distance-based methods is their speed of execution: they work on evolutionary distances of taxa and are therefore computationally less demanding than other methods working directly on molec-

ular sequences (character-based methods). A general trend in bioinformatic and computational biology is the growing demand for methods that can cope with massive datasets of DNA sequences. Distance-based methods are a possible answer to this demand, not only for phylogenetic inference, but also for related tasks such as sequence identification and gene orthology inference [46].

A limitation of distance-based methods lies in the fact that if the distances are estimated from pairwise sequence comparisons only, then it may be impossible to infer some parameters common to the evolution of all the sequences [47]. For example, we cannot trace the evolution of individual sites on a tree, we only have an overall estimate of the relationship between tree and data. There may be some loss of information inherent in converting sequences to distances.

Character-based methods:
These methods consider each nucleotide site (or some function of each site) directly. These methods operate directly on the sequences, or on functions derived from the sequences, rather than on pairwise distances. Discrete data methods examine the nucleotide variation in each column of the alignment separately. These methods are quite time-consuming because all the sequence information is used for the evaluation of the best phylogenetic tree.

- Maximum parsimony (MP) [48]: chose the tree (or trees) that require the fewest evolutionary changes, i.e., that explains the alignment with the least possible substitutions. MP is based on the principle that the best hypothesis to explain a process is the one that involves the smallest number of events (Lex parcimoniae). The parsimony methods are not based on an explicit evolution model. These methods are less used, but they offer a great simplicity associated with fast algorithms.

- Maximum likelihood (ML): proposed by Felsenstein in 1981 [49], is a statistical method that chose the tree (or trees) that of all trees is the one that is most likely to have produced the observed data. It is a frequentist method that interpret the probability in terms of hypothetical frequencies of events. ML requires an explicit model of sequence evolution, a tree and the observed data. The phylogeny is a rooted tree. This method is claimed to be very accurate, because the analysis relies heavily on the evolutionary model. ML uses all the information in the sequence to calculate the maximum likelihood value but has a high computational cost and is not suitable for large data sets.

- Bayesian inference [50] is the newest method: this method relies on Bayes's theorem, which transform data and input probabilities (prior probabilities) into output probabilities (posterior probabilities). Bayesian phylogenetic inference uses a Markov chain Monte Carlo (MCMC) algorithm to search for the best tree. Some popular software are MrBayes [51], and BEAST [52].

Most phylogenetic reconstruction methods generate unrooted trees. Unrooted trees do not specify evolutionary relationships in terms of ancestors or descendant, they can only imply different relationships between taxa depending on the location of the root. In other words, they don't have an evolutionary

direction, there is no temporal dimension in unrooted trees. In order to root an unrooted tree, we need some other source of information. An outgroup can be used to root the tree. An outgroup is a taxon and its data, chosen by the user, to root the tree. Often a user will choose an outgroup that is a closely related strain to the study group. Alternatively, a user might use an isolate of the same strain collected earlier than the study group as an outgroup.

Tree evaluation
Trees are often reconstructed from several methods in the software on the market to increase the confidence in the final result. The bootstrap [53] is a general method in statistics to evaluate the uncertainty of an estimate due to incomplete sampling, this method was applied to phylogenetic trees by Felsenstein in 1985 to evaluate the robustness of the inference. The idea is to perturb the initial data (sequence alignment) in a random way, and for each data set a phylogenetic tree is built with the same method. But the bootstrap is not able to overcome an inappropriate analysis of the data, and many other sources of error are possible, such as sampling errors or choice of an unsuitable evolution model.

Trees are hypotheses of evolutionary relationship, and are inference made on limited data using often greatly simplified models of evolution, that nevertheless pose immensely challenging computational problems.

3 Notions of graph theory

Before getting into the details of the thesis, it is important to have some notions of graph theory used throughout the thesis.

The basic assumption of phylogenetic reconstruction is that the evolutionary history of species takes place through successive speciation events. According to this hypothesis, an ancestral lineage can, by speciation, give rise to two new lineages, and a phylogeny can be represented by a tree. This paragraph introduces several concepts that are frequently used throughout this thesis. Some of the notions mentioned below are based on elementary notions of Graph theory.

Definition 1 *A graph is a triplet (V, E ,ϕ), where*

- *V is a finite set where the element's constituents are called vertices or nodes.*
- *E is a finite set where the element's constituents are called edges.*
- *ϕ is an incidence function associating every edge with one or two nodes.*

Definition 2 *A directed graph is a triplet (V, E, ϕ), where*

- *V is a finite set where the element's constituents are called vertices or nodes.*
- *E is a finite set where the element's constituents are called edges.*
- *ϕ is an incidence function associating every edge with an ordered couple of nodes. (The edge can only be used in one way)*

Definition 3 *Two vertices incident to the same edge is said to be adjacent or neighbors.*

Definition 4 *The degree of a node represents the number of incident edges to this node.*

Definition 5 *Two graphs (V, E, ϕ) and (V', E', ϕ') are Isomorphic if there exists bijective functions $f : V \to V'$ and $g : E \to E'$ such that:*

$$\phi(e) = \{u, v\} \text{ iff } \phi(g(e)) = \{f(u), f(v)\}$$

Definition 6 *A path is a sequence of vertices $x_0, x_1, \ldots, x_n$ such that every consecutive vertex represents an edge of the graph. Meaning that for all $0 < i \leq n$ we have $(x_{i-1}, x_i) \in E$. A path can contain multiple times the same node. It is called a closed path if $x_0 = x_n$.*

In link with the path, let's denote the notation

$$x_i P x_j := x_i, \ldots, x_j$$

as the path between node x_i and x_j.

Definition 7 *A cycle is a closed path in which every node and edge are unique.*

Definition 8 *A graph is called connected iff, for each pair of vertices, there exists a path that connects them.*

Definition 9 *Let us call a path an Eulerian path if it traverses every edge of the graph exactly once. A graph is Eulerian if it admits a closed Eulerian path.*

3.1 Trees

An acyclic graph, one not containing any cycles, is called a forest. A connected forest is called a tree. (Thus, a forest is a graph whose components are trees.) The vertices of degree 1 in a tree are its leaves, and the others are its inner vertices.

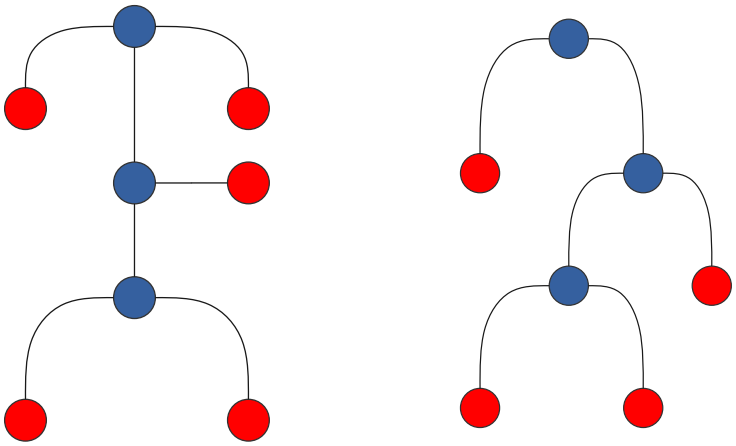
Definition 10 *A tree is said to be bifurcating if all non-root internal nodes are attached to precisely three edges and the root is connected to either one or two edges.*

Definition 11 *A rooted binary tree is a binary tree in which only the root is allowed to have degree 2. The remaining nodes have degrees equal to either 1 or 3. The nodes with degree 1 are called leaves and the nodes with degree 3 are the internal vertices.*

Definition 12 *An unrooted binary tree is a tree whose leaves (degree 1 vertex) are labeled bijectively by a (species) set* Γ*, and such that each non-leaf vertex is unlabelled and has degree three. We let UBT(n) denote the set of such trees for* $\Gamma = \{1, ..., n\}$*. [54]*

We now have enough background to define properly what a phylogeny is.

Definition 13 *Consider a set* $\Gamma = \{1, 2, ...n\}$ *for* $n \geq 3$ *of distinct molecular sequences referred as taxa. A phylogeny* T *is an ordered triplet* $(U, \phi, \boldsymbol{w})$ *where* U *is an unrooted binary tree with* n *leaves,* ϕ *is a bijection between the leaves of* U *and the taxa of* Γ *and* $\boldsymbol{w}$ *is a vector of weights associated to the edges of* U*.*



(a) Unrooted binary tree

(b) Rooted binary tree

Figure 2

For introduction purposes, we will define two specific phylogenetic trees. These two topologies will be used throughout the thesis.

The first notable graph is the three-leaf phylogeny or the three-star phylogeny. It is composed of only one internal vertex and has three leaves. This is the only topology representing a phylogenetic tree, as we have defined it for three taxa. It will be used as a base for a lot of developments. Its representation can be seen in Figure 3.

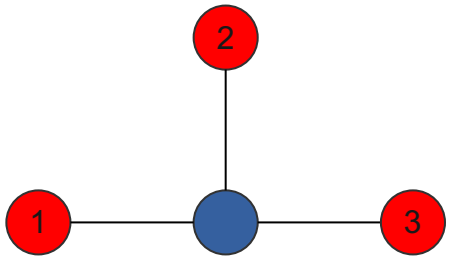


Figure 3: three-star phylogeny

The second notable phylogeny is called "the comb". It is composed of as many leaves as we want, but every internal vertex is adjacent to two other vertices and one leaf, except for the two most extreme vertices, which are adjacent to one vertex and two leaves. An example of this type of tree for five taxa can be seen in Figure 4.

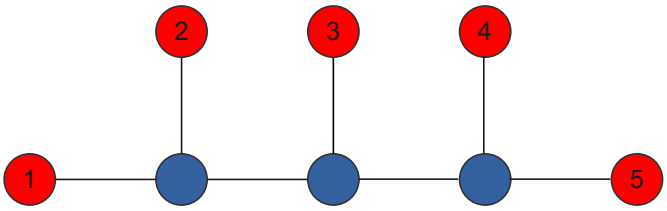


Figure 4: Representation of the comb phylogeny for five taxa

3.2 Meaning for phylogenetic trees

Definition 14 *The topology is the branching structure of the tree. It is of particular biological significance because it indicates patterns of relatedness among taxa, meaning that trees with the same topology and root have the same biological interpretation.*

We will use phylogenetic trees as unrooted binary trees later in this thesis due to some assumptions and aims of the model.
For unrooted trees, there is no ancestral root. Unrooted trees represent the branching order but do not indicate the root of the location of the last common ancestor [55]. So the aim is not to find the parent-child relations but to find the speciation events between the species.

The tree is also not oriented because it does not represent a historical evolution. It only gives relationships between taxa and speciation events. It also comes from the fact that it does not aim to represent a parent-child relationship.

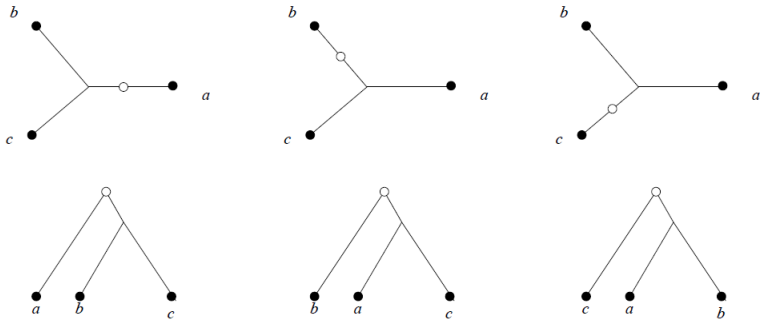


Figure 5: Making rooted binary trees from unrooted binary trees [56]

Another advantage of working with unrooted binary trees is that it is less computationally demanding. This can be easily understood since that in order to make a rooted binary tree from an unrooted binary tree. One can add a root to any of the edges of the unrooted binary tree. Since there are 2n - 3 edges to a rooted binary tree with n leaves, the following development follows [56].

$$UBT(n) = (2n-5)!! \tag{1}$$

$$RBT(n) = UBT(n) * (2n-3) = (2n-5)!! * (2n-3) = (2n-3)!! \tag{2}$$

So working with RBT with n leaves results in the same number of trees as working with UBT with n+1 leaves. Therefore, choosing to work with UBT is better from a computational point of view.

4 Estimating phylogenies from molecular data

Re-introduce this
There are multiple models that aim to reconstruct phylogenies but they can be separated into two main archetypes. The first one is character based and focuses more on morphological data, the two main models for this approach are the Maximum parsimony and the maximum likelihood.
The second approach, on which the model hereafter is part of, is based on a distance between species. that is, the best evolutionary trees are the ones that minimize the number of changes along the edges of the tree. [57]
The most widely used methods are the maximum parsimony (MP; [14 , 15]) and maximum likelihood (ML; [16 , 17]) methods.

Now that the problem is introduced, the question of how to actually construct the optimal phylogeny still resides. How to translate the problem on paper and which criteria of construction of the phylogeny to choose. This section will address some of these questions and give a brief overview of all the different choices available.
The general problem of the estimation of a phylogeny is stated as [58, 41]:

Problem 1 *Given a set* Γ *of n taxa, find the phylogeny* T^* *that solves the problem*

$$\begin{aligned} &\textit{Optimize} && \Lambda_\Gamma(T) && (3)\\ &\textit{s.t.} && \Omega_\Gamma(T) \geq 0 && (4)\\ & && T \in \mathcal{T} && (5) \end{aligned}$$

Where

- $\mathcal{T}$ is the set of $(2n-5)!!$ phylogenies of Γ.
- $\Lambda_\Gamma : \mathcal{T} \to \mathbb{R}$ is a function that models the criterion of selection for the phylogeny to find.
- $\Omega : \Gamma \times \mathcal{T} \to \mathbb{R}^m$ for some $m \geq 1$, is an additional (not always required) function that imposes additional constraints on the structure of the phylogenies based on some evolutionary assumptions.

Finding an optimal solution to this kind of problem means finding the tree T that minimizes (or maximizes) the objective function while respecting the constraints.
There are different criteria proposed in the literature for different purposes, differing from each other by the type of observed data, the evolutionary hypotheses, and the assumptions. This has been further discussed in the introduction part in much further detail.
In this thesis, we will be focused on one evolution criteria based on distance methods, namely the Balanced Minimum Evolution Problem (BMEP), which is presented hereafter.

4.1 The BMEP formulation

The BMEP is formulated as follows:

Problem 2 *(The balanced minimum evolution problem)*
Given a set Γ *of taxa, find an phylogeny* $T \in \mathcal{T}$ *having* Γ *as leaf set and minimizing the following length function:*

$$min \qquad L(T) = \sum_{i \in \Gamma} \sum_{j \in \Gamma \setminus i} \frac{d_{ij}}{2^{\tau_{ij}}} \tag{6}$$

where τ_{ij} represents the path length between taxon i and taxon j in T i.e. the number of edges belonging to the path connecting these two taxa and d_{ij} represents the distance between taxon i and j i.e the dissimilarity metric between the two taxa [41].
This formulation is a particular case of Problem 1 where $\Lambda_\Gamma(T)$ is $L(T)$, without any additional constraint on the set apart the implicit fact that $T \in \mathcal{T}$ hence $\Omega_\Gamma(T) = \emptyset$.

One easy way to understand the objective function is by looking at the terms that compose it. The term d_{ij} represents a measure of dissimilarity or a genealogical distance between the taxa i and j. We can understand the terms $2^{-\tau_{ij}}$ as a discount factor that decreases when the distance between taxa increases. Simply said, minimizing this function means trying to make taxa that lie nearby each other be as similar as possible (having a small τ_{ij} with a small d_{ij}) and making taxa that are more different be far away from each other in the binary tree (big d_{ij} implies big τ_{ij}).

Now that this intuitive description is given, we will take a deeper look and give more details about where this originates from.

4.2 Combinatorial and optimization aspects of the BMEP

Before explaining in detail the description of the optimization problem, there are two main properties of a phylogenetic tree. They are stated hereunder: [59]

(i) A binary phylogenetic tree with n leaves has $2n - 3$ edges and $n - 2$ internal vertices.
(ii) For a fixed set Γ of size at least three, the number of binary phylogenetic trees is

$$\frac{(2n-4)!}{(n-2)!2^{n-2}} = 1 \times 3 \times \cdots \times (2n-5) = (2n-5)!! \tag{7}$$

where $n = |\Gamma|$
An easy way to understand *(ii)* is by seeing that to increase the size of a phylogenetic tree, one can add a taxon on every edge of the tree. Doing so actually means that a speciation event happened in between either a taxon and a speciation event (leaf and internal vertex) or in between two speciation events (two internal vertices), and in both cases, it resulted in a new specie.
By applying this idea, the following equation results:

$$|\mathcal{T}_{n+1}| = |\mathcal{T}_n|(2n-3)$$

which gives the same results under $n \geq 3$. The following graph shows the idea behind this explanation and by knowing that $|\mathcal{T}_3| = 1$.
A visual representation of the proof is shown in Figure 6.

