An analysis of the Geodesic Distance and other comparative metrics for tree-like structures

Bernardo Lopo Tavares
bernardo.lopo@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

December 2018

Abstract

Graphs are interesting structures: extremely useful to depict real-life problems, extremely easy to understand given a sketch, extremely complicated to represent formally, extremely complicated to compare. Phylogeny is the study of the relations between biological entities. From it, the interest in comparing tree graphs grew more than in other fields of science. Since there is no definitive way to compare them, multiple distances were formalized over the years since the early sixties, when the first effective numerical method to compare dendrograms was described. This work consists of formalizing, completing (with original work) and give a universal notation to analyze and compare the discriminatory power and time complexity of computing the thirteen here formalized metrics. We also present a new way to represent tree graphs, reach deeper in the details of the Geodesic Distance and discuss its worst-case time complexity in a suggested implementation. Our contribution ends up as a clean, valuable resource for anyone looking for an introduction to comparative metrics for tree graphs.
Keywords: Geodesic Distance, Comparative Metrics, Graph Theory, Complexity, Phylogeny.

1. Introduction

Phylogeny is the study of the relations between biological entities. From it, the need to compare tree-like graphs has risen and several metrics were established and researched, but since there is no definitive way to compare them, its discussion is still open nowadays. All of them emphasize different features of the structures and, of course, the efficiency of these computations also varies.

The topic of this work is comparison metrics. Given two classifications, the challenge is to generate some parameter that expresses how similar these classifications are. A **classification** is some *data structure* that expresses relations between the data. For instance:

One of the big applications of comparison metrics is **phylogenetics**. **Phylogenetics** is the study of relations between biological entities: may it be genes, species or individuals. If we consider Darwin's theory of evolution, we can think of every species as having a direct ancestor (or multiple direct ancestors, but for sake of this example, lets assume there's only one) and multiple successors, and then, it's only natural to suggest that the whole map of species evolution can be described by a *tree-like graph*. We say that one specific tree hypothesis for the arrangement of species is a **classification**.

One of the first *data structures* considered for the first known problem of classification were the dendrograms. **Dendrograms** are tree graphs where the data is concentrated in the leaves, while the rest expresses the relation between it. This was a way to relate data by hierarchical clustering: close data is on the same cluster (cluster as a *bulk* of closely related data), and how close they are on the tree dictates how alike the information stored in the leaves is. The first effective numerical method to compare dendrograms was developed by Sokal and Rohlf in 1962, is known as the **cophenetic correlation** and it will be defined in the appropriate section. At the time, it was a method created to compare dendrograms generated from numerical taxonomic research.

However, since these methods were born out of a necessity to compare *tree-like structures*, a lot of other metrics were proposed, some of them formalized for quirky data structures and to emphasize different properties (based on topology, edge weight, and other parameters). This happened since there was no definitive method to compare trees, from a discriminative point of view and also from lack of efficiency: in computer science, graphs are a complicated structure to work with, and with the growth of the field of application and amount of available data over time, it's no surprise the need for an efficient and personalized way to deal with these prob-

lems rises.

In recent years this field of study has widened and our knowledge of this problem deepened since more people started working on it. Probably the most relevant **metric** (or **distance**) is *Robinson Foulds* since it can generate a parameter in linear time in the number of vertices of our trees (which is fairly good), but, as stated before, this does not mean that this metric suits all problems, hence the need to create others.

2. Background

2.1. Methodology and relevant aspects

As stated before, no metric should be considered as default for all problems. Depending on the problem at hand, choosing a metric over the other can be an advantage given the goal we want to achieve. However, this idea revolves around two important concepts. Since graphs are difficult to handle from a computation point of view, it is an advantage to know how fast are we able to compute the metric. On the other hand, since the output parameter describes how close two trees are the metric might benefit certain properties over others. These two can be referred to as **efficiency** and **discriminatory power**, respectively.

Efficiency can be seen from a **complexity** point of view, but complexity varies with the implementation of the algorithms. Although most of the times the description for the metrics might describe an algorithm, it might exist an equivalent algorithm implementation that is not the literal *translation* from that description but outputs the same values for the same inputs, although has a lower complexity.

