Comparison of phylogenetic reconstruction methods

Lívia Qian

KTH Royal Institute of Technology

School of Electrical Engineering and Computer Science

liviaq@kth.se

Abstract

The aim of this work is to briefly describe and compare the performance of five methods for phylogenetic reconstruction. Instead of trying to be a comprehensive guide summarizing the theoretical background of phylogenetics, it focuses on the main principles of creating simple phylogenetic trees and the conclusions I reached while testing these methods. My primary goal was to measure the time complexity of each algorithm besides estimating the accuracy of the generated trees. In the end, the results were compared those of the paper *Confirming the Phylogeny of Mammals by Use of Large Comparative Sequence Data Sets* (Prasad et al., 2008).

1 Introduction

Phylogenetics is the study of the evolutionary history of species and groups of species. There are multiple ways to express the relationship between different groups; one common method is the construction of a phylogenetic tree (or evolutionary tree), where all taxa are present as leaves. Common traits can be observed using DNA sequences or morphology, the latter being generally more complex due to the many characteristics a certain species can have. Phylogenetic trees can be rooted and unrooted; the former is a version representing not only the similarity between each taxon but the ancestral relationships as well, while the latter form only illustrates the connection between the taxonomical units.

2 Methods

DNA sequences need to be of the same length, therefore a proper sequence alignment is needed before constructing phylogenetic trees. Regarding tree-creating algorithms, there are two main categories to choose from [1]: there are distance-based methods, which measure the differences between pairs of sequences and character-based methods, which use the columns of a sequence alignment directly. Examples of distance-based algorithms include neighbor joining (NJ) [2], UPGMA [3], WPGMA [3] and BIONJ [4], while the two most widely used character-based methods are the maximum parsimony (MP) and maximum likelihood (ML) methods. While all these techniques are popular, distance methods are usually recommended when computational time is an important factor; in other cases, parsimony and likelihood are preferred as they are more rigorous and have the ability to explore different combinations.

2.1 Sequence alignment

DNA is a molecule that has two polynucleotide chains forming a double helix and carries information about an organism's development and

reproduction. Genomes — genetic materials that make up an organism's body — consist of both protein-coding DNA (genes) and non-coding DNA. There are four different types of nucleoids — adenine (A), guanine (G), cytosine (C) and thymine (T) — that define the characteristics of the DNA containing them; DNA sequencing is the process of determining the order of such nucleoids in order to gain relevant information about the molecule at hand. Sequence alignment is a way of arranging multiple DNA sequences into a matrix so that regions that show a high degree of similarity in multiple sequences are placed underneath each other. As this process makes the characters in the sequences shift, a special character (commonly a "-") is used to fill in the gaps.

2.2 Neighbor joining

Neighbor joining is a distance-based method created by Saitou and Sei in 1987 [2]. A distance matrix is needed for all of the calculations in this method; this matrix should contain the pairwise distances between the input sequences according to a predefined distance metric. This matrix is reduced in size after each step because of the clustering nature of the method.

In each iteration, the following steps are repeated until there are only two nodes left:

1. Calculate the divergence (r) for every element. Divergence for element a is the sum of the distances from a to all the other elements.

2. Calculate the new, alternative distance matrix using the following formula:

$$d_{new}(i, j) = d(i, j) - \frac{r(i) + r(j)}{N - 2} \qquad (1)$$

where N is the number of elements in the current iteration.

3. Take the smallest value in the new distance matrix and the elements belonging to it (let them be denoted by a and b). Let p

be a new node that represents the common parent of a and b. The length of the new branches can be calculated as

$$\delta(a, p) = \frac{d(a, b)}{2} + \frac{r(a) - r(b)}{2(N - 2)} \qquad (2)$$

$$\delta(b, p) = d(a, b) - \delta(a, p) \qquad (3)$$

4. The rows and columns belonging to p need to be added to the original distance matrix, while the ones belonging to a and b need to be deleted. After this, the difference between p and the remaining nodes need to be calculated. Let's take an arbitrary element c as an example:

$$d(c, p) = \frac{d(a, c) + d(b, c) - d(a, b)}{2} \qquad (4)$$

Branches (a, p) and (b, p) are now part of the tree that is being built.

After this, the remaining two elements can be joined by taking the ancestral node created in the last iteration and making the other node its child with their distance — that was specified in the last iteration — as the branch length.

Although it was originally intended for unrooted trees, NJ can easily be extended to rooted trees. This is one of the simplest and fastest methods in phylogeny, but it may not produce accurate results.

2.3 WPGMA

Originally attributed to Sokal and Michener [3], WPGMA (weighted pair group method with arithmetic mean) is a hierarchical clustering method that uses a distance matrix in a similar fashion as the neighbor joining method. Each step consists of the following and is repeated until there is only one node left:

1. Find the smallest element in the distance matrix. Suppose that it is $d(a, b)$, therefore it belongs to elements a and b (both of them can be clusters). Create a parent node called p.

2. Calculate branch lengths $\delta(a, p)$ and $\delta(b, p)$. If both a and b are leaves, these lengths are simply

$$\delta(a, p) = \delta(b, p) = \frac{d(a, b)}{2} \qquad (5)$$

If a or b are clusters, the height of the subtree belonging to them needs to be deducted from the amount mentioned before. Suppose that a is a cluster and that c is a leaf of the subtree belonging to it. Then

$$\delta(c, p) = \frac{d(a, b)}{2} \qquad (6)$$

$$\delta(a, p) = \delta(c, p) - \delta(c, a) =$$
$$= \frac{d(a, b)}{2} - \delta(c, a) \qquad (7)$$

In the case where a is a leaf, c could be represented by a itself and $\delta(c, a)$ would be zero, therefore this is a generalized formula that can be used in any situation.

3. After cluster $p = (a, b)$ is created, it should be integrated into the distance matrix as a new element, therefore both a new column and a new row should be created (additionally, the columns and rows belonging to a and b can be deleted). The distance between q and any arbitrary element q can be calculated as

$$d(p, q) = d((a, b), q) = \frac{d(a, q) + d(b, q)}{2} \qquad (8)$$

This step is not needed in the last iteration.

2.4 UPGMA

Also created by Sokal and Michener, UPGMA (unweighted pair group method with arithmetic mean) is highly similar to WPGMA. The only difference lies in that it keeps track of the number of elements in each cluster, making the resulting calculations unweighted. In Step 3, UPGMA uses cluster weights, resulting in the following distance matrix update rule:

$$d((a, b), q) = \frac{d(a, q) \cdot m + d(b, q) \cdot n}{m + n} \qquad (9)$$

where a consists of m and b consists of n elements. This is called *proportional averaging.*

There are situations in which it is not recommended to use UPGMA, e.g., when the three-point criterion is not fulfilled. This criterion demands that for any three elements $d(a, c) \leq max(d(a, b), d(b, c))$, i.e., that the two greatest distances be equal. This is needed because UPGMA assumes a constant rate of evolution across lineages, therefore none of the leaves can be further away from the root than the others.

