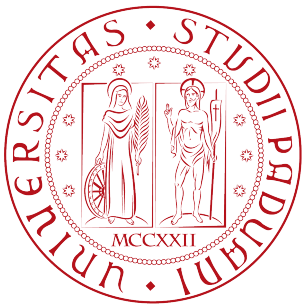

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Laurea magistrale in Ingegneria Informatica

Genome composition with mismatches and its application to phylogeny

Laureando: *Giulio Fracasso*

Relatrice: *Dott.ssa Cinzia Pizzi*

Anno Accademico 2013/2014

Alla mia famiglia

Contents

Abstract

Most molecular phylogenies are based on sequence alignments. Consequently, they fail to account for modes of sequence evolution that involve frequent insertions or deletions. Here we present a method for generating accurate species phylogenetic trees from whole genome sequence. Our approach makes use of the frequency of l-mers with mismatches and calculates a distance matrix for a given dataset of s taxa, of average length n and a pattern length l in $O(s^2 \cdot n^2)$ (so it is independent from l). It is computed according to the number of l-mers of each taxon occuring also in another one with at most k mismatches. It is still not clear if, given a set of sequences of average length n, there is a pattern length and relative number of mismatches which best fits with this dataset and, in that case, how to derive these parameters. However, our results demonstrate over two datasets (mitochondrial DNA and retroid viruses) that reliable phylogenetic trees are obtained for a certain range of values of both pattern length and maximum number of mismatches rather than for a single combination (l, k) of these parameters. As concerning the mitochondrial dataset, our method has been compared to the traditional (single gene or protein based) maximum likelihood method, to the Average Common Substring approach and to a method which uses the information theoretical concept of Kolmogorov complexity to deduce a distance between the species. The trees constructed from the dataset of retroid viruses has been evaluated looking at the taxonomy descripted in the International Committee on Taxonomy of Viruses. In both cases our results are as good as or better than those presented in these other works in terms of accuracy.

Sommario

La maggior parte delle filogenesi molecolari sono basate sull'allineamento di sequenze. Di conseguenza, esse non riescono a spiegare le modalità di evoluzione di sequenze che coinvolgono frequenti inserzioni o cancellazioni di caratteri. Qui viene presentato un metodo per generare accurati alberi filogenetici delle specie a partire dall'intera sequenza dei genomi. Il nostro approccio fa uso della frequenza di l-mers con mismatches e calcola una matrice delle distanze per un dato dataset di s taxa di lunghezza media n e una lunghezza del pattern l in $O(s^2 \cdot n^2)$ (indipendente dunque da l). Essa viene calcolata in base al numero di l-mers di ciascun taxon che è presente anche in un altro taxon con al più k mismatches. Non è ancora chiaro se, dato un insieme di sequenze di lunghezza media n , vi sia una lunghezza di pattern e un relativo numero massimo di mismatches che meglio si adatti a questo dataset e, in tal caso, come calcolare tali valori. Tuttavia, i nostri risultati mostrano su due dataset (il primo costituito da sequenze di DNA mitocondriale, il secondo relativo invece ai retroid virus) come si siano ottenuti alberi filogenetici corretti per un certo intervallo di valori sia di lunghezza del pattern che del numero massimo di mismatch, anziché per una sola combinazione (l, k) di questi parametri. Per quanto riguarda il dataset mitocondriale, il nostro metodo è stato paragonato al metodo della massima verosimiglianza (basato su singoli geni o proteine), all'approccio *Average Common Substring* e ad un metodo che utilizza le informazioni relative al concetto teorico di complessità di Kolmogorov per dedurre una misura di distanza tra le specie. Gli alberi costruiti dal dataset dei retroid virus sono stati valutati esaminando la tassonomia descritta nell' International Committee on Taxonomy of Viruses. In entrambi i casi i nostri risultati sono paragonabili a quelli presentati in questi altri lavori, e talvolta anche migliori in termini di *accuracy*.

1 Introduction

The fast advancement of worldwide genome sequencing has raised a fundamental and challenging question to modern biological science: "how do we compare genomes?" A phylogenetic tree is the only figure in *On the Origin of Species*, evidence of the central importance of such trees to evolutionary biology. A phylogenetic tree is a graphical representation of the evolutionary relationships among entities that share a common ancestor. Those entities can be species, genes, genomes, or any other operational taxonomic unit (OTU). More specifically, a phylogenetic tree, with its pattern of branching, represents the descent from a common ancestor into distinct lineages. It is critical to understand that the branching patterns and branch lengths that make up a phylogenetic tree can rarely be observed directly, but rather they must be inferred from other information.

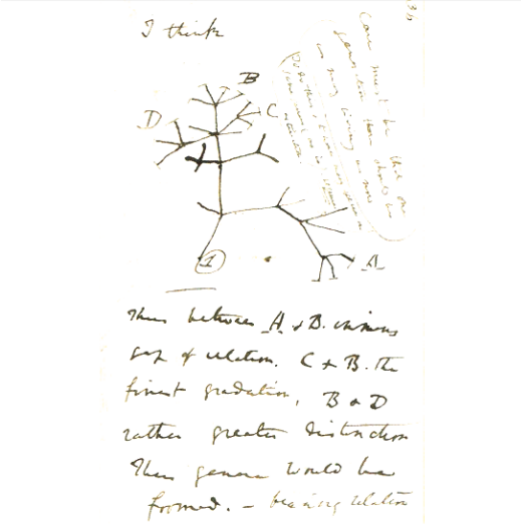

Figure 1.1: "I think" tree as depicted by Charles Darwin.

Existing methods such as multiple alignment and various sequence evolutionary models do not directly apply to complete genomes where such events as rearrangements make traditional full length alignments unfeasible. Traditional methods such as maximum parsimony or maximum likelihood are inapplicable for determinate datasets: for example viruses can be split up into several different families which have very few genes in common.
The desire to infer the evolutionary history of a group of species should be more viable now that a considerable amount of multilocus molecular data is available. However, the great majority of current molecular phylogenetic paradigm still reconstructs gene trees to represent the species tree.
In this thesis we propose a method which uses whole genome information and the distance between two species is calculated accordine on the mutual occurrences of l-mers with mismatches.
The thesis is organized as follows: Chapter 2 gives an overview of the background

of phylogeny, with the main techniques utilized to infer relationships between the species. In particular we concentrated on alignement-free methods, also to describe the works we referred to in order to compare our results.
Chapter 3 describes the tools and the libraries that have been used and the specifications of the development environment. Particularly there are reported the settings for the software `Phylip` [44], which we utilized to calculate the RF-distance between the species.
Chapter 4 describes the details of our approach and there we found the pseudocodes for our algorithms and the instruction to run it and to use the graphic user interface we provided in order to execute pairwise analysis.
In Chapter 5 we presented the achieved results and their comparison with other methods.
Finally, in Chapter 6 we sum up our observation and give a trace for possible future improvements of the present work.

2 Phylogeny basics

Evolution is a topic that has been subject of many studies during the last two centuries, since the great contribution of the naturalist Charles Darwin and his publication *"On the Origin of Species"*. The advent of molecular sequencing techniques has marked the transition from morphological observations to a kind of research of relationships among groups of organisms, that is based on similarities or differences that are intrinsic to the genetic inheritance of the species.
The result of phylogenetic studies is an hypothesis about the evolutionary history of taxonomic[1] groups i.e their *phylogeny*. The phylogenetic process describes an alteration of the species over time: indeed populations may split into separate branches, hybridize together, or terminate by extinction and this evolutionary branching process may be depicted as a *phylogenetic tree*, also called *dendrogram*. In such a diagram there are two kinds of nodes: external nodes represent the taxa of the current dataset while an internal node indicates an unknown common ancestor from which the species of its subtree split up; the distance of one group of species from the others indicates its degree of relationship, so closely related groups are located on branches close to one another[45]. Different dendrograms allow different analysis: for example *chronograms* explicitly represent evolutionary time through their branch spans, while *cladograms* and *phylograms* describe the distance between the taxa, with branch spans that are respectively indipendent or proportional to the amount of character change.
Trees can also be *rooted*, when a common predecessor is present, or *unrooted*, when they describe the evolutionary relationships between the taxonomic unities. We used unrooted trees are used since it is not a trivial problem nor to assess or to demonstrate that a particular taxon is the common ancenstor for a dataset, especially when big datasets are treated.

In the last decades the explosive growth of nucleotide and amino acid sequenced data and the consequent availability of whole genome and proteome sequences of different species has allowed Computer Science to investigate new techiques and has lead to the creation of huge datasets capable of storing this growing amount of data. NCBI[2], founded in 1988, houses a series of databases relevant to biotechnology and biomedicine, including *GenBank* for DNA sequences, an open access, annotated collection of all publicly available nucleotide sequences and their protein translations, and *PubMed*, a bibliographic database for the biomedical literature. All these databases are available online through the Entrez search engine [49].

Phylogenetic analysis are mostly based on these techniques:

- analysis of single genes or proteins;
- combination of a few gene trees to a species tree [25] in such a method single genes are utilized to recover the genealogy of taxa, individuals of a population, etc.;

[1] Taxonomy: the science of classification of organisms
[2] National Center for Biotechnology Information

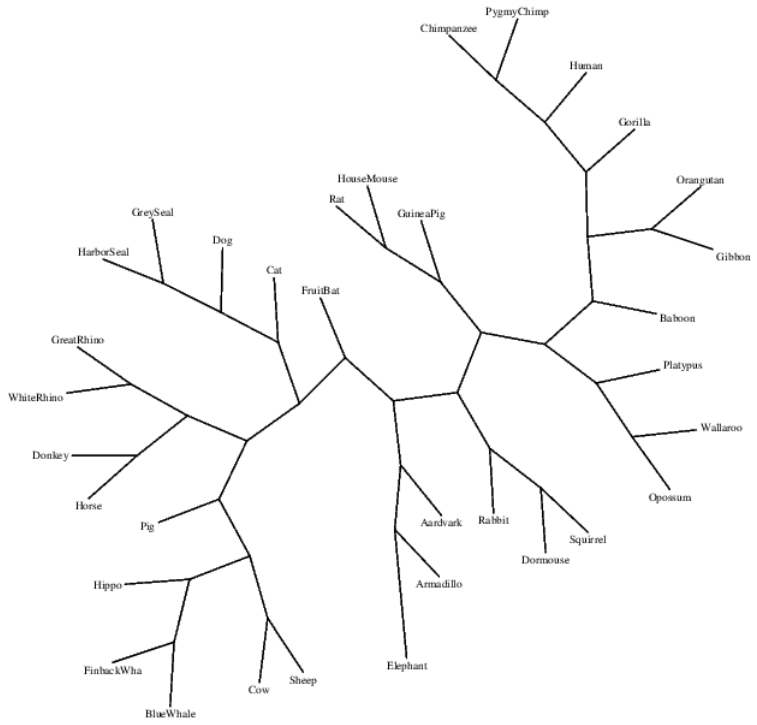

Figure 2.1: Example of a phylogenetic tree.

- quartet methods[29][4], which directly decompose the n datasets into subsets of four taxa each. If every quartet tree is computed correctly from noiseless data, then there is a single tree compatible with all $\binom{n}{4}$ resolved quartets and that is the true tree. In practise, of course, not all the resolved quartets are correct and consequently there may not be a single tree that is compatible with them [24];

- super tree methods [5]. Generally we call *consensus tree* the maximum likelihood tree derived from input trees with identical leaf set. Super tree methods are particular techniques that can combine information from input trees with nonidentical leaf sets, too.

Let us quickly explain some of the principal methods that are used for deriving a phylogenetic tree.

2.1 Phylogenetic tree reconstruction

We will now give a rapid overview of the main techniques to infer a phylogenetic tree, with particular enphasis on the distance-based approach (see section 2.1.4) which is the one used by the tool `Phylip` [44] that we have utilized in the current work.

2.1.1 Maximum parsimony

Maximum parsimony analysis is a character-based method that lies upon the principle that *"simpler hypotheses are preferable to more complicated ones"*. Thus a phylogenetic tree is inferred by minimizing the total number of evolutionary steps required to explain a given set of data, or in other words by minimizing the total tree length. The steps may be base or amino-acid substitutions for sequence data, or gain and loss events for restriction site data.
Since the problem of identifying the most parsimonious tree is known to be NP-hard [17], naïve approach (the enumeration of all possible trees and the following evaluation and identification of the most "convienient") has given way to various heuristics. The core of such technique is the selection of the so called "informative sites". A position in the relevant set of sequences is said to be informative if it favors one or more trees between all possible trees and if it contains at least two different characters, each of which is present on at least two sequences.
The steps performed are:

1. select the informative sites;
2. calculate the minimum number of substitutions required by each of the possible unrooted trees that describe the phylogenetic relationships between the taxonomic units concerned;
3. the tree (or trees) of maximum parsimony is the one that requires the minimum number of substitutions among all informative sites considered.

We underline that for this analysis only the information that present in informative sites is used.
Maximum parsimony presents some limitations due to the absence of a model of evolution and to the fact that it does not correct for multiple mutations or parallel substitutions. It also has to be said that these limitations particularly affect nucleotide sequences so this method is generally used for the analysis of amino acid sequences.

2.1.2 Maximum likelihood

Maximum likelihood assumes a certain model of evolution (generally a Markov model is used) and aims at finding an answer to the question:

"What is the probability to observe certains data, given a particular model of evolution?"

In some ways this method is not dissimilar from maximum parsimony: indeed maximum likelihood finds a tree based on probability calculations that best accounts for the large amount of variations of the dataset. The idea is that the more mutations are needed to explain a phylogeny, the less that evolution, and so that tree, will be likely. Obviously the probability of making some observations is deeply related to the model behind our assumptions. The critical difference between maximum likelihood and maximum parsimony is that the latter minimizes the amount of evolutionary

change required for data explanation, while the former attempts to estimate the actual amount of change according to the evolutionary model in place. So, given the original data, maximum likelihood works with a prior nucleotide substitution model to compute a likelihood score for each tree. Before beginning the procedure of tree derivation, either an evolutionary model must be specified that can account for the conversion of one sequence into another or parameters must be selected that can be estimated from the data. Then the probability that the selected evolutionary model will have generated the observed sequences is evaluated.
The main disadvantage in using this method is given by its computational complexity, which prevents to apply it to datasets with more than 20-30 sequences [38].

2.1.3 Bayesian methods

Bayesian methods are based on a probabilistic model that explains how the observed data have been produced. Here every parameter of the model has its own probability value and Bayesian inferece works comparing the *a priori* probabilitity of the model with the *a posteriori* probability i.e. the probability that the value of a parameter is equal to the observed probability.
Usually Markov chain Monte Carlo (MCMC) algorithms are used for the Bayesian inference approach, since the great advances that have been made in this field [18].

2.1.4 Distance-based methods

Distance-based approach for phylogenetic trees reconstruction relies upon the observation that relationships among the trees can be described through a distance matrix, which can come from a number of different sources such as morphometric analysis, pairwise distance formulae between the input organisms relying upon morphological data or, as often occurs, genetic distance. The basis of these methods is the assumption that, according to some biological criterion, a distance $d(S_i, S_j) = d_{ij}$ is associated with a set of sequences $(S_1, S_2, ...S_N)$, such that the following properties are verified:

- **Positivity**: $d_{ij} \geq 0 \ \forall i, j$
- **Zero-diagonality**: $d_{ij} = 0 \iff i = j$
- **Symmetry**: $d_{ij} = d_{ji}$
- **Triangle inequality**: $d_{i,j} + d_{jw} \geq d_{iw}$

After distance estimates have been computed, a phylogenetic tree can be reconstructed using precisely a distance-based reconstruction method. Most of such techniques perform a bottom up reconstruction using a greedy clustering algorithm. Initially, each input organism is put in its own cluster which corresponds to a leaf node in the resulting tree. Next, pairs of clusters are iteratively joined into higher level clusters, which correspond to connecting two nodes in the tree with a new parent node. When a single node remains, the tree is reconstructed.
In particular there are two clustering methods we need to cite, since they are the phylogenetic trees reconstruction techinques adopted by the tool `Phylip`, whose description is given in section 3.1: UPGMA and Neighbor-Joining.

2.1.4.1 UPGMA

Unweighted Pair Group Method with Arithmetic mean (UPGMA) uses an iterative clustering algorithm which proceeds through $N-1$ steps (where N is the size of the dataset) by gradually associating couple of sequences, or clusters of sequences, which are more similar.
UPGMA works under the assumption of the *molecular clock*, i.e. the hypothesis of constancy in time of the number of substitutions per site.

2.1.4.2 Neighbor-Joining

Another distance-based method, which we will use to build our trees, is *Neighbor-Joining*. This is a bottom-up clustering approach, created by Naruya Saitou and Masatoshi Nei in 1987, which is particularly suited for datasets comprising lineages with largely varying rates of evolution since it does not require the data to be ultrametric[3]. With the rapid growth of sequence databases it is still one of the few methods that allows the rapid inclusion of all homologous sequences present in the database in a single tree. In Figure 2.2 we report a diagram which describes how Neighbor-Joining works. An operational taxonomic unit (OTU) is an operational definition of a species or group of species often used when only DNA sequence data is available[7]. Starting from a completely unresolved star-like phylogeny tree, all the internal branches are determined through $N-3$ steps, where N is the size of the dataset, so that the overall length of all branches is the smallest possible.

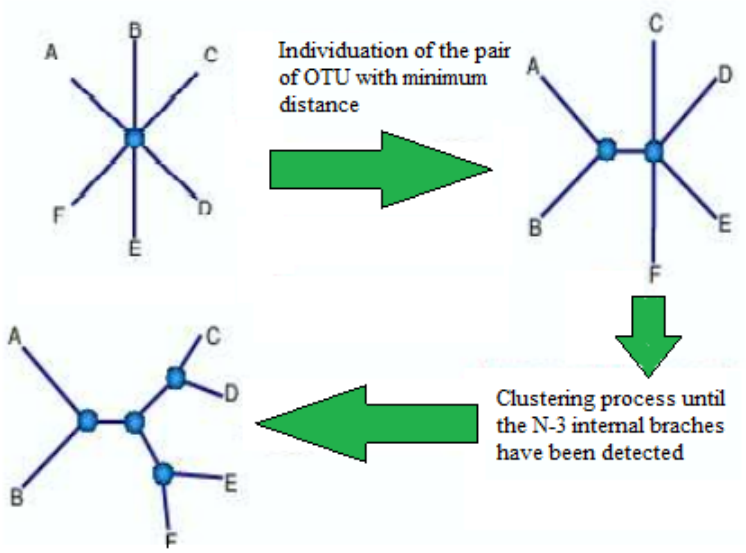

Figure 2.2: Flow chart of the Neighbor-Joining method.