The **Discriminatory Power** will depend on the metric. For instance, some metrics might benefit the tree topology over edge length (or weight) and some might have associated errors or output the same value for some types of trees. Being aware of these properties it is important to recognize the best metric to solve a problem at hand, however (and unlike efficiency) it is not a parameter that we can quantify, so we will go over the discriminatory power of each metric on the appropriate section.

The following work is structured according to the analyzed metrics. First, we will go over the most commonly used metrics and then the others. The latter have a less common use since they may be formalized for a different type of data structures (e.g.: *Hybridization number*) or aren't that practical to compute in present days (e.g.: *Subtree Prune and Regraft*). In each subsection our main focus (besides defining the metrics) will be their **efficiency** and **discriminatory power**.

2.2. Basic definitions

Next, we'll go over some definitions needed to define the metrics later on.

Definition 2.1. ***Graph theory basic definitions*** *Let $G = (V, E)$ where V is a set of vertices (or nodes) and E a set of edges. An edge is a pair (v_1, v_2) of vertices from V (for a lighter notation, one may write v_1v_2 instead of (v_1, v_2)). G is a **graph**.*

*A **path** is a subset $P \subseteq E$ of size k that can be ordered in a way that for the i-th element of P $(v_{i,1}, v_{i,2})$: $v_{i,1} = v_{i-1,2}$, $v_{i,2} = v_{i+1,1}$. We say that a path is a **cycle** if, on top of being a path, $v_{0,1} = v_{k,2}$. We name a connected graph without cycles a **tree graph** (or just **tree**). The length of a path P with no cycles is $k = |P|$.*

*In **undirected graphs** the edges (a, b) and (b, a) are equal. We will work with undirected graphs unless is differently stated*

*Let $u, v \in V$. We say that v is **neighbor** of u if $(u, v) \in E$ and we write $u \sim v$. In an undirected graph this relation is reflexive. The **degree** of a vertex is the number of neighbors it has.*

For the next set of concepts, it is important we assume that graph vertices can have **labels**, this means that exists a function $label : V \longrightarrow \mathrm{STR} \cup \{'\mathrm{NULL}'\}$ that for each vertex returns a *string* (that we call *label*) or NULL. The same way we define a concept of edge **weight** (or **length**), as a function $weight : E \longrightarrow \mathbb{R}$. That's actually needed to be considered for some problems and could represent, building over the *phylogeny* application, for example, how many years are between species. In both cases, they are just ways to hold information in these data structures, if needed.

Definition 2.2. ***Tree specific concepts*** *Let $T = (V, E)$ be a tree graph.*

- *A **leaf** is a vertex with degree 1.*
- *If exists one (and only one) vertex $v \in V$ such that $label(v) =' \mathrm{root}'$ then we say that v is the **root vertex** and that T is a **rooted tree**. If a root vertex does not exist, then T is an **unrooted tree**. Assuming T is rooted we can now define a new set of concepts:*
  - *The **depth of a vertex** v is the number of edges on the path (that on a tree is singular) from the root vertex to v. The **depth of a tree** is the maximum depth between all nodes. (We will refer to the depth of a vertex v as $depth(v)$, and the depth of a tree T as $depth(T)$)*
  - *Let $u, v \in V$. We say that u is an **direct ancestor** of v if $(u, v) \in E$ and $depth(u) < depth(v)$. In this case, v is also a **direct descendant** of u. Also, we generally say that u is an **ancestor** of v if there is a path of direct ancestors from v*

*to u (similarly we define the same general concept for **descendant**).*

  - *A **clade** consists of a vertex and all its lineal descendants.*

- *A **dendrogram** is a tree where only leaves (and, in case of a rooted tree, the root) have labels.*
- *A **binary tree** is a tree in which every vertex has degree at most* 3.
- *A **forest** is a collection* $F = \{T_1, T_2, ..., T_k\}$ *where for every* i T_i *is a tree.*

Definition 2.3. *Let* $d : X \times X \longrightarrow \mathbb{R}_0^+$ *be an injective function. We say that* d *is a **metric** over* X *(and* d *is called the **distance function**) if: **(1)*** $d(x, y) = 0$ *if and only if* $x = y$; ***(2)*** d *is symmetrical, that is* $d(x, y) = d(y, x)$; ***(3)*** d *satisfies the triangular inequality, that is for all* $x, y, z \in X$ $d(x, z) \leq d(x, y) + d(y, z)$.