2.5 Maximum parsimony

Maximum parsimony is an umbrella term for all the methods that are based on the principle that a model is better than another if it implies fewer evolutionary events. The number of evolutionary events is represented by a cost called the *parsimonious score.* There are two major categories: *exact* methods are the ones that consider the score of all possible trees, thus giving a global minimum, and *heuristic* methods are those that use some kind of technique to find a significant subset of all the trees and produces a somewhat sub-optimal result. As exact methods are often computationally infeasible, it is recommended to use a heuristic method and let it run for a sufficient amount of time.

Fitch's algorithm [5] can be used to calculate the score of an input tree [6] [7]: the first-pass of the algorithm starts from the leaves and goes up to the root to determine the parsimonious score and possible sets of bases at each internal node, while the second-pass goes from top to bottom and helps in determining a hypothetical tree. In each step of the first phase, the algorithm takes a look at the bases belonging to the children nodes (which consist of one element in the case of the leaves), and then, assigns their intersection to the parent node if they have bases in common. If they do not have any bases

in common, the parent node gets the union of the children's sets and the parsimonious score is increased by 1. This algorithm is repeated for each site — that is, position in the sequences — in all input sequences and the score is summed up in the end.

There are a number of heuristic methods that can help in exploring new trees, for example nearest neighbor interchange (NNI), subtree pruning and regrafting (SPR), or tree bisection and reconnection (TBR). NNI is the simplest one but is used for unrooted binary trees; the current tree may produce two new trees if all four branches connecting to one of its internal branches are swapped in every possible combination. In SPR, a certain subtree is removed from the current tree and gets reinserted elsewhere. TBR does the same as SPR, but instead of reinserting the subtree in the form it was in originally, it tries every possible way of connecting the two subtrees; similarly to NNI, it works for unrooted trees only. These methods are stochastic, but they should be able to approximate the original distribution of trees given that they have enough time to run.

Branch and bound, an exact method that filters out a certain number of trees, may reduce the running time of the exhaustive search — an exact method that goes through all possible trees — significantly. This makes use of the fact that adding a new edge to a tree increases its parsimonious score, thus avoids going for trees that are bigger than the current one in each step.

2.6 Maximum likelihood

Maximum likelihood, like in many other cases where probabilistic methods can be used, makes use of the basics of Bayesian statistics and examines the probability of a certain sequence given a model (the model is arbitrarily chosen). It may have a high algorithmic complexity as evaluating one sequence in itself may be computationally intensive, let alone multiple sequences. There are also Bayesian methods (e.g.,

MCMC) that build upon ML; the major difference between these two groups of tree-building algorithms is that Bayesian methods take prior knowledge into account.

2.7 Self-growing tree algorithm

Self-growing tree algorithm (or self-organizing tree algorithm) is based on a paper published in 1997 by Dopazo and Carazo [8] and is a combination of the Kohonen self-organizing map [9] and the growing cell structures algorithm of Fritzke [10]. This is an unsupervised learning network that first initializes a tree consisting of a small number of nodes (the paper mentions two sister nodes) and then alternates between growing the tree and adapting it to the input sequences until it is fully grown and every taxonomic unit is assigned a proper place. The strictest exit condition guarantees that every input sequence is associated to a unique cell.

First, the input sequences are converted to one-hot encoded values. Secondly, the tree's initial node(s) and the corresponding weight matrices that the encoded input sequences will ultimately be compared to need to be created; in order to distinguish inner nodes from leaves, the authors of the paper mention that it would be best to call them *nodes* and *cells*, respectively. In each step, nodes are considered "closed", meaning that they cannot be assigned sequences after they transition from being a cell to a node. After the first few cells are initialized with numbers ranging from 0 to 1, the alternating phases begin to take place. The first phase is called adaptation, which is basically what characterizes Kohonen's self-organizing maps: the input points are compared to all available cells and those that are closest to each of the inputs are updated, along with their neighborhood (in the paper, finding the winning cell and updating its neighborhood is referred to as a *presentation*). After running this for a number of epochs, the inputs are mapped to appropriate positions in the output space, namely, the cells that they are the closest to according to a predefined metric.

4

This distance metric is defined as

$$d_{S_j C_j} = \frac{\sum_{r=1}^{A} S_j(r,l) \cdot C_i(r,l)}{L} \qquad (10)$$

where S_k is sequence k, C_i is cell i, A is the number of characters in the alphabet (in our case, four) and L is the length of the sequences.

The concept of neighborhood is not as intuitive as when the output space is a 1D or 2D space; in the case where the winning cell's sister is a cell, the winning cell, the sister cell and their common ancestor (mother cell) all get updated — it should be noted that all three of them are assigned different update rates. However, if the sister is actually a subbranch of the tree and consists of multiple nodes and cells, only the winning cell is updated.

The second phase is the growing of the tree. As it is mentioned in the paper [8], "the growing of the network takes place in the cell having higher resources. This cell gives rise to two new descendant cells [...] and transforms itself into a node. At this moment, its upper node becomes a grandmother node, and thereafter, it does not receive any more updating. The two new cells are, in principle, identical to the node which generated them".

The tree stops growing when the resource value of the cell with the highest value is smaller than a predefined threshold — this threshold should be zero if the goal is to map each sequence to a unique cell, or rather, a sufficiently small number in order to avoid inaccuracies in numerical calculations. The concept of resource value is related to the distance between cells and input points, and is defined as

$$R_i = \frac{\sum_{k=1}^{K} d_{S_k C_i}}{K} \qquad (11)$$

where R is the resource value belonging to cell i, S_k is sequence k, C_i is cell i and the summation is done over the K sequences associated to cell i.

The criteria used for monitoring the convergence of the network relies on the definition of the total error, ϵ, "defined as the summation of the distances of each sequence to the corresponding winning cell after an epoch" [8]. This helps in setting an exit condition for the adaptation process — e.g., the process can be ended when the relative increase of the error falls below a small threshold. The relative increase in error is

$$\left| \frac{\epsilon_t - \epsilon_{t-1}}{\epsilon_{t-1}} \right| < E \qquad (12)$$

What is also interesting is that there is an extended update rule that is based on Kohonen's original formula:

$$C_i(\tau + 1) = C_i(\tau) + \eta_{t,\tau,i} \cdot [S_j - C_i(\tau)] \quad (13)$$

and $\eta_{t,\tau,i}$ is defined as a factor dependent on the number of cycles among others:

$$\eta_{t,\tau,i} = \alpha_i \cdot \frac{1-t}{M_t} \cdot (1 - b\tau) \qquad (14)$$

where α_i is a coefficient dependent on the role of the current cell, t is the total number of presentations, M_t is the maximum number of presentations allowed ($\mu \times A \times L$) and b is the slope for the reduction of the interaction as the number of presentations, τ, increases within a cycle.

3   Implementation

I implemented the algorithms mentioned in the previous section using Python in combination with NumPy and Pandas. For reference and sanity tests, I used Biopython's implementation of some of the algorithms mentioned above. Since some of the data structures needed were not feasible to implement within the scope of the project, I decided to use Biopython's corresponding classes to facilitate the work; these are `Bio.SeqIO`, `Bio.`

`Align.MultipleSeqAlignment`, `Bio.Alphabet` and `Bio.Phylo.BaseTree`. They were needed to read in data, create sequence alignments in the format needed, fill in the gaps in the sequence alignments and create the trees in a form that can be visualized easily, respectively.