[3]Data are said to be ultrametric when all lineages have diverged by equal amounts.

2.2 Alignment-free methods

The comparison of two or more closely related genomes at the base-by-base nucleotide sequence level is accomplished by sequence alignment; molecular sequence alignment is, in essence, a procedure by which we can recognize and describe potential homology among nucleotide or amino acid positions. Literature lists several well-known algorithms which operate optimal alignment and are the very foundation of sequence alignment such as *Smith-Waterman* algorithm [36] and *Needleman-Wunsch* algorithm [27]. Clearly it is impractical to find a perfect alignment between several sequences so dynamic and heuristic algorithms, mostly based on the recognition of alignment *seeds*, are used.
There are several popular tools for sequence alignment, among which `BLAST`[4] [12][46], one of the most widely used bioinformatics programs. BLAST's heuristic approach for performing pairwise alignment consists in a process called *seeding*: it consists in finding similar sequences, not by comparing either sequence in its entirety, but rather by locating short matches between the two sequences. Once these first matches has been individuated, BLAST begins to make local alignment. This heuristic method runs much faster than calculating an optimal alignment.
Other well-known tools are `FASTA` [42] and programs like `ClustalW` [47] which perform multiple alignment, i.e. the comparison of many sequences in one go.
However, since species diverge extensively over time, insertions or deletions (*indels*) and genomic rearrangements make straightforward sequence alignment unreliable. The overwhelming majority of biological sequence comparison methods rely on first aligning reference homologous sequences and deriving a score for the alignment of individual units, typically the logarithm of the odds ratio, which is then used to optimize the alignement of new sequences, thus reducing sequence dissimilarity evaluation to a comparison bewteen candidate alignments and reference alignmnent of well-studied sequences.
Moreover, we must refer to the computational complexity of alignment-based approaches, which makes them unfeasible when dealing with large-scale sequence data[3].

In the last two decades alignment-free methods have been studied as an alternative to estimating evolutionary distance from multiple alignment, in order to overcome the previously cited limitations.
We can differ between two principal kinds of approaches: the one based on k-mer frequency and the one based on substrings.

2.2.1 Methods based on l-mer frequency

This first category of methods is based on the statistics of word frequency and includes procedures which are based on metrics defined in coordinate space of l-mer count vectors, like the Euclidean distance, but also on the textitHamming distance. From now on, when referring to a l-mer, we will mean a substring of fixed lenght l. At a high level, the distance between two sequences is defined by first collecting the set of l-mers (subsequences of length l) occuring in the two sequences. From these two sets, the evolutionary distance between the two organisms is now defined

[4]Basic Local Alignment Search Tool

by measuring how different the two sets are. The more the two sets look alike, the smaller is the evolutionary distance. The main motivation for estimating evolutionary distance based on l-mers, is that it is computationally much faster than first constructing a multiple alignment[6]. It is noteworthy that such methods, although alignment-free, are still length dependent in the sense that the comparisons are made for fixed word length. This could even be viewed as a weak departure from the original idea of alignment since sharing l-mers is equivalent to recognizing an alignment between identical segments. For this reason, various methods have been proposed in order to derive combined distance metrics that contain information about all resolutions. Thus it is possible to achieve complete independence from the contiguity of conserved segments [41].
All of the l-mer based distance measures completely ignores the ordering of the l-mers inside the input sequences.

2.2.2 Methods based on substrings

In this category methods look for similarities and differences among the substrings of the genomes or proteomes that are studied. We particularly want to the present two approaches which we have used to compare our results.

2.2.2.1 Average Common Substring (ACS)

The *Average Common Substring (ACS)* method, described by Ulitsky *et al.* in [40], calculates the pairwise genome sequence distances $L(X, Y)$ between each couple of sequences X and Y by finding the average length of the longest substrings starting at every sequence position that are shared between them. Clearly, the shortest the average common substring is, the less the two sequences are similar and viceversa. The distance between two sequences X and Y is calculated as follows:

$$d_s(X,Y) = \left[\frac{log|Y|}{L(X,Y)} + \frac{log|X|}{L(Y,X)} - \left(\frac{log|X|}{L(X,X)} + \frac{log|Y|}{L(Y,Y)} \right) \right] /2 \tag{1}$$

where the subtraction at the second member of the equation consists in a correction factor which aims to ensure the zero-diagonality of the distance matrix. Being the simple distance, for instance, from X to Y, defined as

$$d(X,Y) = \frac{log|X|}{L(Y,X)} - \frac{log|X|}{L(X,X)} \tag{2}$$

we can recognize in formula (1) the correction of the non-symmetry of formula (2) by averaging $d(X, Y)$ and $d(Y, X)$:

$$d_s(X,Y) = \frac{d(X,Y) + d(Y,X)}{2} \tag{3}$$

Then the construction of the generalized suffix tree is performed in $O(|X| + |Y|)$, by which the ACS distance can also be calculated. The overall work, for comparing s sequences of length up to n, requires $O(s^2 \cdot n)$ time.
Ulitsky *et al.*, in their work, have used the "lightweight suffix array"[26] as data structure leading to an overall time complexity of $O(s^2 \cdot n \cdot log(n))$ with particular advantage in terms of smaller space requirement and in faster search of subsequences, with a temporal complexity of $O(log(n))$.
In terms of space complexity, each suffix array requires $O(n)$ space, and an additional $O(n/\sqrt{\log n})$ is required in the construction stage and then reclaimed.

2.2.2.2 An information-based sequence distance

In a series of two papers, Chen *et al.* [14] and Li *et al.* [22] develop tools that are inspired by Kolmogorov complexity to compress biosequences, and then to compute pairwise distance based on the compression outcome. Since Kolmogorov complexity is incomputable, what their *GenCompress* algorithm actually uses is a generalizion of the Lempel-Ziv algorithm [21][43]. This compression algorithm reportedly outperforms other DNA compression methods. It has been applied to construct a whole mitochondrial genome phylogeny. The distance in [22] is defined as follows:

$$d(X, Y) = 1 - \frac{K(X) - K(X|Y)}{K(XY)}$$

where $K(X|Y)$ is the conditional Kolmogorov complexity (or algorithmic entropy) of X given Y. $K(X|Y)$ is defined as the length of the shortest program causing a standard universal computer to output X on input Y, and $K(X)$ is defined as $K(x|\epsilon)$, where ϵ is the empty string. So we may say that it computes the randomness of X given Y.
So the numerator indicates the amount of information Y knows about X, and we know that

$$K(X) - K(X|Y) \approx K(Y) - K(Y|X)$$

[23] i.e. there is a mutual algorithmic information between X and Y. The denominator $K(XY)$, being the amount of information in the string X concatenated with Y, serves as a normalization factor such that the distance $d(X, Y)$ ranges between 0 when Y "knows" all about X and 1 when Y "knows" nothing about X.

3 Software and tools

In this section we will describe the tools that have been used in this work and the machines over which our algorithms have run.

3.1 Phylip

`Phylip`[5] is a tool, developed by Joseph Felsenstein of the Department of Genome Science at the University of Washington, used for inferring phylogenies. It is available free over the Internet, and written to work on as many different kinds of computer systems as possible. The source code is distributed (in C), and executables are also distributed, for different operating systems: Windows (95/98/NT/2000/me/xp/Vista), Mac OS X, and Linux systems.
It consists of 35 individual programs, broadly grouped into these categories:

- molecular sequence methods;
- distance matrix methods;
- gene frequencies and continuous characters analysis;
- tree drawing, consensuns, tree editing, tree distances.

Often the output of a program is the input for other programs within the package. Therefore for a typical analysis the user makes choices regarding each aspect of an analysis and chooses specific programs accordingly.
Programs are run interactively via a text-based interface that provides a list of choices and prompts users for input.
The programs we have used in order to perform our test are:

- **consense**: this program has been used to build some consensus tree from our own results, in order to compare it with our reference consensus, taken from [50], and used as meter of comparison in [40]. It provides four ways to derive the consensus tree:
  - Strict: a set of species must appear in all input trees to be included in the strict consensus tree.

[5] PHYLogeny Inference Package

– Majority Rule extended (MRe): any set of species that appears in more than 50% of the trees is included. The program then considers the other sets of species in order of the frequency with which they have appeared, adding to the consensus tree any which are compatible with it until the tree is fully resolved. This is the default setting.

– Maximum likelihood (Ml): the user is asked for a fraction between 0.5 and 1, and the program then includes in the consensus tree any set of species that occurs among the input trees more than that fraction of then time. The Strict consensus and the Majority Rule consensus are extreme cases of the Ml consensus, being for fractions of 1 and 0.5 respectively.

– Majority Rule: a set of species is included in the consensus tree if it is present in more than half of the input trees.

We have used MRE method since it has been empirically proved that it builds the most likely tree. In Figure 3.1 we report the settings used for this program.

```
Consensus tree program, version 3.695

Settings for this run:
 C         Consensus type (MRe, strict, MR, Ml):  Majority rule (extended)
 O                              Outgroup root:  No, use as outgroup species  1
 R                   Trees to be treated as Rooted:  No
 T             Terminal type (IBM PC, ANSI, none):  ANSI
 1                   Print out the sets of species:  Yes
 2           Print indications of progress of run:  Yes
 3                              Print out tree:  Yes
 4                 Write out trees onto tree file:  Yes
```

Figure 3.1: Settings used for the program `consense`

- **neighbor**: this program implements the Neighbor-Joining method of Saitou and Nei [34 and the UPGMA method of clustering (see sections 2.1.4.1 for a better description) and the built trees do not assume an evolutionary clock, so that the program actually performs unrooted trees.
  The input of the program is the distance matrix, which must be written in a particular form: the first row reports the number l of species of the dataset, then the following l rows report the name of each species (it must be 10 characters long so, longer words have to be shortened, while shorter ones have to padded with spaces up to the achievement of the ten characters) and the distance from each of the other taxa; the output of `neighbor` is written in the Newick notation[48]. Figure 3.2 describes the settings utilized.

- **treedist**: the input of this program is a set of trees among which we want to compute the Robinson-Foulds (RF) distances. It provides two kinds of distance: the `Branch Score Distance`, defined by Kuhner and Felsenstein, and the more widely known `Symmetric Difference` or Robinson-Foulds (RF) distance, which we have used since it does not use branch length information, but

```
Neighbor-Joining/UPGMA method version 3.695

Settings for this run:
  N       Neighbor-joining or UPGMA tree?  Neighbor-joining
  O                       Outgroup root?  No, use as outgroup species  1
  L         Lower-triangular data matrix?  No
  R         Upper-triangular data matrix?  No
  S                       Subreplicates?  No
  J     Randomize input order of species?  No. Use input order
  M           Analyze multiple data sets?  No
  0   Terminal type (IBM PC, ANSI, none)?  ANSI
  1    Print out the data at start of run  No
  2  Print indications of progress of run  Yes
  3                       Print out tree  Yes
  4       Write out trees onto tree file?  Yes
```

Figure 3.2: Settings used for the program `neighbor`

only the tree topologies. This distance counts how many partitions there are, among each couple, that are on one tree and not on the other and viceversa, so it can range from 0 to twice the number of internal branches.
Since we are dealing with fully bifurcating trees, the RF-distance proceeds by hops of two.

```
Tree distance program, version 3.695

Settings for this run:
 D                       Distance Type:  Symmetric Difference
 R         Trees to be treated as Rooted:  No
 T    Terminal type (IBM PC, ANSI, none):  ANSI
 1  Print indications of progress of run:  Yes
 2                Tree distance submenu:  Distances between all possible
                                            pairs in tree file.
```

Figure 3.3: Settings used for the program `treedist`

- **drawtree**: this program receives in input the output of `neighbor` and provides the user with a graphical rappresentation of the phylogenetic tree.

```
Unrooted tree plotting program version 3.695

Here are the settings:

 0  Screen type (IBM PC, ANSI)?  ANSI
 P       Final plotting device:  Postscript printer
 (Preview not available)
 B          Use branch lengths:  No
 L             Angle of labels:  branch points to Middle of label
 R            Rotation of tree:  90.0
 I     Iterate to improve tree:  Equal-Daylight algorithm
 D  Try to avoid label overlap?  No
 S      Scale of branch length:  Automatically rescaled
 C   Relative character height:  0.3333
 F                         Font:  Times-Roman
 M          Horizontal margins:  1.65 cm
 M            Vertical margins:  2.16 cm
 #           Page size submenu:  one page per tree
```

Figure 3.4: Settings used for the program `drawtree`

3.2 OpenMP

OpenMP (Open Multi-Processing) is an API that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran, on most processor architectures and operating systems, including GNU/Linux, Mac OS X, and Windows platforms.
We used it to parallelize our algorithms with a great advantage in time performances.

3.3 Qt

We used `Qt` library to realise a graphic user interface (GUI) for our software. `Qt` is a cross-platform and open source application framework developed by Nokia's Qt team.

It uses the C++ programming language and allows the development of GUI programs through the use of graphic elements (widgets). There are several considerations which led to this implementative choice: particularly `Qt`-based graphical interfaces are very intuitive, user-friendly and the library has a huge, well-done documentation.

3.4 Development Environment

Our algorithms, which will be described in chapter 4, have run on the IBM p570 server, a multiprocessor developed by IBM[6], featuring 24 POWER5 cores. Its specifications are:

- **Processor**: 24 IBM POWER5 @ 1.9 GHz
- **RAM**: 48 GB
- **Hard disk**: 38.7 TB
- **Operating system**: AIX 5L V5.3

The POWER5 is a dual-core microprocessor, with each core supporting one physical thread and two logical threads, for a total of two physical threads and four logical threads, so the architecture makes the chip appear as a four-way symmetric multiprocessor to the operating system. It supports the 64-bit PowerPC architecture. A single die contains two identical processor cores, each supporting two logical threads.

We also used the server BLADE of the Department of Information Engineering of the University of Padova. It consists in 14 DELL PowerEdge M600 computing "blades", each equipped with:

- 2 Processors quad core Intel Xeon E5450 (12MB Cache, 3.00 GHz)
- 16 GB RAM
- 2 72GB hard drive in RAID-1 (mirroring)

Though this processor outperforms the IBM POWER5, when using all its cores the IBM p570 server runs faster than the server BLADE so we utilized it for performing our tests.

The machine over which the tool `Phylip` has run is an HP Pavilion dv6, with the following specifications:

- **Processor**: AMD Turion(tm) II P520 Dual-Core Processor @ 2.4 GHz
- **RAM**: 3.6 GB
- **Hard disk**: 500 GB
- **Operating system**: Ubuntu 13.10 (64-bit)

[6]International Business Machines Corporation

4 Methods and materials

In this chapter we will present and describe the techniques that have been studied to infer phylogeny. Our approaches are alignment-free, and particularly they are based on l-mers frequency.

4.1 Approach based on l-mers with mismatches analysis

The main idea of this work derives from the fact that one of the factors that most contribute to the evolutionary process is the presence of errors, rearrengements and mutations in the transmission of genetic inheritance. Mutations in fact determine the so-called genetic variability, or the condition for which the organisms differ in one or more characters. On this variability through recombination, natural selection operates, which promotes favorable mutations at the expense of unfavorable or even death.
A mutation modifies the genotype[7] of an individual and can possibly alter the phenotype[8] depending on its characteristics and interactions with the environment. A DNA sequence can be altered in a number of different ways: point mutations, for example, exchange a single nucleotide for another; also insertions or deletions can occur, both in small and large scale.

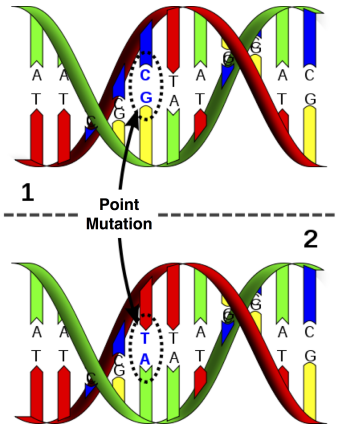

Figure 4.1: Point mutation on a DNA sequence.

The presence of a certain percentage of substrings of a genome into another is certainly evidence of a specific similarity between the two sequences, but the small local mutations are likely to prevent the recognition of an adequate degree of relationship. The key idea is therefore to allow that two substrings of l charachters (l-mers) belonging to different genomes can be recognized as a mutual occurrence of the l-mer in a genome in the other one, even though a certain number of differences occurs, since it can be due to the evolutionary process. So we hypothesized that

[7]The set of all genes that make up the DNA of an organism or a population.

[8]The set of all observable characteristics of a living organism, so its morphology, its development, its biochemical properties and physiological inclusive of behavior.

approximate string matching could provide a greater amount of information.
The problem of approximate string matching is widely studied in Computer Science and different kinds of distances are available: two of the most popular distances are the *edit distance* as defined by Vladimir Levenshtein, where insertion, deletion and mismatches are allowed and the *Hamming distance*, where only mismatches, or differences, are allowed.
Dealing with k-difference problem, we used *Hamming distance*, which, for two sequences $X = x_0x_1...x_{k-1}$ and $Y = y_0y_1...y_{k-1}$ defined over an alphabet Σ, has the following definition:

$$d_{Hamming}(X, Y) := \sum_{i=0}^{k-1} neq(x_i, y_i) \tag{4}$$

and $neq(a, b)$ is defined as:

$$neq(a, b) := \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases} \tag{5}$$

First of all we started by considering the k-difference problem, which finds all the occurrences (with at most k mismatches) of a given word w of length k in a text T of length n. Well-known algorithms achieve solutions in $O(n\sqrt{k \cdot \log k})$ [2], $O(n \cdot k)$ [20] and $O(n\sqrt{l \cdot \log k})$ [1].
We are interested in finding the occurrences with at most k mismatches of all the words in a text X in another text Y so we extended the work presented by Pizzi in [28], which efficiently calculate the occurrences of all the words of a text in the text itself in $O(n^2)$, in order to apply it to different sequences.