In regard to efficiency, we now define a notation that will be useful to talk about program complexity.

Definition 2.4. *Big-O Notation*
If f *and* g *are two functions from* $\mathbb{N}$ *to* $\mathbb{N}$*, then we: **(1)** say that* $f = O(g)$ *if there exists a constant* c *such that* $f(n) \leq c \cdot g(n)$ *for every sufficiently large* n*, **(2)** say that* $f = \Omega(g)$ *if* $g = O(f)$*, **(3)** say that* $f = \Theta(g)$ *if* $f = O(g)$ *and* $g = O(f)$*, **(4)** say that* $f = o(g)$ *if for every* $\epsilon > 0$*,* $f(n) \leq \epsilon \cdot g(n)$ *for every sufficiently large* n*, and **(5)** say that* $f = \omega(g)$ *if* $g = o(f)$.
To emphasize the input parameter, we often write $f(n) = O(g(n))$ *instead of* $f = O(g)$*, and use similar notation for* o, Ω, ω, Θ.

When one refers *complexity* it is common to use *Big-O Notation*, but there are other notations. This will be important to understand how the computation time varies with the size of the input. So if we say that an implementation runs in $O(n)$ time it means time grows linearly with input growth. As expected, *how fast* a program runs will depend on its complexity. Given two functions f and g, $f = O(p_f(n))$ and $g = O(p_g(n))$ the function with higher complexity is determined by which of $p_f(n)$ and $p_g(n)$ as a higher growth rate. For example, if $p_f(n) = e^n$ and $p_g(n) = n^5 + n^3 + 10$, then f has a higher complexity.

2.3. Robinson Foulds, Robinson Foulds Length

Definition 2.5. *Given a tree* $T = (V, E)$ *we define* $S = \{x \in \text{STR} : \exists v \in V \text{ s.t. } label(v) = x\}$ *(that is* $S = label(V)$*) as the **set of labels** of* T*. A **labeled tree** consists of a 4-tuple* $T_l = (V, E, label, S)$ *where* (V, E) *is a tree, label a labeling function* $label : V \longrightarrow S$ *and* S *the corresponding set of labels.*

Definition 2.6. *The set of all labeled trees with* S *as the set of labels is defined as:*

$$\gamma_S = \{(V, E, label, S) : label : S \longrightarrow V\} \quad (1)$$

For sake of simplicity, we will omit the label function from the elements of γ_S.

One should realize that, given this definition, **dendrograms** are in fact trees $T \in \gamma_S$ such that the function *label* is injective and $label(V_T) = \{v \in V_T : degree(v) = 1\}$.

Definition 2.7. *We say that two trees are **identical** if there is a bijective map between them that preserves labeling, meaning that, for two identical labeled trees* $A, B \in \gamma_S$ *exists* $h : V_A \longrightarrow V_B$ *bijective, such that* $x, y \in V_A$ *and* $xy \in E_A$ *if and only if* $h(x)h(y) \in E_B$ *and* $label(x) = label(h(x)) \wedge label(y) = label(h(y))$.

Remark 1. *In all extension of our work we assume that* S *is the set of labels of the leaves (and usually consists in all natural numbers until some* $k \in \mathbb{N}$*), meaning that leaves of trees in* γ_S *must be exactly* $|S|$ *and its labels will be non repeated labels from* S*. The label 'root' is not in* S.

Definition 2.8. *A **weighted labeled tree** consists of a 4-tuple* $T_{wl} = (V, E, S, w)$ *where* (V, E, S) *is a labeled tree and* w *the corresponding weight function* $w : E \longrightarrow \mathbb{R}_0^+$*. The set of all weighted labeled trees with* w *as weight function and* S *as the set of labels is denoted by* γ_S^w*. We say that two trees are **weight-identical** if there is a bijective map between them that preserves labeling and weight of the edges, meaning that, for two weight-identical trees* $A, B \in \gamma_S^w$ *exists* $h : V_A \longrightarrow V_B$ *bijective, such that* $x, y \in V_A$ *and* $xy \in E_A$ *if and only if* $h(x)h(y) \in E_B$*,* $label(x) = label(h(x)) \wedge label(y) = label(h(y))$ *and* $w(xy) = w(h(xy))$.