I decided to use rooted trees because they are generally more accurate and more easily comparable in a lot of cases. Biopython's `BaseTree` can easily be visualized with `Bio.Phylo.draw`, a function that takes into account features like branch length and custom labels.

The methods I implemented are neighbor joining, UPGMA, WPGMA, maximum parsimony and SOTA. In the case of MP, I implemented an exact algorithm that uses exhaustive search to find all the trees (with an additional option for branch and bound) and a heuristic algorithm that uses SPR. SOTA contains a few hyper-parameters; these can be tuned before extensive testing. I found that the following parameters provided a relatively fast convergence: $\alpha_w = 0.5$ (current cell's update rate), $\alpha_m = 0.02$ (mother cell's update rate), $\alpha_s = 0.01$ (sister cell's update rate), $b = 0$ (slope), $\eta = 0.5$ (learning rate).

3.1 Maximum parsimony — versions

While the exact method doesn't need any fine-tuning (branch and bound produces the same result as the full version), heuristic MP with SPR needs the maximum number of iterations — that is, the number of trees to test — as input. The heuristic version needs to be run for a relatively long time to find a sufficiently good result; as it is stochastic, the use of optimization methods may boost its performance or speed. For the remaining experiments, I decided to test the exact method instead of the heuristic one as it is able to produce exact results.

Figure 1 shows a tree resulting from a 50-iteration heuristic MP run on 9 randomly selected species; although the algorithm managed to find some smaller clusters (*baboon* and *squirrel monkey*, *mouse* and *bat*), it is still far from

optimal.

4 Data

The first dataset I used is a protein-coding sequence alignment [11] consisting of 44 sequences of different vertebrate taxa. The input is first processed by Biopython's sequence alignment class before being passed to any of the algorithms implemented. Since this dataset is evaluated in a 2008 paper by Prasad et al. [12], I decided to refer to it and compare my results with the findings of the authors of this publication. The other dataset I ran my experiments on uses non-coding DNA sequences but is identical to the coding dataset in every other aspect, including the fact that it can be found in Prasad's work.

5 Experiments

5.1 Comparison of running time

All the experiments were run on CPU (Intel Core i7-8650U, 1.9 GHz); none of the algorithms were optimized for GPU which made testing hard to a certain extent. Some methods were computationally intensive (e.g, maximum parsimony with 9 species and SOTA with 44 species), therefore I will include the result of only one test run in these cases. The following results were produced on the coding dataset; it can be assumed that the relationship between the speed of these algorithms holds for different datasets.

Table 1: Running time of the distance-based methods on the coding dataset. 44 taxa are used. Numbers are based on three test runs.

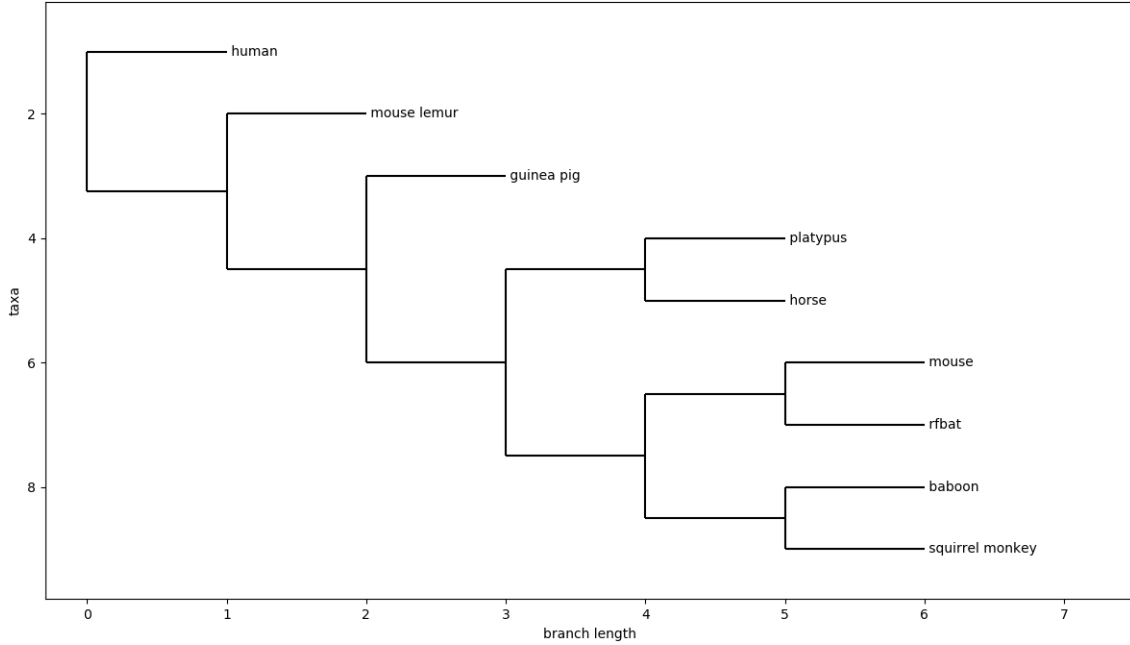| Method	Mean (seconds)	Std.
NJ	3.3105	0.3885
UPGMA	0.6767	0.0446
WPGMA	0.5359	0.0526

Figure 1: The result of MP run on 8 elements of the coding dataset. The result is the best tree after 50 iterations.

Table 2: Running time of the exact maximum parsimony method on the coding dataset. Uses branch and bound. Based on three test runs (except for the case with 9 taxa).

Nr. of taxa	Mean (seconds)	Std.
6	43.9988	2.7520
7	916.6252	661.2072
8	8613.8869	5145.0788
9	228420.7188	-

Table 3: Running time of the self-growing tree algorithm on the coding dataset. Based on three test runs (except for the case with 44 taxa).

Nr. of taxa	Mean (seconds)	Std.
4	53.8339	4.49524
5	93.3756	3.4764
15	2399.81665	464.18335
44	36669.3622	-

It can be seen that the three distance-based methods are immensely fast, especially since they do not entail convoluted tree-manipulation steps. SOTA runs in a reasonable amount of time, and it may be reasonable to think that it can provide a better solution than NJ, UPGMA and WPGMA while still being able to keep its running time under control.