Given two sequences X and Y of length n_X and n_Y, a fixed pattern length l and a maximum number of allowed mismatches k ($k \leq m$), we aim to build a matrix M of size $(n_X - l + 1) \times (n_Y - l + 1)$ in which the cell M_{ij} contains the *Hamming* distance between the substring $x_i = X[i, i + l - 1]$ and $y_j = Y[j, j + l - 1]$.
If X and Y are the same sequence, M is a symmetric matrix and we do not need to calculate it all but only its upper triangular portion. But in our case ($X \neq Y$) the entire matrix has to be computed.
Figure 4.2 describes the procedure: red cells are those to be computed with classical method. The first row of the matrix is built using the naïve method which checks each position of the substrings. We then proceed by filling M row by row: the first element of each row need to be computed naively, while element M_{ij} ($i, j > 0$) is initially set to $M_{i-1,j-1}$. Then, sliding a window of size l over the row, if $x_{i+l-1} \neq y_{j+l-1}$ we acquire a difference and so M_{ij} is incremented while if $x_{i-1} \neq y_{j-1}$ M_{ij} is decremented since an error is lost. Time complexity is $O(n^2)$ with a small constant factor.
As all of the l-mer based distance measures, also our technique completely ignores the ordering of the l-mers inside the input sequences. Hence, if the selected l value (the length of the substrings) is too small, very distantly related organisms may be assigned a small evolutionary distance (in the extreme case where l is 1 , two organisms will be treated as being identical if the frequency of each nucleotide/amino-acid is the same in the two corresponding sequences). In the other extreme, the l-mers

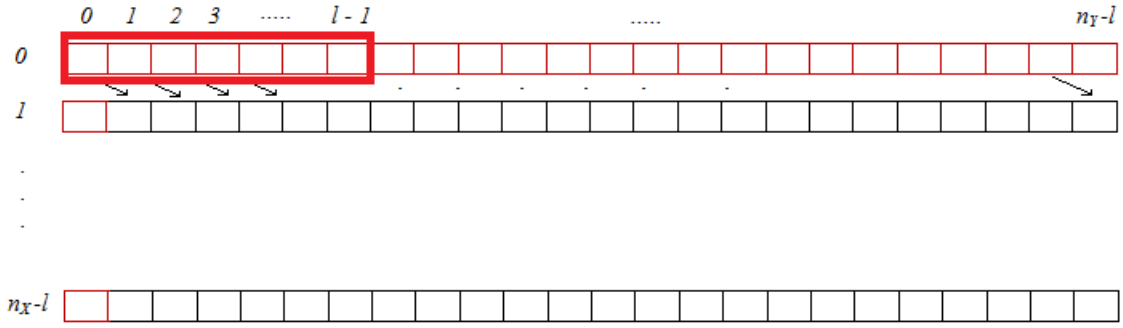

Figure 4.2: Graphical description of algorithm 1

should have a length that is somewhat below the average distance between mismatches if the input sequences were aligned. So the selected l value should not be too large nor too small; the identification of the best patter length *a priori* is a non-trivial problem.

Algorithm 1 Matrix_of_mismatches

```
1: INPUT: sequences X and Y
2: OUTPUT: matrix of Hamming distances
3: n_X ⟵ size of X
4: n_Y ⟵ size of Y
5: compute first row vector with classic k-mismatch algorithm
6: for i ⟵ 1 to n_l do
7:    calculate first element with classic algorithm
8:    for i ⟵ 1 to n_1 − m do
9:       M_ij ⟵ M_{i−1,j−1}
10:      if x_{i+l−1} ≠ y_{j+l−1} then
11:         M_ij ⟵ M_ij + 1
12:      end if
13:      if x_{i−1} ≠ y_{j−1} then
14:         M_ij ⟵ M_ij − 1
15:      end if
16:   end for
17: end for
```

Since the final purpose is to build the phylogenetic tree we need to translate the information given by algorithm 1 into a distance between the species. We defined two measures.

4.1.1 Distance based on coverage

Given a set S of s sequences, a fixed pattern length l and a maximum number of allowed mismatches k ($k \leq m$), algorithm 2 considers each ordered couple of sequences

X and Y of length n_X and n_Y respectively and computes how many mismatches we have to allow for each substring of the first sequence to be found in the second one. So we build a vector whose i-th element is the minimum number of mismatches with which $x_i = X[i, i + l - 1]$ is present in Y and the output of the overall algorithm is the set of s^2 of such vectors (one for each possible pair).
For doing so, we need to build the matrix M of differences as described in algorithm 1 keeping a vector min of integers of length n_X; the first element of row i of M is used as the initialization value for $min[i]$. Then we only need to compare each element of the row with the current minimum value and replace it if it is smaller.

Algorithm 2 single_occurrences

INPUT: set of the species
pattern size l
OUTPUT: set of vectors of minimum number of mismatches with which each word of one species is present an other
for each couple X, Y of the dataset **do**
  $n_X \longleftarrow$ size of X
  $n_Y \longleftarrow$ size of Y
  initialize the vector min of minimum element of each row
  compute first row vector with classic k-mismatch algorithm
  find the minimum element $min[0]$ of the first row vector
  for $i \longleftarrow 1$ **to** $n_X - l$ **do**
    copy the first $n_Y - l$ elements of the previous row in the last $n_Y - l$ elements of the current row
    calculate first element M_{i0} with naïve method
    $min[i] \longleftarrow M_{i0}$
    for $j \longleftarrow 1$ **to** $n_Y - l$ **do**
      if $x_{i+l-1} \neq y_{j+l-1}$ **then**
        $M_{ij} \longleftarrow M_{ij} + 1$
      end if
      if $x_{i-1} \neq y_{j-1}$ **then**
        $M_{ij} \longleftarrow M_{ij} - 1$
      end if
      if $M_{ij} < min[i]$ **then**
        $min[i] \longleftarrow M_{ij}$
      end if
    end for
  end for
end for

In order to improve the time performances we parallelized the algorithm by dividing the set of rows of between the processors; thus, if p processors are used, each one of them is assigned to a set of $\frac{n_X}{p}$ rows. The drawback of such an escamotage is that there will be p rows computed in the naïve way since the first row of each section can not be derived from the previous one, but this does not affect the overall improvements in the runtime as it can be seen in Figure 4.3 where we reported the execution time required for determinating this measure of similarity between the 34 taxa of the dataset (see sec. 5.1.1 for details on it), for a pattern length $l = 100$ and varying the number of processors involved. Let us underline that this algorithm is independent of the number of mismatches we are willing to allow, so we only need to run it once for each pattern length l we want to test.
Tests have been performed on the IBM p570 server which allows to use up to 24 processors.

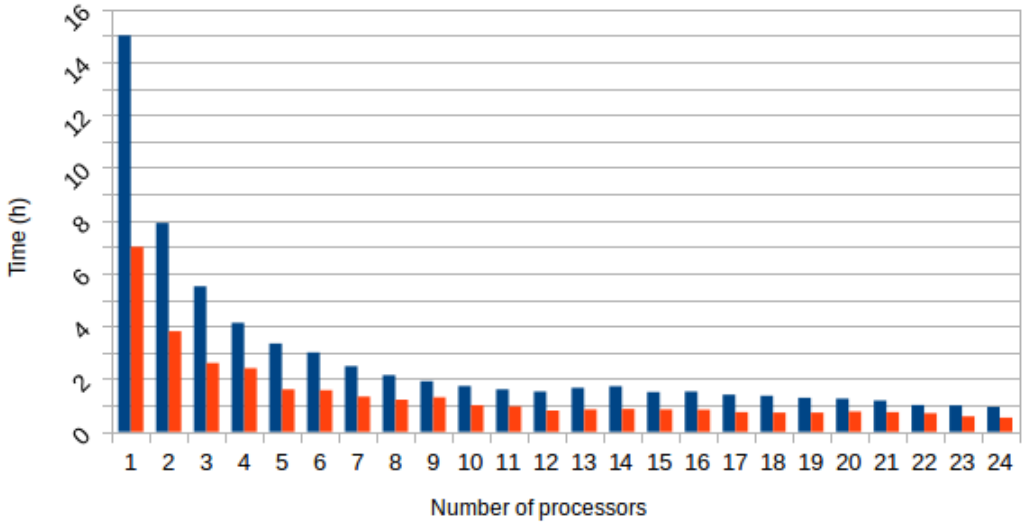

Figure 4.3: Improvement in time due to parallelization on the IBM p570 server. The blue bars refers to the implementation described in alg 2, while the orange bars represent the execution times for directly calculating the distance matrix for a specific couple of values (l, k).

Values for a small number of processors are quite elevated due to specifications of the IBM p570 server. The cluster BLADE would take about 3 hours to execute the program with a single core but it scales a little slower as we can see in Figure 4.4.

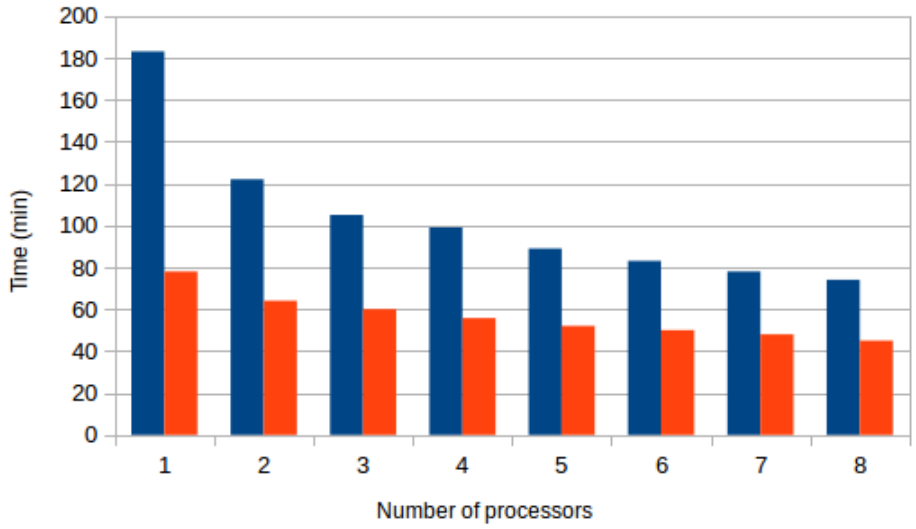

Figure 4.4: Improvement in time due to parallelization on the cluster BLADE. The blue bars refers to the implementation described in alg 2, while the orange bars represent the execution times for directly calculating the distance matrix for a specific couple of values (l, k).

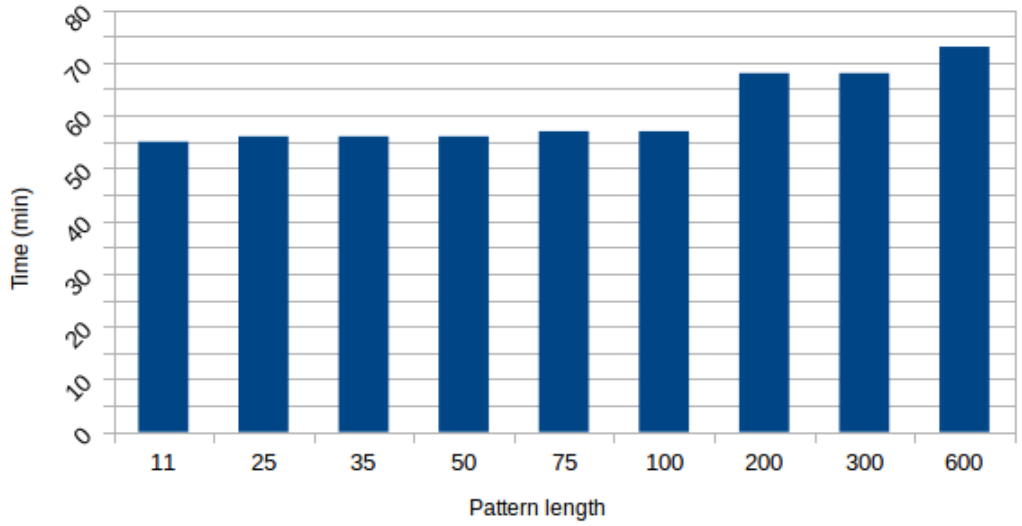

Figure 4.5: Small time dependency on the pattern length.

The algorithm's time complexity for each pair of genomes is quadratic in the length of the sequences, due to the double **for** cycle. So, for a dataset of s strings of length up to n and a running machine with p processor, it takes $O(\frac{s^2 \cdot n^2}{p})$.
With regard to the space required, we do not need to store all the matrix M but

just the current and the previous rows. So space complexity is $O(p \cdot n)$.
As can be seen in the graphic of Figure 4.5, algorithm 2 is slightly dependent on the length of the pattern l. The small increment is due to the calculation of the first row of each parallel block: indeed it costs $O((n-l+1) \cdot l)$, which grows with $O(nl)$.
We also considered the implementation by diagonals, in which instead of distributing to each processor a set of rows of the matrix M, we assigned them $\alpha = \frac{n_Y - l}{p}$ diagonals of the upper triangular matrix first and then $\beta = \frac{n_X - l}{p}$ diagonals of the lower triangular matrix. However this lead to two problems:

1. while rows have all the same length, diagonals do not: so work is not generally equally distributed among the various processors.

2. suppose we are operating on the upper triangular portion of M: processor i starts calculating $M[0][\alpha \cdot i]$ so there are p simultaneous attempts to access the location $min[0]$ in order to find the minimum and thus it is necessary to enter in a region of mutual exclusion, ruled by a `lock`. Then the same happens for the next locations.
   Consequently a lot of time is spent waiting for that locked region to be accessible and in this case, the more processors are used, the more time it takes the algorithm to run. For $l = 100$ the following execution times have been measured on the same dataset on the IBM p570 server:

   - **Implementation row-by-row**: $\sim 55 - 70$ minutes
   - **Implementation by diagonal**: ~ 5 hours

Next step is the calculation of the matrix of distances itself, since we will use Neighbor-Joining method for deriving the tree.
We now have s^2 vector (s for each species); as described in algorithm 3, for calculating the distance between X of size n_X and Y of size n_Y we need to load the vector v_{XY} which refers a sort of presence of of each l-mer of X on Y. Once a maximum number of mismatches k has been chosen, we counted how many positions of the vector are smaller or equal to k and thus obtain a measures, r_{XY}, which we then normalized on the size of X in order to get a sort of percentage of the presence of a sequence on the other one.

$$cov^k_{XY} := \frac{r_{XY}}{n_X}$$

Now we have a measure of similarity while we are interested in a *distance*. Since $cov^k_{XX} \in [0, 1]$ we consider

$$d^k_{XY} = 1 - cov^k_{XY}$$

When considering the same species, each substring of X is obviously present in X with 0 mismatches and so we will have that

$$\begin{gathered} r_{XX} = n_X \quad \forall k \\ \Downarrow \\ cov^k_{XX} = 1 \\ \Downarrow \\ d^k_{XX} = 0 \end{gathered}$$

This last measure d^k_{XY} is not symmetric, so we compute

$$dist_{XY} = \frac{d^k_{XY} + d^k_{YX}}{2}$$

which is our final distance measure between the two sequences.

Algorithm 3 calculate_distance

1: **INPUT:** s^2 vectors
minimum number k_{min} of allowed mismatches
maximum number k_{min} of allowed mismatches
patter size l
2: **OUTPUT:** matrix of distances $dist$
3: **for** $k \longleftarrow k_{min}$ **to** k_{max} **do**
4: **for** $x \longleftarrow 1$ **to** s **do**
5: **for** $y \longleftarrow i$ **to** s **do**
6: Load the vector v_{XY} of size n_X between species x and y
7: Load the vector v_{YX} of size n_Y between species x and y
8: $r_{XY} \longleftarrow 0$
9: $r_{YX} \longleftarrow 0$
10: Calculate how many cells of v_{XY} are less or equal to k and save this number in r_{XY}
11: Calculate how many cells of v_{YX} are less or equal to k and save this number in r_{YX}
12: $cov^k_{XY} \longleftarrow r_{XY}/n_X$
13: $cov^k_{YX} \longleftarrow r_{YX}/n_Y$
14: $d^k_{XY} \longleftarrow 1 - cov^k_{XY}$
15: $d^k_{YX} \longleftarrow 1 - cov^k_{YX}$
16: $dist[x, y] \longleftarrow \frac{d^k_{XY}+d^k_{YX}}{2}$
17: $dist[y, x] \longleftarrow dist[x, y]$
18: **end for**
19: **end for**
20: **end for**

The time complexity of algorithm 3 is $O(s \cdot n \cdot (k_{max} - k_{min} + 1))$ while space complexity is $O(n + s^2) \sim O(n)$. Algorithm 3 has been proven to be practically independend from the pattern length and, on a standart notebook, it takes ~ 1.8 seconds for each number of number of mismatches we are interested to test.
Since we believe it is possible that, given a certain dataset, there exists at least a set of values of pattern length l and maximum number of mismatches k which ensure the best reconstruction of the phylogenetic tree, we also provide an algorithm which directly calculate the distance matrix for a pair of values (l, k) which it receives in input. In this case two vectors V_x and V_y are initialized: one for the current X string and the other for the current Y string. If substring $X[i, i + 1...i + l - 1]$ matches with $Y[j, j + 1...j + l - 1]$, then $V_x[i]$ and $V_y[j]$ are set to 1. The number of 1's of V_ι is the number of words of species ι which occur in the other species. This way we

do not need to consider each ordered pair any more: so the procedure is executed $\frac{s^2}{2}$ times.

Instruction for running

Both algorithm 2 (single_occurrences.cpp) and 3 (calculate_distance.cpp) are written in `C++`. Here we provide the instruction to run them.

- **single_occurrences.cpp**

  In order to compile the code on a Unix-like operating system, here is the command:

  ```
  g++ single_occurrences.cpp -o <executable_name> -fopenmp -O3
  ```

  where the flag `-fopenmp` load the OpenMP library, while `-O3` is the optimization flag.
  The executable must be in the same directory that contains the files of the dataset in the format `<name>.fna` (the genomes or proteomes must be in the FASTA format). Moreover we need to provide a text file `infile` that lists the names of the species of the dataset, one per line and without the extension `.fna`.
  The command to execute the program is:

  ```
  ./<executable> <infile> <pattern_length> <method> [<opt> k_max]
  ```

  The flag `<methods>` can be **-m** if we need to build the vectors of minimum number of mismatches according to algorithm 2 because the correct number of mismatches is not known nor guessable; otherwise it is set to **-k**. In this two more parameters are needed: the first is the flag `<opt>` which can be **-v** if we want the program to consider the maximum number of allowed mismatches (the next and last parameter k_{max}) as the number it represents, or **-p** if istead we want to consider it in terms of percentage. The vectors which constitute the output are saved in the subdirectory `<pattern_length>/` which has to be created by the user.
  They are named `minNumberOfOccurrences_`*SpeciesX*`_VS_`*SpeciesY*`.txt`.

- **calculate_distance.cpp**

  To compile type:

  ```
  g++ calculate_distance.cpp -o <executable_name>
  ```

  Also this executable must be in the same directory of the dataset. Command for execution is:

```
> ./<executable> <infile> <pattern_length> <option> k_min k_max
```

If flag `option` is **-v**, the program calculates the matrices of distances for all the values $k \in [k_{min}, k_{max}]$. If instead `option` is **-p**, k_{min} and k_{max} are intepreted as the minimum and maximum percentage of allowed mismatches.