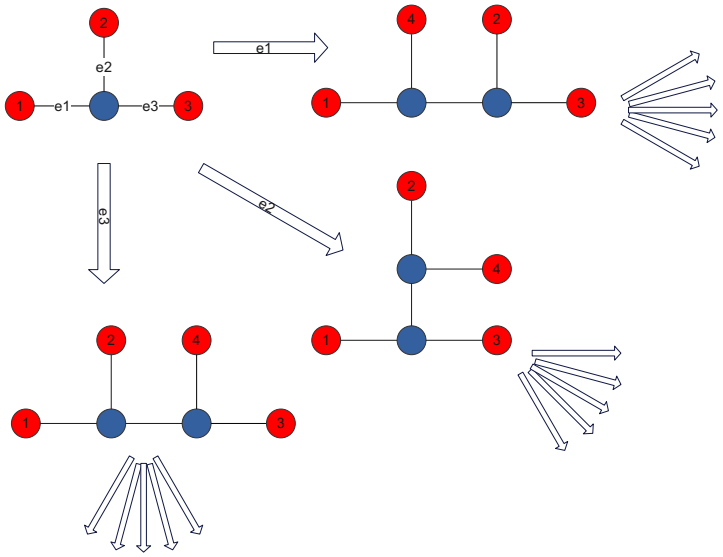


Figure 6: Different possibilities to add a taxon to a tree

4.3 Development of objective function with circular orderings applied to phylogenetic trees

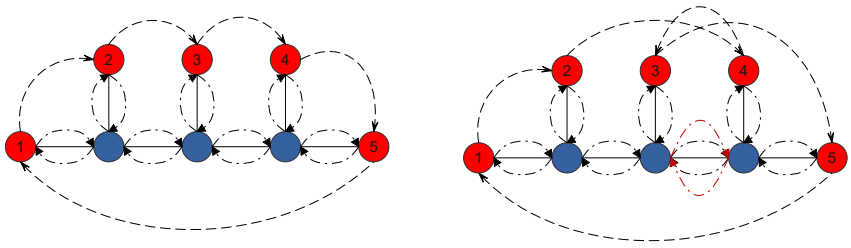
To have a deeper understanding of how the function $L(T)$, which represents the length of the phylogenetic tree, evolved to this form and the development that led to it so as how to characterize formally the set Θ, which is the set of path-length matrices corresponding to a phylogenetic tree, some notions and bases must be introduced.

The particular form of the length function $L(T)$ originates from a concept called circular ordering [60]. To form a circular permutation π of a phylogeny T of the set $\mathcal{T}$ made by n taxa, you start by representing the phylogeny on a planar space and take a first leaf/taxa and set it as the first element of π. After that, you add the next leaf that comes in the circular order, i.e. the next one that comes clockwise (or counter-clockwise) on the graphical representation.

Some examples illustrate what an actual circular order is and what it is not in figure 7. The subfigure (a) represents an actual circular ordering of a phylogeny T made by five taxa, the permutation of the circular order is $\pi_a = (1, 2, 3, 4, 5)$. We can see that each edge is visited twice, whereas the ordering on the subfigure (b) $\pi_b = (1, 2, 4, 3, 5)$, one edge is visited four times.

We can see that these orders are related to Eulerian circuits because they pass only twice on each edge and once on each leaf before returning to the initial leaf.

Let $\pi(i)$ denote the i^{th} element of a permutation π of Γ and by convention, assume that $\pi(n+1) = \pi(1)$ to be able to completely describe the circuit in an easy way. Then with these notions introduced, two demonstrations were made



(a) (b)

Figure 7: (a) : circular order, (b) : not a circular order

[59]:

(i) A circular order decomposes T in precisely n paths ${}_{\pi(i)}P_{\pi(i+1)},\ i \in \{1, \cdots, n\}$

(ii) Every edge occurs only twice in the paths ${}_{\pi(1)}P_{\pi(2)}, \cdots, {}_{\pi(n)}P_{\pi(n+1)}$; it is a necessary and sufficient property of π to be a circular order of T.

A further observation is that out of the $n!$ permutations of π for a set Γ, only 2^{n-2} are circular orders on T [59, 61, 62].

The term d_{ij} represents the measure of dissimilarity but is also a notion of evolutionary distance between taxa i and j $\in \Gamma$.
If we suppose that d_{ij} it is equal to the sum of the weights of the edges on the path between taxa i and j, i.e.

$$d_{ij} = \sum_{e \in E(p_{ij})} w_e \quad \forall\, i, j \ \in \Gamma \tag{8}$$

where p_{ij} represents the shortest path between taxa i and j. This is an important property since we can calculate the distances d_{ij} between taxa with different methods, whereas the weights of the edges w_e are difficult to estimate and change in each phylogenetic tree.
Under the assumption that (8) holds true and by knowing that $L(T)$ is the length function of the tree then:

$$2L(T) = 2\sum_{e \in E} w_e = \sum_{i=1}^{|\Gamma|} d_{\pi(i)\pi(i+1)} \tag{9}$$

It was also proved by Semple & Steel [59] that the length of a tree does not depend on the choice of π. Therefore the following holds true:

$$L(T) = \frac{1}{2}\sum_{i=1}^{|\Gamma|} d_{\pi(i)\pi(i+1)} = \frac{1}{|\Pi(T)|} \sum_{\pi \in \Pi(T)} \frac{1}{2}\Big(\sum_{k=1}^{|\Gamma|} d_{\pi(k)\pi(k+1)}\Big) \tag{10}$$

Where $\Pi(T)$ represents the set of the circular orders for a specific phylogeny T. As a reminder, it was proven that this set has 2^{n-2} different sequences in it.

By introducing μ_{ij} as the fraction of circular orders in which taxon j immediately follows taxon i and by noticing that the number of internal vertices not belonging to the path p_{ij} is equal to $(n-2)-|V(p_{ij})\setminus\{i,j\}| = (n-2)-(\tau_{ij}-1)$ [41]; n-2 being the number of internal vertices and $|V(p_{ij})\setminus\{i,j\}|$ being the number of vertices not belonging to the path between taxon i and j, which is equal to the number of edges separating them - 1.

The following results comes from [59] in which he proved the following propositions:

Proposition 1 *Let T be a phylogenetic Γ-tree with at least one interior vertex. Then the number of distinct circular orderings for T is:*

$$\prod_{v\in\overset{\circ}{V}(T)} (d(v)-1)!$$

Where $\overset{\circ}{V}(T)$ denotes the set of interior vertices in a graph T.

Furthermore, for all distinct elements $x, y \in \Gamma$, the proportion of these circular orderings for which y immediately follows x is

$$\prod_{v\in V(p_{ij})} (d(v)-1)^{-1}$$

Since we know that there are n-2 internal nodes in an unrooted binary graph and that each internal node is of degree 3 we have enough information to derive the number of circular orders in which taxon j immediately follows i:

$$\prod_{v\in\overset{\circ}{V}(T)} (d(v)-1)! \prod_{v\in V(p_{ij})} (d(v)-1)^{-1} = 2^{n-2}2^{-(\tau_{ij}-1)} \tag{11}$$

We now have all the tools to derive the mathematical expression of μ_{ij}, the fraction of circular orderings in which taxon j immediately follows taxon j:

$$\frac{\mu_{ij}}{2} = \frac{2^{(n-2)-(\tau_{ij}-1)}}{2|\Pi(T)|} = \frac{2^{(n-2)-(\tau_{ij}-1)}}{2^{n-1}} = 2^{-\tau_{ij}}$$

We now have all the elements to explicitly describe the function L(T) as formulated in (6).

$$L(T) = \frac{1}{|\Pi(T)|} \sum_{\pi\in\Pi(T)} \frac{1}{2}\Big(\sum_{k=1}^{|\Gamma|} d_{\pi(k)\pi(k+1)}\Big) = \sum_{i\in\Gamma}\sum_{j\in\Gamma\setminus\{i\}} \frac{\mu_{ij}}{2} d_{ij} \tag{12}$$

The transformation in equation (12) comes from a change in the summation order. Instead of doing the summation over the circular orderings, we do the summation over the distances present in the circular orderings, which are multiplied by how often they are present in all the cycles. Since μ_{ij} is a proportion, $\sum_i^n \sum_{j\neq i}^n \mu_{ij} = 1$, this means that the objective function is a weighted average of the half distances. This is where the term balanced in the model's name

comes from.
and by developing the term μ_{ij} it finally becomes the objective function of the BMEP (6):

$$L(T) = \sum_{i \in \Gamma} \sum_{j \in \Gamma \setminus \{i\}} \frac{d_{ij}}{2^{\tau_{ij}}} \tag{13}$$

Which gives the same length formula as in the BMEP formulation [61, 62].

4.4 Description of the set describing path-length matrices of phylogenetic trees

As a reminder $\mathcal{T}$ represents the set of the $(2n-5)!!$ different phylogenies made upon the set Γ where $n = |\Gamma|$. We will now describe all the known properties that the set Θ, which represents all the path-length matrices associated to the phylogenetic trees.

4.4.1 Conditions inherent to path-length matrices

The question of how to characterize the set Θ rose and is (was) an open problem. The first observations are straightforward, since τ is a path-length matrix of an UBT(Γ), its entries must respect the next three necessary constraints [41, 61]:

$$\tau_{ij} \in \{2, ..., n-1\} \ \ \forall\, i \neq j \tag{14}$$

$$\tau_{ii} = 0 \ \ \forall\, i \in \Gamma \tag{15}$$

$$\tau_{ij} = \tau_{ji} \ \ \forall\, i, j \in \Gamma \tag{16}$$

These are the first conditions of the set.
The first property, equation (14), indicates that the number of edges separating two taxa is at least two and at most n-1.
The null diagonal property, equation (15), comes from the fact that there are no loops in a tree.
The symmetric property, the equation (16), comes from the fact that an unrooted binary tree is not directed. An edge can be used on both sides, so the distance to go from taxon i to taxon j is the same as the distance to go from taxon j to taxon i.

The other nontrivial necessary properties to characterize the set of path-length matrices of unrooted binary trees are explained hereafter.

4.4.2 Kraft equality applied to phylogenetic trees

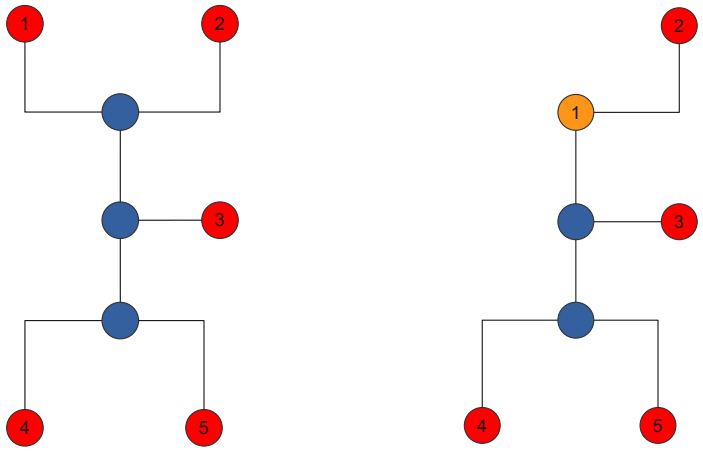
This next equation originates from a property applied to rooted binary trees, which is widely used with Huffman codes. The theorem called the Kraft Equality is stated as follows [63]:

Theorem 1 *For all $n \leq 1$, ${}_rP_1, {}_rP_2, \ldots, {}_rP_n$ is a sequence of path lengths between the leaf i and the root of the tree (r) in a rooted binary tree iff*

$$\sum_{i=1}^{n} 2^{-{}_rP_i} = 1 \tag{17}$$

By finding an analogy between rooted binary trees and unrooted binary trees, the latter being what is of interest for this model, we can adapt the Kraft equality to our problem.

The analogy between rooted and unrooted binary trees comes by seeing that a contraction of a leaf in an UBT to its adjacent vertex results in a rooted binary tree, to which we can apply the equality [61].
This operation can be visualized in Figure 8. If we denote τ_{ij} as the number of edges separating the taxon i and the taxon j, let's denote $\tau_{ij}^{c_i}$ the distance between the node i, which is the node resulting from the contraction of the leaf i to its adjacent vertex, and taxon j. We can now apply the Kraft equality to this tree since it is an UBT.



(a) T (b) T^{c_1}

Figure 8: (a): Unrooted binary tree T, (b) Contraction of the tree T on taxa 1

$$\sum_{j=1 \ j\neq i}^{n} 2^{-\tau_{ij}^{c_i}} = 1 \tag{18}$$

We can apply this operation on every leaf of an unrooted binary tree independently, which leads to the following:

$$\sum_{j=1, j\neq i}^{n} 2^{-\tau_{ij}^{c_i}} = 1 \ \ \forall i \in \Gamma \tag{19}$$

This implies that every path-length sequence between two taxa if reduced by one, respects the Equality. By noting that $\tau_{ij} = \tau_{ij}^{c_i} + 1$, we can expand the original equation to another equation that can be applied to phylogenetic trees.

$$\sum_{j=1,\, j\neq i}^{n} 2^{-\tau_{ij}^{c_i}} = 1$$

$$\sum_{j=1,\, j\neq i}^{n} 2^{-\tau_{ij}} = \sum_{j=1,\, j\neq i}^{n} 2^{-(\tau_{ij}^{c_i}+1)} = \frac{1}{2} \sum_{j=1,\, j\neq i}^{n} 2^{-\tau_{ij}^{c_i}} = \frac{1}{2}$$

Which results in this general relation for all the taxa:

$$\sum_{j=1\ ,j\neq i}^{n} 2^{-\tau_{ij}} = \frac{1}{2} \quad \forall i \in \Gamma \tag{20}$$

This equation is independent of the row we select and states that for any taxon we contract to its closest vertex, the obtained result is a rooted binary tree, which incorporates the notion of connectivity and respect for the degree of internal nodes of an unrooted binary tree. However, since each row can represent a different unrooted binary tree, more constraints are needed to ensure that the complete matrix τ encodes a tree. This is, therefore, a necessary condition but is not sufficient.

This makes up for a fourth necessary condition that defines the set Θ.

4.4.3 phylogenetic manifold

Another condition on Θ, known as the *phylogenetic manifold*, is expressed as [64, 61]:

$$\sum_{i\in\Gamma} \sum_{j\in\Gamma\setminus i} \tau_{ij} 2^{-\tau_{ij}} = (2n-3) \tag{21}$$

It originates from the work of Pauplin [38], in which he established the relation between an edge set and the edge weights of a phylogenetic tree. It states that for any phylogeny T with edge set $\mathcal{E}(T)$ and for any set of edge weights $\{w_e : e \in \mathcal{E}\}$, the following condition holds: $\sum_{e\in\mathcal{E}} w_e = \sum_{i\in\Gamma} \sum_{j\in\Gamma\setminus i} \delta_{ij} 2^{-\tau_{ij}}$, where δ_{ij} is equal to the sum of the weights w_e along the path from taxon i to taxon j, for all $i \in \Gamma$ and $j \in \Gamma \setminus i$. When setting $w_e = 1$, for all $e \in \mathcal{E}(T)$, we obtain $\delta_{ij} = \tau_{ij}$ and the statement (21) follows naturally since we know the exact number of edges in a phylogeny of size n [64].

4.4.4 Triangular inequality

The last condition comes from the triangular inequalities of a path-length sequence applied to phylogenetic trees. The normal condition is stated as follows:

$$\tau_{ij} + \tau_{jk} - \tau_{ik} \geq 2 \ \ \forall\, i,j,k \in \Gamma : i \neq j \neq k \tag{22}$$

Because any path-length matrix $\tau \in \Theta$ must encode a tree, the triangular inequalities (13) can be further generalized by means of Buneman's additive property [65, 41], which states that:

Theorem 2 *a given graph G = (V,E) is acyclic and connected (i.e., a tree) if and only if*

$d_{ij}+d_{pq} \leq max\{d_{iq}+d_{jp}, d_{ip}+d_{jq}\} \ \forall \ i,j,p,q \in V$ *at least three of which distinct.*

When Buneman's additive property is applied to UBTs, any quartet i, j, p, and q of its vertices (at least three of which are distinct) must satisfy exactly one of the following conditions:

$$\begin{cases} \tau_{ij} + \tau_{pq} + 2 \leq \tau_{iq} + \tau_{jp} = \tau_{ip} + \tau_{jq} \\ \tau_{iq} + \tau_{jp} + 2 \leq \tau_{ij} + \tau_{pq} = \tau_{ip} + \tau_{jq} \\ \tau_{ip} + \tau_{jq} + 2 \leq \tau_{ij} + \tau_{pq} = \tau_{iq} + \tau_{jp} \end{cases} \tag{23}$$

Another result that can be used from Buneman is that given an arbitrary distance matrix d, we can determine whether d is additive without attempting to obtain a suitable tree. [56]:

Theorem 3 *A distance matrix D on* Γ *is additive, iff for every quartet of (not necessarily distinct) taxa* $x_i, x_j, x_k, x_l \in \Gamma$ *the so-called four-point condition holds:*

$$d(x_i, x_j) + d(x_k, x_l) \leq \max(d(x_i, x_k) + d(x_j, x_l), d(x_i, x_l) + d(x_j, x_k))$$

The essential difference between Buneman's theorem and the additivity of a matrix is that for Buneman's four-point condition, he is verifying the condition for all the nodes of the graph. Here the matrices we are using only represent matrices of distances between tree leaves. Therefore the acyclicity of the graph to which the distance matrix is associated cannot be verified by only checking the distances between taxa. However, if the matrix does not respect the four-point condition, it cannot be represented by a graph, reason why it is only a necessary condition of the set. This is a question still pending. Bringing an answer to the question of whether checking Buneman's property to only the leaf set is sufficient to have a graph without a cycle for the matrix τ.
This can be applied to the matrix τ in which case all the edges have a weight of 1 (topological distance) or on the distance matrix d that represents the evolutionary distance or dissimilarity between taxa. This property has a double use, since it is a necessary property of a tree, it verifies that the matrix of path length and the matrix of relative distance can both be represented as trees. In fact, it was first used to verify that the matrix d could be encoded by a tree since it was already used before introducing the matrix τ in the previous models but, is also handy for the definition of the matrix τ.