This means that, looking back on our Robinson Foulds distance, the $|C_A \cap C_B|$ term of the expression can be rewritten as $|(C(A, B))'|$. The algorithm defined in Day's paper is, in fact, an algorithm to calculate the strict consensus tree between $T_1, T_2, ..., T_k \in \gamma_S$ with $|S| = n$ and the conclusion is that this algorithm is capable of doing it in $O(kn)$ time. Implementation, complexity reasoning and respective empirical verification are available in the article [2].

Definition 2.9. *A **weighted labeled tree** consists of a 4-tuple* $T_{wl} = (V, E, S, w)$ *where* (V, E, S) *is a labeled tree and* w *the corresponding weight function* $w : E \longrightarrow \mathbb{R}_0^+$*. The set of all weighted labeled*

*trees with w as weight function and S as the set of labels is denoted by γ_S^w. We say that two trees are **weight-identical** if there is a bijective map between them that preserves labeling and weight of the edges, meaning that, for two weight-identical trees $A, B \in \gamma_S^w$ exists $h : V_A \longrightarrow V_B$ bijective, such that $x, y \in V_A$ and $xy \in E_A$ if and only if $h(x)h(y) \in E_B$, $label(x) = label(h(x)) \wedge label(y) = label(h(y))$ and $w(xy) = w(h(xy))$.*

Definition 2.10. ***Partitioning Function***
*Let $A \in \gamma_S^w$ such that $A = (V, E, S, w)$ and Z_S the set of all proper partitions of S into two subsets. Let $f : E \longrightarrow Z_S$ such that, for every edge $e \in E$, $f(e)$ returns the set of Z_S that corresponds to the partition of S given by (according to Day's notation, Notation **??**) the clusters of both connected components of $(V, E\backslash\{e\}, S, w)$. We say that f is the **partitioning function** of A.*

Definition 2.11. *Let $A, B \in \gamma_S^w$ with edge sets E_A and E_B and partitioning functions f_A and f_B. The edges $e_A \in A$ and $e_B \in B$ are **matched** if and only if*

$$f_A(e_A) = f_B(e_B) \tag{2}$$

This last definition can help us build concepts for matching functions from E_1 to E_2, and actually, one of those is needed for the definition of the *Robinson Foulds Length* distance. Consider that $h_{(A,B)} : E_A \longrightarrow E_B$ is a function that, given $e_A \in E_A$, if exists $e_B \in B$ such that $f_A(e_A) = f_B(e_B)$ then $h_{(A,B)}(e_A)$ is defined and equals e_B, undefined otherwise.

Definition 2.12. ***Robinson Foulds Length distance***
*Let $A, B \in \gamma_S^w$, E_A, E_B, $E_{C(A,B)}$ the respective sets of edges, f_A and f_B the respective partitioning functions and $h_{(A,B)}$ the matching function from A to B. The **Robinson Foulds Length distance** is defined as*

$$d_{RFL}(A, B) = \left(\sum_{e \in (E_A \backslash E'_A)} w(e) \right) + \tag{3}$$

$$+ \left(\sum_{e \in (E_B \backslash E'_B)} w(e) \right) + \left(\sum_{e \in E'_A} |w(e) - w(h_{(A,B)}(e))| \right) \tag{4}$$

where:

$$E'_A = \{e_A : e_A \in E_A, \exists e_B \in E_B \; s.t. \; f_A(e_A) = f_B(e_B)\}$$

$$E'_B = \{e_B : e_B \in E_B, \exists e_A \in E_A \; s.t. \; f_B(e_B) = f_A(e_A)\}$$

One interesting thing to note is the limitations of d_{RFL}, since can give multiple results for the same input and it isn't symmetric.

It should be trivial to understand that:

- The edge sets E'_A and E'_B equal E_A and E_B respectively;
- There are two possible functions $h_{(A,B)} : E_A \longrightarrow E_B$: one that maps a_1 into b_1 and other that maps a_1 into b_2;
- $h_{(B,A)} : E_B \longrightarrow E_A$ maps b_1 and b_2 in a_1.

Given this, one should also realize that we have two different values for $d_{RFL}(A, B)$ depending on the chosen $h_{(A,B)}$ function. These are results of several problems in [5]: Theorem 4 proves the existence of h matching function between identical trees, however, there's no unicity assured; The authors define (differently to Day's Notation) T'_1 and T'_2 as the trees generated by collapsing edges in $E_1\backslash E'_1$ and $E_2\backslash E'_2$ respectively as identical, but in the example we just exposed that does not hold; Definition 5 assumes a unique h between T'_1 and T'_2 but that might not be the case, as we just exemplified.