4.1.2 Method based on multiple occurrences

We implemented another measure, which counts the number of times each substring of X is present in Y. We provide as input the maximum number of allowed mismatches k_{max} we are interested in for a given pattern length l. Intuitively we will create a matrix Γ of $k_{max} + 1$ rows of length $n_X - l + 1$.Location $\Gamma[i][j]$ contains the number of occurrences of $X[j, j+1...j+l-1]$ in Y with exactly k mismatches. Since we are not interested in knowing the composition of the vector but only in the total "presence" of X in Y, Γ is reduced to a single column whose values are the sums of all the cells of each row.
Considering occurrences with exactly k mismatches allows to increment one single value at a time, instead of $k + 1$; it is then sufficient to sum the first $k + 1$ values of the column Γ in order to get the number of occurrences with *at most* k errors. This algorithm has been parallelized, too, with an improvement comparable to that shown in Figure 4.3. It is dependent on the choice of k_{max}, especially when it assumes values higher than the 50% of the pattern length, as can be seen in Figure 4.6.

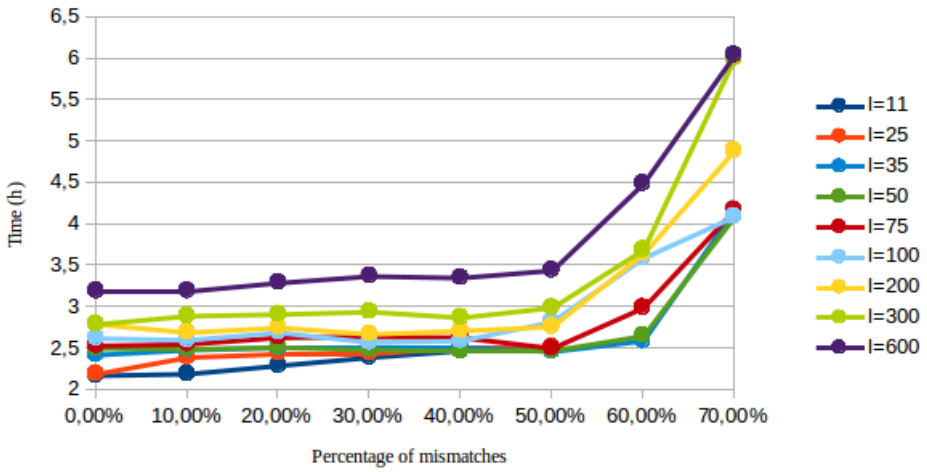

Figure 4.6: Dependence of algorithm 4 on k_{max}.

For building the matrix D^k of distance the following steps are performed:

1. Let R^k_{XY} be the total number of occurrences of any words of X in Y with at most k mismatches. In order to ensure that the condition $D^k_{XX} = 0$ is verified, we calculate
$$dist^k_{XY} := 1 - \frac{R^k_{XY}}{R^k_{XX}}$$
2. Analogously calculate $dist^k_{YX}$
3. Correct the asymmetry of this measure and so set
$$D^k_{XY} = D^k_{YX} = \frac{dist^k_{XY} + dist^k_{YX}}{2}$$

For a dataset of s sequences of length up to n, this algorithm takes $O((s^2 \cdot n^2)/p)$ in time and requires $O(k_{max} + n \cdot p) = O(n \cdot p)$ space.
This kind of measure has revealed to be inefficient since it does lead to reliable phylogenetic trees.

Instruction for running

- **multiple_occurrences.cpp**

  Compilation is done through the command

  ```
  > g++ multiple_occurrences.cpp -o <executable_name> -fopenmp -O3
  ```

  while execution requires four parameters: a text file with a list of the species of our dataset, the pattern length, an option (**-v** or **-p**) and the maximum number of mismatches interpreted as absolute value or percentage according to the option, as described in the previous section.
  The command is:

  > `> ./multiple_occurrences <infile> <option>` k_{max}

  Here the output has the format `multiple_`*`SpeciesX`*`_VS_`*`SpeciesY`* and it is stored in the subdirectory `/<pattern_length>`.

- **calculate_distance2.cpp**

  The specifications and parameters for this algorithm are the same of `calculate_distance.cpp`.

Algorithm 4 multiple_occurrences

1: **INPUT:** set of the species
maximum number k_{max} of allowed mismatches
pattern size l
2: **OUTPUT:** coverage between each pair of species
3: **for** each couple X, Y of the dataset **do**
4: $n_X \longleftarrow$ size of X
5: $n_Y \longleftarrow$ size of Y
6: initialize the vector v of length $k_{max} + 1$
7: **for** $i \longleftarrow 0$ **to** $n_Y - l$ **do**
8: calculate the number of differences M_{0i} between word $X[0]$ and $Y[i]$ with the classic textitk-mismatch algorithm
9: **if** $M_{0i} \leq k_{max}$ **then**
10: $v[M_{0i}] \longleftarrow v[M_{0i}] + 1$
11: **end if**
12: **end for**
13: **for** $i \longleftarrow 1$ **to** $n_X - l$ **do**
14: copy the first $n_Y - m$ elements of the previous row in the last $n_Y - m$ elements of the rrent row
15: calculate first element M_{i0} of the current row with naïve method
16: **if** $M_{i0} \leq k_{max}$ **then**
17: $v[M_{i0}] \longleftarrow v[M_{i0}] + 1$
18: **end if**
19: **for** $j \longleftarrow 1$ **to** $n_Y - l$ **do**
20: **if** $x_{i+l-1} \neq y_{j+l-1}$ **then**
21: $M_{ij} \longleftarrow M_{ij} + 1$
22: **end if**
23: **if** $x_{i-1} \neq y_{j-1}$ **then**
24: $M_{ij} \longleftarrow M_{ij} - 1$
25: **end if**
26: **if** $M_{ij} < k_{max}$ **then**
27: $v[M_{ij}] \longleftarrow v[M_{ij}] + 1$
28: **end if**
29: **end for**
30: **end for**
31: **end for**

4.1.3 Graphical user interface (GUI)

We provide the final user with a graphical interface (Figure 4.7) intended for the pairwise analysis.
It is composed of three components:

- ***Control panel***: this area is dedicated to the control of the parameters for executing the programs and visualizing the results. In the upper part there are two buttons, `Choose sequences` and `Load results`: the former loads a couple of sequences (either DNA or amino-acids sequences can be treated) in

the FASTA format (**.fna*) while the latter loads the results file of a previous analysis. The user can also load a single subsequence; in this case the analysis performed assumes the sense of an autocorrellation and can help in understanding which are the parts that more characterize the string. An error occurs if more than two strings are loaded.
The control panel is then constituted by two pages:

  - `Parameters`: here we find the parameters for choosing which algorithm to execute, the pattern length and the number of mismatches. The button `PLAY` starts the execution while clicking `RESET` the application is set for a new analysis.
  - `Visualization`: here the users can choose the display settings and it is the only used page where results are loaded.

- ***Visualization window***: it provides a graphical description of the results, which goes beyond the task of the phylogenetic trees reconstruction and allows to *a posteriori* analysis on the composition of the sequences.

- ***Console***: this element provides the user with information regarding the actions performed and a textual description of the graphics generated.

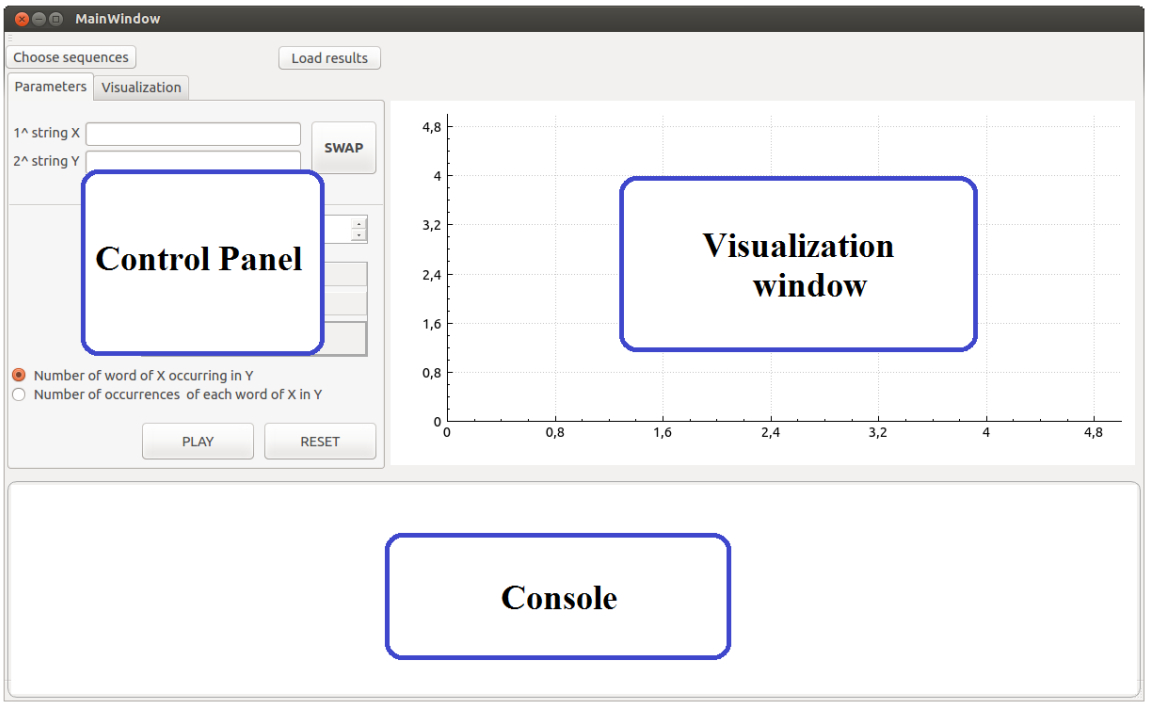

Figure 4.7: Graphic user interface

At the top of the GUI, on the ***Menu bar***, the path *File*⟶*Settings* open a widget that allows to set the directory of the dataset and the one where to store the results. The results' directory must be provided with subdirectories named as the

pattern lengths, before running the application. The button `Restore` set the default directories i.e. the directory where the GUI files are stored.

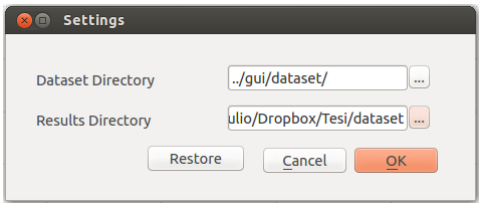

Figure 4.8: Settings' widget.

4.1.3.1 Working mode

In order to help the user using the application, warning messages are displayed that prevent from performing wrong or non-sense actions. Moreover, according to the kind of analysis the user has chosen to make, some buttons are enabled or disabled.

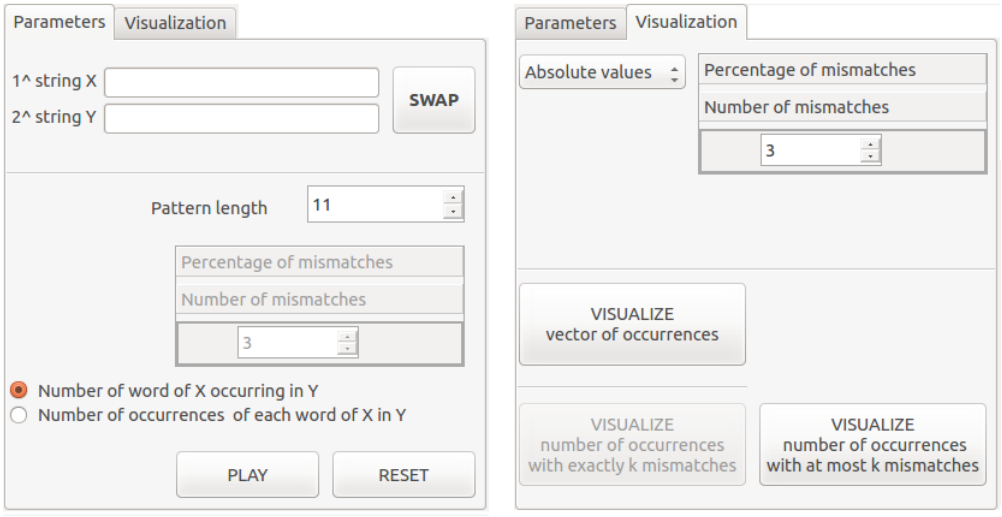

Figure 4.9: Control Panel

After choosing a couple of strings and a pattern length (expressed as a pure number or a percentage), the analysis described by algorithm 2 can be perfomed by clicking on the voice `Number of words of X occurring in Y`. Since this analysis is independent on the number of mismatches, the part of the GUI relative to its setting is disabled as can be seen in Figure 4.9. The output is saved in the same format of that produced by algorithm 2 and executed for example on the p570 server. This allow the user even to load a results he/she already has without executing the algorithm again for that particular couple of sequences.
Then, in the visualization page, a maximum number of mismatches k_{max} can be set, and it is shown how many words of X are present in Y with exactly or at most k mismatches, for each $k \leq k_{max}$, also in terms of frequency.
Since the algorithm described in section 4.1.2 has proved to be inefficient for the purpose of phylogeny derivation (as we will describe in the next chapter), it is has not been implemented in the GUI. Instead another kind of analysis can be performed: indeed it could be interesting, given a pattern length and a maximum number of mismatches k, how many times each word of X occurs in Y with at most k mismatches. Thus algorithms 5 is executed.

Algorithm 5

1: **INPUT:** species X and Y
maximum number k of allowed mismatches
pattern size l
2: **OUTPUT:** vector of the number of occurrences of each word of X in Y
3: $n_X \longleftarrow$ size of X
4: $n_Y \longleftarrow$ size of Y
5: initialize the vector v of length n_X
6: **for** $i \longleftarrow 0$ **to** $n_X - l$ **do**
7: **for** $j \longleftarrow 0$ **to** $n_Y - l$ **do**
8: calculate the number of differences k_i between $X[i]$ and $Y[j]$ with dynamic programming described in algorithm 1
9: **if** $k_i \leq k$ **then**
10: $v[i] \longleftarrow v[i] + 1$
11: **end if**
12: **end for**
13: **end for**

The vector produced by the algorithm can be displayed clicking the button `VISUALIZE vector of occurrences`. The button `VISUALIZE number occurrences with at most k mismatches`, instead, shows how many substrings occur never, once, twice etc.
This allows to do some considerations concerning the structure of the sequence: in particular it is interesting to see how fast the progressive increment of k makes the sequences closer.

5 Experimental results

Here we present our results on two dataset, one constituted by 34 mitochondrial DNA sequences of average length $n \approx 16000$ characters and the other one, which considers 84 retroid viruses of average length $n \approx 8000$ characters. Our analysis focused on the first dataset and we eventually performed some tests with another dataset in order to ensure that our results are not dependent on a particular set of data.
Method described in section 4.1.1 has been used. Table 1 shows the time required for running algorithm 2 over different pattern lengths. Tests have been performed on the IBM p570 server using all its 24 processors.

Pattern length	**Mitochondrial dataset**	**Viruses' dataset**
$l = 25$	56 min.	80 min.
$l = 50$	56 min.	80 min.
$l = 75$	58 min.	81 min.
$l = 100$	62 min.	81 min.
$l = 200$	68 min.	87 min.

Table 1: Time for running algorithm 2 over different pattern lengths.

As we said in section 4.1.1, tests can also be performed for a given pattern length l and an arbitrary maximum number of mismatches k. Table 2 shows the running time for this implementation. Being practically independent from k, we did not report its values in the table.

Pattern length	**Mitochondrial dataset**	**Viruses' dataset**
$l = 25$	33 min.	50 min.
$l = 50$	34 min.	50 min.
$l = 75$	36 min.	50 min.
$l = 100$	35 min.	52 min.
$l = 200$	39 min.	56 min.

Table 2: Time for running algorithm 2's variation for a single couple (l, k).

5.1 A Tree Based on Mitochondrial DNA

5.1.1 Description of the dataset

Tests have been performed over a dataset including the complete mitochondrial DNA sequences of 34 mammals, whose features are reported in Table 3.

This same dataset has been used by Li *et al.* in [22] and Utilsky *et al.* in [40]. Also Cao *et al.* used part of this dataset (the part relative to the Eutherian order) in [10].

Class	Super-order	Short name	Full name	Size
EUTHERIA	**Euarchontoglires**	GuineaPig	*Cavia Porcellus*	16801 ch
		Rat	*Rattus Norvegicus*	16313 ch
		Housemouse	*Mus Musculus*	16299 ch
		Rabbit	*Oryctolagus Cuniculus*	17245 ch
		Squirrel	*Sciurus Vulgaris*	16507 ch
		Dormouse	*Myoxus Glis*	16602 ch
		Baboon	*Papio Hamadryas*	16521 ch
		Gibbon	*Hylobates lar*	16472 ch
		Orangutan	*Pongo Pygmaeus*	16389 ch
		Gorilla	*Gorilla Gorilla*	16364 ch
		Human	*Homo Sapiens*	16569 ch
		PygmyChimpanzee	*Pan Paniscus*	16563 ch
		Chimpanzee	*Pan Troglodytes*	16554 ch
	Laurasiatheria	Dog	*Canis Familiaris*	16727 ch
		Cat	*Felis Catus*	17009 ch
		GreySeal	*Halichoerus Grypus*	16797 ch
		HarborSeal	*Phoca Vitulina*	16826 ch
		WhiteRhino	*Ceratotherium Simum*	16832 ch
		GreatRhino	*Rhinoceros Unicornis*	16829 ch
		Donkey	*Equus Asinus*	16670 ch
		Horse	*Equus Caballus*	16660 ch
		FruitBat	*Artibeus Jamaicensis*	16651 ch
		BlueWhale	*Balaenoptera Musculus*	16402 ch
		FinbackWhale	*Balaenoptera Physalus*	16398 ch
		Hippo	*Hippopotamus Amphibius*	16407 ch
		Pig	*Sus Scrofa*	16613 ch
		Sheep	*Ovis Aries*	16616 ch
		Cow	*Bos Indicus*	16341 ch
	Afrotheria	Elephant	*Loxodonta Africana*	16866 ch
		Aardvark	*Orycteropus Afer*	16816 ch
	Xenarthra	Armadillo	*Dasypus Novemcinctus*	17056 ch
METATHERIA	**Ameridelphia**	Wallaroo	*Macropus Robustus*	16896 ch
		Opossum	*Didelphis Virginiana*	17084 ch
AUSTRLOSPHENIDA	**Monotremata**	Platypus	*Ornithorhynchus Anatinus*	17019 ch

Table 3: Table of the species of the dataset

5.1.2 Results

In this subsection we demonstrate the performance of our method. Since the multiple alignments for 13 mitochondrial proteins (ATP6, ATP8, COX1, COX2, COX3, CYTB, ND1, ND2, ND3, ND4, ND4L, ND5, and ND6) of the species reported in Table 3 are available in NCBI genomes, we can compare our trees to trees that are build from these proteins using the maximum likelihood method.
A majority consensus tree has been derived from the 13 ML trees of the proteins using the program `CONSENSE` from the `PHYLIP` package. The quality of the obtained trees is judged according to RF distance with the consensus tree and to the average distance from the 13 ML trees. Moreover a visual feedback is given by the splits of the trees i.e. the correct clustering of the species:

- **Marsupials and Monotremes**: Platypus, Wallaroo and Opossum.