5.2 Comparison of accuracy

5.2.1 Coding dataset

I first tested all five algorithms on the coding dataset. Regarding the result of neighbor joining (Figure 2), we can see that *cats* and *ferrets* are closely related (Carnivora, Laurasiatheria, Boreoeutheria), but they are further away from *dogs* and *pigs* which should be closely related to them according to the point of reference [12]. *Sheep* and *cows* are related (Cetartiodactyla), as well as species that are close to *humans* (baboon, orangutan, chimp, gorilla, colobus monkey, macaque, gibbon). Neighbor
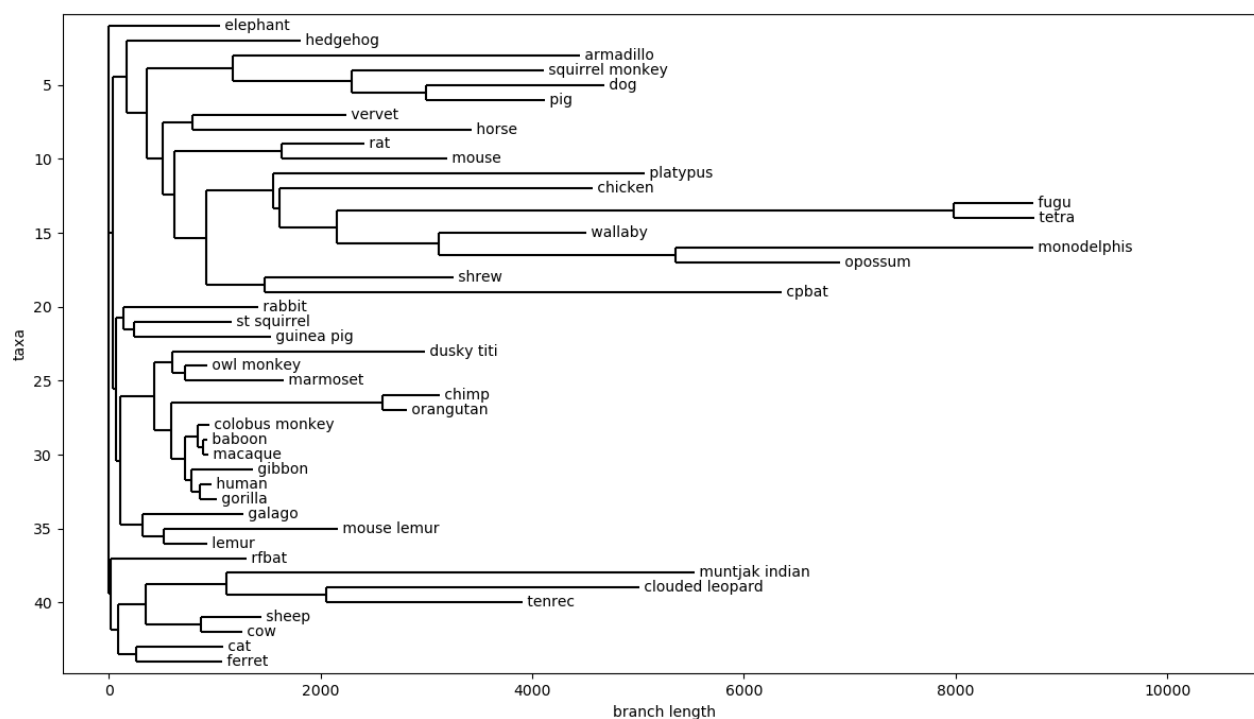
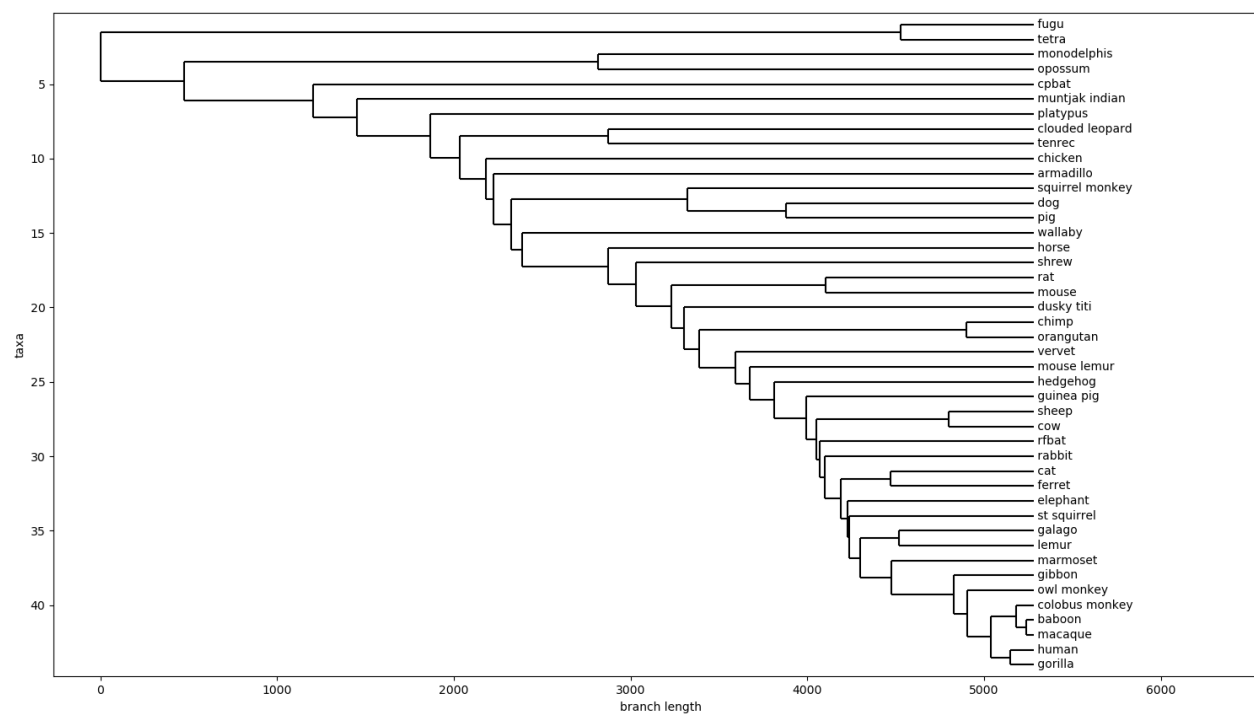Figure 2: The result of neighbor joining run on the coding dataset.

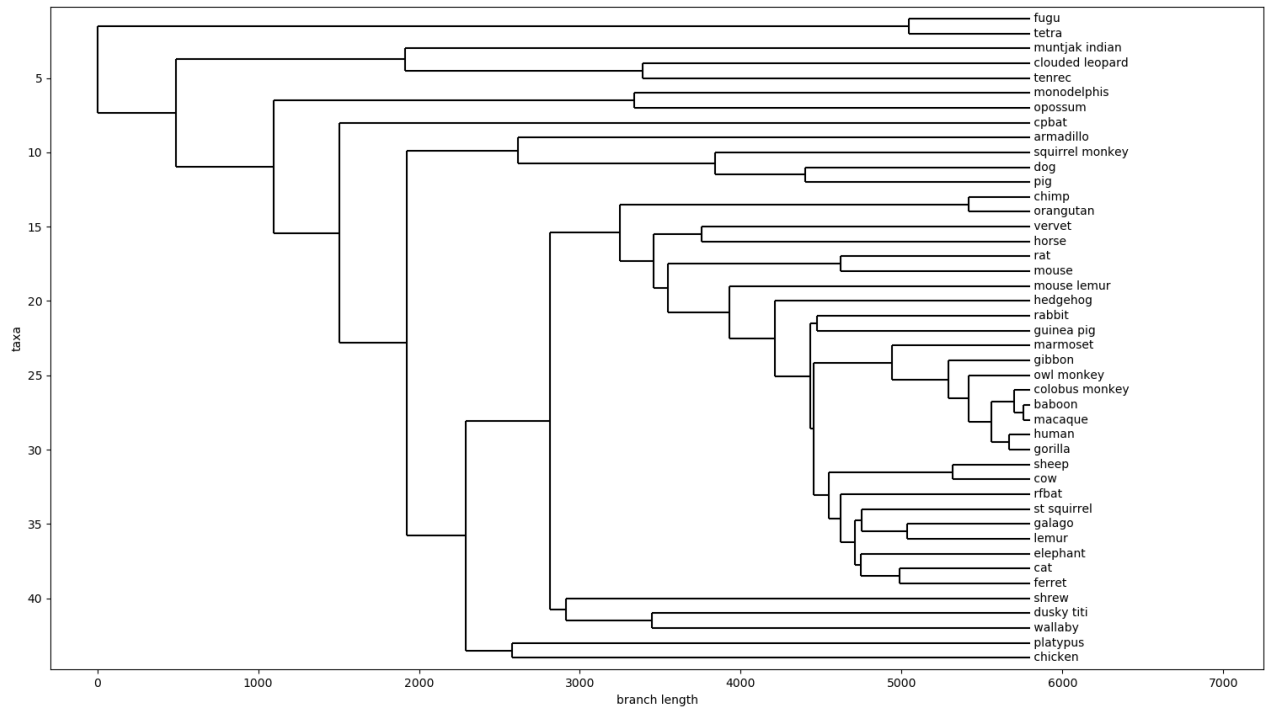Figure 3: The result of UPGMA run on the coding dataset.