But the problems don't end here. We ask the reader to check if the symmetry and identity of indiscernibles holds in d_{RFL} for all $A, B \in \gamma_S^w$, which the second can be easily refuted by considering the weight of every edge in A and B of our example as 1.

Definition 2.13. *Let $A \in \gamma_S^w$ and B a subtree of A. $S\,|_B$ is the subset of S in which its elements are labels of some vertex in B. Also, let $v \in V_A$. We define $A(v)$ as the subtree of A that consists of v and all its lineal descendants.*

Assume as well that for every edge $uv \in E_X$ for $X \in \gamma_S^w$, v is deeper than u.

Theorem 2.1. *Let $A, B \in \gamma_S^w$, f_A, f_B the respective partitioning functions and $C(A, B)$ their strict consensus tree. Assuming (†) there's a one-to-one correspondence between edges of A and B and their respective clades and (††) for all $a \in E_A$ there's no $a' \in E_A$ such that $S\,|_{A(a)} = S\backslash(S\,|_{A(a')})$, there's bijective matching functions $h_{(C(A,B);A')}$ and $h_{(C(A,B);B')}$.*

2.4. Geodesic distance

We are first left with the question of how many minimal (with no edges $ab, bc \in E$ such that $degree(b) = 2$ and b is not the root) non-identical (Definition 2.7) binary trees (Definition 2.2) exist. This will be a key factor for the space we will later formalize.

Theorem 2.2. *The number of minimal non-identical binary trees with n leaves is $(2n-3)!!$ (where !! stands for the **double factorial**).*

Most literature points to [6] for a proof, but this can be instead done in the following way: given trees with n leaves ($S = \{1, 2, ..., n\}$) and root r, if

we identify all the possible $n-2$ internal vertices (except the root) as $a_1, a_2, ..., a_{n-2}$, the Prüfer code (you can find a description in Reference [3]) gives us a bijection between those trees and sequences of size $2n-3$ with one r and two of each a_i (for $1 \leq i \leq n-2$). Therefore, we can conclude that the number of sequences is

$$\frac{(2n-3)!}{2^{n-2}} \tag{5}$$

Finally, we need to remove from this counting the trees with permutations of labels on the internal vertices $a_1, a_2, ..., a_{n-2}$, which are $(n-2)!$. Then we obtain the desired result:

$$\frac{(2n-3)!}{2^{n-2}(n-2)!} = (2n-3)!!. \tag{6}$$

That leaves us with the task of primarily formalizing the space where the distance will be built. Take into consideration that an **internal edge** is any edge that is not connected to a leaf of a tree. Do not forget that when we refer minimal trees we are specifically referring to our context previously explained (that these have no edges $ab, bc \in E$ such that $degree(b) = 2$ and b is not the root).

Definition 2.14. ***Space of trees with*** n ***labels,*** $\mathcal{T}_n$
*Consider S a set of labels and $|S| = n$. For every minimal non-identical (Definition 2.7) binary tree (Definition 2.2) $B_i \in \gamma_S$ (there is a total of $(2n-3)!!$ minimal non-identical binary trees [6]) generate an $(n-2)$-dimensional space $\mathcal{T}_B^o$ (that we designate as **orthant**) such that every component c_e is identified to one (and only one) internal edge $e \in E_B$ and takes real values between $[0, \infty[$. For all pairs of spaces $\mathcal{T}_{B_1}^o$ and $\mathcal{T}_{B_2}^o$ (with $e_1 \in E_{B_1}$ and $e_2 \in E_{B_2}$) identify components c_{e_1} and c_{e_2} if and only if the cluster representation of the clades associated with the removal of e_1 and e_2 from their respective trees match, that is, if $(C_{e_1})' = (C_{e_2})'$ (or, according to Definition 2.11, e_1 and e_2 are matched edges). The **Space of trees with** n **labels** is the result all the orthants $\mathcal{T}_B^o$ with this identification. A point $(t_1, t_2, ..., t_k) \in \mathcal{T}_n$ specifies a unique tree $A \in \gamma_S^w$ with internal edges e_i such that $w(e_i) = t_i$ for all $t_i \neq 0$.*

