Computing Geodesic Tree Distances with the GTP Algorithm

Miguel Caires
Instituto Superior Técnico, Lisboa, Portugal

December 2021

Abstract

The geometry of tree space $\mathcal{T}_n$ introduced by Billera, Holmes, and Vogtmann provides an effective way of comparing phylogenetic trees in the form of the geodesic distance, i.e. the length of the shortest path from one tree to another. We study and implement the first polynomial-time algorithm for finding geodesic paths in tree space $\mathcal{T}_n$, proposed by Owen and Provan. We focus on other graph problems and algorithms related to the Geodesic Treepath Problem, namely the maximum flow problem. Our results show that the geodesic distance can be efficiently computed in practice with the correct choice of algorithms and data structures, confirming the theoretical results we derived.
Keywords: Geodesic distance, phylogenetic trees, Geodesic Treepath Problem, Complexity, Graph Theory

1. Introduction

Phylogenetic trees are trees which depict evolutionary relationships between entities, which are typically biological species or strains. These trees are often constructed by algorithms which compare DNA sequences from selected parts of the genome using some distance measure (e.g.: the Hamming distance). However, the resulting trees change depending not only on the selection of genes or coding regions for the DNA sequences but also on the choice of the tree-building algorithm. From this uncertainty arises the need for comparing phylogenetic trees, and to this effect several measures have been proposed [6]. The geometry of tree spaces proposed by Billera et al. [1] indeed provides us with one such distance measure, which is the length of a geodesic path in a defined tree space $\mathcal{T}_n$, a quantity that shall henceforth be called geodesic distance. The geodesic distance seems to be the most appropriate quantitative comparison, since it incorporates aspects of tree topology and numerical edge lengths in a single measure, whereas other measures often lose information by focusing exclusively on tree topology and cannot be computed efficiently.

The Geodesic Treepath Problem (GTP) is the problem of finding the geodesic path between two trees in tree space $\mathcal{T}_n$. The first polynomial-time algorithm able to solve this problem was presented by Owen and Provan [5], seeing as two previous algorithms by Owen [4] and by Kupczok et al. [3], had exponential time complexity. In this thesis we intend to examine and implement this polynomial-time algorithm, to be referred to as the GTP algorithm. We will also perform experimental analysis of the time and space complexity of different versions of the algorithm, and verify whether our implementation is consistent with the theoretical analysis.

The GTP algorithm relies on an iterative approach that aims to optimize path length at each step, given some adequate initial path. The conditions for a path between two trees to be the geodesic path were introduced in [4], and in [5] they were formulated as the problem of finding a minimum weight vertex cover in subgraphs of a specially defined bipartite graph called the incompatibility graph. In our thesis we describe the details of reducing the minimum weight vertex cover problem in a bipartite graph to a maximum flow problem in a flow-equivalent network. From there we introduce and explore some exact maximum flow algorithms and their worst-case time complexities for our specially defined networks, which is different than the worst-case time complexity in general networks. The exact algorithms we study are Dinitz's algorithm and Edmonds-Karp, two algorithms that derive from the Ford-Fulkerson method. Assuming that the input $G = (V, E)$ is a flow-equivalent network of some incompatibility graph, we derive a bound of $\mathcal{O}(|V||E|)$ for the time complexity of both these algorithms. In addition, we briefly overview some $(1 - \varepsilon)$-approximation algorithms which improve this time complexity at the expense of a relative error.

2. Background

2.1. Tree Space

Let $\mathcal{T}_n$ be the set of trees which have exactly n leaves. Any tree $T \in \mathcal{T}_n$ has at maximum $n-2$ internal edges, i.e. edges whose endpoints are not leaves. Therefore, a given tree with variable internal edge lengths can be represented as a point in $n-2$ dimensional space with positive coordinates, which we call an orthant of $\mathbf{R}^{n-2}$.