8

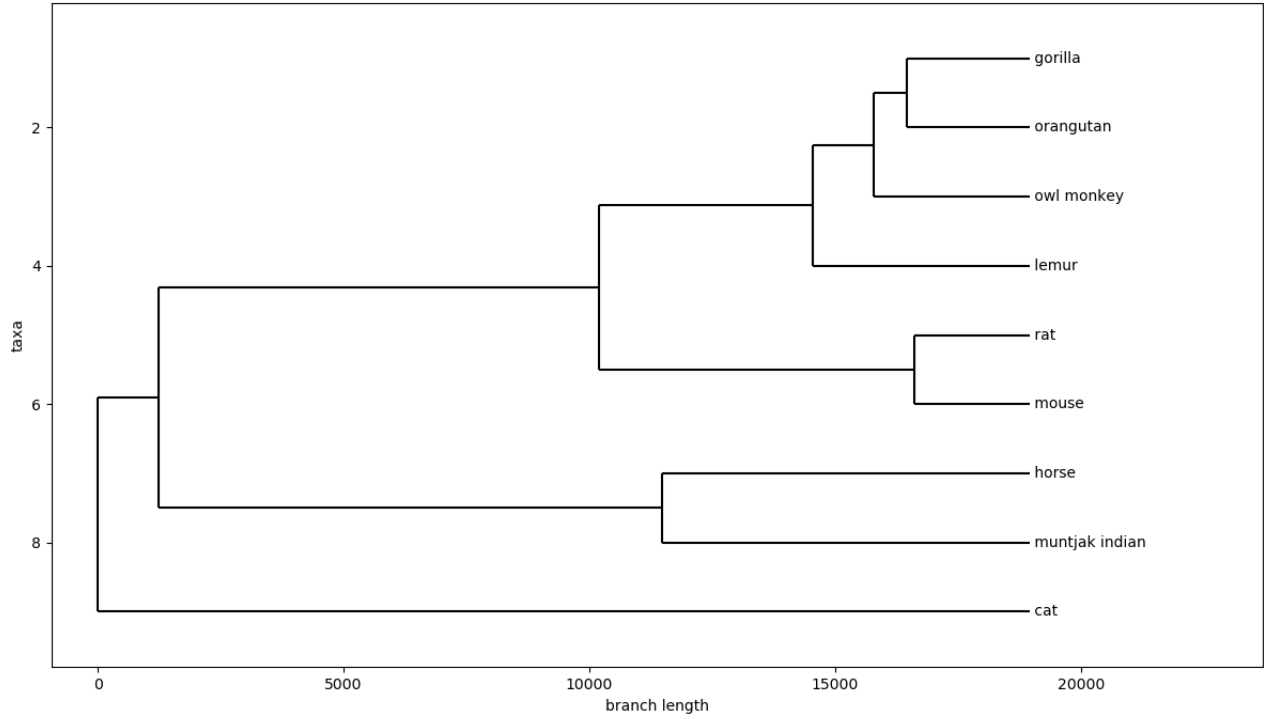Figure 4: The result of WPGMA run on the coding dataset.

Figure 5: The result of exact MP with branch and bound run on 9 elements of the coding dataset. Branch length have been adjusted to the parsimonious scores of the nodes.
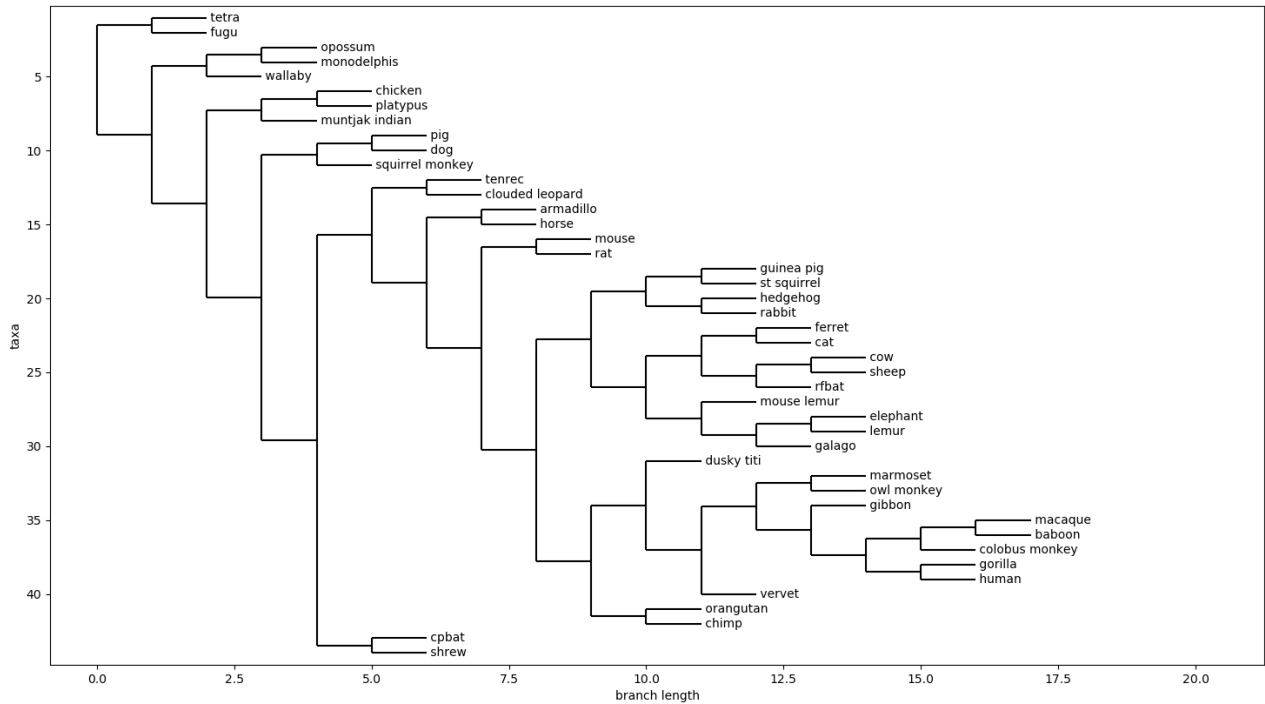
Figure 6: The result of SOTA run on the coding dataset.

joining managed to position other primates in the same cluster as well. The algorithm did well by placing some rodents in the vicinity of primates; it, however, put *rats* and *mice* into a different high-level category as humans. Other discrepancies include putting mostly dissimilar species like *muntjak* and *clouded leopard* next to each other but it can be said that it performed fairly well, especially since it is the most basic method.