- **Primates**: Baboon, Gibbon, Orangutan, Gorilla, Human, PygmyChimpanzee, Chimpanzee.
- **Afrotheria and Xenarthra**: Elephant, Aardvark, Armadillo.
- **Rodents**: Guinea Pig, Rat, House Mouse, Rabbit, Squirrel, Dormouse.
- **Laurasiatheria**: Dog, Cat, Grey Seal, Harbor Seal, Hippo, White Rhino, Great Rhino, Donkey, Horse, Pig, Sheep, Cow, Finback Whale, Blue Whale, FruitBat.

We tested our algorithm over 9 different pattern length (11, 25, 35, 50, 75, 100, 200, 300 and 600) and for each of them we admitted a number of mismatches varying from 0% to 75% of the pattern length. We used `PHYLIP` to calculate the RF distance, according to the settings displayed in subsection 3.1. As concerning the Eutherian order, it has been debated which two of the three main groups of placental mammals (*Primates, Laurasiatheria*, and *Rodents*) are more closely related. Indeed by the maximum likelihood method, some proteins support the (*Laurasiatheria*, (*Primates, Rodents*)) grouping while other proteins support the (*Rodents*, (*Laurasiatheria, Primates*)) grouping.
Our work re-confirm the hypothesis of (*Rodents*, (*Primates, Laurasiatheria*)) grouping , as Cao *et al.* and Li *et al.* did.
Considering now the whole dataset, Figure 5.1 shows the distance of our trees from the reference consensus. We can notice the improvement given by the introduction of mismatches in the l-mers instead of considering exact occurrences; for pattern length $l \in [50, 100]$ and number of mismatches $k \in [20\%-25\%, 35\%]$, several of our trees have distance 16 from the reference consensus, the same results presented in [40]. The couple $(l, k) = (75, 30)$ is even closer with distance 14. Good results are obtained also for shorter pattern lengths while increasing l over 100 seems to lead to worse results.
The graphic also shows that the longer the substrings, the more mismatches (in percentage) we have to allow in order to obtain good results. In some ways this aspect seems to confirm the work presented by Cunial and Apostolico in [15], where the quality of phylogenies reconstructed is measured by comparing suitably defined sets of gapped motifs that occur in mitochondrial proteomes. Here it is asserted that the average performance of suitably defined sets of gapped motifs is comparable to that of popular string-based alignment-free methods and that extremely long and sparse motifs produce phylogenies of the same or better quality than those produced by short and dense motifs.
The maximum possible RF distance, for s taxa, is $2s-6$ which is equal to 62 for our dataset of 34 mammalian species. Indeed in the Figure 5.1 this is highest reached distance and it indicates that, for those couple (l, k), none of the branches of the majority consensus tree is found in the current tree. As we can see, all the curves reaches a stady distance from the consensus tree, when the number of mismatches gets too high: in this case indeed all the species have null distance from the others and so the tree is built simply putting in output the taxa in the same order of the input. Since the input provided is already clustered, some branches that are present in the consensus tree may be hit also in these trees.

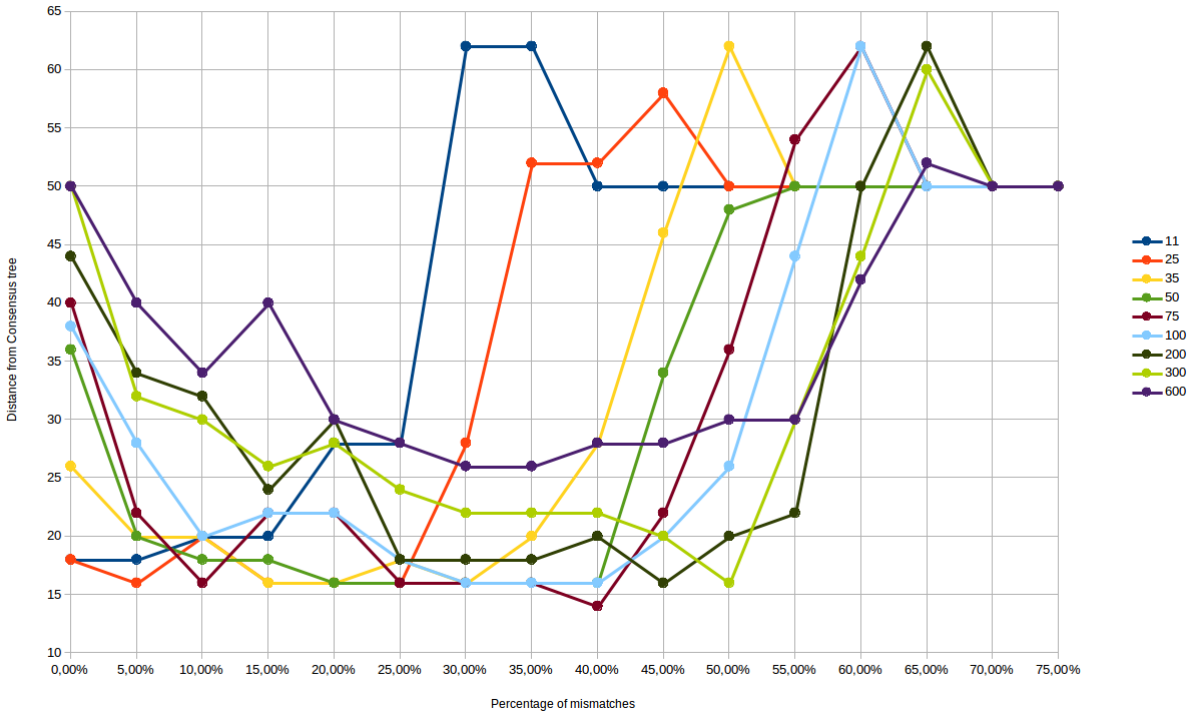

Figure 5.1: Distance from the consensus tree for different pattern lengths and percentage of mismatches.

Figure 5.6 shows the majority consensus tree that we used as reference tree, Figure 5.7 reports the tree built by Ulitsky *et al.* with the Average Common Substring method. In Figure 5.8 we can see the tree obtained by Li *et al.* while Figures 5.9, 5.10 and 5.11 show the tree obtained with our method respectively for $l = 35$, $k = 7$, $l = 75$, $k = 30$ and $l = 100$, $k = 35$.
In general the group of the *Laurasiatheria* is derived correctly and we can easily recognized the clusters of the different sub-orders:

- carnivora (cat, dog, grey seal, harbor seal);
- perissodactyla (white rinho, great rhino, donkey, horse)
- chitophera (fruit bat)
- cetacea (blue whale, finback whale)
- artiodactyla (hippo, pig, sheep, cow)

The pig is clustered closer to the *cetartiodactyls* (blue whale, finback whale, hippo, sheep, cow) than the *perisodactyls* (white rinho, great rhino, donkey, horse) according to what concluded Ulitsky in [40] and Reyes in [31] and opposed to the results found by Li in [22].

The separation between the *Ameridelphia super-order (Wallaroo and Opossum)* and the *Monotremata super-order (Platypus)* are emphasized in the Newick format of the trees: indeed Ulitsky, Li and even the consensus tree present the following grouping:

$$(((Wallaroo, Opossum), Platypus))(other\ species))$$

while our results support the division:

$$((Wallaroo, Opossum), (other\ species), Platypus)$$

Although the graphical position of this group in the trees does not differ from the results of the other works, the presence of different branches increases the distance between our trees and the consensus tree.
The group of *Primates* is generally built correctly and so it is the group of *Afrotheria and Xenarthra*. Both our approach and ACS method present some differences with respect to the majority consensus tree for these two orders:

1. the consensus tree supports the division

   $$((elephant, armadillo), aardvark)$$

   while ACS and our method support the

   $$((elephant, aardvark), armadillo))$$

   grouping, which is supposed to be more correct since *elephant* and *aardvark* belong to the same super-order of *Afrotheria*.

2. the consensus tree estimates that there is a branch which divides *orangutan* and *gibbon* from an unknown common ancestor while our method and ACS do not. As reported in [51] and [37], gibbons differ from great apes (chimpanzees, gorillas, orangutans, bonobos and humans) in several aspects: they are smaller, exhibit low sexual dimorphism, do not make nests, and certain anatomical details advice that they superficially more closely resemble monkeys than great apes do. As we can see in Figure 5.2, molecular evidence indicates that the family of *Hylobatidae*, to whom *gibbons* belong, diverged from great apes between 18 and 12 million years ago, and that of *orangutans* (subfamily *Ponginae*) diverged from the other great apes at about 12 million years. Consequentely it is not likely the evolution scheme proposed by the consensus tree for these two species.

The section of the *rodents* is slightly more uncertain: we can clearly recognize two groups: the one formed by *rat* and *house mouse* and that formed by *squirrel, dormouse, rabbit* and *guinea pig*. For pattern lengths stricly shorter than 50 the two subclusters are positioned close to each other in the tree (see Fig. 5.9), while for larger pattern lengths there appears an additional branch and we find the *Marsupials and Monotremes* group between them (Fig. 5.10 and 5.11).
In one case ($l = 75$, $k = 30$, Figure 5.10) the *rabbit* is misplaced since it results to be closer to the group of *Primates*; in all other cases it is well positioned.

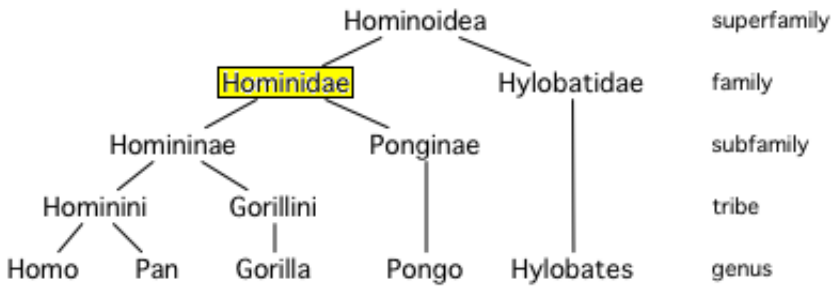

Figure 5.2: Evolutionary tree of the Hominoidea : after an initial separation from the main line of *Hylobatidae*, to whom the lineage of *gibbons* belong, the line of *Pongidae* broke away, leading to the current *orangutan*, while the *Hominidae* split later in *Gorillini* and *Hominini*.

The *guinea pig* is closer to the *leporidae* group than to the *muridae* one while in [40] and also in the consensus tree it is more related to *rat* and *housemouse*. In [22], instead, the *guinea pig* is neither close to *leporidae* nor to *muridae*. However none of these discrepancies are unreasonable hypotheses since the phylogenetic posistion of *guinea pig* is one of the most controversial topics in systematic biology [19,16,11, 39, 32].
Figure 5.3 represents the distance matrix for the pair $(l, k) = (75, 30)$. The matrix has been colored accordingly to its distance values: for a given row, the cells that refer to closer taxa are colored in red. The more distant the taxa, the more tenuous the color gets. An opportune order of the species allows to identify the various clusters where there are reddish regions. Figures 5.4 and 5.5 show the vector of the distances for two taxa: *human* and *horse*. It is possible to see how distances from species of the same sub-order are extremely lower with respect of the others. So for example human is closely related to all the *Primates*, especially to the *pygmy-Chimpanzee*, the *chimpanzee* and the *gorilla*. The species most distant to him are *platypus*, *opossum* and *wallaroo* which, indeed, belong to a different order from that of Eutheria.
The *horse* is very close to the *donkey* as we expected; then gradually we can recognize the different orders it belongs to, as the distance grows: first the perissodactyla (*white rhino* and *great rhino*), than the order of *Lausiatheria*, than the *Eutherian* class.

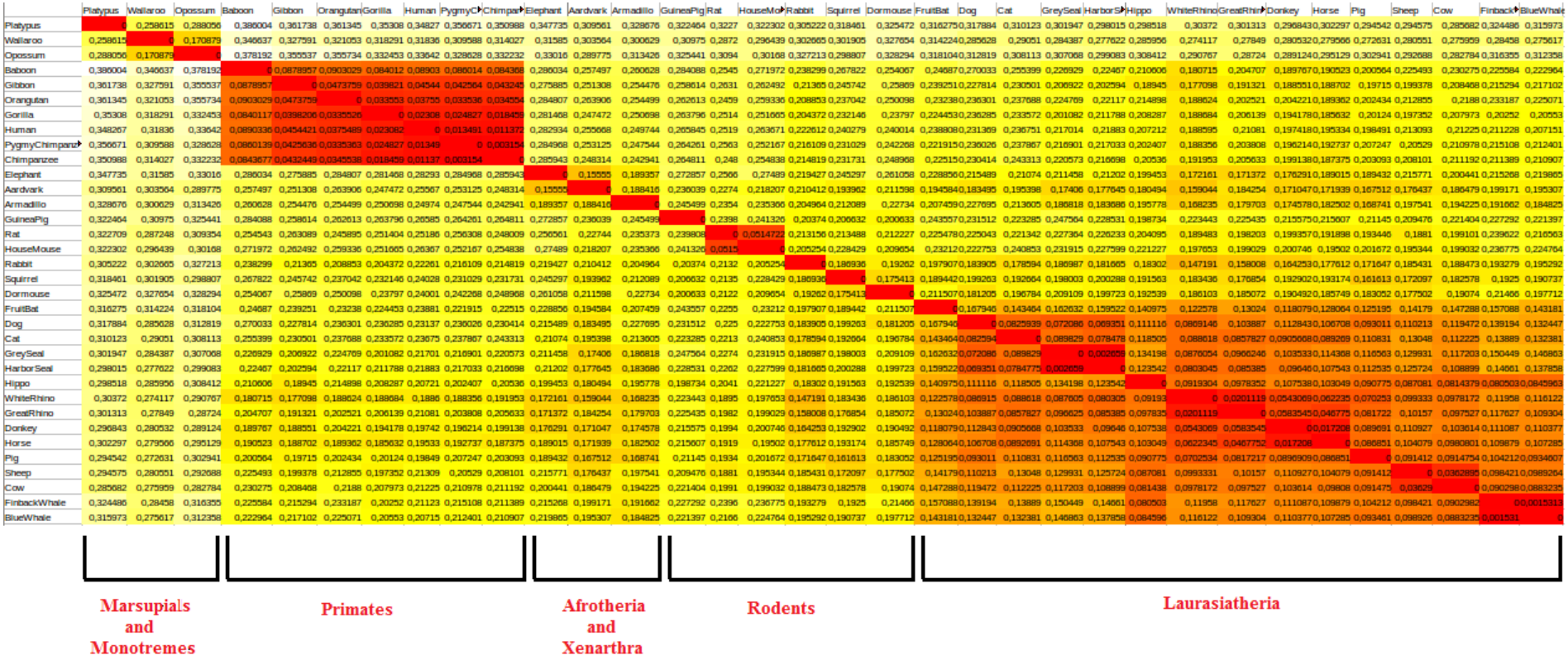

Figure 5.3: Distance matrix of the 34 taxa, for $l = 75$ and $k = 30$.

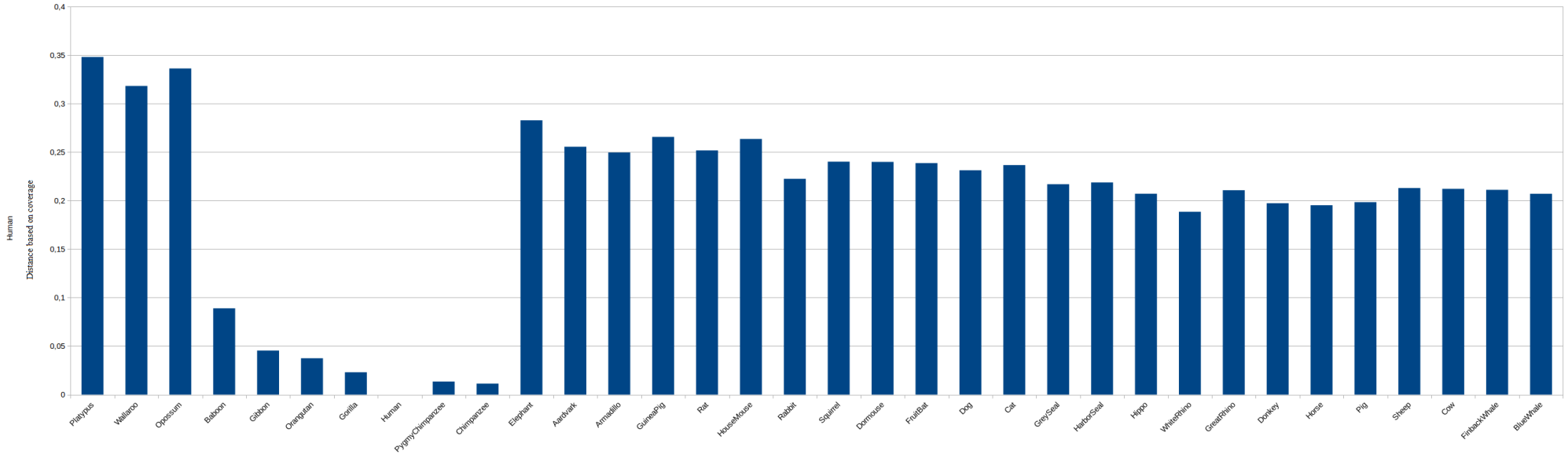

Figure 5.4: Vector of the distance between Human and the other taxa.