4.4.5 Independence of conditions

An important remark about the conditions on Θ is that they are independent of each other. This can be shown with the following three examples:

$$\begin{pmatrix} 0 & 3 & 3 & 3 \\ 3 & 0 & 3 & 3 \\ 3 & 3 & 0 & 3 \\ 3 & 3 & 3 & 3 \end{pmatrix} \tag{24}$$

$$\begin{pmatrix} 0 & 5 & 5 & 5 & 4 & 5 \\ 5 & 0 & 2 & 2 & 3 & 4 \\ 5 & 2 & 0 & 2 & 3 & 4 \\ 5 & 2 & 2 & 0 & 3 & 4 \\ 4 & 3 & 3 & 3 & 0 & 4 \\ 5 & 4 & 4 & 4 & 4 & 0 \end{pmatrix} \tag{25}$$

$$\begin{pmatrix} 0 & 3 & 5 & 5 & 2 & 4 \\ 3 & 0 & 3 & 4 & 4 & 3 \\ 5 & 3 & 0 & 2 & 5 & 4 \\ 5 & 4 & 2 & 0 & 5 & 3 \\ 2 & 4 & 5 & 5 & 0 & 3 \\ 4 & 3 & 4 & 3 & 3 & 0 \end{pmatrix} \tag{26}$$

The three previous matrices all satisfy all the different conditions stated above except one. Indeed matrix (24) satisfies all the properties except the phylogenetic manifold (21):

$$\sum_{i\in\Gamma}\sum_{j\in\Gamma\setminus i} \tau_{ij}2^{-\tau_{ij}} = \frac{15}{2} \neq 7 = (2n-3)$$

matrix (25) satisfies the properties but not the Kraft equality (20):

$$\sum_{j=1\ ,j\neq 1}^{n} 2^{-\tau_{1j}} = 0.1875 \neq \frac{1}{2}$$

and the property that matrix (26) does not satisfy is the triangular inequality (22):

$$\tau_{12} + \tau_{23} - \tau_{13} = 1 < 2$$

The fact that all the properties are independent implies that there is no overlap between the conditions. This will be important in the second section of this thesis, where we will enumerate different sets. It is essential to notice that the addition of conditions results in more restrictive sets.
This gives us the necessary state-of-the-art constraints for a matrix to be an element of Θ. However, the question that still needs to be answered is if this set of equations is sufficient to properly describe this set.

Conjecture 1 *The equations (15,16,20,21,23) are necessary and sufficient to characterize Θ the set of path-length matrices encoding phylogenetic trees.*

This conjecture is still open, and one part of this thesis will try to give a partial answer to it. The question that still needs an answer to prove this conjecture is whether Buneman's four-point condition applied only to the leaves of a tree results in the same acyclicality of a tree.

4.5 Interesting properties of model

4.5.1 statistical consistency

One essential property that the BMEP holds is called statistical consistency [40, 66].

In statistics, an estimator is said to be statistically consistent if the value it produces as an estimate of a parameter converges in probability to the true value of the parameter as the sample size used to compute the estimate increases. As the number of data points used to compute the estimate increases, the estimate becomes more accurate and less variable.

For example, suppose we want to estimate the mean of a population using a sample of data. If we use a statistically consistent estimator for the mean, then as the sample size increases, the estimate of the mean produced by the estimator will become more and more accurate and less and less variable. This is because the estimator is able to accurately capture the true mean of the population as the number of data points used to compute the estimate increases.

Statistical consistency is a desirable property of an estimator because it means that the estimator will become more accurate as the sample size increases. This is important in many applications where it is essential to have accurate estimates of population parameters.

So applied to the BMEP, we know that the more taxa are used, the closer the optimal solution given by the model will be to the actual/real phylogenetic tree. This alone provides an excellent reason to study this model.

Definition 15 *A real phylogeny is seen as the phylogeny that describes the real evolutionary process of taxa occurring in nature and the true phylogeny with respect to the considered criterion as the phylogeny that one would obtain if all molecular data from taxa were available [58, 64].*

The balanced minimum evolution is an improvement of the minimum evolution problem under ordinary least squares (OLS) [41]. That model tried to associate weights to the edges of the phylogeny. The first problem that occurred was that for some solutions, the weights could be negative, which does not hold any significance in biology. The next try was by adding the constraint that all weights had to be positive, but that did not solve all the problems [40]. What happened in the OLS model was that certain edges of a phylogeny may be weighted more than others, and this fact, in general, is not supported by any biological evidence or rationale.

This led the researchers to the balanced minimum evolution problem (BMEP). The attribute "balanced" assigned to the Problem indicates that the number of times each evolutionary distance appears in the summation of all edge weights is normalized by reflecting that all of the edge weights are weighted similarly. The optimal tree produced by the BMEP is the one with the shortest balanced minimum evolution length and its edge weights are always positive whenever the distance matrix $\mathbf{d}$ is an actual distance matrix (satisfies the triangular equality).

4.5.2 NP-hardness

In computer science, there are different classifications of problems based on their computational complexity, P, NP, and NP-hard. These classifications are used to describe the difficulty of solving a problem and the number of resources (such as time and space) required to solve it.

P is the class of problems that can be solved efficiently by a computer, meaning that an algorithm exists for solving the problem that runs in polynomial time. Polynomial time means that the algorithm's running time is a polynomial function of the input data size (for example, $O(20n^2+5n)$). Examples of problems in the P class include sorting algorithms, linear search, and matrix multiplication.

NP is a class of problems for which a proposed solution can be checked efficiently by a computer, but it is unknown whether an efficient algorithm for finding a solution exists. In other words, verifying a solution to an NP problem is easy, but it is not known whether it is easy to find a solution. Examples of NP problems include the traveling salesman problem.

NP-hard is a class of problems that are at least as hard as the hardest problems in NP. In other words, if an NP-hard problem can be solved efficiently, then all problems in NP can also be solved efficiently. However, it is unknown whether this is possible, and many researchers believe it is not. NP-hard problems are considered some of the most difficult problems in computer science.

A famous question in computer science theory is to prove P=NP In other words, the question is whether a problem can be in both the P and NP classes.

If P=NP, it would mean that all NP problems can be solved efficiently and that there is an efficient algorithm for finding a solution to every problem in NP. This would have significant implications for various problems in computer science and other fields and could potentially lead to the development of more efficient algorithms for solving various problems.

On the other hand, if $P \neq NP$, which is widely believed, it would mean that there exist problems for which it is easy to verify a solution but hard to find a solution and that there is a fundamental limit to the efficiency of algorithms for solving these problems.

In the case of the BMEP, it has been proven that it is an NP-hard problem; a problem difficult to find and verify whether an answer is a solution [67].

5 Branch and Bound

In this first section, we will implement a first algorithm that finds the optimal solution of the BMEP, i.e., finding the phylogenetic tree that minimizes the length function.
As a reminder, the BMEP problem is stated as follows:

Given a set Γ of taxa, find an phylogeny T $\in \mathcal{T}$ having Γ as leaf set and minimizing the following length function:

$$\min \qquad L(T) = \sum_{i \in \Gamma} \sum_{j \in \Gamma \setminus i} \frac{d_{ij}}{2^{\tau_{ij}}} \tag{27}$$

The approach to solve this is by using a branch and bound paradigm.

5.1 General conception of a branch and bound algorithm

A Branch and bound algorithm is a method for solving optimization problems, such as the traveling salesman person. In that problem, the salesperson must visit several cities, each at a certain distance from one other. The goal is to find the shortest route that visits each city exactly once and returns to the starting city.

Here is an example of a branch and bound algorithm:

One way to solve the traveling salesperson problem using branch and bound is to divide the problem into smaller subproblems. For example, we could divide the cities into groups: North, South, East, and West. Then, for each group, we could find the shortest possible route that visits each city in that group exactly once and returns to the starting city.

Next, we would combine the solutions to the subproblems to find the shortest possible route for the entire problem. For example, if the North group has cities A, B, and C, and the South group has cities D, E, and F, then the shortest possible route for the entire problem would be the shortest route that visits A, B, and C in the North group, followed by the shortest route that visits D, E, and F in the South group, followed by the route between the last city in the North group and the first city in the South group.

By dividing the problem into smaller subproblems and then combining the solutions to those subproblems, we can often find the optimal solution to the traveling salesperson problem more quickly than if we had tried to solve the entire problem at once. This is the key idea behind the branch and bound algorithms.

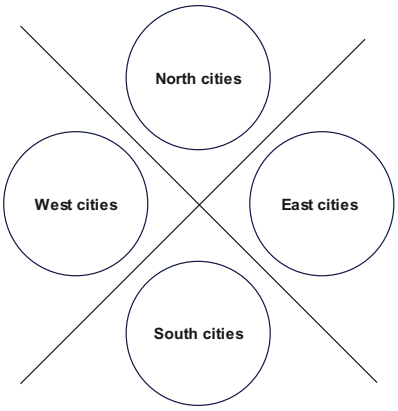


Figure 9: Traveling salesperson problem

5.2 Computational limitation of the problem

The first limitation of the problem is due to the floating point error (ϵ) of a computer. It is the difference between the smallest possible representable number and the value that is actually represents. Since we are working with small numbers, $2^{-\tau_{ij}}$, it is essential to understand the highest path length the computer can represent without any loss of information.
In C++, the machine epsilon for a value stored as a double is $2.22045 * 10^{-16}$ [68]. With the following development, we will check the highest amount of taxa we can process.

$$2^{-\tau_{ij}} \geq 2.22045 * 10^{-16}$$

$$-\tau_{ij} \geq \log_2(2.22045 * 10^{-16}) = -16 \log_2(2.22045 * 10)$$

$$-\tau_{ij} \geq -16 \log_2(2.22045 * 10)$$

$$\tau_{ij} \leq 71.39$$

So by design, in the best world possible, without any time limitations, the problem would be able to optimal tree with taxa separated by at most 71 edges. All the taxa lying beyond that threshold would be disregarded. It would be as if the computer did not see them and did not take them into account. This would take away the idea of the program, which is finding the optimal solution to the problem. [67]
However, other models can work with more taxa, but they give an approximation of the solution, not the optimal one.

5.3 Branch and bound algorithm for BMEP

The general idea of solving the BMEP with a branch and bound algorithm is progressively constructing trees starting from the 3-star topology, which is the origin of all phylogenetic trees we are exploring. We will progressively explore

all its descendants and asses at each step if the descendants of the current tree have the potential to give a lower objective value than the one already found. We can add a new taxon to extend a current tree on every edge. We have seen that for a tree with h taxa, there are (2h-3) edges in the graph.

An important property that we want for this method is that every tree we explore is unique so that there is no redundancy or overlapping. The method of starting from the three-star tree and expanding a taxon to every different edge and doing this for all the intermediate trees we encounter has the property of having a unique sequence of addition of taxa.
Note that adding a taxon k to an edge e creates a new leaf with taxon k at its end, splitting edge e into two new edges. This only creates bifurcating trees since we are adding taxa to the interior of the tree; it will never be added to an existing leaf.

To assess the potential of the descendants of a tree, we have to understand how adding a taxon to a tree changes the balanced length of the tree. If we can deduct from this how to formulate a lower bound that represents the minimum balanced length for the complete topologies (finding a way to see how adding all the taxa will change the objective value), this algorithm will work. Note that a complete topology is a topology that includes all of the taxa in the set Γ. This explanation will be done in the following sections by introducing two different bounds that can be used.
The general idea is to be able to express a bound such that:

$$L(T^+) \geq \beta(T)$$

Where T^+ represents all the complete trees that can be constructed with the tree T as its root tree and $\beta(T)$ the lower bound associated with the tree T.
When the bound $\beta(T)$ is calculated, it is compared with the best optimal value already found among the complete topologies encountered; let's call it L^*. If the lower bound for T is higher than that optimal value previously calculated ($L^* > \beta(T)$), then every complete tree descendent of T will have a higher objective value than the best tree already found. Therefore there is no reason to explore all the different cases underlying the tree T. We can then save the calculation of these trees. In the branch and bound terminology, we say that this part of the problem is pruned.

The last part to discuss is how we add the taxa. There are two different ways to do it. The first is known as the static order, and the second is the dynamic order. The static order is when we already know when each taxon will be added to the tree before starting the expansion of the trees. The dynamic order is when we calculate which next taxon will be added to the tree at each step based on the current topology. Dynamic ordering might be useful to converge quicker to better result by picking the taxon that will produce the smallest change in the objective function. However, the drawback of this method is that it has to be calculated at every step of the algorithm. It might produce more prunings and, therefore, fewer calls to the main processing algorithm, but we have to calculate which taxon to add each time. This addition in the calculation does not always lead to faster results. In this thesis, we will only look at static orders.

So the first step of the algorithm is to construct a complete tree to compare its objective value with all the different topologies. To do this, we use a greedy algorithm; as explained, we start with the three-star tree and add the taxon defined in the static order to the edge where it will result in the smallest objective value for the next tree. In essence, we add the next taxon to the edge on which the next tree will produce the lowest objective value. This operation is repeated until the first tree is complete (the n taxa are present in the tree).

Algorithm 1 First tree (d)

construct the 3-star tree
for $i = 4$ `to n` **do**
    add taxon i to the edge s.t. L(T^{+1}) is smallest
end for
return $L(T)$

Now that we have the first completed tree, the question of how to process the rest of the trees to find the exact result remains. The general approach is shown in the algorithm below 2.

Algorithm 2 Branch and bound (d)

Ł* = First tree(d)
construct 3 star tree
if T contains all taxa in $\{1, 2, \ldots, n\}$ **then**
    Calculate L(T)
    if $L(T) < L^*$ **then**
        $L^* = L(T)$
        $\theta = \{\tau(T)\}$
    end if
else
    Calculate lower bound $\beta(T, d)$
    if $\beta(T, d) > L^*$ **then**
        Remove the last taxon
    else
        choose edge to add next taxon
        add next taxon to the edge
    end if
end if

What the algorithm tries to achieve is to select potential new optimal trees. A visual representation of the aim is shown in Figure 10. Suppose we have to partial trees (trees that do not have all the taxa as leaves) T_1 and T_2. If we can find a lower bound based on the current tree, we are able to assess the potentiality that one of their complete subtrees will produce a better solution than the one currently found. In the example, the lowest possible length value of T_1 is lower than the value currently stored. This means that one of its descendants might lead to a new better solution, which is not assured. In contrast, for the graph T_2, even in the best possible scenario, none of its descendants will produce a better result than what is already found; thus, there is no reason to explore this tree further. There are two ways to improve an algorithm like that, firstly,

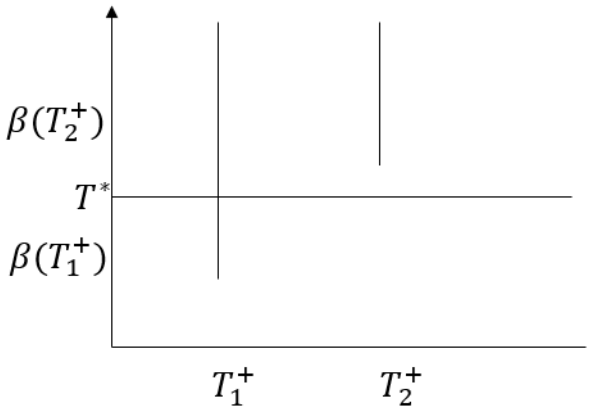


Figure 10: Representation of the use of the lower bound for the algorithm

finding the lowest upper bound T^*, and secondly, finding the best estimate $\beta(T)$ for a lower bound. In the following sections, we will see how this lower bound $\beta(T)$ can be found.

5.4 Adding a taxon to a tree

To come up with a lower bound, the first step is to understand how adding a taxon affects the balanced length of the tree. In the BMEP, adding a taxon can decrease the length function associated with a partial tree. So adding a taxon does not always result in a bigger objective function. Therefore, we cannot use the objective value of the current phylogenetic tree as a lower bound. We will now look at what happens when a taxon is added to a tree. This will introduce two lower bounds used for the branch and bound algorithm.