When it comes to UPGMA (Figure 3), cats and ferrets are still far from dogs and pigs but munjaks and leopards moved further from each other. Besides this, it performed a little worse than neighbor joining; *chimps* and *orangutans* have moved away from other primates, and other high-level categories remained mixed or obscure. Closely tied species still go together — *tetraodon-fugu, sheep-cow, baboon-macaque, rat-mouse, human-gorilla* —, but other highly similar species — *muntjak-cow, horseshoe bat and short-tailed bat, mouse lemur and right-tailed lemur, owl money and squirrel monkey,*

short-tailed opossum (monodelphis) and North American opossum (opossum) — are separated from each other.

In the case of WPGMA (Figure 4), only the mid-level clusters seem to have moved around compared to the structure of the tree produced by UPGMA. Most of the statements that apply to UPGMA are true of WPGMA as well. Slight improvements can be seen in certain relationships, e.g. *platypuses* and *chickens* got paired up and *guinea pigs* got closer to *rats* and *mice*; these changes, however, are not sufficient to claim that there is a significant difference in UPGMA and WPGMA.

As mentioned before, time is an important factor when using maximum parsimony. The maximum number of taxa I ran the exact method on was 9, therefore only a subtree of the result is shown in Figure 5. The exact method has the capacity to evaluate every possible combination of branches, therefore the solution it provides is optimal. It can be said that the resulting tree bears a high degree of resemblance

to Figure 1 in Prasad's paper [12].

Finally, SOTA (Figure 6) created a tree that has guessed the major clusters right, but there are minor disparities in less obvious cases: *muntjak* is grouped with *chicken* and *platypus*, *squirrel monkey* is with *pig* and *dog* and *tenrec* belongs to the same branch as *clouded leopard*. It was generally good at identifying the similarities between different primate and rodent species, but there is still room for improvement, for example, *shrew* and one of the *bat* species have diverged from the other species of *Laurasiatheria*.

5.2.2 Non-coding dataset

Neighbor joining performed quite well (Figure 7); primates are in the same cluster except for *squirrel monkey* and *Glires* (hares, rabbits and rodents) are also together except for *rabbit*, which was instead linked with primates. Cats are still far from dogs, and rats and mice are far from humans, too.

UPGMA (Figure 8) produced slightly better results that neighbor joining; the main categories seem stable, although some elements in Laurasiatheria and Glires have been moved around. The tree created with WPGMA (Figure 9) is also remarkable, only some species like *horse* and *vervet*, *dusky titi* and *elephant* deviated from their expected places.

For the sake of simplicity, I decided to make two test runs on 8 taxa instead of one test run on 9 taxa in the case of maximum parsimony. Figure 10 is very similar to Figure 5; species were chosen to coincide with the ones shown in the non-coding run and as it can be seen there is little or no difference between the two types of sequences in the case of the species selected. Figure 11, although there is no point of comparison from the coding dataset, shows a tree that seems to contain mostly bad taxonomic relationships: according to the tree, primates have diverged early on while *squirrel monkey* is closer to the *guinea pig* than any other primate. This may be due to the lack of information in the non-coding dataset, which contradicts what has been mentioned in correlation with Figure 10; in order to determine if it is really the case, further experimentation with different combinations of species is needed.

6 Conclusion

Among distance-based methods, WPGMA performed slightly better than NJ and UPGMA on the two datasets. Exact MP with branch and bound could be a good choice as it has produced mostly accurate results, but its running time creates an obstacle in cases with a high number of taxa; heuristic MP with SPR could be a good substitute but there is no guarantee that the sub-optimal result it finds is a sufficiently good one. SOTA is not significantly better than the distance-based methods and its computational complexity is a hindrance to accuracy.

Combining these methods by e.g. starting out from a tree pre-calculated with neighbor joining and then finding better solutions with heuristic MP could result in a possible improvement in performance; such optimization techniques are often used in the field of phylogenetics [13]. Additionally, it is possible that ML methods give better results than MP methods under special circumstances; finding out which one is better than the other for specific datasets needs extensive experimentation, which I leave for future work.