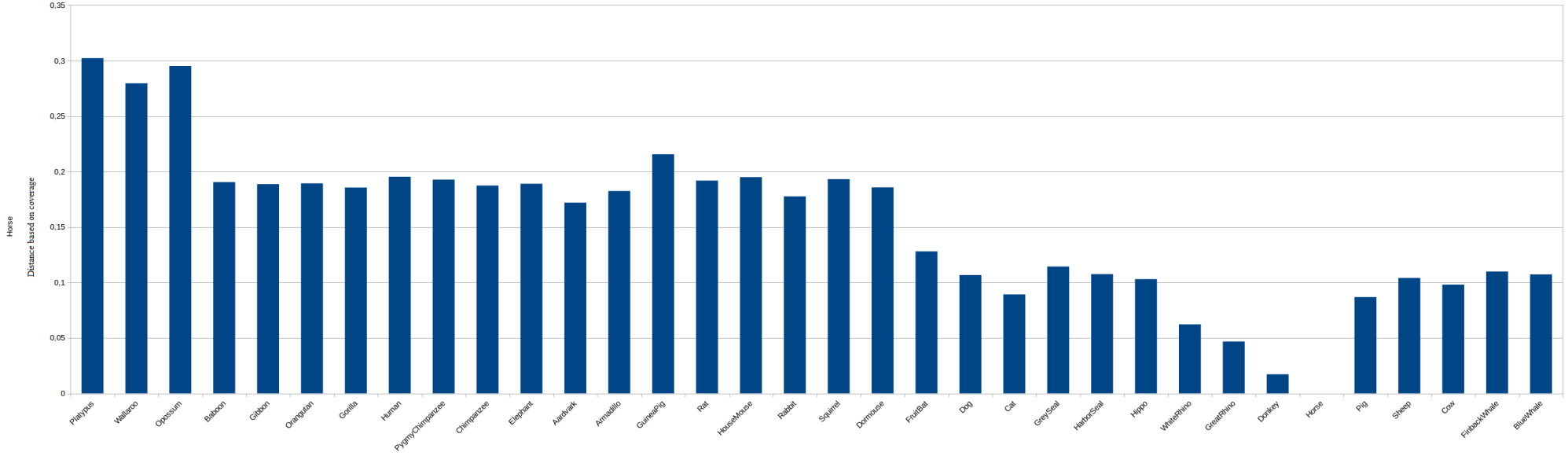

Figure 5.5: Vector of the distance between Horse and the other taxa.

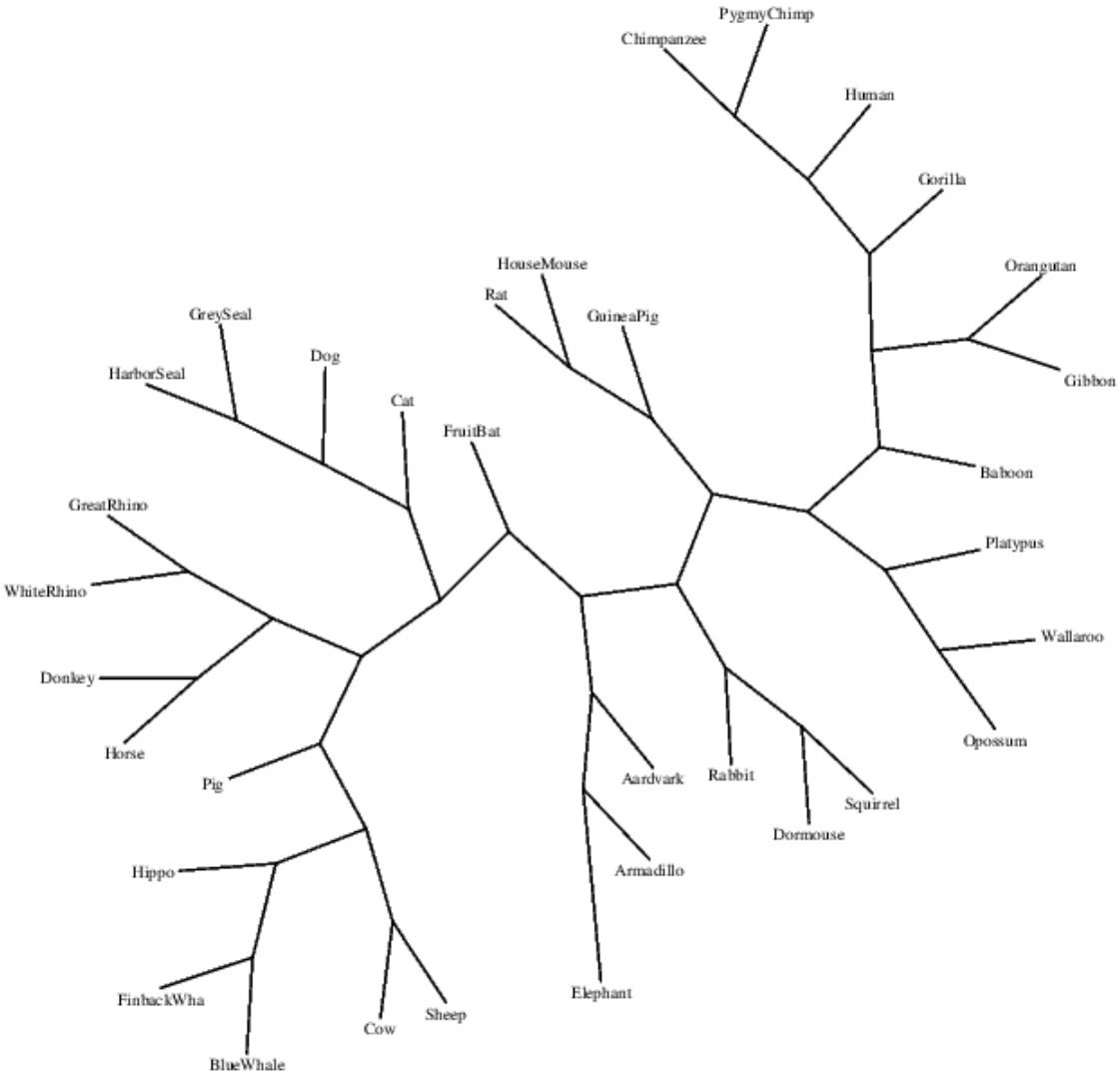

Figure 5.6: Majority consensus tree derived from the ML trees of the 13 mitochondrial proteins from NCBI.

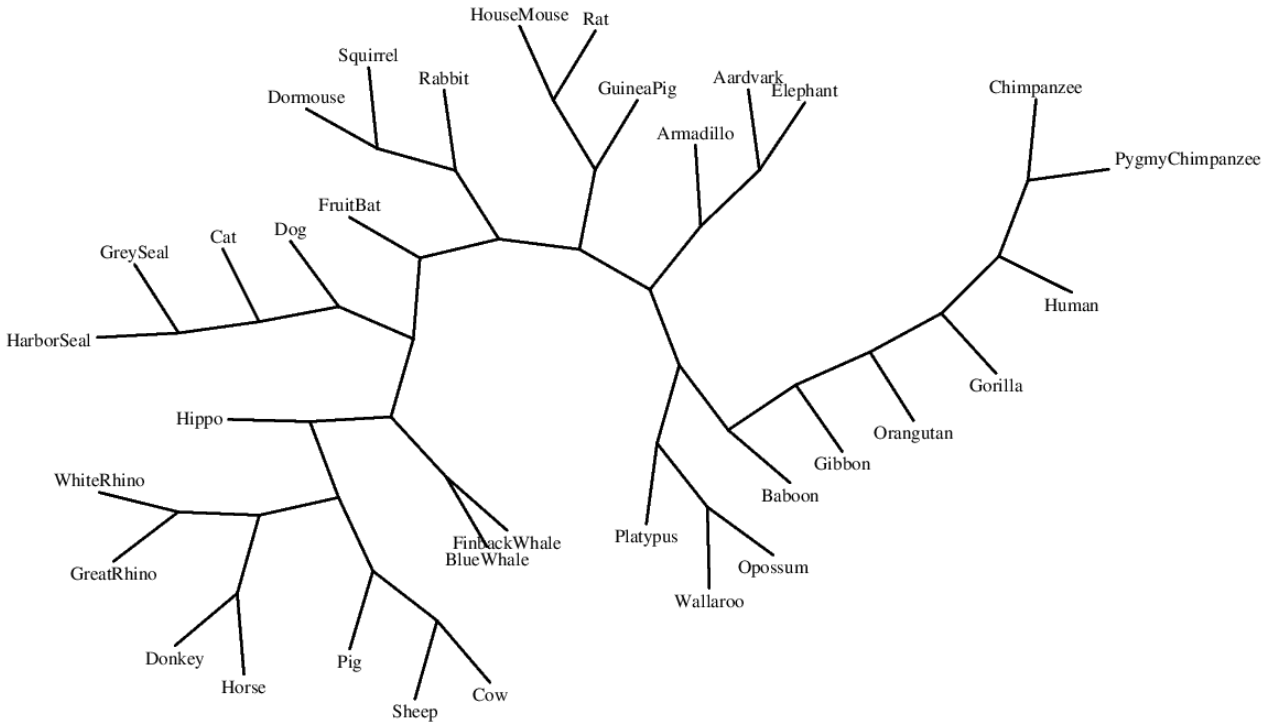

Figure 5.7: Phylogenetic tree obtained with the *Average Common Substring* approach.

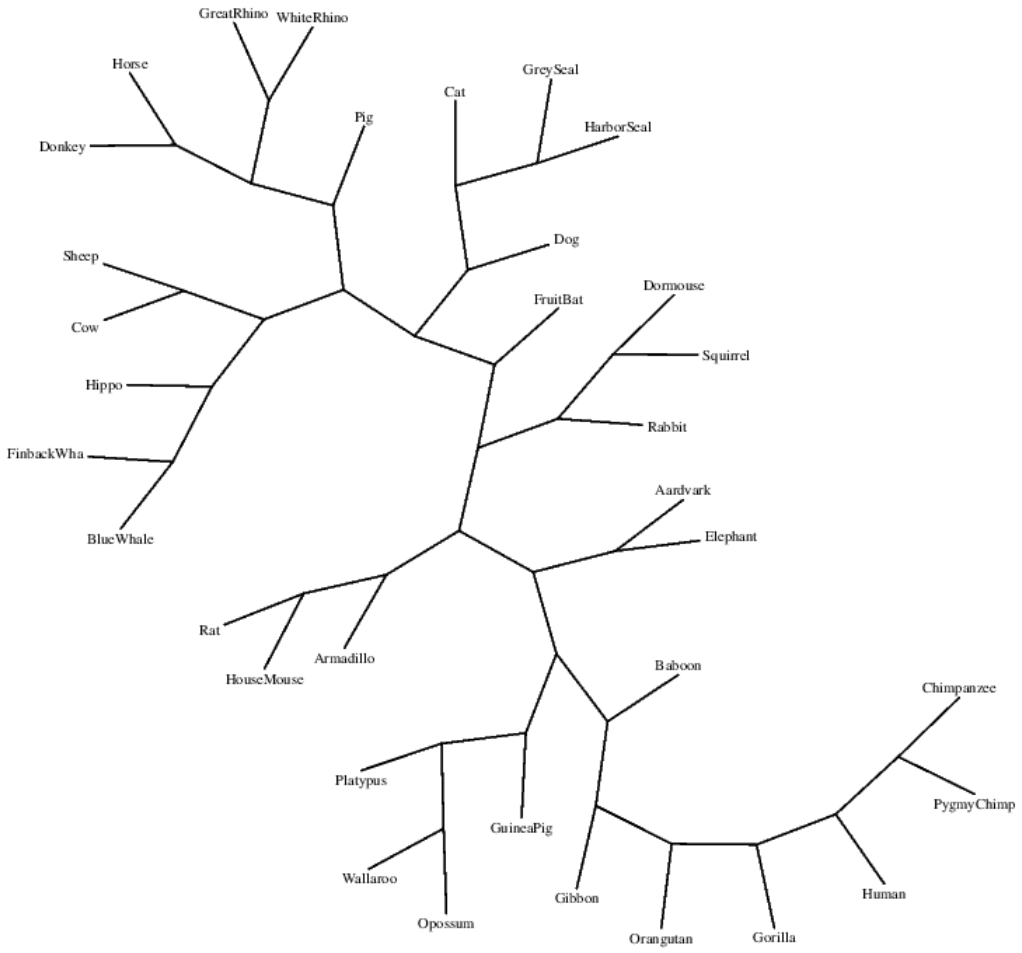

Figure 5.8: Phylogenetic tree of Li *et al.*

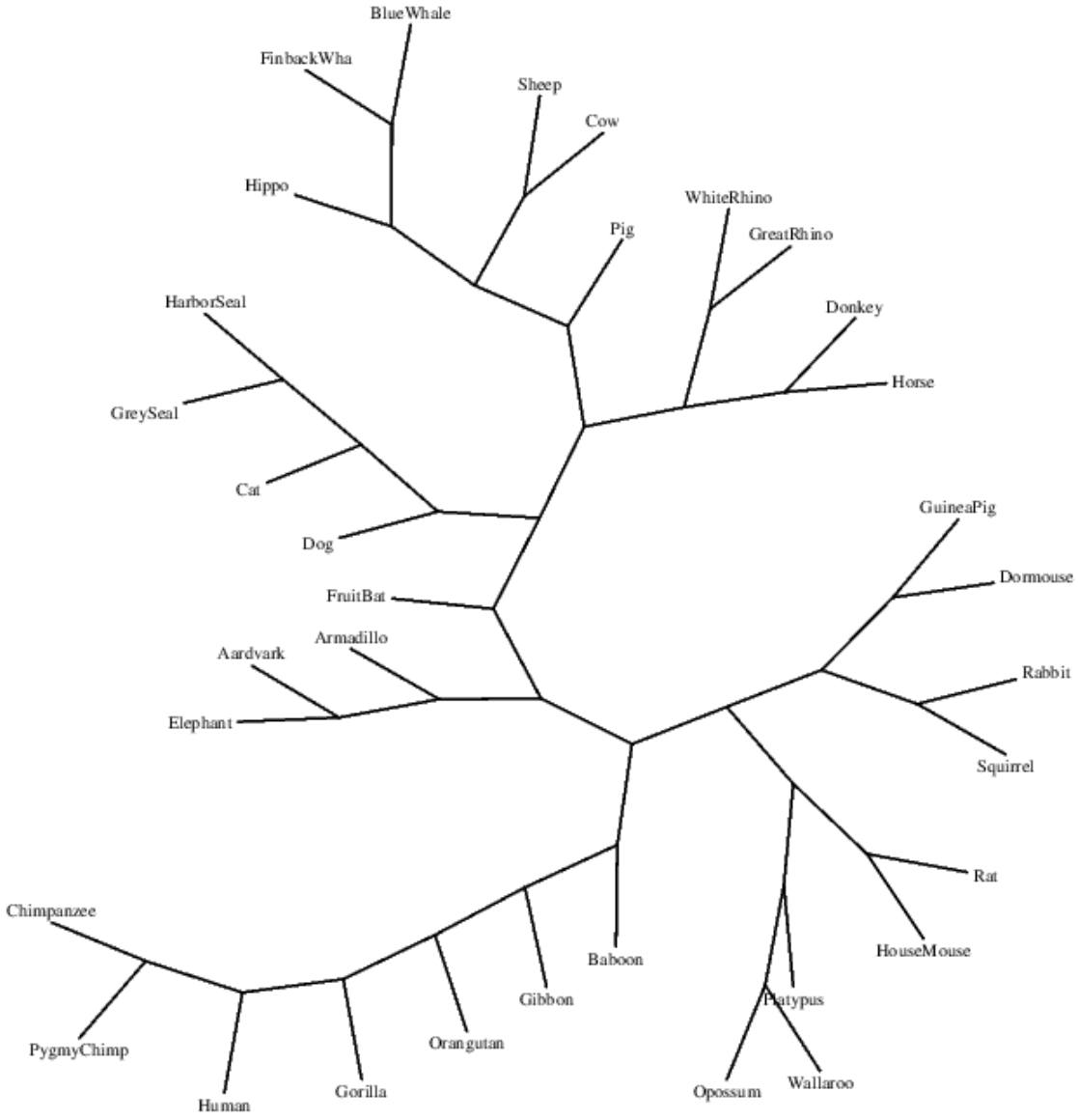

Figure 5.9: Phylogenetic tree of the mitochondrial dataset,for $l = 35$, $k = 7$.

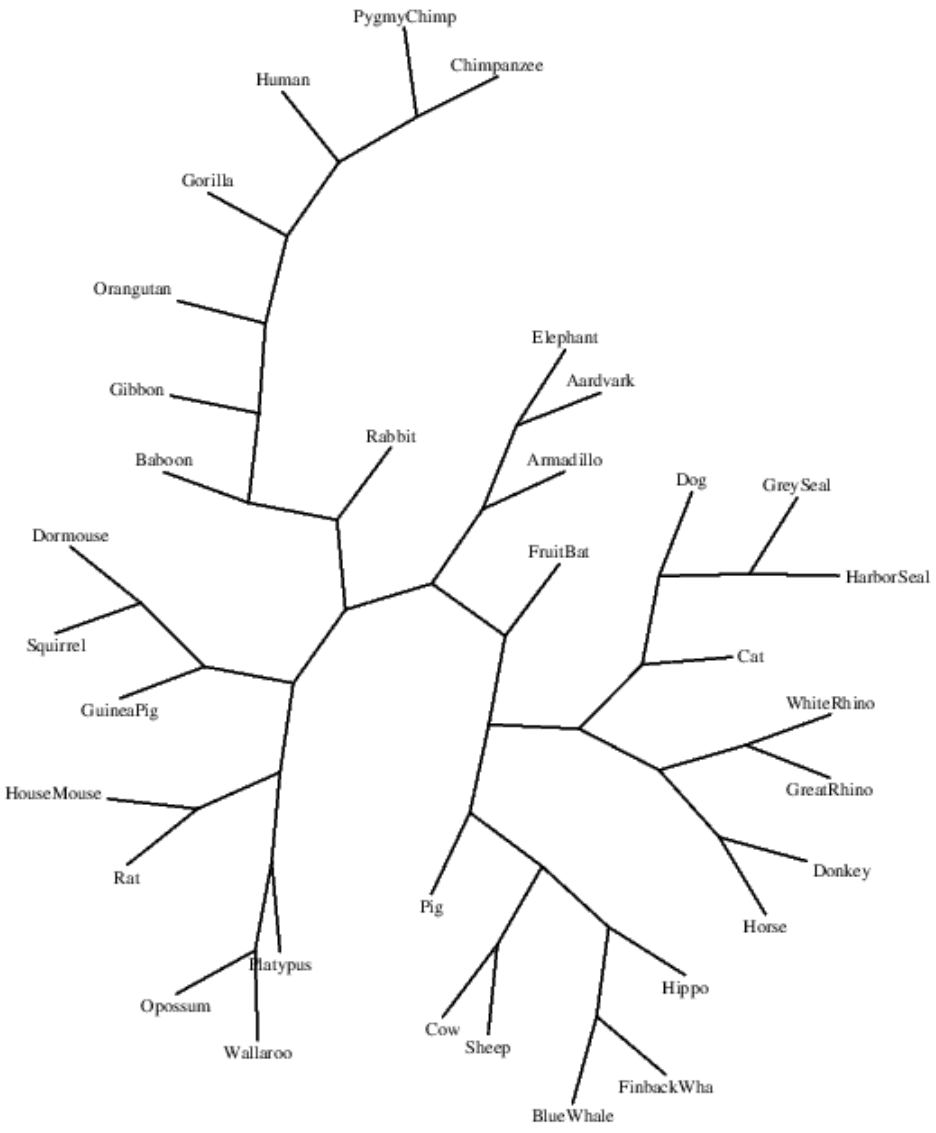

Figure 5.10: Phylogenetic tree of the mitochondrial dataset, for $l = 75$, $k = 30$.