Given an edge e belonging to a tree T a *split* $\sigma_e = X_e|\overline{X_e}$ is defined as a partition of the tree's leafset into two disjoint subsets X_e and its complement $\overline{X_e}$, resulting from the removal of edge e from T. Given two edges e, f (from the same or from two different trees with n leaves each), the splits σ_e and σ_f are said to be compatible if at least one of the following

$$X_e \cap X_f, X_e \cap \overline{X_f}, \overline{X_e} \cap X_f, \overline{X_e} \cap \overline{X_f}$$

is the empty set. Let $\mathcal{X}, \mathcal{Y}$ be distinct sets of splits. $\mathcal{X}$ is said to be a compatible set of splits if any two splits in $\mathcal{X}$ are compatible. $\mathcal{X}$ is said to be compatible with $\mathcal{Y}$ if x is compatible with y for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

A tree can therefore be defined by a set of leaves and by a correspondence between splits and the respective edge length. Hence we shall denote a tree $T \in \mathcal{T}_n$ as $T = (X, \mathcal{E})$, where X is the set of leaf labels and $\mathcal{E}$ is a set of splits/edges, with the addition of a function $|.|_T : \mathcal{E} \to \mathbf{R}^+$ that assigns a positive length to each edge. It can be shown that there are exactly $(2n-3)!!$ non-identical trees in $\mathcal{T}_n$,[6, 2] non-identical trees being trees which do not have the exact same set of splits.

$\mathcal{T}_n$ is path-connected [1], which means that through continuous contraction and expansion of these unique edges one can transform any $T_1 \in \mathcal{T}_n$ into a different $T_2 \in \mathcal{T}_n$, along a continuous path $\Gamma = \{T(\lambda) \in \mathcal{T}_n : 0 \leq \lambda \leq 1\}$. For any given tree $T \in \mathcal{T}_n$ the set of its $n-2$ splits is compatible, and any set of $n-2$ compatible splits (of a leafset of n elements) defines a valid tree. In other words, two splits are compatible if and only if they can coexist in the same tree. If two trees each have an internal edge corresponding to the same split we say that this edge is common between the two trees. If there are no common edges we say the trees are disjoint. Geometrically, the tree space $\mathcal{T}_n$ can be seen as a collection of $(2n-3)!!$ orthants of dimension $n-2$.

Given $T_1, T_2 \in \mathcal{T}_n$, the cone path between these trees corresponds to uniformly contracting all edges in T_1 until their lengths are zero and then uniformly expanding them until arriving at tree T_2. This path may or may not be the geodesic path, as evidenced in Figure 1, adapted from [5].

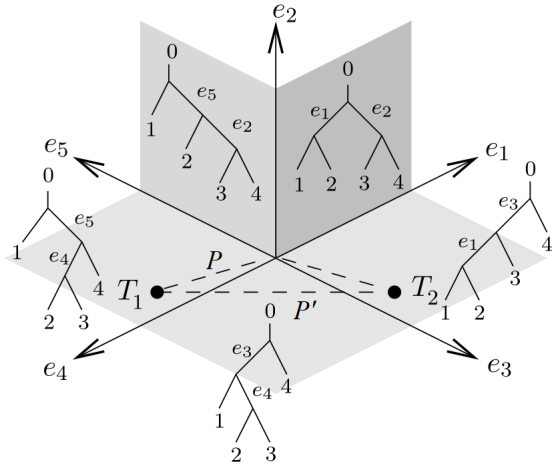


Figure 1: Embedding of $\mathcal{T}_4$ on $\mathbf{R}^3$. Path P is the cone path between T_1 and T_2 while P' is the geodesic path.

2.2. Geodesic Path properties

Given $T = (L, \mathcal{E}), T' = (L', \mathcal{E}') \in \mathcal{T}_n$ with disjoint set of splits and $\mathcal{A} = (A_1, ..., A_k)$ and $\mathcal{B} = (B_1, ..., B_k)$ partitions of $\mathcal{E}$ and $\mathcal{E}'$ respectively, assume that

(P1) For each $i > j$, A_i and B_j are compatible sets.

Then $B_1 \cup ... \cup B_i \cup A_{i+1} \cup A_k$ is a compatible set for all $1 \leq i \leq k$, and therefore defines a tree T_i belonging to the orthant generated by this set, denoted by $\mathcal{O}_i = \mathcal{O}(B_1 \cup ... \cup B_i \cup A_{i+1} \cup ... \cup A_k)$. $\mathcal{P} = \cup_{i=1}^{k} \mathcal{O}_i$ forms a connected space and we call $\mathcal{P}$ a *path space* with *support* $(\mathcal{A}, \mathcal{B})$. The shortest path from T to T' in $\mathcal{P}$ is called a *path space geodesic* for $\mathcal{P}$. $Gamma$ is a path space geodesic with support $(\mathcal{A}, \mathcal{B})$ if Γ satisfies **(P1)** and the following property:

$$\textbf{(P2)}\ \frac{||A_1||}{||B_1||} \leq \frac{||A_2||}{||B_2||} \leq ... \leq \frac{||A_k||}{||B_k||}.$$

where $||A_i|| = \sqrt{\sum_{e \in A_i} |e|}$. A path space satisfying (P1) and (P2) is called a proper path space, and the respective path space geodesic is called a proper path. Furthermore, if all the inequalities in the above equation are strict, Γ is assured to be the geodesic path between T and T'. In other words, Γ is the geodesic between T and T' if it satisfies (P1), (P2) and the following condition:

(P3) For each pair (A_i, B_i), there is no partition $C_1 \cup C_2$ of A_i and $D_1 \cup D_2$ of B_i such that C_1, C_2, D_1, D_2 are all non-empty, C_2 is compatible with D_1 and $\frac{||C_1||}{||D_1||} < \frac{||C_2||}{||D_2||}$.

Of the necessary and sufficient conditions for finding the geodesic path between two trees, (P3) suggests a procedure for iteratively improving upon

a starting proper path, such as the cone path. The condition is only satisfied if for each support pair no such partition exists in the conditions described by (P3).

The incompatibility graph $G(A, B)$ is a bipartite graph $G = (A \cup B, E)$ such that $A \subseteq \mathcal{E}$, $B \subseteq \mathcal{E}'$ correspond to node sets on the left and right sides of $G(A, B)$ respectively, and an edge $(a, b) \in E$ exists if $a \in A$ and $b \in B$ are incompatible edges, i.e. the splits they induce in their respective tree are incompatible.

In essence, the Extension Problem rewrites condition (P3) as the problem of finding an independent set in the incompatibility graph. We show that this can be solved by performing a maximum flow computation in a related graph, with a crucial intermediary step provided by the max-flow min-cut theorem.

3. The GTP Algorithm

3.1. Overview

As explained in the previous section, the (P3) condition hints at a procedure for iteratively improving upon a chosen initial proper path, until the length is minimal. Provided that two given trees are disjoint (the corresponding set of splits is disjoint), the procedure detailed in Algorithm 1 does precisely this.

Given input $T_1, T_2 \in \mathcal{T}_n$ such that $T_1 = (L, \mathcal{E})$ and $T_2 = (R, \mathcal{E}')$, the procedure begins by building a simple path between the two trees, represented as a support pair $(\mathcal{A}, \mathcal{B})$ where $\mathcal{A}$ and $\mathcal{B}$ are ordered vectors representing partitions of $\mathcal{E}$ and $\mathcal{E}'$ respectively. Initially $\mathcal{A} = (\mathcal{E})$ and $\mathcal{B} = (\mathcal{E}')$ represent the cone path, and the incompatibility graph G of T_1 and T_2 is built.

The algorithm enters a loop that will partition the sets $\mathcal{E}$ and $\mathcal{E}'$ depending on the existence of solutions to the Extension Problem, modifying the contents of the support pair $(\mathcal{A}, \mathcal{B})$ in the process. When the sets cannot be partitioned further, the algorithm terminates returning $(\mathcal{A}, \mathcal{B})$

The procedure for dealing with pairs of trees whose edge sets are not necessarily disjoint is done by splitting each tree at those common edges to form two forests of disjoint subtrees, where subtrees in each forest are indexed by their parent edge. This parent edge can be the root edge of the respective tree, which is always present in the set of common edges between any two trees. This allows us to form a collection of pairs of disjoint subtrees $\mathcal{P} = \{(T_1(e), T_2(e)) : e \in \mathcal{E} \cap \mathcal{E}'\}$, and we apply the previously explained disjoint trees version of the GTP algorithm for each of these pairs. The support pairs of the geodesic path between each pair of subtrees is then used to form the full geodesic path between T_1 and T_2, and the corresponding distance. Assuming $|\mathcal{E} \cap \mathcal{E}'| = r$, let $(A_1(l), ..., A_{k_l}(l))$,$(B_1(l), ..., B_{k_l}(l))$ be the support

Algorithm 1: The GTP Algorithm for disjoint trees.