The repository is available on GitHub. Since the project provides a simplified overview of various phylogenetic methods, I hope that it can serve as reference for others who are interested in this field.
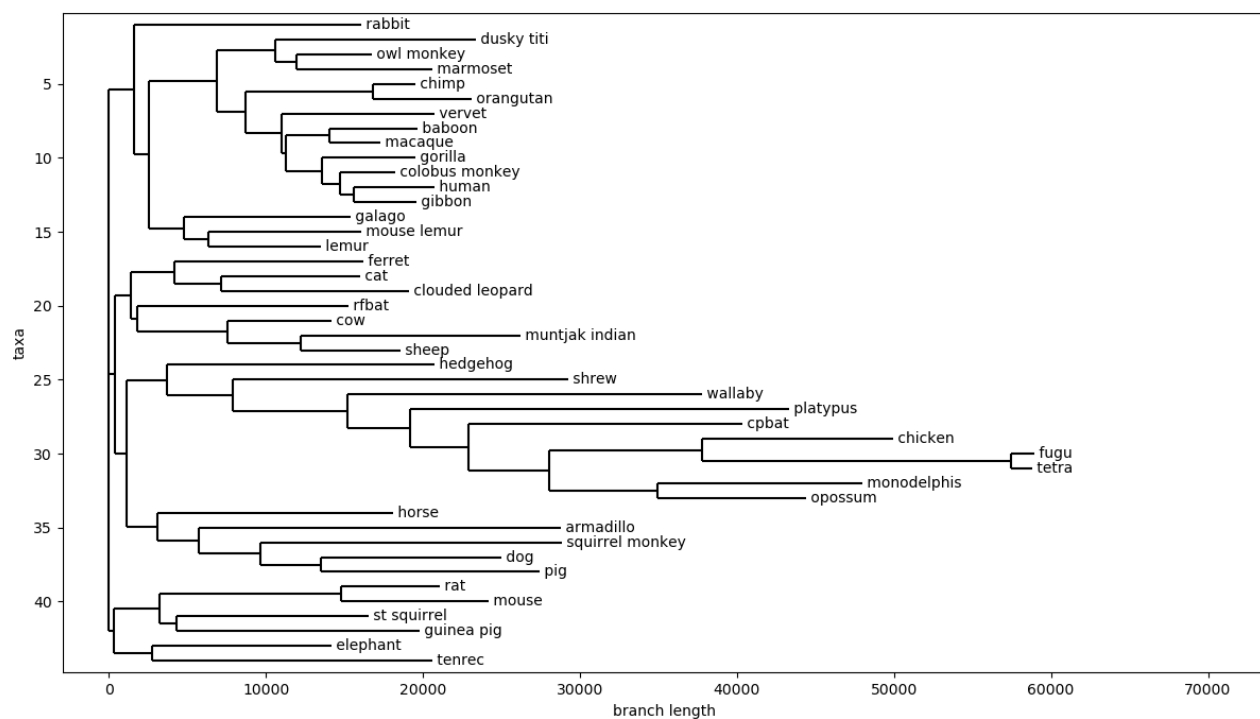
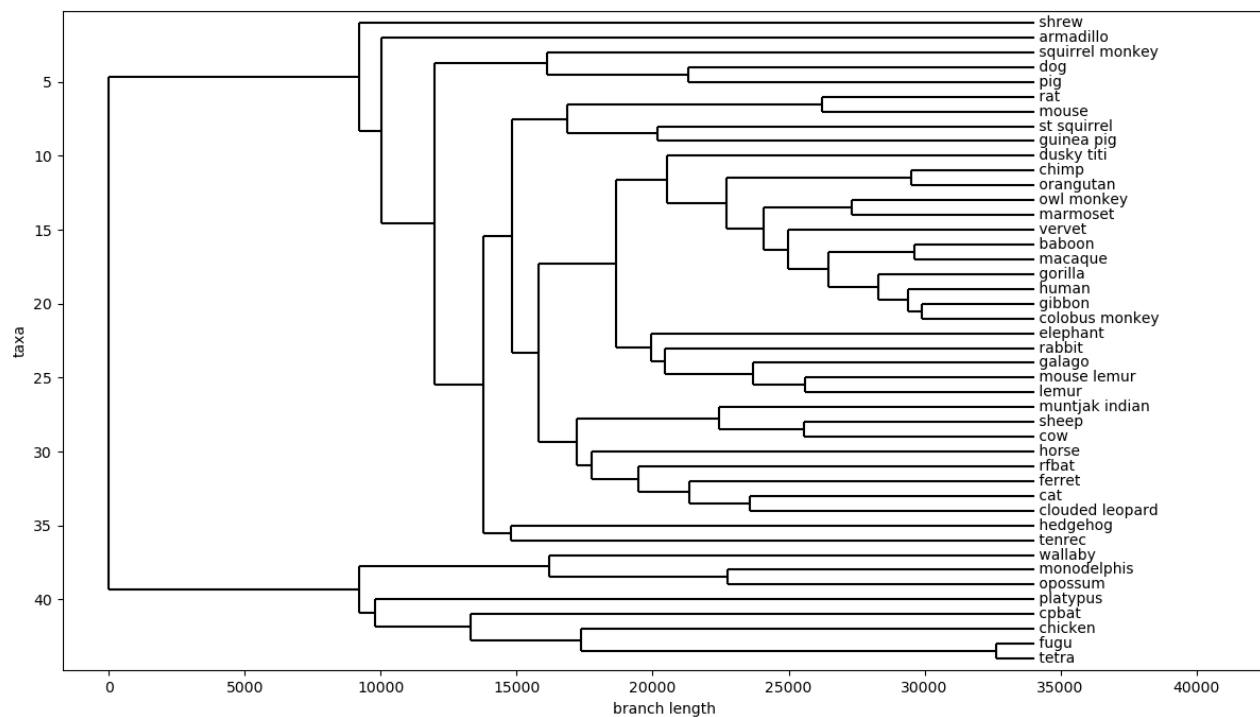Figure 7: The result of neighbor joining run on the non-coding dataset.

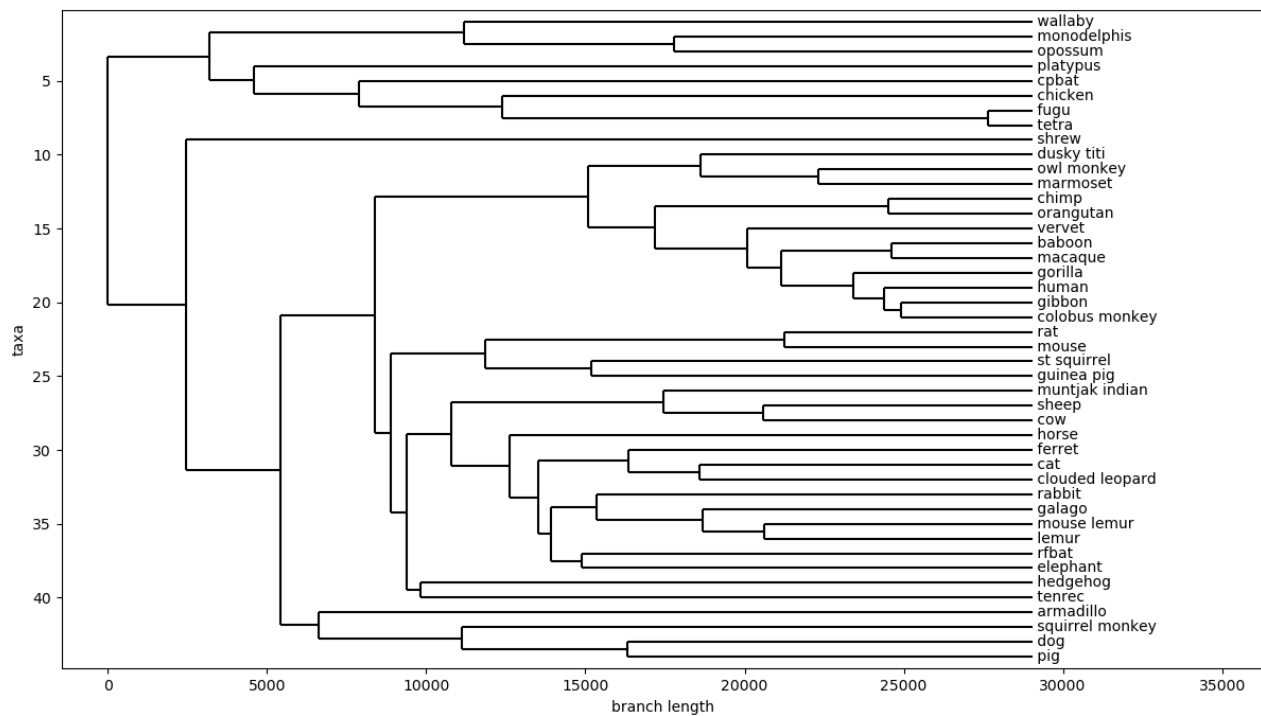Figure 8: The result of UPGMA run on the non-coding dataset.