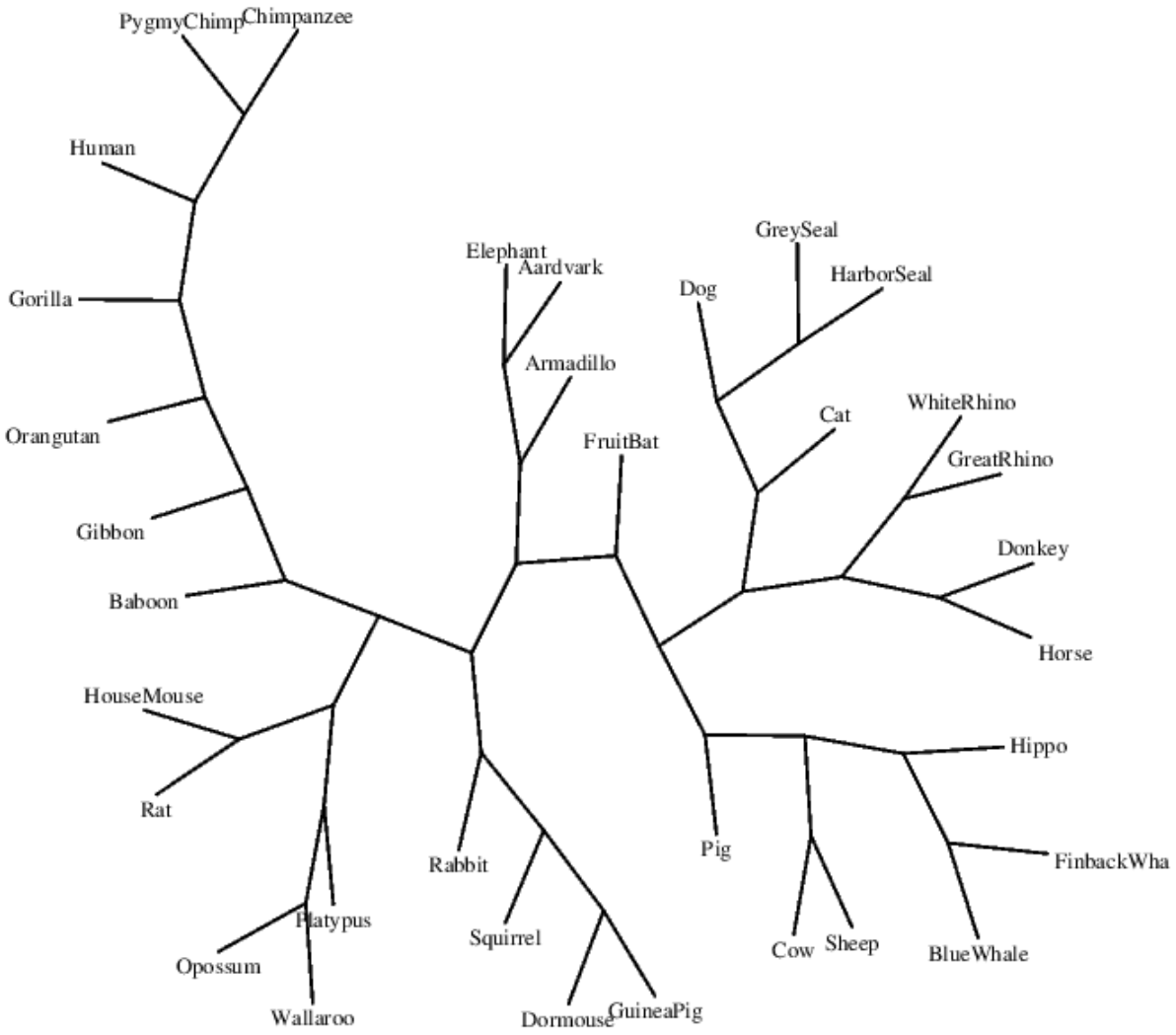

Figure 5.11: Phylogenetic tree of the mitochondrial dataset,for $l = 100$, $k = 35$.

In order to quantitatively compare the trees produced using whole-genome methods to the trees obtained using maximum likelihood with single gene sequence inputs, we have computed the RF distances between the trees generated. The first 13 columns of Table 4 describe the distance from each of the 13 proteins previously cited. We abbreviated the protein names as follows: Ai stands for ATPi, Ci for COXi, CB for CYTB, NL for ND4L, and Ni for NDi. Then the column named "ACS" is the distance from the evolutionary tree proposed by Ulitsky in [40], column "Li" is the distance from the tree proposed in [22], "Cons" refers to the majority consensus tree of Figure 5.6 and the last column, "**Avg**", is the average of the first 13 columns. The first three rows are reported as meter of comparison for our results.

	A6	A8	C1	C2	C3	CB	N1	N2	N3	N4	NL	N5	N6	ACS	Li	Cons	**Avg**
Cons	36	38	22	38	38	26	22	18	32	28	34	18	28	16	18	0	**29.07**
ACS	40	42	28	42	42	30	30	24	28	28	36	18	28	0	14	16	**32**
Li	40	42	26	40	42	24	30	24	30	24	40	18	32	14	0	18	**31.69**
A6	0	42	34	46	40	44	32	40	44	40	48	40	40	40	40	36	**40.83**
A8	42	0	42	48	38	46	42	40	46	42	50	44	40	42	42	38	**43.33**
C1	34	42	0	40	40	32	30	26	34	26	44	28	36	28	26	22	**34.33**
C2	46	48	40	0	48	42	36	40	40	40	50	38	44	42	40	38	**42.66**
C3	40	38	40	48	0	46	42	40	46	44	48	42	36	42	42	38	**42.5**
CB	44	46	32	42	46	0	38	34	32	34	44	30	38	30	24	26	**38.33**
N1	32	42	30	36	42	38	0	28	38	32	44	28	30	30	30	22	**35**
N2	40	40	26	40	40	34	28	0	38	30	40	16	32	24	24	18	**33.66**
N3	44	46	34	40	46	32	38	38	0	34	48	32	44	28	30	32	**39.66**
N4	40	42	26	40	44	34	32	30	34	0	44	28	36	28	24	28	**35.83**
NL	48	50	44	50	48	44	44	40	48	44	0	40	38	36	40	34	**44.83**
N5	40	44	28	38	42	30	28	16	32	28	40	0	32	18	18	18	**33.16**
N6	40	40	36	44	36	38	30	32	44	36	38	32	0	28	32	28	**37.16**
Result with k-mer approach																	
l=11, k=0	38	42	22	40	42	26	30	24	30	22	44	22	32	18	16	18	**31.8**
l=11, k=1	40	40	24	40	42	26	32	26	30	26	46	24	34	20	18	18	**33.07**
l=25, k=1	40	38	24	40	42	26	30	24	30	24	42	22	30	14	16	16	**31.69**
l=25, k=7	40	40	22	38	40	30	28	16	34	26	44	16	32	22	20	16	**31.23**
l=35, k=7	40	40	20	40	40	26	30	20	30	24	44	20	32	18	16	16	**31.23**
l=50, k=10	40	40	26	38	42	28	28	22	32	26	44	20	34	22	20	16	**32.3**
l=50, k=13	40	38	24	38	40	30	28	14	34	26	44	16	32	22	20	16	**31.07**
l=50, k=15	40	38	24	38	40	30	28	14	34	26	44	16	32	22	20	16	**31.07**
l=50, k=17	40	40	24	38	40	30	28	16	34	26	44	16	32	22	20	16	**31.38**
l=75, k=19	40	38	24	38	40	30	28	14	34	26	44	16	32	22	20	16	**31.07**
l=75, k=23	40	40	26	40	40	30	28	14	36	28	42	16	30	22	20	16	**31.54**
l=75, k=27	40	40	22	38	40	30	28	16	34	26	44	16	32	22	20	16	**31.07**
l=75, k=30	40	42	24	38	40	28	26	14	34	26	42	14	30	20	18	14	**30.62**
l=100, k=30	40	38	24	38	40	30	28	14	34	26	44	16	32	22	20	16	**31.07**
l=100, k=35	40	38	24	38	40	30	28	14	34	26	44	16	32	22	20	16	**31.07**
l=100, k=40	40	42	22	38	40	30	26	16	34	26	42	16	30	20	18	16	**31.23**

Table 4: Summary of the distances between evolutionary trees built from complete mtDNA of 34 mammalian taxa and some of our trees.

Our method seems to better describe the phylogenetic truth of the complete mtDNA with respect to the ML trees derived from single proteins, Li's approach and ACS method. Indeed the average RF distance of almost all our trees is lower than that of all the other trees, included Li's and Ulitsky's and the distances from the consensus tree achieved the same or a lower value (in one case) than Average common substring approach.

5.2 Retroid viruses

Viruses are partitioned into a small number of superfamilies, according to their nucleic acid type: DNA or RNA, double strand or single strand. We considered one of these super families: the retroid viruses.

5.2.1 Description of the dataset

The dataset, shown in Table 5 is formed by 84 retroid viruses, and we aim at the evaluation of the consistency of its phylogenetic tree . In this case we have no reference tree expressed in the Newick notation so that we can calculate the distance between it and our trees. We compared our results with those exposed by Ulitsky in [40] and with the taxonomy described in the NCBI Taxonomy and ICTV (International Committee on Taxonomy of Viruses) [52].

5.2.2 Results

The phylogenetic tree obtained for $l = 20$ and $k = 4$ is reported in Figure 5.12. Also with this dataset, various pairs (l, k) properly build the evolution tree. We can see that *Retroviridae* Family is correctly separated according to the *Orthoretrovirinae* and *Spumaretrovirinae* subfamilies. The *Orthorerovirinae* are ulteriorly split up into *Alpharetrovirus*, *Betaretrovirus*, *Gammaretrovirus*, *Deltaretrovirus*, and *Epsilonretrovirus* viruses. The Avian Endogenous Retrovirus (*Avia.endo*) is clustered near to the *Alpharetrovirus* group, thus supporting the results presented in [40] and [33]. In [40] the *EpYVV* is clustered with the *Alpharetrovirus*, in our case it is associated with *Epsilonretrovirus*, while it actually belongs to the Family of *Caulimoviridae* as can be seen in Table 5.
The Lentivirus genus members are clustered together, with a clear separation of the *Primate* (containing the *HIV1, HIV2, SIV1, SIV2, SHIV*) from the *Avian* and *Bovine* species of the viruses (*Ovi.lenti, BIV, Eq. Anemia et cetera*).
The Family of *Hepadnaviridae* are correctly divided into its two sub-genera: *Orthohepadnavirus* (mammalian) and *Avihepadnavirus* (avian).
As concerning the *Caulimoviridae* Family, it is divided into *Badnaviruses* (bacilliform DNA viruses) and *Caulimoviruses* in accordance with the taxonomy of ICTV. ACS method puts the Petunia Vein Clearing Virus (*PVCV*) inside the *Caulimoviruses* while our method correcly inglobe it into the *Badnaviruses* sub-family.

Short name	Full header	Family
SbCMV	GI—11344952—REF—NC_001739.2— SOYBEAN CHLOROTIC MOTTLE VIRUS, C. G.	
PVCV	GI—14575752—REF—NC_001839.2— PETUNIA VEIN CLEARING VIRUS, C. G.	
SCBV	GI—15029533—REF—NC_003031.1— SUGARCANE BACILLIFORM VIRUS, C. G.	
BSV	GI—18450258—REF—NC_003381.1— BANANA STREAK VIRUS, C. G.	
Tobac.VC	GI—18450263—REF—NC_003378.1— TOBACCO VEIN-CLEARING VIRUS, C. G.	
Citru.mosa	GI—18450268—REF—NC_003382.1— CITRUS YELLOW MOSAIC VIRUS, C. G.	
CERV	GI—19919889—REF—NC_003498.1— CARNATION ETCHED RING VIRUS, C. G.	
FMV	GI—20143424—REF—NC_003554.1— FIGWORT MOSAIC VIRUS, C. G.	
BRRV	GI—21263121—REF—NC_003138.2— BLUEBERRY RED RINGSPOT VIRUS, C. G.	
MiMV	GI—21450043—REF—NC_004036.1— MIRABILIS MOSAIC VIRUS, C. G.	
TaBV	GI—27228718—REF—NC_004450.1— TARO BACILLIFORM VIRUS, C. G.	**Caulimoviridae**
EpYVV	GI—27573297—REF—NC_004515.1— EUPAT. YELLOW VEIN VIRUS-ASSOC. DNA BETA, C. G.	
KTSV	GI—27819377—REF—NC_004540.1— KALANCHOE TOP-SPOTTING VIRUS, C. G.	
CmYLCV	GI—32453808—REF—NC_004324.3— CESTRUM YELLOW LEAF CURLING VIRUS, C. G.	
ComYMV	GI—9625564—REF—NC_001343.1— COMMELINA YELLOW MOTTLE VIRUS, C. G.	
CaMV	GI—9626938—REF—NC_001497.1— CAULIFLOWER MOSAIC VIRUS, C. G.	
CSSV	GI—9627246—REF—NC_001574.1— CACAO SWOLLEN SHOOT VIRUS, C. G.	
PCSV	GI—9627957—REF—NC_001634.1— PEANUT CHLOROTIC STREAK VIRUS, C. G.	
CsVMV	GI—9627994—REF—NC_001648.1— CASSAVA VEIN MOSAIC VIRUS, C. G.	
SVBV	GI—9628900—REF—NC_001725.1— STRAWBERRY VEIN BANDING VIRUS, C. G.	
RTBV	GI—9630630—REF—NC_001914.1— RICE TUNGRO BACILLIFORM VIRUS, C. G.	
Stork.HepB	GI—18071209—REF—NC_003325.1— STORK HEPATITIS B VIRUS, C. G.	
HepB	GI—21326584—REF—NC_003977.1— HEPATITIS B VIRUS, C. G.	
Wood.HepB	GI—22256030—REF—NC_004107.1— WOODCHUCK HEPATITIS B VIRUS, C. G.	
Sheld.HepB	GI—48696569—REF—NC_005890.1— SHELDGOOSE HEPATITIS B VIRUS, C. G.	
Ross.HepB	GI—48696604—REF—NC_005888.1— ROSS' GOOSE HEPATITIS B VIRUS, C. G.	
Snow.HepB	GI—49246207—REF—NC_005950.1— SNOW GOOSE HEPATITIS B VIRUS, C. G.	**Hepadnaviridae**
Duck.HepB	GI—9625568—REF—NC_001344.1— DUCK HEPATITIS B VIRUS, C. G.	
Gs.HepB	GI—9626714—REF—NC_001484.1— GROUND SQUIRREL HEPATITIS VIRUS, C. G.	
Heron.HepB	GI—9626719—REF—NC_001486.1— HERON HEPATITIS B VIRUS, C. G.	
Arct.HepB	GI—9628827—REF—NC_001719.1— ARCTIC GROUND SQUIRREL HEPATITIS B VIRUS, C. G.	
Wool.HepB	GI—9630370—REF—NC_001896.1— WOOLLY MONKEY HEPATITIS B VIRUS, C. G.	
Oran.HepB	GI—9634216—REF—NC_002168.1— ORANGUTAN HEPADNAVIRUS, C. G.	
Porc.endo	GI—15187162—REF—NC_003059.1— PORCINE ENDOGENOUS RETROVIRUS, C. G.	
PTLV3	GI—18071203—REF—NC_003323.1— PRIMATE T-LYMPHOTROPIC VIRUS 3, C. G.	
SIV2	GI—27311166—REF—NC_004455.1— SIMIAN IMMUNODEFICIENCY VIRUS 2, C. G.	
Enzo.goat	GI—33354433—REF—NC_004994.2— ENZOOTIC NASAL TUMOUR VIRUS OF GOATS, C. G.	
Avia.endo	GI—49248517—REF—NC_005947.1— AVIAN ENDOGENOUS RETROVIRUS EAV-HP, C. G.	
FrMLV	GI—9626096—REF—NC_001362.1— FRIEND MURINE LEUKEMIA VIRUS, C. G.	
Mur.sarv	GI—9626100—REF—NC_001363.1— MURINE SARCOMA VIRUS, C. G.	
SFV	GI—9626103—REF—NC_001364.1— SIMIAN FOAMY VIRUS, C. G.	
ACV	GI—9626152—REF—NC_001402.1— AVIAN CARCINOMA VIRUS, C. G.	
FuSV	GI—9626154—REF—NC_001403.1— FUJINAMI SARCOMA VIRUS, C. G.	
Y73SV	GI—9626156—REF—NC_001404.1— Y73 SARCOMA VIRUS, C. G.	
RSV	GI—9626196—REF—NC_001407.1— ROUS SARCOMA VIRUS, C. G.	
ALV	GI—9626201—REF—NC_001408.1— AVIAN LEUKOSIS VIRUS, C. G.	
BIV	GI—9626219—REF—NC_001413.1— BOVINE IMMUNODEFICIENCY VIRUS, C. G.	
BLV	GI—9626225—REF—NC_001414.1— BOVINE LEUKEMIA VIRUS, C. G.	
HTLV1	GI—9626453—REF—NC_001436.1— HUMAN T-LYMPHOTROPIC VIRUS 1, C. G.	
Eq.anemia	GI—9626530—REF—NC_001450.1— EQUINE INFECTIOUS ANEMIA VIRUS, C. G.	
Visna	GI—9626546—REF—NC_001452.1— VISNA VIRUS, C. G.	
Carp.erth	GI—9626651—REF—NC_001463.1— CAPRINE ARTHRITIS-ENCEPHALITIS VIRUS, C. G.	
FIV	GI—9626701—REF—NC_001482.1— FELINE IMMUNODEFICIENCY VIRUS, C. G.	
HTLV2	GI—9626726—REF—NC_001488.1— HUMAN T-LYMPHOTROPIC VIRUS 2, C. G.	
Abelson	GI—9626914—REF—NC_001494.1— OVINE PULMONARY ADENOCARCINOMA VIRUS, C. G.	
AbMLV	GI—9626953—REF—NC_001499.1— ABELSON MURINE LEUKEMIA VIRUS, C. G.	
Spleen.foc	GI—9626955—REF—NC_001500.1— SPLEEN FOCUS-FORMING VIRUS, C. G.	
MLV	GI—9626958—REF—NC_001501.1— MURINE LEUKEMIA VIRUS, C. G.	
MoMSV	GI—9626962—REF—NC_001502.1— MOLONEY MURINE SARCOMA VIRUS, C. G.	
MMTV	GI—9626965—REF—NC_001503.1— MOUSE MAMMARY TUMOR VIRUS, C. G.	**Retroviridae**
Mur.osteo	GI—9626984—REF—NC_001506.1— MURINE OSTEOSARCOMA VIRUS, C. G.	
Ovi.lenti	GI—9627001—REF—NC_001511.1— OVINE LENTIVIRUS, C. G.	
WMSV	GI—9627014—REF—NC_001514.1— WOOLLY MONKEY SARCOMA VIRUS, C. G.	
SIV	GI—9627204—REF—NC_001549.1— SIMIAN IMMUNODEFICIENCY VIRUS, C. G.	
MPMV	GI—9627210—REF—NC_001550.1— MASON-PFIZER MONKEY VIRUS, C. G.	
ASV	GI—9627732—REF—NC_001618.1— AVIAN SARCOMA VIRUS, C. G.	
Jembrana	GI—9628091—REF—NC_001654.1— JEMBRANA DISEASE VIRUS, C. G.	
MurC	GI—9628654—REF—NC_001702.1— MURINE TYPE C RETROVIRUS, C. G.	
HIV2	GI—9628880—REF—NC_001722.1— HUMAN IMMUNODEFICIENCY VIRUS 2, C. G.	
SnRV	GI—9628892—REF—NC_001724.1— SNAKEHEAD RETROVIRUS, C. G.	
HFV	GI—9629127—REF—NC_001736.1— HUMAN FOAMY VIRUS, C. G.	
H.spuma	GI—9629258—REF—NC_001795.1— HUMAN SPUMARETROVIRUS, C. G.	
HIV1	GI—9629357—REF—NC_001802.1— HUMAN IMMUNODEFICIENCY VIRUS 1, C. G.	
STLV2	GI—9629498—REF—NC_001815.1— SIMIAN T-LYMPHOTROPIC VIRUS 2, C. G.	
Rauscher	GI—9629514—REF—NC_001819.1— RAUSCHER MURINE LEUKEMIA VIRUS, C. G.	
BFV	GI—9629644—REF—NC_001831.1— BOVINE FOAMY VIRUS, C. G.	
AMCV	GI—9629900—REF—NC_001866.1— AVIAN MYELOCYTOMATOSIS VIRUS, C. G.	
WDSV	GI—9629902—REF—NC_001867.1— WALLEYE DERMAL SARCOMA VIRUS, C. G.	
SHIV	GI—9629914—REF—NC_001870.1— SIMIAN-HUMAN IMMUNODEFICIENCY VIRUS, C. G.	
FFV	GI—9629925—REF—NC_001871.1— FELINE FOAMY VIRUS, C. G.	
GALV	GI—9630311—REF—NC_001885.1— GIBBON APE LEUKEMIA VIRUS, C. G.	
FeLV	GI—9630707—REF—NC_001940.1— FELINE LEUKEMIA VIRUS, C. G.	
STLV1	GI—9632565—REF—NC_000858.1— SIMIAN T-LYMPHOTROPIC VIRUS 1, C. G.	
Eq.foamy	GI—9634977—REF—NC_002201.1— EQUINE FOAMY VIRUS, C. G.	