Input: $T_1 = (L, \mathcal{E}), T_2 = (R, \mathcal{E}') \in \mathcal{T}_n$ disjoint n-trees
Output: Geodesic distance between T_1 and T_2

$G :=$ incompatibility graph of T_1 and T_2;
$w :=$ dictionary of vertex weights;
$\mathcal{A} := (\mathcal{E}), \mathcal{B} := (\mathcal{E}')$;
while *True* **do**
  for ($i = 0; i < |\mathcal{A}|; i++$ **)** {
    $A :=$ i'th element of $\mathcal{A}$;
    $B :=$ i'th element of $\mathcal{B}$;
    $G(A, B) :=$ subgraph of G induced by A and B;
    for ($e \in A$ **)** {
      $w[e] = \frac{|e|^2}{\|A\|^2}$
    for ($e \in B$ **)** {
      $w[e] = \frac{|e|^2}{\|\mathcal{E}'\|^2}$;
    $G' :=$ flow-equivalent network of G;
    $R :=$ residual matrix of applying max flow algorithm to G';
    $C_1 \cup D_2 :=$ min. weight vertex cover of $G(A, B)$ computed from R;
    // $(C_1 \subseteq A, D_2 \subseteq B)$
    $C_2 = A \setminus C_1, D_1 = B \setminus D_2$;
    $w' :=$ total weight of $C_1 \cup D_2$;
    if $w' < 1$ **then**
      Replace A with C_1, C_2 in $\mathcal{A}$;
      Replace B with D_1, D_2 in $\mathcal{B}$;
      break;
  if $w \geq 1$ **then**
    return
    $[\|A_1\| + \|B_1\|, ..., \|A_k\| + \|B_k\|]$;

for the geodesic path beteen a given subtree pair $(T_1(e), T_2(e)) \in \mathcal{P}$. Then the length of the geodesic path between T_1 and T_2 is given by the following value:

$$||(||A_1(1)|| + ||B_1(1)||, ..., ||A_{k_1}(1)|| + ||B_{k_1}(1)||,$$
$$...,$$
$$||A_1(r)|| + ||B_1(r)||, ..., ||A_{k_r}(r)|| + ||B_{k_r}(r)||,$$
$$|e_1|_{T_1} - |e_1|_{T_2}, ..., |e_r|_{T_1} - |e_r|_{T_2})||$$

Algorithm 2: The GTP Algorithm admitting common splits.

Input: $T_1, T_2 \in \mathcal{T}_n$
Output: The geodesic distance between T_1 and T_2
$C := \mathcal{E}_1 \cap \mathcal{E}_2$ //$T_1(i), T_2(i)$ are subtrees of T_1, T_2 indexed by the same $e \in C$
$r := |C|$;
$\mathcal{P}_1 := \{T_1(i)\}_{i=1}^r$;
$\mathcal{P}_2 := \{T_2(i)\}_{i=1}^r$;
$v :=$ empty list;
for ($i = 1; i < r; i++$ **)** {
  $p :=$ result of Algorithm 1 applied to $T_1(i)$ and $T_2(i)$;
  Concatenate list v with list p;
  if e_i *is not the root edge of* T_1 *and* T_2 **then**
    diff := $||e_i|_{T_1} - |e_i|_{T_2}|$;
    Append diff to list v;
$L :=$ length of list v;
return $\sqrt{\sum_{i=1}^{L} v[i]^2}$

3.2. Time and Space Complexity

Taking $T_1, T_2 \in \mathcal{T}_n$ as input, the algorithm first determines the set of splits belonging to both trees. Each tree has $n-2$ internal edges, and so the time for computing each split set is $\mathcal{O}(n)$. Set intersection can then be implemented in linear time using hash tables. Each tree is partitioned according to the common split set, and the version of the GTP algorithm for disjoint trees is applied afterwards to each pair of subtrees indexed by a given split. Assume that the partitions induced by slicing the trees at their common edges are given respectively by $\{T_1(i)\}_{i=1}^r$ and $\{T_2(i)\}_{i=1}^r$. Let n_i be the number of leaves of $T_1(i)$ and $T_2(i)$, other than a possible root node of degree 1.

Given $T_1(i)$ and $T_2(i)$, the disjoint trees version of the GTP algorithm determines their incompatibility graph. Since they both have $n_i - 2$ internal edges each, the worst case is when $\mathcal{O}(n_i^2)$ pairs of corresponding splits are incompatible. Determining the compatibility of two splits using bitwise operations on the split bitmasks we describe in the implementation details takes time $\mathcal{O}(n_i)$, meaning that the worst case time complexity for the construction of these graphs is of $\mathcal{O}(n_i^3)$. On the other hand, space needed to perform the relevant computations is just $\mathcal{O}(n_i^2)$ in the worst case, which is the space needed to store a representation of the resulting incompatibility graph along with the respective vertex weights. Thus the flow-equivalent network should also take space $\mathcal{O}(n_i^2)$ in the worst case. Since our implementation represents this network as an adjacency list and a matrix of capacities, it is expected that space complexity will be $\mathcal{O}(n_i^2)$.