Before getting into the details, we have to introduce the concept of clades and how they will be used in the rest of the chapter. A phylogenetic tree can be divided into subtrees called clades.
A clade in a topology T is defined as follows [69]:

Definition 16 *It is any set of taxa lying on one side of a branch in T.*

Figure 11 gives a visual representation of different clades in a tree.
The BMEP has a property that is stated as follows [69]:

Theorem 4 *Suppose the taxa in a phylogeny T can be partitioned in $X_1, X_2, \ldots, X_m$ where each X_i is a clade of T. Then*

$$L(T) = \sum_{h=1}^{m} L(X_h) + \sum_{1 \le h \le k \le m} d_{X_h X_k} \prod_{v \in P_{hk}} \frac{1}{deg(v) - 1}$$

Where

$$L(X_h) = \sum_{i \in \Gamma(X_h)} \sum_{j \in \Gamma(X_h) \setminus j} \frac{d_{ij}}{2^{\tau_{ij}}}$$

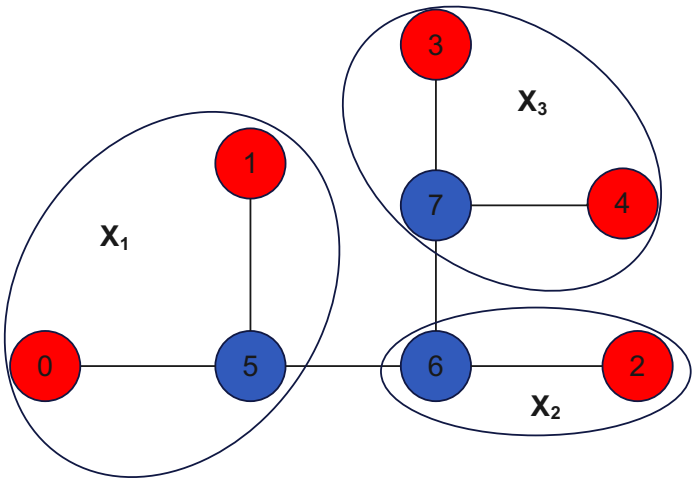


Figure 11: Three different possibilities of clades

$$d_{X_h X_k} = \sum_{i \in X_h} \sum_{j \in X_k} d_{ij} \prod_{v \in P_{ij} - P_{hk}} \frac{1}{deg(v) - 1}$$

and P_{hk} represents the edges that separate the root of clades X_h and X_k.
If we denote T, the current phylogenetic tree, then $T\left|\frac{A}{B}\right. k$ is the tree produced by the addition of the taxon k between the clades A and B of the tree T.
The notation τ_{iX} represents the distance between taxon i and the root of clade X. For example, the distance τ_{3B} in figure 12a between the taxa number 3 and the vertex 6 equals 2.

Suppose we were to construct the sequences τ_{iA} and τ_{iB} based on the tree represented in Figure 12a. They would represent the number of edges separating taxon i and the root of the clade. The root of clade A would be vertex 5 and for clade B, it would be vertex 6. It would give the following result:

$$\tau_{iA} = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad \tau_{iB} = \begin{bmatrix} 2 & 2 & 1 \end{bmatrix}$$

From the theorem 4 applied to two clades, where the two clades are separated by the edge where the next taxon will be added (12a as an example), the length function can be expressed as:

$$L(T) = L(A) + L(B) + d_{AB}$$

and

$$L(T\left|\frac{A}{B}\right. k) = L(A) + L(B) + \frac{1}{2}d_{AB} + \frac{1}{2}d_{A\{k\}} + \frac{1}{2}d_{B\{k\}}$$

Therefore the change in length by adding taxon k between the clades A and B is given by:

$$L(T\left|\frac{A}{B}\right. k) - L(T) = \frac{1}{2}(d_{A\{k\}} + d_{B\{k\}} - d_{AB}) \tag{28}$$

Where

$$d_{A\{k\}} = \sum_{i\in A} \frac{1}{2^{\tau_{iA}}} d_{ik} = \sum_{i\in A} \frac{1}{2^{\tau_{iA}}} d_{ik} \sum_{j\in B} \frac{1}{2^{\tau_{jB}}} = \sum_{i\in A}\sum_{j\in B} \frac{1}{2^{\tau_{iA}+\tau_{jB}}} d_{ik}$$

$$d_{B\{k\}} = \sum_{j\in B} \frac{1}{2^{\tau_{jB}}} d_{ik} = \sum_{j\in B} \frac{1}{2^{\tau_{jB}}} d_{jk} \sum_{i\in A} \frac{1}{2^{\tau_{iA}}} = \sum_{i\in A}\sum_{j\in B} \frac{1}{2^{\tau_{iA}+\tau_{jB}}} d_{jk}$$

$$d_{AB} = \sum_{i\in A}\sum_{j\in B} \frac{1}{2^{\tau_{iA}}} \frac{1}{2^{\tau_{jB}}} d_{ij} = \sum_{i\in A}\sum_{j\in B} \frac{1}{2^{\tau_{iA}+\tau_{jB}}} d_{ij}$$

By substituting these equations in the equation 28 we obtain:

$$L(T\,|^{A}_{B}\,k) - L(T) = \frac{1}{2} \sum_{i\in A,\ j\in B} \frac{1}{2^{\tau_{iA}+\tau_{iB}}} (d_{ik} + d_{jk} - d_{ij}) \tag{29}$$

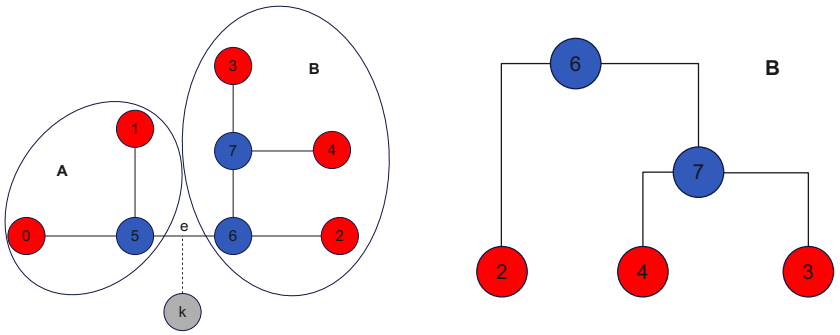
By introducing the notation

$$\lambda_k^{ij} = \frac{1}{2}(d_{ik} + d_{jk} - d_{ij})$$

We can simplify the equation 29 and it becomes:

$$L(T\,|^{A}_{B}\,k) - L(T) = \sum_{i\in A,\ j\in B} \frac{1}{2^{\tau_{iA}+\tau_{iB}}} \lambda_k^{ij} \tag{30}$$

Since every clade is a rooted binary tree, the sequence τ_{iA} satisfies the original Kraft equality. This comes from a property of Huffman codes applied to Rooted binary trees, as explained in Chapter 3. We explained how to change from Kraft equality, based initially on rooted binary trees, to an equivalent property applied to unrooted binary trees. In our case, we use the original equality, coming from [63]. An example can be seen in Figure 12 where we can see that the clade B in Figure 12b is a rooted binary tree.



(a) Topolgy T divided into two clades

(b) Clade B seen as a rooted binary tree

Figure 12

Therefore the following equation holds:

$$\sum_{i\in X} \frac{1}{2^{\tau_{iX}}} = 1$$

When summed, the factors multiplying λ_k^{ij} are equal to 1.

$$(\sum_{i\in A}\frac{1}{2^{\tau_{iA}}})(\sum_{j\in B}\frac{1}{2^{\tau_{iB}}}) = \sum_{i\in A,\, j\in B}\frac{1}{2^{\tau_{iA}+\tau_{iB}}} = 1$$

This means that expression (30) can be seen as a weighted average of the terms λ_k^{ij}.
We have seen how adding one taxon changes the length function of the problem. On this basis, it is possible to expand the addition of any number of taxa. This makes the Branch and bound algorithm applicable to this problem.
The two following bounds that will be explained will be applied in a static ordering heuristic. This means that the order in which each taxon will be added to the tree is predefined. This reduces the computation needed at each step of the program. The other method is called dynamic order when we decide which will be the next taxon to add at each step. Dynamic ordering might produce more "prunings" and converge to the optimal solution in fewer iterations, but each iteration is more costly. In this thesis, we only consider static ordering because it is easier to implement and both methods converge in approximately the same time complexity [69].

If we had to use the same bounds for dynamic ordering, the central assumption that is required is to assume that all the taxa are already in the tree except the one that we are trying to add. Regarding the rest of the development, it is quite similar.

5.5 α as a lower bound

Since equation (30) can be seen as a weighted average of the terms λ_k^{ij}, the first bound that comes to mind is to take the minimum value among these. This simplifies the equation in the following way:

$$L(T\,|\tfrac{A}{B}\,k) - L(T) \geq \min_{i,j\in T: i\neq j}\lambda_k^{ij} \tag{31}$$

Since we work with a static order, we know when which taxa will be added to the tree. This means that depending on the number of taxa already present in the topology, we already know what taxa are present and when the other taxa will be added. This enables the algorithm to find the value $\min_{i,j\in T:i\neq j}\lambda_k^{ij}$ as soon as the order is available.
If we assume that the order is the following: we add taxon 4, then taxon 5, then the sixth, and so on until the tree is complete. Then we can add taxon k ($4 \leq k \leq n$) onto the partial topology T. In that case the set of taxa in T is $\{1, 2, \ldots, k-1\}$ and we can rewrite $\min_{i,j\in T:i\neq j}\lambda_k^{ij}$ as:

$$\alpha_k = \min_{i,j:1<i<j<k}\lambda_k^{ij} \tag{32}$$

Adding taxon k will at least change the objective function by α_k. If we denote T^+ the subspace containing all the complete topologies of T, the next equation can be deduced:

$$L(T^+) \geq L(T) + \sum_{k\notin T}\alpha_k \tag{33}$$

As explained before, the computation of all the α_k can be done before the branch and bound process because there is no need for any information regarding the topology T to which the taxon k is added. The only need is to know which taxa are included in the topology and which taxon will be added next. This is possible because the static order gives us this information. Based on the number of taxa in the tree, we know which taxa the tree is composed of and what the next taxon is added.

5.6 γ as a lower bound

By taking the minimum of λ_k^{ij} in the α_k bound, we are sure that the value is lower than the weighted average of the different λ_k^{ij}. In this part, we will explore another tighter bound than the first one. This is done by considering more terms in the average and by understanding the different topologies of A and B, the clades where the taxon k is placed in between.
However, an essential property that has to be kept is that the bound must be independent of the clades since, to be efficient, the bound has to be computed easily without knowing the topology of the tree T without the taxon k. The only information we have at our disposition is the order in which the taxon is added. We use the same assumption as in the last lower bound: we assume that the taxa are numbered 1,2,...,n following the predefined static order. We, therefore, can assume that the taxon k will be added to a topology T, including the set of taxa $\{1, 2, \ldots, k-1\}$.
The concept behind this bound is to find the worst case (where the change in the balanced length is the smallest, whatever the current topology is). To express it clearly, the following definition has to be added:

Definition 17 *Denote by $\lambda_k^{(h)}$ the h-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.*

There are only a limited number of ways to add a taxon to a topology. Let's explore the different possibilities by looking at the alternatives for trees with an increasing number of taxa.

For three taxa, there is only one way a taxon can be added: on an edge separating a taxon from the central vertex. This is an easy case; there is only one way to add a taxon. Depending on which edge the taxon is added, the sum of the weights in that multiply λ_k^{ij} terms are either equal to:

$$\frac{1}{2}\lambda_4^{01} + \frac{1}{2}\lambda_4^{02}$$

$$\frac{1}{2}\lambda_4^{01} + \frac{1}{2}\lambda_4^{12}$$

$$\frac{1}{2}\lambda_4^{02} + \frac{1}{2}\lambda_4^{12}$$

The first equation is when the fourth taxon is added to the branch e_1, the second is when it is added to the edge e_2 and the third is for e_3. Since we are ordering the λ_k^{ij} by smallest to greatest, we can express the minimum among these values as:

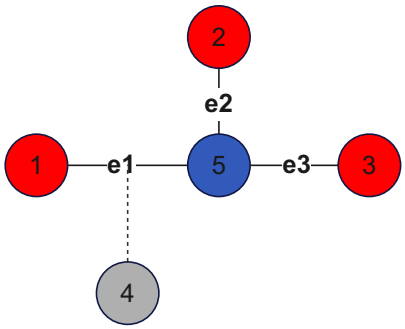


Figure 13: Addition of a taxon to the three-star phylogeny

$$\frac{1}{2}\lambda_3^{(1)} + \frac{1}{2}\lambda_3^{(2)} \tag{34}$$

Doing this gives an optimally tight bound independent of where the fourth taxon is added.

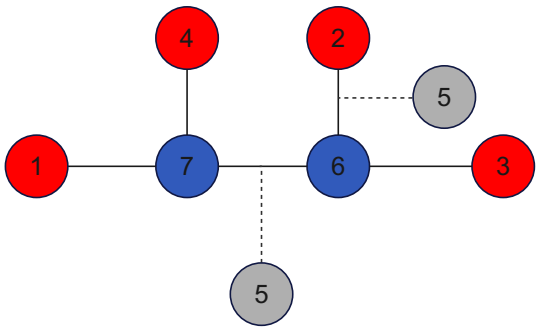


Figure 14: Addition of the fifth taxon to the phylogeny

When looking at how to add the fifth taxon, adding the taxa can define clades in two different ways. Either it splits the current topology into two similar clades, or it creates a clade with three taxa on one side and one on the other (see Figure 14).
If the fifth taxon is added to the internal edge between the two vertices, the change in balanced length is the following:

$$\frac{1}{4}\lambda_5^{i_1 j_1} + \frac{1}{4}\lambda_5^{i_1 j_2} + \frac{1}{4}\lambda_5^{i_2 j_1} + \frac{1}{4}\lambda_5^{i_2 j_2}$$

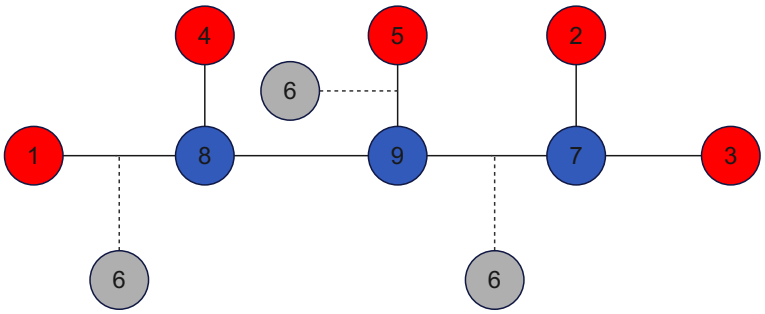


Figure 15: Addition of the sixth taxon to the phylogeny

If the taxon is added to an edge that links a vertex and its taxon, then we have:

$$\frac{1}{2}\lambda_5^{ij_1} + \frac{1}{4}\lambda_5^{ij_2} + \frac{1}{4}\lambda_5^{ij_3}$$

We can easily see that a lower bound for the fifth taxon is given by:

$$\frac{1}{2}\lambda_5^{(1)} + \frac{1}{4}\lambda_5^{(2)} + \frac{1}{4}\lambda_5^{(3)}$$

For the case where we add the sixth taxon, there is only one unlabelled tree [41], which is represented by the comb with five taxa, a case still easy to check. Figure 15 represents the three ways the sixth taxon can be added to the tree. There are two edges where the clades A and B have, respectively, two and three taxa in their set, these edges are the two that link the central vertex with the other vertex. An addition in this situation gives a change of balance of:

$$\frac{1}{8}\lambda_6^{i_1j_1} + \frac{1}{8}\lambda_6^{i_1j_2} + \frac{1}{4}\lambda_6^{i_1j_3} + \frac{1}{8}\lambda_6^{i_2j_1} + \frac{1}{8}\lambda_6^{i_2j_2} + \frac{1}{4}\lambda_6^{i_2j_3}$$

The second case is when the taxon is added to an edge linking a taxon to a vertex on the side of the tree. Another way to see it is by adding a taxon to a terminal branch except for the one that links the central vertex and its taxon. This divides the graph into two clades of one and four taxa. In this case, we get the next equation:

$$\frac{1}{2}\lambda_6^{ij_1} + \frac{1}{4}\lambda_6^{ij_2} + \frac{1}{8}\lambda_6^{ij_3} + \frac{1}{8}\lambda_6^{ij_4}$$

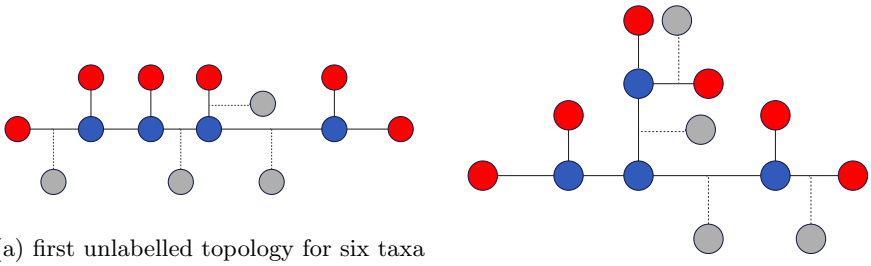
The last case happens when the sixth taxon is added to the edge connecting the central vertex and its taxon. This also divides the two clades into sets, where they include one and four taxa. The difference lies in how they are connected to the root of the clade. This case gives the following result:

$$\frac{1}{4}\lambda_6^{ij_1} + \frac{1}{4}\lambda_6^{ij_2} + \frac{1}{4}\lambda_6^{ij_3} + \frac{1}{4}\lambda_6^{ij_4}$$

These are the three different possibilities to add a taxon to a five-taxa topology. We can see that for the addition of the fifth taxon, the lower bound is:

$$\frac{1}{2}\lambda_6^{(1)} + \frac{1}{4}\lambda_6^{(2)} + \frac{1}{8}\lambda_6^{(3)} + \frac{1}{8}\lambda_6^{(4)} \tag{35}$$

Looking at the seventh taxon takes more time since we have to look at all the



(a) first unlabelled topology for six taxa

(b) second unlabelled topology for six taxa

Figure 16: Addition of the seventh taxon to the phylogeny

different cases that it can be done on a six-taxa tree. For this kind of tree, there are two different unlabelled trees to which the seventh taxon can be placed (see Figure **??**). If we were to check every possibility, there would be eight cases to analyze. We can already understand intuitively from the previous developments what the lower bound will ultimately be. However, the resulting bound for the seventh taxon would be:

$$\frac{1}{2}\lambda_7^{(1)} + \frac{1}{4}\lambda_7^{(2)} + \frac{1}{8}\lambda_7^{(3)} + \frac{1}{16}\lambda_7^{(4)} + \frac{1}{16}\lambda_7^{(5)} \tag{36}$$

We notice from this that the weight associated with the λ_k^{ij} always happens in the same topology and at the same place where the next taxon is added. Proving this enables the bound to work without taking into account the place the taxon is added and without taking into account the topology to which the taxon is added.