12

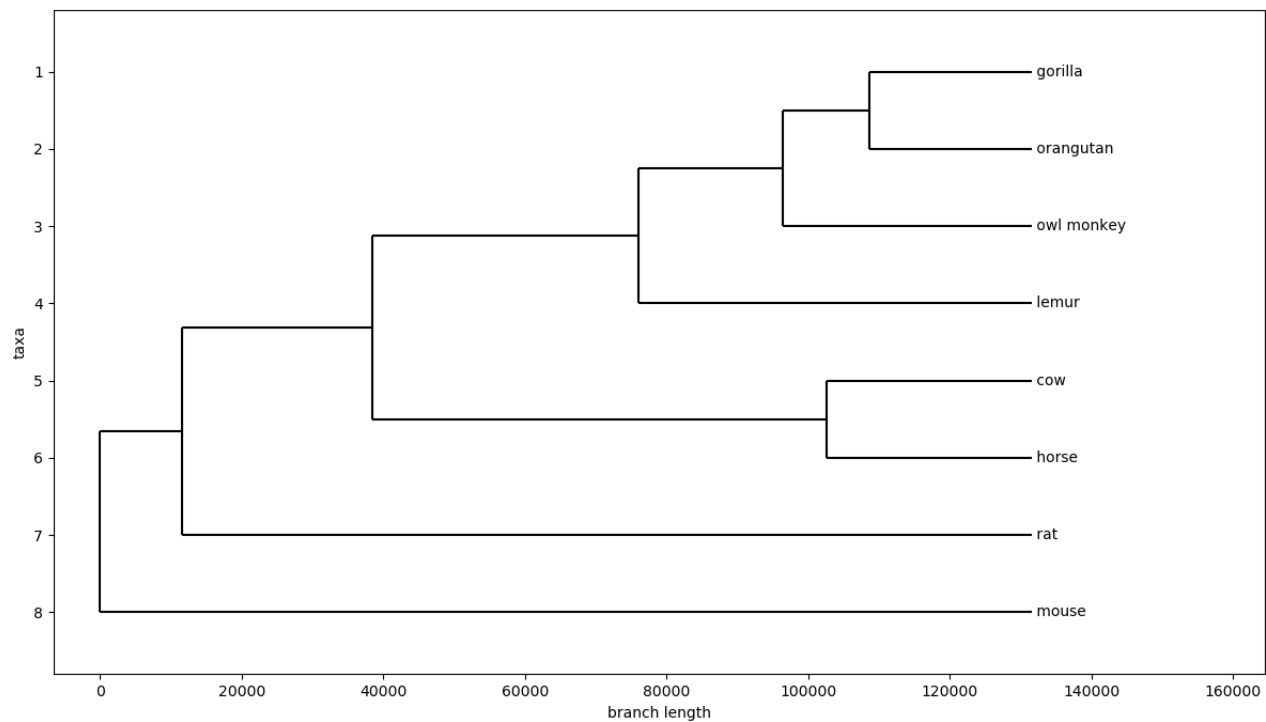Figure 9: The result of WPGMA run on the non-coding dataset.

Figure 10: The result of exact MP with branch and bound run on 8 elements of the non-coding dataset. Branch length have been adjusted to the parsimonious scores of the nodes.
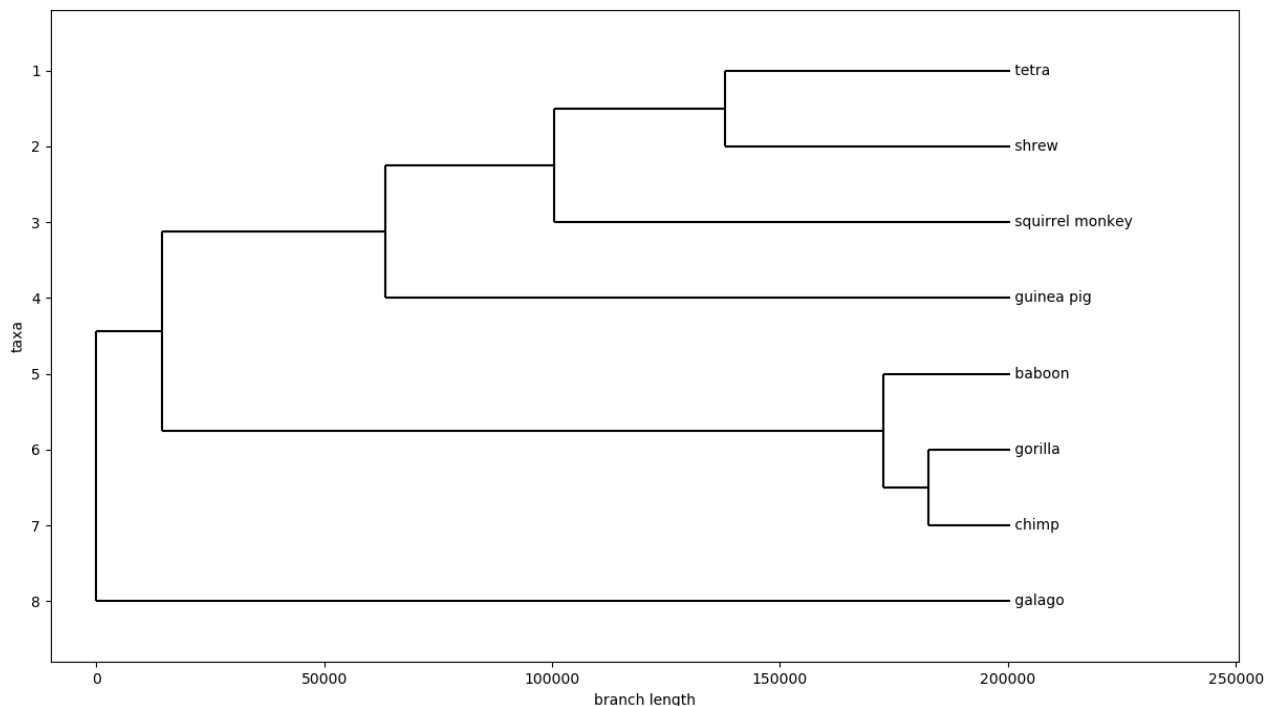
Figure 11: The result of exact MP with branch and bound run on 8 elements of the non-coding dataset. Branch length have been adjusted to the parsimonious scores of the nodes.

References

[1] Alexandre De Bruyn, Darren P. Martin, and Pierre Lefeuvre. Phylogenetic reconstruction methods: An overview. In *Methods in Molecular Biology*, pages 257–277. Humana Press, December 2013.

[2] The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, July 1987.

[3] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[4] O. Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution*, 14(7):685–695, July 1997.

[5] Walter M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20(4):406, December 1971.

[6] Karla Esmeralda Vazquez Ortiz. *Advanced methods to solve the maximum parsimony problem*. PhD thesis, Université d'Angers, 2016. Computational Complexity [cs.CC]. English. NNT: 2016ANGE0015ff. fftel-01479049.

[7] Martin Vingron, Jens Stoye, Hannes Luz, and Roland Wittler. Algorithms for phylogenetic reconstructions. February 2002-2009. Lecture notes from Bielefeld University.

[8] Joaquín Dopazo and José María Carazo. Phylogenetic Reconstruction Using an Unsupervised Growing Neural Network That Adopts the Topology of a Phylogenetic Tree. *Journal of Molecular Evolution*, 44(2):226–233, February 1997.

[9] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[10] Bernd Fritzke. Growing cell structures — a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, January 1994.

[11] NIH Intramural Sequencing Center. NISC Publications with Supplemental Data — Supplementary Data for Prasad et al. 2008. [Online; accessed 1-June-2020].

[12] Arjun B. Prasad, Marc W. Allard, and Eric D. Green and. Confirming the phylogeny of mammals by use of large comparative sequence data sets. *Molecular Biology and Evolution*, 25(9):1795–1808, May 2008.

[13] Marketa Zvelebil. *Understanding Bioinformatics*, page 286. Garland Science, August 2007.