Entering the main `while` loop in the disjoint version of GTP, given $A \in \mathcal{A}$ and $B \in \mathcal{B}$, the incompatibility graph $G(A, B)$ is a subgraph of G, and its adjacency list can be constructed simply by scanning the entries of the adjacency list of G corresponding to the nodes in $A \cup B$, taking $O(n_i^2)$ time.

In the flow-equivalent network, we have that $|V| = \mathcal{O}(n_i)$ and $|E| = \mathcal{O}(n_i^2)$ and thus time complexity of maximum-flow algorithms we studied is $\mathcal{O}(|V||E|) = \mathcal{O}(n_i^3)$ for the case of the flow-equivalent network of the bipartite graphs. However, the space needed is also the space taken by the representation of the flow network given as input. Since we chose to represent edge capacities in a capacity matrix, the space needed in our case is of $\mathcal{O}(n_i^2)$.

Determining the set of reachable nodes from the source in the residual graph is done with a BFS in time $\mathcal{O}(|E|) = \mathcal{O}(n_i^2)$, and computing the vertex cover from this cut takes time $\mathcal{O}(|V|) = \mathcal{O}(n_i)$.

If this vertex cover forms a valid solution to the Extension Problem the support pair is reconstructed accordingly, and since the number of internal edges of $T_1(i)$ and $T_2(i)$ is $n_i - 2$, there exist $\mathcal{O}(n_i)$ solutions of the Extension Problem throughout the execution of the disjoint trees GTP algorithm in the worst-case.

Therefore each execution of the disjoint trees version of the GTP algorithm should have worst-case time complexity in $\mathcal{O}(n_i^4)$. Observe that $\sum_{i=1}^r n_i = n + (r-1) < 2n$, since there are $r-1$ common edges between T_1 and T_2 and $r < n-2$. Due to these constraints the complexity of the GTP algorithm remains unaltered when applied to all pairs of subtrees. In other words, if the execution time of the disjoint GTP algorithm with $T_1(i), T_2(i) \in \mathcal{T}_{n_i}$ as input is given by $f(n) = an^4 + \omega(n^4)$ for some $a > 0$ then the complexity of GTP will be given by:

$$\sum_{i=1}^{r} f(n_i) = a\sum_{i=1}^{r} n_i^4 + \sum_{i=1}^{r} \omega(n_i^4)$$
$$\leq a\left(\sum_{i=1}^{r} n_i\right)^4 + \omega(n^4)$$
$$= an^4 + \omega(n^4) = \mathcal{O}(n^4)$$

Using a similar argument, space complexity of the GTP algorithm should be of $\mathcal{O}(n^2)$, which is the space needed to store and represent the capacity matrices of the incompatibility graphs needed for computations at any given point in the execution of algorithm. What is meant by this is that even though the total size of data structures built and used can be greater than $\mathcal{O}(n^2)$, they can be written over and replaced as they cease to be useful.

3.3. Experimental results

For our experiments we used trees generated from a continuous time birth-death model. The birth-death process simply starts with a root node and has two possible events, speciation(birth) or extinction(death). The time between two consecutive speciation events is given by an exponential distribution with some parameter λ, and likewise with parameter μ for two consecutive extinction events. Here λ and μ are called birth rate and death rate respectively, as they can also be seen as the average number of births or deaths occuring within one time unit.All pairwise comparisons were performed between 10 randomly generated birth death trees with n leaves, for $n = 20, 40, ..., 300$. The cubic root of the average execution time for each these pairs is plotted in Figure 2. Linear regressions in log-log scale for the data in these plots produces a slope of 2.64 with $r^2 = 0.9892$ for the version of GTP with Dinitz's algorithm and slope of 2.68 with $r^2 = 0.9988$ for the Edmonds-Karp version, as expected. If one ignores the obvious outlier for $n = 280$ in the graph corresponding to the Dinitz version of GTP, the linear regression in log-log scale produces slope 2.58 with $r^2 = 0.9955$.

Allocated memory for the GTP algorithm was also measured for different max-flow algorithms, and the square root of these results are presented in Figure 3. Log-log scale linear regression of the original results gave slopes of 1.51 and 1.57 with $r^2 = 0.9798$ and $r^2 = 0.9903$ for the version of GTP using Dinitz's algorithm and the Edmonds-Karp algorithm respectively. Performance of our GTP implementation is once again consistent to the previously established time complexity of $\mathcal{O}(n^4)$ and space complexity of $\mathcal{O}(n^2)$. However, average time complexity seems to conform to $\mathcal{O}(n^3)$, and this may be explained by the number of solutions to the Extension Problem not being as much as assumed

Figure 2: Cubic roots of execution time of the GTP algorithm using different max-flow algorithms.