The general idea of finding this lower bound is multiplying the smallest λ_k^{ij} with the largest possible weight while keeping the number of additions the smallest possible. This always happens when the taxon is added on one of the edges of an extreme taxon of the "comb" topology 17.

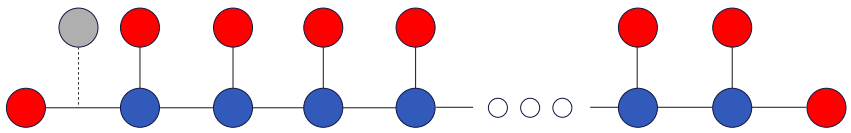


Figure 17: Representation of the place and topology where the taxon is added for the γ bound

Based on this intuition, we can derive a second lower bound which is tighter than the first introduced.

Proposition 2 *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 the two clades separated by exactly one branch in T. Then,*

$$L(T \left|\frac{A}{B}\right. k) - L(T) \geq \sum_{i=1}^{k-3} \frac{1}{2^i} \lambda_k^{(i)} + \frac{1}{2^{k-3}} \lambda_k^{(k-2)} \tag{37}$$

There is a formal proof of this proposition provided in [69], but it is lengthy and unnecessary for this thesis's sake. Getting into the details of it does not provide any relevant information for this thesis.
Now define this bound as γ_k, which is:

$$\gamma_k = \sum_{i=1}^{k-3} \frac{1}{2^i} \lambda_k^{(i)} + \frac{1}{2^{k-3}} \lambda_k^{(k-2)} \tag{38}$$

It is clear that this bound is tighter than the first introduced, simply because $\gamma_k \geq \lambda_k^{(1)} = \alpha_k$ and will help converge to the result quicker by bringing more prunings in the algorithm. It, however, keeps the important property of being able to be calculated before the process of the main algorithm. We can therefore express a bound for a complete extension of a tree T^+:

$$L(T^+) \geq L(T) + \sum_{k \notin T} \gamma_k \tag{39}$$

This explains the two known bounds we can use for the branch and bound algorithm.

5.7 Data structure and algorithm

We now have all the aspects needed to implement the branch and bound. Let's now take a deeper look at the algorithm. The first thing to explain is the representation of a tree T in the program. The only thing in which we are interested in is the edges of the tree and what nodes they connect. Therefore, we only store the edge set in the program. In the array that defines tree a tree with h taxa, the first h elements of the arrays are the edges that connect the taxa to their vertex, and the h-2 following elements are the edges that connect the vertices in between them. An example of the data structure is shown in Figure 18.

The three pieces of information needed for the algorithm are the number of taxa already in the graph, the past edges to which the previous taxa were added, and the edge set. With this base of data, we can access every phylogenetic tree. As a reminder, the lower bounds α and/or γ are calculated before the algorithm's processing. For each
The complete algorithm works as shown in algorithm 3.

This approach makes a parallelization job relatively easy. What needs to be done to do that is to define the starting point and ending point of each processor. For example, if there were three processors, each processor could explore all the subtrees of the direct descendants of the 3-star tree. If they were 15 processors, they could explore all the second generations of the original tree.

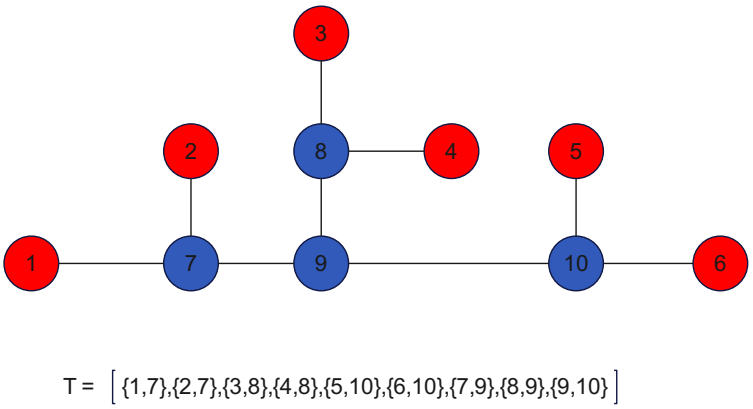


Figure 18: Example of the data structure

Suppose there is a number between these boundaries without overthinking too much about the load distribution among the processors. In that case, some will explore multiple second generations of subtrees some will only explore one.

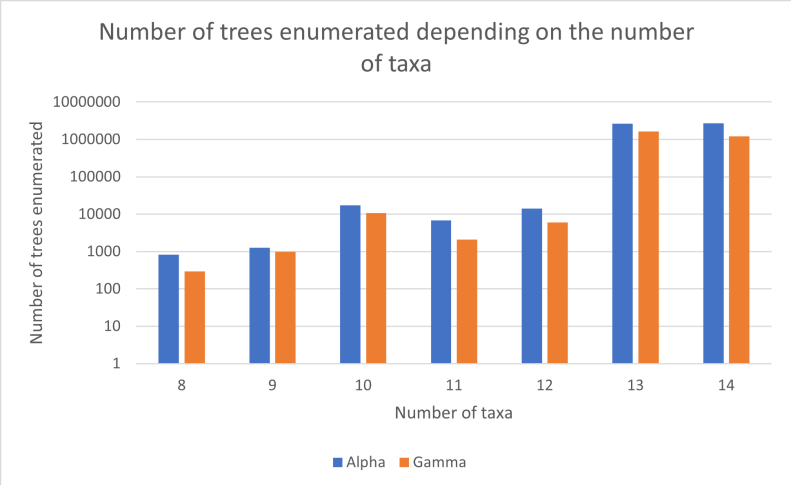


Figure 19: Number of complete trees enumerated

On both of the the Figures 19 and20, we used a logarithmic scale, so even if the differences between the two lower bounds seem to be quite similar, the gamma bounds consitently converges to the same optimal value twice as fast as with the alpha bound. We also only enumerate half of the complete trees, this is where the time difference comes from.

Algorithm 3 Branch and bound (d,α)

```
L* = First tree(d)
construct the 3 star tree
if T contains all taxa in {1, 2, ..., n} then
    compute τ(T)
    calculate L(τ, d)
    if L(T) < L* then
        L* = L(T)
        θ = {τ(T)}
    end if
else if Edge[current taxa] ≤ max number of edges then
    compute τ(T)
    calculate L(τ, d)
    β = L(τ, d) + α_current taxa
    if β > L* then
        Remove the last taxon
        edges[current taxa -1] = edges[current taxa -1] + 1
    else if current taxa != 2 then
        add next taxon
        Edge[current taxon + 1] = 0
    else if current taxon == 2 then
        Return T*
    end if
else
    remove last taxon
end if
```

We see from this that implementing tighter lower bounds really improves the time complexity. If we were able to produce even tighter ones, we could process more and more data. However, this implementation of the branch and bound algorithm already takes up to 21 minutes to process 14 taxa. Since this time grows exponentially (linear relationship on a logarithmic scale) further increasing the size of the problem will soon lead to impossible computational time.

5.8 Improvement of the code

At the moment, the most significant improvement in the temporal complexity that can be made is to improve the construction of the matrix τ based on the edge set. Currently, it works with a depth-first search (DFS) algorithm.
A depth-first search (DFS) algorithm traverses a graph or tree data structure. It involves searching through a node's children before exploring its siblings. In other words, it explores as far as possible along each branch before backtracking. The algorithm starts at the root node and explores as far as possible along each branch before backtracking. It continues this process until all the nodes in the graph have been explored. The computational cost of this method to build the matrix τ is $O(n^3)$.
To get a better result, one analysis that could be interesting is to improve the lower bound that is initially found by the greedy algorithm. One idea is, instead

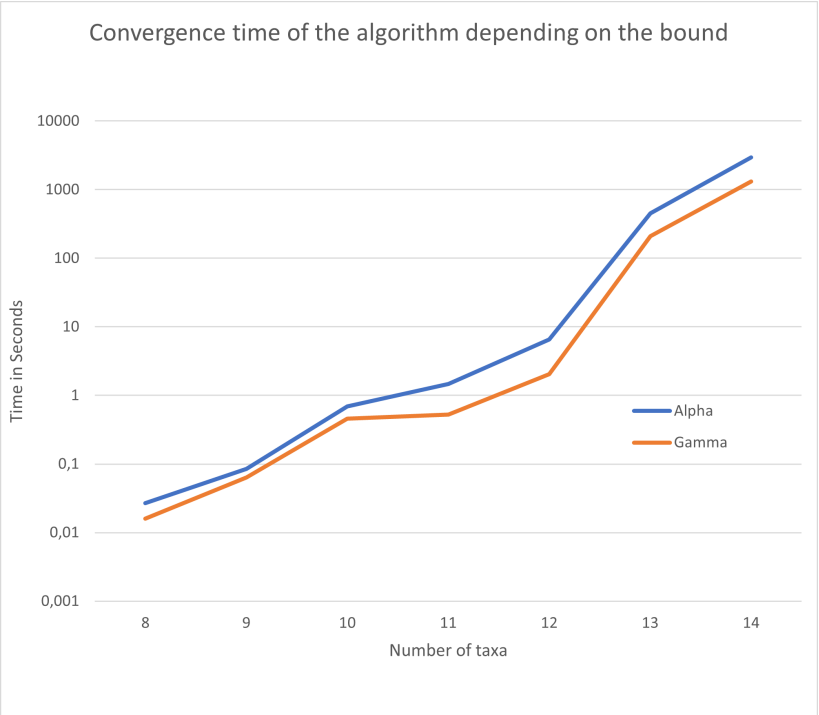


Figure 20: Time of convergence

of using the exact value found, to use a discounted value, for example, using the value $0.95 * L(T)$ instead of L(T). The only problem is that it could possibly lead to no optimal value if the exact solution is in between $L(T)$ and $0.95 * L(T)$. In that case, changing the coefficient or keeping it as original could not lead to better results and is more of a case-by-case. Doing so would however lead to more pruning, it would give a better approximation to the first lower objective function.

For each taxon, it looks at how far each other taxon is located. The improvement that can be made is to reuse previous calculations to reduce future ones. This is the function that takes the most amount of time in the branch and bound process.

There are two different methods to change this part of the execution of the algorithm completely. The first method is to store the trees as a prufer code and then implement the optimal algorithm that reconstructs the path-length matrix explained in this paper [70].

The second approach would be to pre-calculate all the different clades that can be produced by adding a taxon to a branch and then applying the following equation:

$$L(T\left|\frac{A}{B}\right. k) = L(T) + \frac{1}{2}(d_{A\{k\}} + d_{B\{k\}} - d_{AB}) \tag{40}$$

This enables us to skip the calculation of the path-length matrix entirely by using the balanced length of the previous tree and the change created by adding the taxon. This is the method used in the FASTME2 algorithm [45].

6 On the path of a counter-example of Θ

The second part of this thesis is to find a counter-example of the condition that the set Θ satisfies, i.e. trying to find a matrix that follows all the necessary conditions but that is not part of the set defining $\mathcal{T}$ of phylogenies. That would prove that the conditions (15,16,20,21,23) are not sufficient to characterize Θ.

This is essential because we have seen in the previous section that solving this problem for a large number of taxa is exponentially difficult. To counter this problem and try to tackle larger sets, new methods are required. Defining the set of possible solutions completely will help in the development of more advanced representations. One way to improve the branch and bound problem, as previously stated, is to formulate the problem in a way that enables an Integer linear program or mixed integer solver to function. This requires that the set of possible solutions to the problem can be expressed in a multi-dimensional polytope. The definition of Θ is key for this approach [71].

We know that for a problem with n taxa, there will be $(2n-5)!!$ different phylogenetic trees, each represented by a different path-length matrix [41, 19]. The objective is to enumerate all the different matrices that respect the necessary conditions. If we can find $(2n-5)!!+1$ matrices, this would mean that the conjecture is insufficient to define Θ completely and that at least one other constraint must be added to perfectly describe the convex hull.

At the moment there exists a theoretical proof that the conjecture holds up until eleven taxa. The proof is still being reviewed and has not yet been published. So the aim would be to reach the threshold of twelve taxa. This would require an enumeration of at least 654 729 075 12×12 matrices.

This chapter will explain how to find this counter-example. An analogy of the method would be trying to find all the prime numbers between 1 and 1 000 000. This analogy will be used during the whole chapter to give an understandable and relatable example of what is done for Θ if it were applied to the set of prime numbers.

If that approach was applied to the set of prime numbers, there would not be any sufficient way to describe a prime number. The only info there would be is the number of prime numbers to find and some known conditions for a number to be a prime number. It would be the same as if the definition of a prime number would not be the definition but a condition on the set.

A more rigorous definition of the approach is to define $\widehat{\Theta}$ as the approximation of Θ and the definition of $\widehat{\Theta}$ is composed of the 5 conditions (15,16,20,21,23). Since we know what $|\Theta|$ is equal to $(2n-5)!!$, we are going to test if $|\widehat{\Theta}|$ is composed of the same number of elements. If they are not equal, it would mean that $\widehat{\Theta}$ is not an accurate representation of Θ.

6.1 Simple approach

The first attempt to enumerate is the simplest approach which takes into account the discrete interval of each entry of $\tau_{ij} \in \{2, ..., n-1\}$, the symmetry $\tau_{ij} = \tau_{ji}$ and null diagonal $\tau_{ii} = 0$ properties of a path-length matrix.

The simple enumeration works with the same principle as a regular enumeration from 0 to 10^n but with an enumeration on a different basis. The bounds of each entry lie between 2 and $n-1$ instead of 0 and 9 compared with an enumeration on basis 10. Therefore the « zero » is composed of 2 in each entry, and then the first entry is increased from 2 to $n-1$ before increasing the second entry to 3 and then repeating the process until all the different matrices are enumerated.
That makes up for an algorithm that enumerates $(n-2)^{\frac{n(n-1)}{2}}$ matrices.
It is the same as counting from 0 to 9. Once we reach 9, we use a second digit to keep the enumeration going. We then use 10 to represent the number following 9.

The symmetry and null-diagonal properties are used to reduce the calculation time. With these properties, it is possible to represent a matrix as a vector. For a $n \times n$ matrix, the vector will be of length $\sum_{i=1}^{n} n - i$ which is equivalent to $\sum_{z=0}^{n-1} z = \frac{n(n-1)}{2}$. The first $n-1$ elements of the vector represent all the entries to the right-hand side of the diagonal of the first row, the next $n-2$ are all the right-hand size elements of the second row, and so on up until the penultimate one. Here is an example of how this translates :

$$\begin{bmatrix} 0 & 2 & 6 & 4 \\ 2 & 0 & 9 & 6 \\ 6 & 9 & 0 & 7 \\ 4 & 6 & 7 & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 2 & 6 & 4 & 9 & 6 & 7 \end{bmatrix}$$

The problem with this approach is that with each increment in the size of the matrix, the number of enumerations drastically increases since both the base and the exponent increase. Going from a matrix $n \times n$ to a $(n+1) \times (n+1)$ changes the number of matrix to enumerate from $(n-2)^{\frac{n(n-1)}{2}}$ to $(n-1)^{\frac{n(n+1)}{2}}$. The algorithm has, therefore, a time complexity of $O(n^{n^2})$, which is far from ideal.
With this approach, the time used for the different sizes of the problem is as shown in Figure 21. It struggles for matrices as soon as for 7×7 matrices, for which, even after 24 hours of running, it still had not found one valid matrix. This shows the limitation of this approach and the need for improvement.

It gives us, however, more certainty that the conditions (15,16,20,21,23) are sufficient to define properly the set Θ for at least 6 taxa but to have results for more taxa another method has to be taken into consideration.

What is interesting, however, is that many matrices that are enumerated are unnecessary and could be scrapped since the set of the relevant matrices can be reduced, which is the idea explored in the next chapter.

To link this with the prime number analogy, this approach checks all numbers

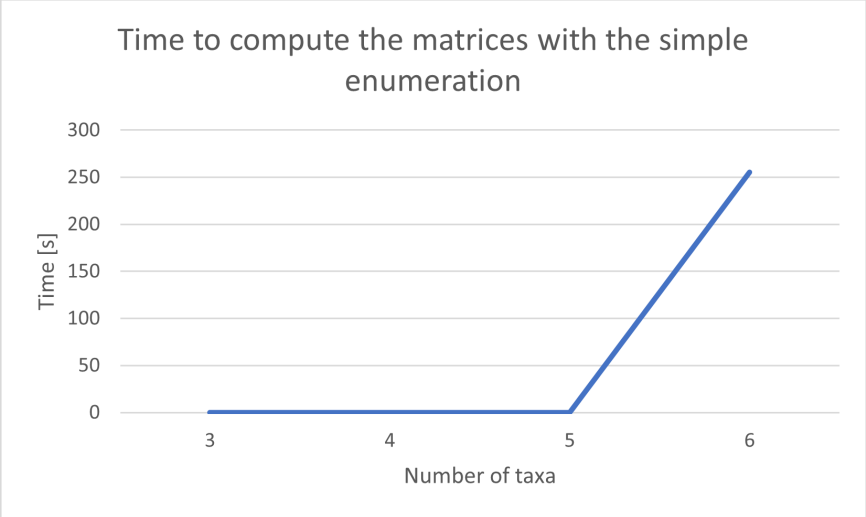


Figure 21: time taken for the simple enumeration process

between 1 and 1 000 000 and checks if every integer is a prime number.