Table 5: Table of the retroid viruses.

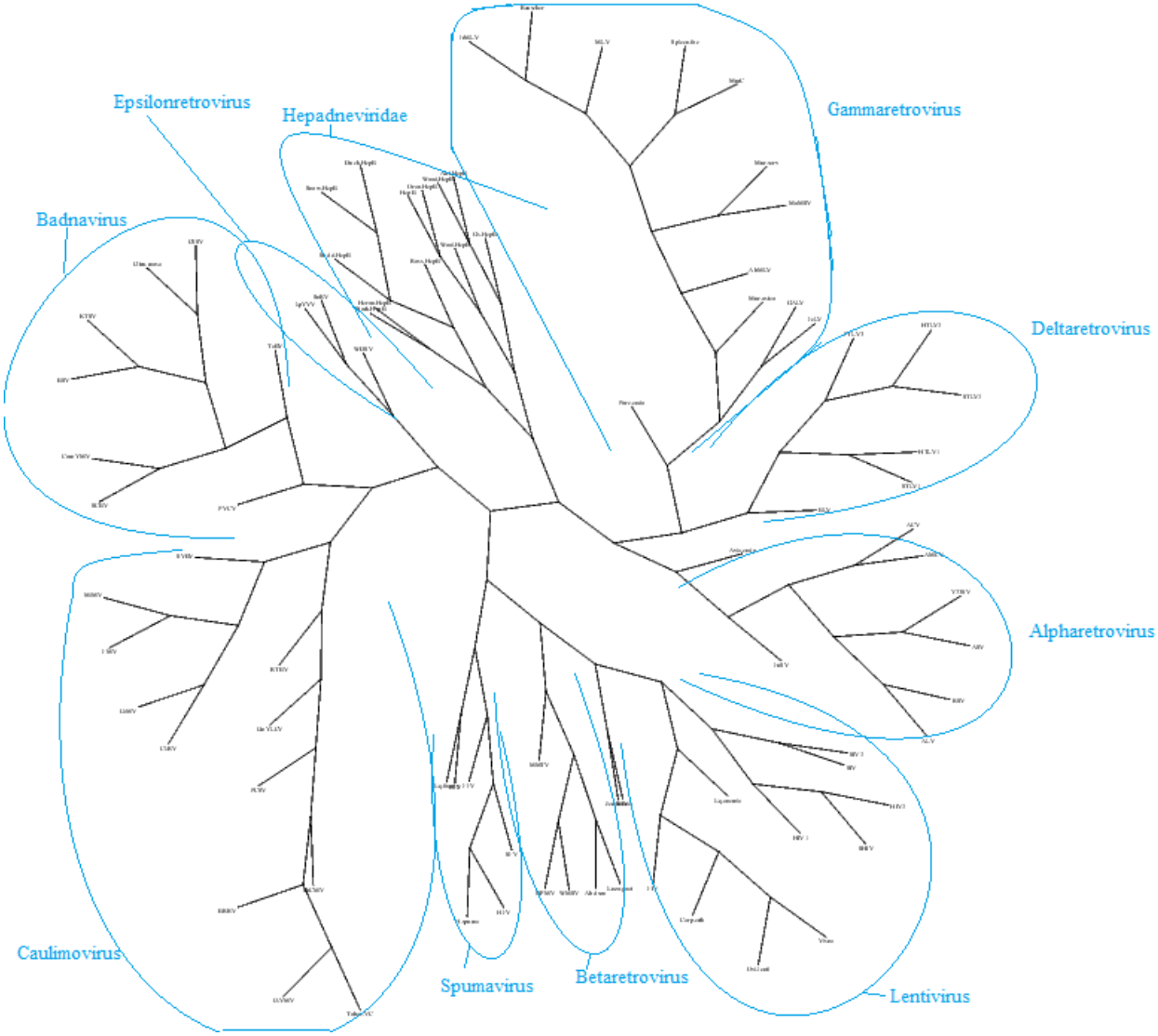

Figure 5.12: Phylogenetic tree of the retroid viruses dataset, for $l = 20$, $k = 4$.

6 Conclusions and future perspectives

In this work we presented a novel approach for phylogenetic reconstruction based on the occurrences of l-mers with mismatches. Several pattern lengths have been tested and in general it seems that longer pattern lengths require a greater fraction of mismatches to be allowed in order to build the tree correctly. For our principal dataset, composed by mitochondrial DNA sequences of average lentgh $n \approx 16000$ characters, substrings longer than 100 lead to wrong phylogenies. It is not clear whether it is possible to infer a relationship between the average length of the sequences of the dataset, the length of the substrings and the optimum number or set of number of mismatches. Analysis should be performed over several datasets of known phylogeny in order to obtain a general rule.
However our method seems to be also robust to variations on these parameters since a wide range of couples of values (l, k) efficiently works for deriving the phylogenetic evolution of the taxa.
The comprehensive comparison with other whole genome methods shows that we can reach an *accuracy* surely comparable and usually as good or better than that presented in such works.
In terms of computational time efficiency, the quadratic cost of our algorithm can not be bypassed when looking for an exact algorithm. The time onerosity for dataset with very long genome or proteome sequences (millions of basepairs) makes our approach unfeasible for such data. In this case we think that an approach with an approximate matching algorithm could provide important time complexity improvements without too much considerable losses in terms of *accuracy*. The results obtained both in the mitochondrial and the retroid viruses' dataset demostrate the value of our method which is not dependent on a particular set of taxa.
The correct clustering of species in orders and super-orders could provide useful information also for classifying taxa that are currently not officially classified.

7 Acknowledgements

Ringrazio la prof.ssa Cinzia Pizzi per l'attenzione, la pazienza, i consigli e il tempo che mi ha dedicato durante questa esperienza di tesi e nella stesura dell'elaborato.

Ringrazio altresì il prof. Fabio Vandin per il prezioso aiuto nell'interpretazione dei risultati, nello sviluppo di nuove idee e per il tempo che mi ha dedicato nonostante un fuso orario non sempre favorevole.

Ringrazio i miei genitori Fulvio e Luigina, silenziosi e attenti autori di piccoli, generosi, eccezionali gesti di quotidiano affetto nelle lunghe giornate e nottate di lavoro.

Ringrazio mio fratello Damiano, per i consigli preziosi da oltre Manica e per essere complice di fantastici momenti di spensieratezza, di viaggi e avventure.

Ringrazio Manuela, per aver creduto in me e perché riesce immancabilmente a farmi sorridere. E anche per avermi fatto correggere i ringraziamenti all'ultimo momento.

Ringrazio i miei Amici, in modo particolare Alberto e Luca, compagni unici di situazioni improbabili, capaci di rendere ogni momento leggendario.

Ringrazio le persone che mi hanno introdotto alla grande passione per la musica, che mi hanno insegnato ad amarla sempre più, a conoscerla e apprezzarla nelle sue infinite sfumature. Quindi grazie ad Elisabetta Illes per avermi insegnato la bellezza del pianoforte e lo spirito di sacrificio di chi vuole ottenere qualcosa di bello con un metodico, costante impegno.
E grazie al Maestro Giuliano Fracasso per avermi dato così tante occasioni di sentire la musica vivere dentro di me e avermi guidato ad una sua espressione ricca di passione, portandomi ad abbattere paure e limiti, aiutandomi a crescere molto e spesso dandomi più fiducia di quanta io stesso me ne sarei concessa.

8 References

References

[1] K. Abrahamson, Generalized string matching, *SIAM J. Comput.* 16 (6) (1987) 1039–1051.

[2] A. Amir, M. Lewenstein, E. Porat, Faster algorithms for string matching with k mismatches, *J. Algorithms* 50 (2) (2000) 257–275.

[3] T.K. Attwood, Genomics: the Babel of bioinformatics. *Science*, 290 (2000) 471–473.

[4] A. Ben-Dor, B. Chor, D. Graur, R. Ophir, and D. Pelleg Constructing phylogenies from quartets: elucidation of eutherian superordinal relationships., *J. comput. Biol.* 5 (1998) 377-390.

[5] O. Bininda-Emonds, Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life. *Kluwer series in Computational Biology.* (2004)

[6] B. E. Blaisdell, Average values of a dissimilarity measure not requiring sequence alignment are twice the averages of conventional mismatch counts requiring sequence alignment for a computer-generated model system. *J. Mol. Evol.*, 29 (6) (1989) 538-47.

[7] M. Blaxter, J. Mann, T. Chapman, F. Thomas, C. Whitton, R. Floyd, E. Abebe, Defining operational taxonomic units using DNA barcode data. *Philos. Trans. R. Soc.Lond. B. Biol. Sci.* 360 (1462) (2005) 1935–43.

[8] A. Brazma, I. Jonassen, I. Eidhammer, D. Gilbert, Approaches to the automatic discovery of patterns in biosequences, *J. Comp. Biol.* 5 (1998) 279–305.

[9] P. Buneman, The recovery of trees from measures of dissimilarity. *Edinburgh University Press*, (1971).

[10] Y. Cao, A. Janke, P. J. Waddell, M. Westerman, O. Takenaka, S. Murata, N. Okada, S. Pabo, M. Hasegawa, Conflict among individual mitochondrial proteins in resolving the phylogeny of eutherian orders. *J. Mol. Evol.* 47 (1998) 307-322

[11] Y. Cao, N. Okada, M. Hasegawa, Phylogenetic position of guinea pigs revisited, *Mol. Biol. Evol.*, 14, (1997), 461–464.

[12] R. M. Casey, BLAST Sequences Aid in Genomics and Proteomics. *Business Intelligence Network* (2005)

[13] L.L. Cavilli-Sforza and A.W.F. Edwards, Phylogenetic analysis: models and estimation procedures. *Evol.* 21 (1967) 550-570.

[14] X. Chen, S. Kwong, M. Li, A compression algorithm for dna sequences and its applications in genome comparison. *RECOMB2000*, (2000) 107-117.

[15] F. Cunial, A. Apostolico, Phylogeny Construction with Rigid Gapped Motifs, *Journal of Computational Biology*, 19(7), (July 2012), 911-927.

[16] A. M. D'Erchia, C. Gissi, G. Pesole, C. Saccone, U. Arnason, The guinea pig is not a rodent, *Nature*, 381, (1996) , 597–599.

[17] J. Felsenstein, Inferring Phylogenies. Sunderland, MA, *Sinauer Associates*, (2004).

[18] C.J. Geyer, Markov chain Monte Carlo maximum likelihood. In Keramidas, E.M. Computing Science and Statistics: Proceedings of the 23rd Symposium of the Interface. Fairfax Station VA: Interface Foundation. (1991) 156–163.

[19] D. Graur, M. Gouy, L. Duret, Evolutionary affinities of the order Perissodactyla and the phylogenetic status of the superordinal taxa Ungulata and Altungulata, *Mol. Phylogenet. Evol.*, 7, (1997) 195–200.

[20] G.M. Landau, U. Vishkin, Efficient string matching with k mismatches, *Theoret. Comput. Sci.* 43 (1986) 239–249.

[21] A. Lempel, J. Ziv, On the complexity of finite sequences. *IEEE Transactions on Information Theory* IT-22, (1976), 75–81 .

[22] M. Li, J. Badger, X. Chen, S. Kwong, P. Kearney, H. Zhang, An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17 (2), (2001) 149-154.

[23] M. Li, P. Vitànyi, An Introduction to Kolmogorov Complexity and its Applications, *Springer*, New York, (1997).

[24] T. Liu, J. Tang, M. Moret, Quartet-Based Phylogeny Reconstruction from Gene Orders. In Proc 11th Annual International Conference on Computing and Combinatorics, of Lecture Notes in Computer Science. Volume 3595. *Springer-Verlag*, (2005) 63-73.

[25] B. Ma, M. Li, and L. Zhang, From gene trees to species trees. *SIAM J. Comput*, 30(3) (2000) 729-752.

[26] G. Manzini , P. Ferragina, Engineering a Lightweight Suffix Array Construction Algorithm, *Algorithmica*, 40 (1) (2004) 33-50.

[27] S.B. Needleman, and C.D. J. Wunsch, *Mol. Biol.*, 48 (1970) 443–453.

[28] C.Pizzi, k-difference matching in amortized linear time for all the words in a text *Theoretical Computer Science*, 410(8-10) (2009) 983-987

[29] P.T. Raul, B. Gordon, and E. Oliver, In Bininda-Emonds, Olaf R.P. (ed), Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, chapter Quartet Supertrees, *Kluwer Academic (In Press)*, (2004) 173–191.

[30] G. Reinert, S. Schbath, M.S. Waterman, Probabilistic and statistical properties of words: An overview, *J. Comp. Biol.* 7 (1–2) (2000) 1–46.

[31] A. Reyes, C. Gissi, G. Pesole, F.M. Catzeflis, C. Saccone, Where do rodents fit? Evidence from the complete mitochondrial 154 genome of Sciurus vulgaris. *Mol. Biol. Evol.*, 17, (2000), 979–983.

[32] A. Reyes, G. Pesole, C. Saccone, Complete mitochondrial DNA sequence of the fat dormouse, *Glis glis*: further evidence of rodent paraphyly. *Mol. Biol. Evol.*, 15, (1998) 499–505.

[33] M. A. Sacco, D. M. J. Flannery, K. Howes, K. Venugopal, Avian endogenous retrovirus eav-hp shares regions of identity with avian leukosis virus subgroup j and the avian retrotransposon art-ch,*J. Virol*, 74(3), (2000), 1296-1306.

[34] N. Saitou, M. Nei, The Neighbor-Joining Method – a new method for reconstructing phylogenetic trees, *Mol. Biol. Evol.*, 4 (1987) 406–425.

[35] D.B. Searls, Reading the book of life. *Bioinformatics*, 17 (2001) 579–580.

[36] T.F. Smith, and M.S.J. Waterman, *Mol. Biol.*, 147 (1981) 195–197.

[37] V.K. Srivastava, Morphology Of The Primates And Human Evolution. *PHI Learning Pvt. Ltd.* (2009) p. 87.

[38] C. Stewart, The Powers and Pitfalls of Parsimony, *Nature 361* (6413), (1993), 603–607.

[39] J. Sullivan, D.L. Swofford, Are guinea pigs rodents? The importance of adequate models in molecular phylogenetics. *J. Mammal. Evol.*, 4, (1997), 77–86.

[40] I. Ulitsky, D. Burstein, T. Tuller,B. Chor, The average common substring approach to phylogenomic reconstruction., *J. Comput. Biol.*, 13 (2006) 336-350.

[41] S. Vinga, J. Almeida, Alignment-free sequence comparison-a review, *Bioinformatics*, 19(4), (2003), 513-523.

[42] J. Xiong, Essential Bioinformatics, I edition published by Cambridge University Press (2006)

[43] J. Ziv, A. Lempel, A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* IT-23, (1977), 337–343

[44] evolution.genetics.washington.edu/phylip.html

[45] McGraw-Hill Dictionary of Scientific & Technical Terms, 6E, Copyright © 2003 by The McGraw-Hill Companies, Inc.

[46] blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download

[47] www.clustal.org/

[48] www.bioperl.org/wiki/Newick_tree_format

[49] www.ncbi.nlm.nih.gov.

[50] www.cs.tau.ac.il/ bchor/whole/html/

[51] en.wikipedia.org/wiki/Gibbon

[52] B.O. Cornelia, Ictvdb (international committee on taxonomy of viruses database). phene.cpmc.columbia.edu/Ictv/index.htm.