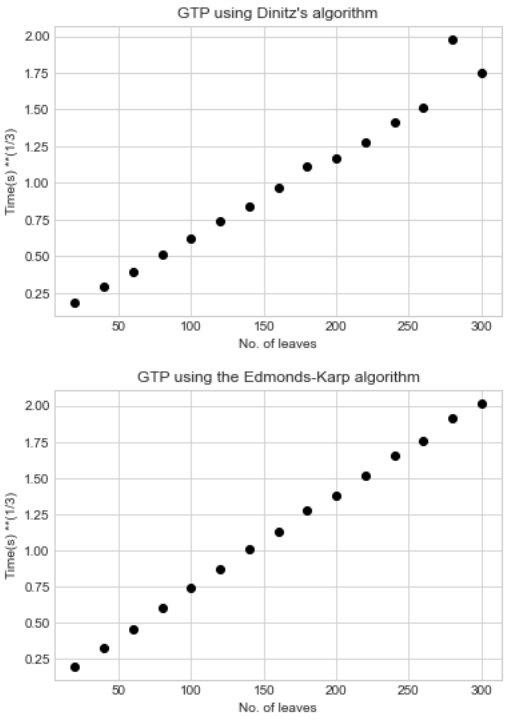
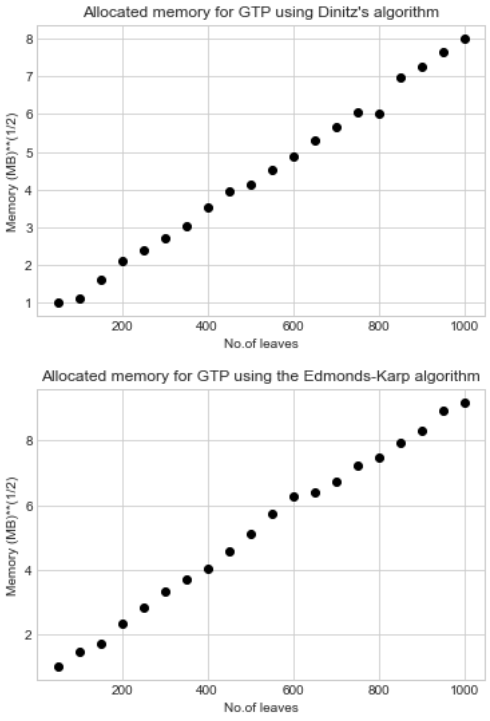


Figure 3: Square root of allocated memory for the GTP algorithm using different max-flow algorithms.



in the worst-case analysis, or the fact that the density of the incompatibility graphs decreases as n increases.

4. Conclusion

We explored the concept of geodesic paths in tree space and their usefulness in comparing tree structures in the field of phylogenetics. We have also explained how to solve the Geodesic Tree Path problem in polynomial time, and have implemented an efficient algorithm to do so. In particular, this involved understanding the procedure for iterative path finding in tree space and graph problems such as the minimum weight vertex cover, and required the knowledge of the best algorithms and data structures to solve such problems. We theoretically analysed all the chosen algorithms, and proposed original proofs of their time complexity in the given context, while also providing extensive detail and explanation of the most important parts of our code. Throughout all of this, we suggested the incorporation of approximation algorithms to achieve a more desirable time complexity. Finally, we performed exhaustive experiments with synthetic datasets to demonstrate the success of our implementation in comparison with the theoretical analysis.

References

[1] L. J. Billera, S. P. Holmes, and K. Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

[2] J. Felsenstein. The Number of Evolutionary Trees. *Systematic Biology*, 27(1):27–33, 03 1978.

[3] A. Kupczok, A. von Haeseler, and S. Klaere. An exact algorithm for the geodesic distance between phylogenetic trees. *Journal of computational biology : a journal of computational molecular cell biology*, 15:577–91, 07 2008.

[4] M. Owen. Computing geodesic distances in tree space, 2011.

[5] M. Owen and J. S. Provan. A fast algorithm for computing geodesic distances in tree space, 2009.

[6] B. L. Tavares. An analysis of the geodesic distance and other comparative metrics for tree-like structures, 2019.