6.2 Reduction of the cardinality of the enumerative set

This section will analyze how to improve the simple algorithm by adding a necessary property of Θ. Namely, the Kraft equality will help to reduce the cardinality of the set that will be enumerated.

This would be equivalent to enumerating all the odd numbers after 2 if we were looking at the prime numbers. The condition on prime numbers would be that except for 2, there are no other even numbers that are prime. Therefore, enumerating even numbers after 2 would be unnecessary. Reducing the enumerative set to only odd integers reduces the cardinality of the set by half.

The choice of integrating the Kraft equality first is due to its independence of row. If we are able to create all the sequences that respect the equality since each row of the matrix is composed of a variation of these sequences, we can construct the matrix row by row. This is not possible with the other equations, none of them enables the construction of a matrix. They are used to check whether or not a matrix is in the set.

The size of this new set is smaller than the one previously enumerated, simply by the fact that the intersection of spaces results in a smaller space. It is, however difficult to compute exactly the cardinality of this set, and we will therefore consider the worst-case scenario even though that will never occur.

If we consider that every combination of the permutation of all the sequences that respects the Kraft equality, which actually means all the matrices that only respect that property and not the symmetry one, it gives the following cardinality.

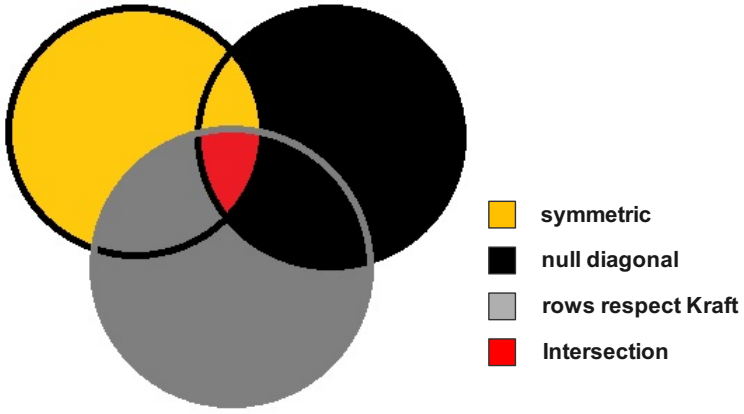


Figure 22: sets

Let's denote $m(n)$, a function of n as the number of different sequences respecting the Kraft equality for n taxa. Since each sequence has (n-1) elements in it that means that each sequence has at most $\frac{(n-1)!}{2!}$ permutations. This is the formula for permutations with repetition. That makes for a total of $m(n)\frac{(n-1)!}{2!}$ sequences It is a worst-case scenario where there is only one repetition of integer since there are at least 2 similar integers in each sequence but there can be more [63]. Then the cardinality of this set is $n * m(n) * \frac{(n-1)!}{2!} = m(n) * \frac{n!}{2!}$ since each row of the matrix can be composed of any ordered sequence or one of its permutation.

The difference between the cardinality of the first set and this one is, therefore, $(n-2)^{\frac{n(n-1)}{2}}$ for the first set and $m(n)\frac{(n)!}{2!}$ for the second. This is an improvement, it can be quickly understood by expanding each expression:

$$(n-2)^{\frac{n(n-1)}{2}} = \underbrace{(n-2) \times (n-2) \times \cdots \times (n-2)}_{\frac{n(n-1)}{2} \text{ times}} \tag{41}$$

This is for the set of the simple enumeration, whereas for the second set the cardinality is

$$m(n)\frac{n!}{2!} = (1 \times 2 \times \cdots \times n)\frac{m(n)}{2} \tag{42}$$

For larger values of n and under the assumption that $m(n)$ does not increase too much, the second set is smaller than the first.
To support the hypothesis that $m(n)$ does not grow too rapidly, that function can be approximated by $m(n) \simeq 0.148(1.791)^{(n-1)}$ for $n \leq 30$ [63].

Something to keep in mind is that not the entirety of this set will be enumerated since a lot of matrices in this set are not symmetric and the enumeration will take advantage of this property of the set Θ.

6.3 Simple algorithm including Kraft

The algorithm works in 3 different stages. Firstly it creates all the ordered path-length sequences that respect Kraft equality. Secondly, it generates and stores all the permutations of these sequences in a file. Lastly, it processes the information to create all the possible matrices.

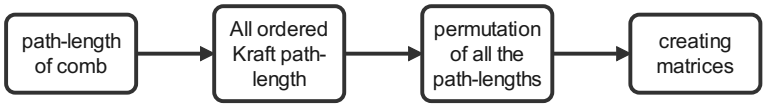


Figure 23: Kraft algorithm steps

The core idea of the algorithm comes from the fact that each row of τ is a Kraft sequence.

$$\sum_{j \in \Gamma \setminus \{i\}} 2^{-\tau_{ij}} = \frac{1}{2} \ \forall i \in \Gamma$$

When looking only at the Kraft equality and its impact on the construction of τ we see that each row must independently respect the equality. There is a way to construct all these sequences, which makes this approach feasible. How to generate these sequences is explained in the following part.

6.3.1 Kraft equality and ternary exchanges to create path-length sequences

The first step is to create all the paths-lengths that respect the Kraft equality, thanks to the work done on Huffman codes where it is stated that with ternary exchanges it is possible to generate all the sequences and by understanding the link between a Huffman code (rooted binary tree) and an unrooted binary tree it is possible to apply this research on our problem [63].

The creation of all the different sequences is done through ternary exchanges applied to an original sequence that already respects Kraft's equality, from which all the other sequences can be derived.

A ternary exchange is a process that works as follows, given any path length sequence:

$$(\cdots \ p \ \cdots\cdots \ (q+1) \ (q+1) \ \cdots)$$

then the change

$$(\cdots \ (p+1) \ (p+1) \ \cdots\cdots \ q \ \cdots)$$

is a ternary exchange.

If the first sequence is a Kraft sequence then all the resulting sequences are also Kraft sequences [63]. This can be shown by seeing that performing a ternary exchange does not affect the result of the following sum:

$$2^{-p} + 2^{-(q+1)} + 2^{-(q+1)} = 2^{-(p+1)} + 2^{-(p+1)} + 2^{-q}$$

$$2^{-p} + 2 * 2^{-q-1} = 2 * 2^{-p-1} + 2^{-q}$$

$$2^{-p} + 2^{-q} = 2^{-p} + 2^{-q}$$

and therefore it does not change the summation of the terms in the Kraft equality:

$$\sum_{j \in \Gamma \setminus \{i\}} 2^{-\tau_{ij}} \quad \forall i \in \Gamma$$

By using minimal balancing exchanges (when $p+1 = q$) and regular ternary exchanges it is possible to generate all the different sequences relevant to the problem.
Doing so results in all the **ordered** sequences that can be used to form any row of τ, this is however not yet sufficient since all the permutations of these sequences can also result in rows of τ.

Concretely, the first part of the algorithm creates all the different Kraft sequences. It starts with the path length from one of the extreme leaves to all the other taxa in the phylogenetic tree called the "comb" represented in Figure 24. For 5 taxa it gives the sequence $\{2, 3, 4, 4\}$. The complete explanation is that the shortest path to go from taxon 1 to taxon 2 is of length 2 (there are 2 edges separating them), 3 to go from 1 to 3, 4 to go from 1 to 4, and 4 to go from 1 to 5.

The general form of that first sequence for n taxa is $\{2, 3, \cdots, (n-1), (n-1)\}$. Based on this, the first part of the algorithm computes all the other sequences by using minimal balancing exchanges.

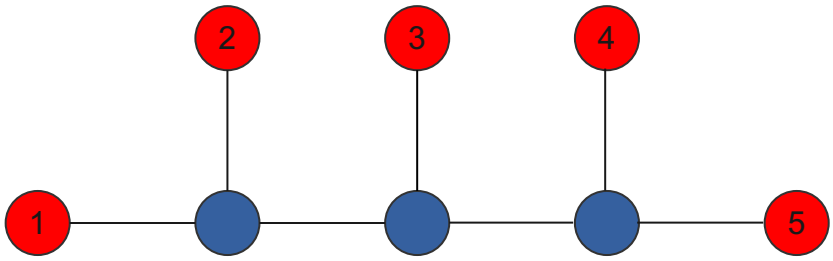


Figure 24: "comb" phylogenetic tree

Doing so creates all the unique ordered sequences of which each row of the matrix τ can be composed.

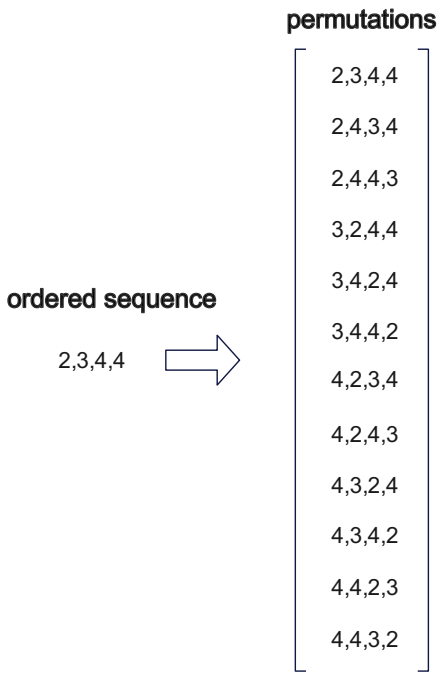


Figure 25: storage of permutations

6.3.2 Storage of permutation of the path length sequences

The next step, after having computed all the ordered path length sequences, is to compute all of their permutations. It is at this point that some reflection is needed. The first algorithmic design decision, choosing between storing the permutations or calculating them progressively without storage. That decision was quick since the larger the size of the problem, the more permutations would be calculated numerous times.

All the sequences are stored in a CSV format. The CSV (comma-separated values) format is chosen in order to be able to make the difference between the sequence of factors $[\cdots, 21, 2, \cdots]$ and the sequence $[\cdots 2, 1, 2, \cdots]$. This is important since the way a .txt is read in c++ is by returning strings. The addition of a parsing delimiter is therefore necessary.
Furthermore, the storage of permutations is done in the following way: for any originally ordered sequence, we create all its permutations in a certain order as well. For example, the sequence $\{2, 3, 4, 4\}$ will be stored as shown in Figure 25.

6.3.3 Processing of the main enumeration algorithm

The general idea of the algorithm is to construct the matrix row by row, starting at the first row. Then at each step/row, it looks what are the constraints induced by the previous rows are already in place due to the symmetry prop-

erty of Θ. If it finds a sequence whose beginning corresponds to the constraint, the sequence is then added to the matrix. If there are none, it looks at the previous row and checks if there is any other sequence that can replace the one that is already in place. That is the general scheme of the algorithm. When the algorithm arrives at the N-1 row, a matrix is constructed, all the degrees of freedom are under constraint, and it checks if the constructed matrix verifies all the properties of Θ. If it does, it increases the value of matrices found. The algorithm stops when all the possible different matrices are built.

The processing of the algorithm can be linked to the simple enumeration algorithm with some abstraction. For the simple enumeration, each entry of the matrix is progressively increased from 2 to $n-1$. In the enumeration using Kraft, we do not work by entry but rather by row. If we link each row of the matrix τ to the index of sequence in the permutation file, it would result in a simple enumeration where each row is increased from 0 to the last index of the permutation file.
Note: That would be the case if the only property of Θ used for generating the matrices was the Kraft equality but by using the symmetry property, a lot of unnecessary matrices can be skipped.
For example, if we consider 5 taxa, there are 13 unique path-length sequences, therefore the number of matrices that would be enumerated (only using Kraft) would be 13^5. A simple observation is that by using symmetry, the last row of τ is actually already constructed, making it 13^4 matrices.

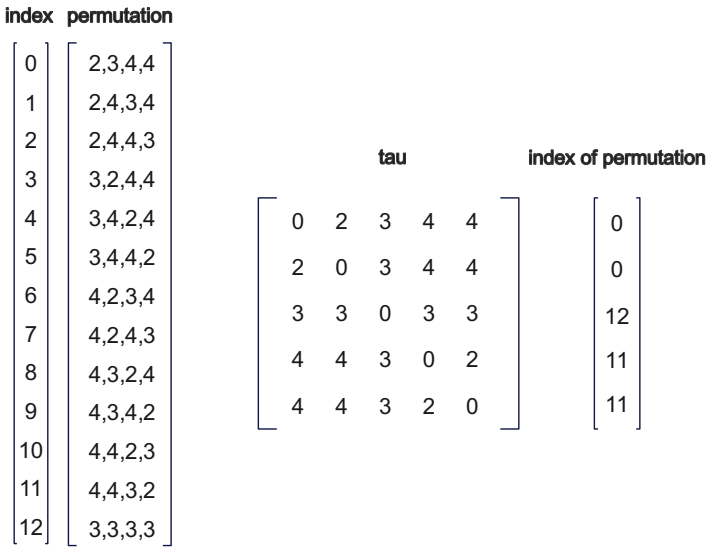


Figure 26: simple Kraft algo

By adding the symmetry property, it further reduces a lot of matrices to enumerate, but it is difficult to quantify. A simple example is that by the addi-

tion of the symmetry property in the whole process, the matrix τ represented in Figure 26 is the first matrix to be completed. The 2 first rows were directly added, but when looking at the third row, the matrix would have looked like the matrix (43), that is if we take symmetry into account. The constraint for row 3 is 3, 3, and the only permutation that has that as its first entry is the permutation at index 12. So simply by adding this constraint, it avoided the program to enumerate $12 * 13 * 13$ matrices, which would have never been part of Θ anyway since there is no other sequence with these same 2 first rows that respects the Kraft equality on row 3.

$$\begin{bmatrix} 0 & 2 & 3 & 4 & 4 \\ 2 & 0 & 3 & 4 & 4 \\ 3 & 3 & 0 & x & x \\ 4 & 4 & x & 0 & x \\ 4 & 4 & x & x & 0 \end{bmatrix} \tag{43}$$

Concretely the implementation part is done in 2 steps. The first one is finding the constraint for the current row, and the second one is finding the sequence that completes the row under the constraint if there is one.
Finding the constraint is quite straightforward, it consists in looking at the values that are on the left of the diagonal.
The second step is however more interesting, it will be the subject of loads of performance improvements. But the simple and first version of the function finds row works as shown in the algorithm 4.

The function loops through the file where the path lengths are stored until the index of the previous row (if there was one) is encountered. Once it reached the start of the comparison it compares each sequence with the constraint already present. If there is a suitable sequence it will add it to the matrix, update the index of the permutation associated with the row and return a true value. If it does not find one it removes the current row (in case there was one already in place), resets the index of permutation, and returns a false value.

The whole enumeration algorithm is included in the algorithm 5. It is done recursively and stops when the function cannot find the next row when looking for the first row.

This approach worked perfectly for small problems 4 or 5 taxa but it was not able to handle larger problems. It was facing an error as soon as 7 taxa were considered.

Algorithm 4 function find row (tau, permutation file, index permutation, current row, last perm)

```
constraint = find constraint(tau, current row)
start = index permutation[current row]
line = 0
while not end of file do
    s = get sequence of file
    if line ≥ start then
        if constraint == first element of s then
            add s to current row of tau
            index permutation[current row] = line
            return true
        end if
    end if
    line++
end while
remove current row
index permutation[current row] = -1
return false
```

Algorithm 5 function enumerate(tau, file, current row, last permutation)

```
if row = N-1 then
    if tau ∈ Θ then
        valid matrices ++
    end if
    return
end if
flag ← find row()  ▷ if sequence is added is true, if sequence is removed false
if flag is true then
    return Enumerate(tau, index permutation, file, current row + 1 )
else if flag is false and current row = 0 then
    return
else if flag is false and current row ≠ 0 then
    return Enumerate(tau, index permutation, file, current row - 1 )
end if
```

The problem encountered when implementing this with a recursive paradigm is called stack overflow. To have a better understanding of the type of problem faced, it is easier to describe the same problem with an easier function such as the calculation of a factorial with a recursive paradigm and how a computer manages that type of data storage.
The code to calculate a factorial is shown in the algorithm 6

Algorithm 6 factorial(N)

Require: $N \geq 0$
  if $N \geq 1$ **then**
    return 1
  else
    return N * factorial(N-1)
  end if

Stack overflow happens when the function is called too many times before returning, so for example if we try to compute the factorial of $1,000,000$ the computer will have to keep in memory the call to the function factorial a million times before returning a value:

$$N \times \text{factorial(N-1)} \rightarrow N \times (N-1) \times (N-2) \times \cdots \times 2 \times \text{factorial(1)}$$

In this example before being able to compute the factorial, the computer has to store in memory N variables. It is in a sense the same as dynamically increasing an array until a length N.

When that occurs the code cannot execute and returns an error of stack overflow, meaning that the program runs out of memory in the call stack i.e. there is no more space to allocate more memory for another call to a function. It does not mean that the code has a syntax error, it can execute the code as intended and give correct results but it just cannot manage a problem of that size due to the obligation of keeping that many calls to the function. This was the exact problem I faced when using recursion on this problem of enumeration of matrices.

To solve this, the function enumerate returns the next row that needs to be filled as an integer instead of calling itself again with a change in the parameter. In a sense the recursion part is taken out of the function and done outside of it, it now only does one step at a time instead of working everything out by itself.

There must be a stopping criterion to indicate when the enumeration is completely done. That happens when the first row of the matrix has been through each of the sequences in the permutation file. When that happens, the index of the permutation is put to -1. That is the stopping criterion of the enumeration problem.

This explains how the basic enumeration is done using Kraft equality in addition to the symmetry and null diagonal property.
There are many ways to improve and make the enumeration more efficient, which will be exposed in the next section.

6.4 Improvements of the simple Kraft enumeration

To compare how the various changes impact the efficiency of the algorithm, the benchmark will be the time to enumerate all the different path-length matrices

Algorithm 7 function enumerate(tau, file, current row, last permutation)

```
if row = N-1 then
    if tau ∈ Θ then
        valid matrices ++
    end if
    return current row - 1
end if
boolean flag
flag ← find row()  ▷ if sequence is added is true, if sequence is removed false
if flag is true then
    return current row +1
else if current row = 0 then ▷ if no sequence was found for first row → end
of enumeration
    return 0
else if flag is false and current row ≠ 0 then
    return current row -1
end if
```

Algorithm 8 procedure(tau, permutation file, index permutation)

```
while index permutation[0] != -1 do
    next row = Enumerate(tau, index permutation, file, next row)
end while
```

associated to 7 taxa. There are 945 different matrices to find. All the calculations will be done on a PC with the following specifications:

CPU	Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz
Ram	16,0 GB (15,6 GB usable)

The algorithm is coded in C++, choice of language is made for performance purposes. The basic algorithm takes 1321.96 seconds or around 22 minutes to do the full enumeration and it enumerates 899210 matrices to find the 945 valid matrices that are part of the set Θ for 7 taxa. This will be the benchmark to compare how each improvement improves the efficiency of the algorithm.

The first and easiest improvement to add to the current algorithm is to change the reading of the permutation file. For the first algorithm, when reading a file, starts at the first line and loops through the first sequence to compare. It reads each row from the beginning, even the unnecessary rows. To change that, instead of looping through the whole file, the function "istream& seekg (stream pos)" is used. That function enables the stream to directly access a specific entry of a file [72].

performance of algorithm		
algorithm	Enumeration time (seconds)	Matrices enumerated
Basic algorithm	1321.96	899210
Improved reading	1074.98	899210
Improved reading and comparison	885.53	899210

The second improvement is linked to the comparison between the constraints and the sequences. In the basic algorithm, the comparison is done between all the constraints and all the first elements in the sequence that corresponds to the constraints. At each comparison, the algorithm compares a number of elements equivalent to the row at which the comparison occurs. Since this is an operation that is happening a lot and is at the core of the algorithm, the optimization of this operation will have a significant impact.

The change made is instead of checking if the whole sequence and the whole constraints are matching, it only first checks the first value. If the first values are equal, it checks the rest of the constraint. This may seem counterintuitive but let's take an example to illustrate how this changes efficiency. If we look at the permutations for 5 taxa, as represented in 26, and the algorithm tries to find a row that has as constraint $2, 3, 2$, then it will only perform a comparison between 3*3 + 10 integers instead of 13*3 integers. These figures come from the fact that there are only 3 sequences beginning with a 2. The comparisons are the ones highlighted in red in Figure 27. This small change has an impact since the comparison operation is performed for every step. Another observation is that this change is more and more important for increasing sizes. This can be shown by looking at the following equation.

$$(N-1)m(x) + (\text{total perm} - m(x))1 < \text{total perm}(N-1)$$

Where $m(x)$ represents the number of sequences starting with the integer x and N is the number of taxa considered.
Something interesting to add can be expressed by looking at the ratio $\frac{m(x)}{\text{total perm}}$. This ratio expresses how many operations are saved with this little change depending on the size of the problem.

Number of taxa	All permutations	Sequences starting with 2	Ratio
5	13	3	0.230
6	75	13	0.173
7	525	75	0.142
8	4347	525	0.121
9	41245	4347	0.105
10	441675	41245	0.093

Interesting observation: number of sequences that starts with 2 for N taxa is the same as the number of permutations for N-1 taxa

This table shows that the greater the problem, the more impact that modification will have relatively speaking. So for 8 taxa and for checking the

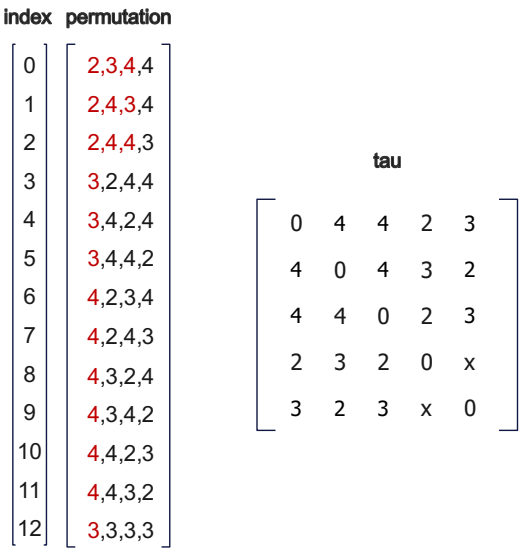


Figure 27: operations for efficient comparison

sixth row that starts, instead of checking $4347 * 5 = 21735$ values, it checks $525 * 5 + 4347 - 525 = 6447$ values at each time a sixth row is constructed which constraints start with a 2. This simple change reduced, in this case, the number of operations by 3.37 times.

The third improvement is the enumeration of the last rows. The idea is to do a simple enumeration, as explained in section 6.1, for the last 2 or three rows, depending on the size of the problem and the efficiency of the comparison algorithm it can reduce the computational complexity of the problem. This idea comes from the observation that the more there are elements in the constraint, the more individual operations are to be done. In other words, exploring the last rows of the matrix to find a sequence that match takes more and more computational time compared to finding rows at the beginning of the matrix. To address this issue, the idea of using simple enumeration for the last rows, i.e. where there are fewer degrees of freedom, is the same as if the algorithm did a simple enumeration for a part of the sub-matrix.

The computational cost of doing so is each time the algorithm is at row N-2 of the matrix, there is $(N-3)^3$ operations happening, $(N-3)^6$ if the simple enumeration starts for the last 3 rows. This approach will make the algorithm faster if the computational cost of finding the last 2 or 3 rows is less than with the Kraft algorithm. With the Kraft algorithm, we have

$$(\text{Current row}) * m(x) + (\text{total perm} - m(x)) * \text{total perm}$$

performance of algorithm		
algorithm	Enumeration time (seconds)	Matrices enumerated
Basic algorithm	1321.96	899210
Improved reading	1074.98	899210
Improved reading and comparison	885.53	899210
Improved reading, comparison and simple enumeration for last 2 rows	672.509	899210

The fourth and last improvement, probably the most significant one, is adding a step before doing the comparison. Instead of browsing through the whole permutation file to check if a sequence fits the constraints, the algorithm will first check in the ordered sequences to see if among their permutations there is a sequence that fits.

Figure 28 is represented the process for 6 taxa with constraints that are 3,3. The algorithm will first search in the ordered sequences if an ordered sequence contains the constraint. If it does, the only permutations that will be checked are those originating from the ordered sequence.

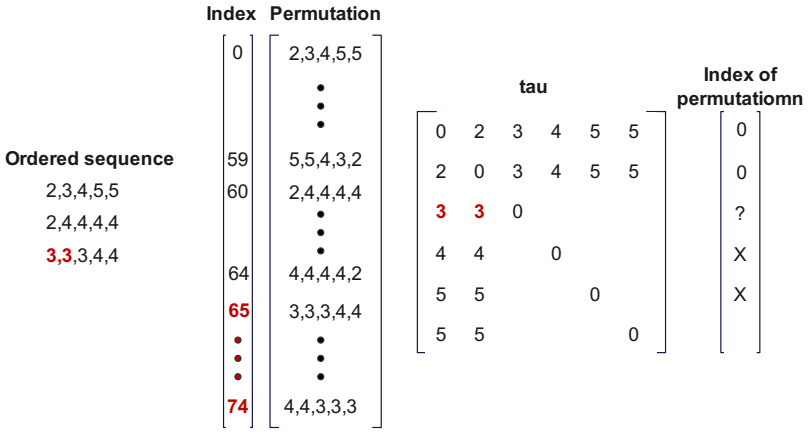


Figure 28: Improved enumeration process

To check if the constraints are in the ordered sequences a special comparison is used for efficiency purposes. The ordered sequences are stored in a different way. They are decomposed into 3 different arrays. The first array stores the unique values that appear in the sequence, the second stores the number of repetitions there is of the unique value, and the last one stores the number of unique values there. The decomposition works as shown in Figure 29.

This decomposition enables the comparison between constraints and order sequences to be faster especially when there are a few unique values in the ordered sequences.

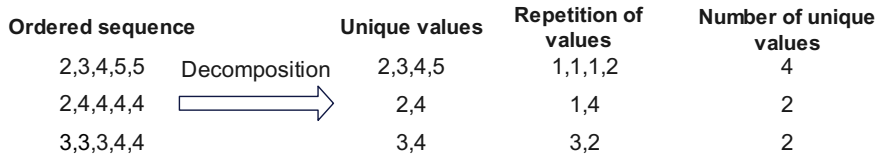


Figure 29: Decomposition of ordered sequences

The comparison of a 3,3 constraint happens in the following way:
Firstly it browses through the array of unique values to check if there is a 3. If there is, it then checks if the number of repetitions of the value is greater or equal to 2 (number of repetitions of 3 in the constraints).
When a sequence has embedded constraints, it only looks at the permutation of the relevant sequence instead of the whole file.

Another small improvement regarding this part is similar to the second improvement discussed before but pushed even further. It is taking advantage of the way permutations are stored. To keep the example of a constraint that is 3,3 if the first element of permutation starts with 2, it goes to the next one, as explained in the first. However, if the constraint is greater than 3, it directly switches to the next permutation of the ordered sequence that contains the constraint if there is one.

This is shown in Figure 30 when there are 5 taxa and the constraint is composed of only 3.

performance of algorithm		
algorithm	Enumeration time (seconds)	Matrices enumerated
Basic algorithm	1321.96	899210
Improved reading	1074.98	899210
Improved reading and comparison	885.53	899210
Improved comparison and decomposition	286.216	899210
Improved comparison, decomposition and simple enumeration for last row	225.286	23180390

There are two main concerns with this approach, the first one is memory allocation. The BMEP problem is considered as NP-Hard; the bigger the size of Γ the more matrices have to be enumerated, and it increases hyper-exponentially (2n-5)!!. The problem with this approach is the balance between computing all the permutations of the sequences respecting the Kraft equality and the memory that it needs to be stored.

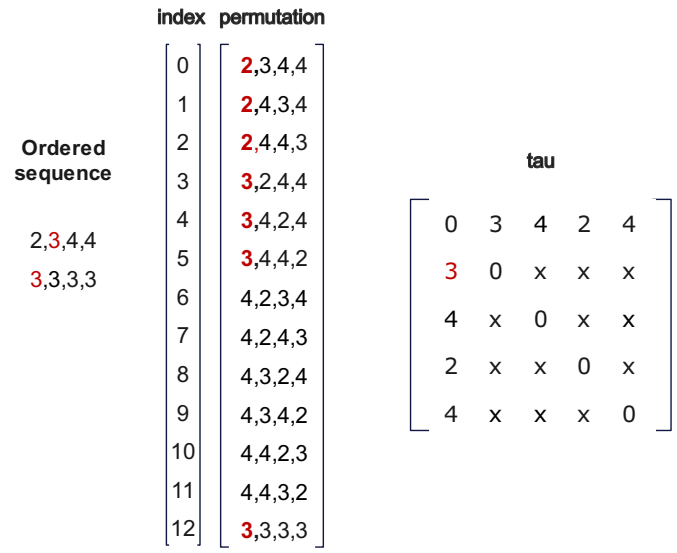


Figure 30: Optimized comparison algorithm

The second concern is the algorithm's time complexity as encountered with the simple method. The algorithm has to be performed in a "reasonable" time frame. Reasonable means different things depending on the problem. Since the code has to be performed only once and not regularly, the maximum amount of time the code has to be executed should not exceed a few days or a week at most if it delivers good results, i.e. works for sufficiently large matrices.
At the moment, this approach struggles for 8×8 matrices. After one hour of running time, it has only found 5 valid matrices of τ. If it were to enumerate all the 11!! or 10395 matrices, it would take 2079 hours or 86 days. This is obviously a big problem. The next section will explain how this problem can be parallelized to reduce the running time.

Adding the Kraft equality to reduce the number of matrices enumerated enabled us to work with 7×7 matrices in a reasonable time frame. Without it, it was not even possible to find one matrix in one complete day whereas now, the whole 945 matrices in τ are found in 21 minutes.

6.5 parallelization

Parallelization is the process of dividing the workload of the algorithm between a number of processors. Implementing it in our case is relatively easy since it is possible to make each processor work independently on a different subsection of the enumerative set. The only thing to do is divide the set into sub-spaces of equal cardinality.
To keep the analogy of finding prime numbers, it would be equivalent to dividing

the interval of interest between the number of processors that are available. So if the interval is 1 000 000 and we have five processors, then processor 1 would search the prime numbers in the interval $[0, 199999]$, processor 2 in the interval $[200000, 399999]$, etc...
In our problem of trying to find all the elements of $\widehat{\Theta}$, the approach is similar, but instead of representing the set on a 10 basis, each entry lies between 0 and 9, the basis of enumeration of the problem is on a total permutation basis, where each entry represent a row in matrix τ. All the set lies in between these 2 representations of the index of the rows of τ.

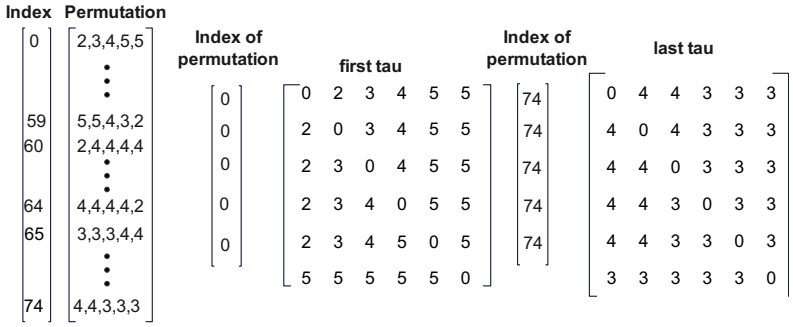


Figure 31: Interval of enumerative Set

In figure 31, we see that the interval, for 6 taxa, lies between the vector $[0, 0, 0, 0, 0]$ and the vector $[74, 74, 74, 74, 74]$. By analogy (or bijection), the vector $[0, 0, 0, 0, 0, 0]$ is similar to the 0 in the set of integers and the vector $[74, 74, 74, 74, 74]$ is similar to the integer 1 000 000. The next question is how to find the value 200 000 in this other set that describes τ. The bijection of 200 000 would give the vector $[15, 0, 0, 0, 0]$, $(15 = 75/5)$. So the boundaries of each subset for each processor works like this, for processor n the enumeration starts at $[\frac{\text{max permutation}}{Numberofprocessors} n, 0, 0, 0, 0]$ and ends at $[\frac{\text{max perm}}{numberofprocessors}(n+1) - 1, \text{max perm}, \text{max perm}, \text{max perm}, \text{max perm}]$.
In Figure 32, we can see the processing time of each processor. The time spent by each processor is different due to the aspect of the problem and the approach. Half of the improvements were to avoid the enumeration of unnecessary matrices, and it is impossible (extremely difficult, at least) to predict how many of these matrices there are to avoid in each sub-space of the set assigned to the processors. This explains why certain processors are done with their sequence faster than others.

6.6 adding the triangular equality

Until now, three out of the five equations defining Θ have been used to enumerate the matrices. The symmetry property, the null diagonal, and the Kraft equality. The next step would be to incorporate the additive property of Buneman into the already created submatrices. The phylogenetic manifold equation will not help us reduce the cardinality of the enumerative set since it only ap-

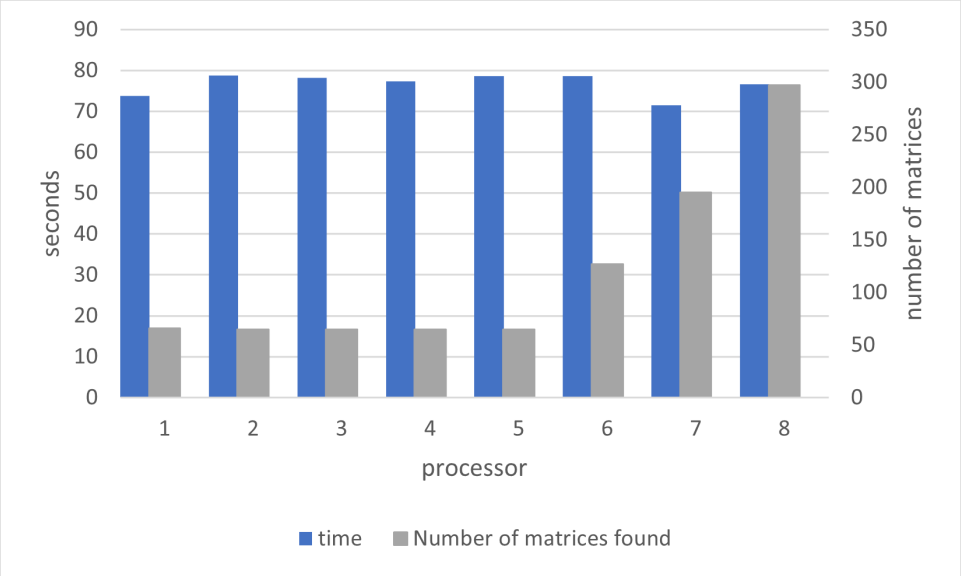


Figure 32: load distribution analysis (time per processor)

plies to fully constructed matrices.

The addition of the Buneman property would be incorporated in the following way:
Since this relation applies to every triplet of already connected leaves, once the matrix has four rows, the property can be checked in the submatrix already fully constructed. An example of a submatrix to which it can be verified is shown in Figure 33. In this example, we could already check if all the entries in the submatrices A and B (C is similar to B because of symmetry), which corresponds to the distance of taxa already connected, are respecting the four-point condition of Buneman.

So if the submatrix does not hold the Buneman additive property, all the matrices that are constructed based on this will never be in Θ since the part of the matrix constructed will not change and is fixed for all its descendants.

One way to see it is that by constructing the matrix with path-length sequences that respect the Kraft equality, we are building a forest of trees. Adding the Buneman property checks whether the connected taxa (the part of the matrix already constructed) can represent a tree and not a network [**?**]. It verifies if there are no cycles in the graph that τ represents.

This would enable is similar to pruning for the branch and bound algorithm. It would enable to enumerate subspaces still relevant to find matrices that fit the restrictions on the set describing Θ.

Adding this would probably enable the enumeration process to matrices of size 8×8, maybe even 9×9 but it is unlikely that it will help achieve an enumer-

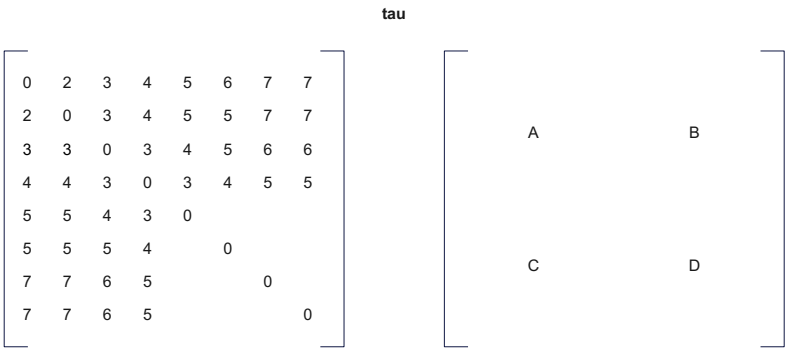


Figure 33: Meaning of the addition of Buneman's four-point property

ation of path-length matrices whose sizes are high enough to possibly find a counter-example of the conjecture we are trying to disprove.

6.7 Possible different approach

The first thing to do is to add the additive property to the algorithm.
Another approach that might produce better results, is to work on the unlabelled trees. To use the kraft equality again in trying to create matrices in Θ but once they are created find all of the matrices that are associated with the isomorphism of the graph. The idea is that isomorph graphs are created with the same sequences but orderred in different ways. For example, a graph with ficve taxa is only composed of three path lengths (2,3,4,4) and one (3,3,3,3), which is linked with the unique unlabelled unrooted tree it can be represented. For six taxa, the two only compositions of a matrix tau are either four sequences (2,3,4,5,5) and two (3,3,3,4,4) or six sequences (2,4,4,4,4).

So the next step in enumerating the matrices would be to incorporate the isomorphisms into the enumeration. This is a step that I could not produce but might reduce the whole process even more. Identifying all the compositions of sequences generating the matrices and enumerating only these possibilities instead of all of them.

7 Conclusion

In this thesis, we have approached the Balanced Minimum Evolution Problem (BMEP) in two different ways. The first axis of this thesis is solving the problem and providing an algorithm that finds an exact solution. Implementing a branch and bound algorithm and explaining the concepts behind the change in the objective function by the addition of a taxon. There exist other approaches that find an approximate solution to the problem but being able to have a precise answer is essential, simply to have a reference point in assessing which other approximate methods are giving the best results. This method is not destined to solve problems with 200 taxa but it will help improve and select the other methods that can do it.

The second axis is about helping the research on the characterization of the path-length matrices of phylogenetic trees. The idea is to computationally find a counter-example to the conjecture that tries to define this set. Up to this day, there is no formal proof that the set of equations that currently characterize this set is sufficient, up until 11 taxa; they are, however, necessary. We tried to use all the equations to their fullest extent to produce the best result. The aim was to enumerate all the elements of the set defined by these equations. If the number of elements in the set were superior to the number of unique path-length matrices ($2n - 5!!$) associated with the problem, it would have meant that there exists a counter-example and that to have a sufficient representation of the problem at least one new relationship has to be expressed. To do this, the main property used to create the different matrices was based on the path-length sequences that respect the Kraft equality. This was the groundwork for the whole algorithm. Built on that, the complexity came from the research of performance in the algorithm and asking ourselves questions on how the other equations can reduce the number of matrices enumerated.

Unfortunately, this approach does not have much future since the publication of an article gives theoretical proof that the actual characterization holds true for problems with at least eleven taxa. The enumeration process was only able to get to 7 taxa in a reasonable time. Due to the hyper-exponentiality of the problem, optimizing it to expand this approach to 12 taxa seems difficult. However, a possible lead could be used to give a more promising result, trying to use the isomorphism of unrooted binary trees. This could possibly be achieved but it seems rather complex. This concept would need more research and work to possibly lead to a new discovery.

References

[1] Charles Darwin. On the origin of species by means of natural selection, or preservation of favoured races in the struggle for life: Murray. *Evolution*, 1859.

[2] Kees van Putten. Trees, coral, and seaweed: An interpretation of sketches found in darwin's papers. *Journal of the History of Biology*, 53(1):5–44, 2020.

[3] Gregor Mendel. Experiments in plant hybridization (1865). *Verhandlungen des naturforschenden Vereins Brünn) Available online*, 1996.

[4] August Weismann. *On germinal selection as a source of definite variation*. Number 19. Open court publishing Company, 1896.

[5] Joseph H Camin and Robert R Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, pages 311–326, 1965.

[6] Willi Hennig. *Phylogenetic systematics*. University of Illinois Press, 1999.

[7] James D Watson and Francis HC Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[8] Kary Mullis, Fred Faloona, Stephen Scharf, Randall Saiki, Glenn Horn, and Henry Erlich. Specific enzymatic amplification of dna in vitro: the polymerase chain reaction. In *Cold Spring Harbor symposia on quantitative biology*, volume 51, pages 263–273. Cold Spring Harbor Laboratory Press, 1986.

[9] Emile Zuckerkandl and Linus Pauling. Molecules as documents of evolutionary history. *Journal of theoretical biology*, 8(2):357–366, 1965.

[10] Walter M Fitch and Emanuel Margoliash. Construction of phylogenetic trees: a method based on mutation distances as estimated from cytochrome c sequences is of general applicability. *Science*, 155(3760):279–284, 1967.

[11] David R Bickel. *Phylogenetic Trees and Molecular Evolution: A Hands-on Introduction with Uncertainty Quantification Corrected*. Springer Nature, 2022.

[12] Roderick DM Page and Edward C Holmes. *Molecular evolution: a phylogenetic approach*. John Wiley & Sons, 2009.

[13] Matt Haber and Joel Velasco. Phylogenetic Inference. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2022 edition, 2022.

[14] Russell Schwartz. Computational models for cancer phylogenetics. *Bioinformatics and Phylogenetics*, pages 243–275, 2019.

[15] Salim Akhter Chowdhury, E Michael Gertz, Darawalee Wangsa, Kerstin Heselmeyer-Haddad, Thomas Ried, Alejandro A Schäffer, and Russell Schwartz. Inferring models of multiscale copy number evolution for single-tumor phylogenetics. *Bioinformatics*, 31(12):i258–i267, 2015.

[16] Louxin Zhang. Clusters, trees, and phylogenetic network classes. In *Bioinformatics and Phylogenetics*, pages 277–315. Springer, 2019.

[17] Sandra L Baldauf. Phylogeny for the faint of heart: a tutorial. *TRENDS in Genetics*, 19(6):345–351, 2003.

[18] Jitra Waikagul and Urusa Thaekham. *Approaches to research on the systematics of fish-borne trematodes*. Academic Press, 2014.

[19] Joseph Felsenstein and Joseph Felenstein. *Inferring phylogenies*, volume 2. Sinauer associates Sunderland, MA, 2004.

[20] Matthew R Helmus, Thomas J Bland, Christopher K Williams, and Anthony R Ives. Phylogenetic measures of biodiversity. *The American Naturalist*, 169(3):E68–E83, 2007.

[21] Stephen D Bentley and Julian Parkhill. Genomic perspectives on the evolution and spread of bacterial pathogens. *Proceedings of the Royal Society B: Biological Sciences*, 282(1821):20150488, 2015.

[22] Daniel Janies. Phylogenetic concepts and tools applied to epidemiologic investigations of infectious diseases. *Microbiology Spectrum*, 7(4):7–4, 2019.

[23] Abraham B Chang, Ron Lin, W Keith Studley, Can V Tran, and Milton H Saier, Jr. Phylogeny as a guide to structure and function of membrane transport proteins. *Molecular membrane biology*, 21(3):171–181, 2004.

[24] Chansotheary Dang, Jeth GV Walkup, Bruce A Hungate, Rima B Franklin, Egbert Schwartz, and Ember M Morrissey. Phylogenetic organization in the assimilation of chemically distinct substrates by soil bacteria. *Environmental microbiology*, 24(1):357–369, 2022.

[25] Sarah Hill, Mark Perkins, Karin J von Eije, Kim Benschop, Nuno R Faria, Tanya Golubchik, Edward Holmes, Liana Kafetzopoulou, Philippe Lemey, Tze Minn Mak, et al. Genomic sequencing of sars-cov-2: A guide to implementation for maximum impact on public health. 2021.

[26] Dean Southwood and Shoba Ranganathan. Genome databases and browsers. *Encyclopedia of bioinformatics and computational biology: ABC of Bioinformatics*, pages 251–256, 2019.

[27] genbank. https://www.ncbi.nlm.nih.gov/genbank/ . Accessed: 2023-01-05.

[28] Da-Fei Feng and Russell F Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of molecular evolution*, 25(4):351–360, 1987.

[29] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma, and Takashi Miyata. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic acids research*, 30(14):3059–3066, 2002.

[30] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[31] Mark A Larkin, Gordon Blackshields, Nigel P Brown, R Chenna, Paul A McGettigan, Hamish McWilliam, Franck Valentin, Iain M Wallace, Andreas Wilm, Rodrigo Lopez, et al. Clustal w and clustal x version 2.0. *bioinformatics*, 23(21):2947–2948, 2007.

[32] Fábio Madeira, Matt Pearce, Adrian RN Tivey, Prasad Basutkar, Joon Lee, Ossama Edbali, Nandana Madhusoodanan, Anton Kolesnikov, and Rodrigo Lopez. Search and sequence analysis tools services from embl-ebi in 2022. *Nucleic acids research*, 50(W1):W276–W279, 2022.

[33] Michael S Rosenberg. *Sequence alignment: methods, models, concepts, and strategies*. Univ of California Press, 2009.

[34] Andrzej Zielezinski, Susana Vinga, Jonas Almeida, and Wojciech M Karlowski. Alignment-free sequence comparison: benefits, applications, and tools. *Genome biology*, 18(1):1–17, 2017.

[35] Thomas H Jukes, Charles R Cantor, et al. Evolution of protein molecules. *Mammalian protein metabolism*, 3:21–132, 1969.

[36] Motoo Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of molecular evolution*, 16(2):111–120, 1980.

[37] Xuhua Xia. *A Mathematical Primer of Molecular Phylogenetics*. Apple Academic Press, 2020.

[38] Yves Pauplin. Direct calculation of a tree length using a distance matrix. *Journal of Molecular Evolution*, 51(1):41–47, 2000.

[39] Andre Rzhetsky and Masatoshi Nei. Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular biology and evolution*, 10(5):1073–1095, 1993.

[40] Olivier Gascuel. *Mathematics of evolution and phylogeny*. OUP Oxford, 2005.

[41] Daniele Catanzaro, Martin Frohn, Olivier Gascuel, and Raffaele Pesenti. A tutorial on the balanced minimum evolution problem. *European Journal of Operational Research*, 300(1):1–19, 2022.

[42] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

[43] Olivier Gascuel and Mike Steel. Neighbor-joining revealed. *Molecular biology and evolution*, 23(11):1997–2000, 2006.

[44] Vincent Lefort, Richard Desper, and Olivier Gascuel. Fastme 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program. *Molecular biology and evolution*, 32(10):2798–2800, 2015.

[45] Richard Desper and Olivier Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. In *International Workshop on Algorithms in Bioinformatics*, pages 357–374. Springer, 2002.

[46] Fabio Pardi and Olivier Gascuel. Combinatorics of distance-based tree inference. *Proceedings of the National Academy of Sciences*, 109(41):16443–16448, 2012.

[47] Mike Steel. A basic limitation on inferring phylogenies by pairwise sequence comparisons. *Journal of Theoretical Biology*, 256(3):467–472, 2009.

[48] Walter M Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[49] Joseph Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, 17(6):368–376, 1981.

[50] Bruce Rannala and Ziheng Yang. Probability distribution of molecular evolutionary trees: a new method of phylogenetic inference. *Journal of molecular evolution*, 43(3):304–311, 1996.

[51] John P Huelsenbeck and Fredrik Ronquist. Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, 2001.

[52] Alexei J Drummond, Marc A Suchard, Dong Xie, and Andrew Rambaut. Bayesian phylogenetics with beauti and the beast 1.7. *Molecular biology and evolution*, 29(8):1969–1973, 2012.

[53] Joseph Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *evolution*, 39(4):783–791, 1985.

[54] Benjamin Allen and Mike A. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5:1–15, 2001.

[55] Tandy Warnow. *Bioinformatics and phylogenetics: seminal contributions of Bernard Moret*, volume 29. Springer, 2019.

[56] Daniel Huson. Introduction to algorithms in phylogeny. 2005.

[57] Guohua Jin, Luay Nakhleh, Sagi Snir, and Tamir Tuller. Inferring phylogenetic networks by the maximum parsimony criterion: a case study. *Molecular Biology and Evolution*, 24(1):324–337, 2007.

[58] Daniele Catanzaro. The minimum evolution problem: Overview and classification. *Networks: An International Journal*, 53(2):112–125, 2009.

[59] Charles Semple and Mike Steel. Cyclic permutations and evolutionary trees. *Advances in Applied Mathematics*, 32(4):669–680, 2004.

[60] Vladimir Makarenkov and Bruno Leclerc. Circular orders of tree metrics, and their uses for the reconstruction and fitting of phylogenetic trees. In *Mathematical hierarchies and Biology*, pages 183–208, 1996.

[61] Daniele Catanzaro, Raffaele Pesenti, and Laurence Wolsey. On the balanced minimum evolution polytope. *Discrete Optimization*, 36:100570, 2020.

[62] Martin Frohn and Daniele Catanzaro. *Models and methods for computational phylogenetics under minimum evolution.* PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2021.

[63] D Stott Parker and Prasad Ram. The construction of huffman codes is a submodular (" convex") optimization problem over a lattice of binary trees. *SIAM Journal on Computing*, 28(5):1875–1905, 1999.

[64] Daniele Catanzaro, Martine Labbé, Raffaele Pesenti, and Juan-José Salazar-González. The balanced minimum evolution problem. *INFORMS Journal on Computing*, 24(2):276–294, 2012.

[65] Peter Buneman. A note on the metric properties of trees. *Journal of Combinatorial Theory, Series B*, 17(1):48–50, 1974.

[66] Richard Desper and Olivier 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(3):587–598, 2004.

[67] Samuel Fiorini and Gwenaël Joret. Approximating the balanced minimum evolution problem. *Operations research letters*, 40(1):31–35, 2012.

[68] C++ reference. https://en.cppreference.com/w/cpp/types/climits. Accessed: 2022-12-23.

[69] Fabio Pardi. *Algorithms on phylogenetic trees.* PhD thesis, University of Cambridge Cambridge, 2009.

[70] Daniele Catanzaro and Raffaele Pesenti. Enumerating vertices of the balanced minimum evolution polytope. *Computers & Operations Research*, 109:209–217, 2019.

[71] Stefan Forcey, Gabriela Hamerlinck, Logan Keefe, and William Sands. The minimum evolution problem in phylogenetics: Polytopes, linear programming, and interpretation. In *Algebraic and combinatorial computational biology*, pages 319–349. Elsevier, 2019.

[72] C++ reference. https://cplusplus.com/reference/istream/istream/seekg/. Accessed: 2022-09-29.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**