Definition 2.15. ***Geodesic distance***
*Let $A, B \in \gamma_S^w$, $\mathcal{T}_{|S|}$ the space of trees with $|S|$ labels and $d_{\mathcal{T}_{|S|}}$ the associated distance function. Then, the **Geodesic Distance** $d_{Geo}(A, B) = d_{\mathcal{T}_{|S|}}(A, B)$.*

Definition 2.16. *Let $T = (V, E, S, w) \in \gamma_S^w$ be a weighted rooted tree where every leaf and root are labeled and have no duplicate labels (there are no two distinct vertices $v_1, v_2 \in V$ such that $label(v_1) = label(v_2)$). Let $V' \subseteq V$ be the set of labeled vertices and $X = \bigcup_{v \in V'} \{label(v)\}$ (note that it is always the case that $S \subseteq X$). Given a set of labels L, a L-split is a two set structure $A|B$ such that $\{A, B\}$ is a partition of L of two non-empty sets and, in our work, if we omit the set and refer only split we are referring to X-splits of the tree at hand. It is straightforward to understand that each edge $e \in E$ induces a partition of the set X, building the sets A and B from the labeled vertices contained in each of the connected components generated from removing e from T. We will denote the X-split induced by $e \in E$ as $\sigma_e = \sigma_e^X = X_e|\bar{X}_e$, Σ will denote an arbitrary collection of X-splits and $\Sigma(T) = \bigcup_{e \in E} \{X_e|\bar{X}_e\}$. These are similarly established to any subtree $T' \subseteq T$ with label set $X' \subseteq X$ of vertices of degree at most 2.*

Definition 2.17. ***(Compatible*** X***-Splits and Compatible*** X***-Split Sets)*** *Let Σ be any non-empty collection of X-splits and $A|B, C|D \in \Sigma$. We say that X-splits $A|B$ and $C|D$ are **compatible** if at least one of the sets $A \cap C$, $A \cap D$, $B \cap C$, $B \cap D$ is empty. Additionally, we say that Σ is a **compatible** X**-split set** if every pair of X-splits is compatible.*

Definition 2.18. ***(Compatible Edge Sets)***
Let $T_1, T_2 \in \gamma_S^w$ and $A \subseteq E_{T_1}$ and $B \subseteq E_{T_2}$.

- *A set C is a **compatible edge set** if, for all $c_1, c_2 \in C$, the X-splits $X_{c_1}|\bar{X_{c_1}}$ and $X_{c_2}|\bar{X_{c_2}}$ are compatible;*
- *We say that A and B are **compatible edge sets** if for every $a \in A$ and $b \in B$ the X-splits $X_a|\bar{X_a}$ and $X_b|\bar{X_b}$ are **compatible**.*

Theorem 2.3. *(**Split Equivalence Theorem**, [7, Theorem 3.1.4])*
*Let Σ be a set of X-splits. Σ uniquely defines a tree if and only if Σ is a **compatible** X**-split set**.*

Definition 2.19. ***(Path Space and Path Space Geodesic)***
Let $T_A, T_B \in \gamma_S^w : T_A, T_B$ non-identical and $\mathcal{A} = (A_1, ..., A_k)$ and $\mathcal{B} = (B_1, ..., B_k)$ partitions of E_{T_A} and E_{T_B} such that the pair $(\mathcal{A}, \mathcal{B})$ satisfies:

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

*Then, for all $1 \leq i \leq k$, $B_1 \cup ... \cup B_i \cup A_{i+1} \cup ... \cup A_k$ is a compatible set, hence, from the **splits equivalence theorem**, uniquely defines a binary $|S|$-tree T_i and an associated orthant $\mathcal{T}_{T_i}^o$. The connected space $\mathcal{P} = \bigcup_{i=1}^k \mathcal{T}_{T_i}^o$ is the **path space** with support $(\mathcal{A}, \mathcal{B})$ and the shortest path from T_A to T_B contained in $\mathcal{P}$ the **path space geodesic** for $\mathcal{P}$.*

Theorem 2.4. *(From Billera et al. [1, Proposition 4.1])*

For trees $T_1, T_2 \in \gamma_S^w$ with disjoint edge sets, the geodesic between T_1 and T_2 is a path space geodesic for some path space between T_1 and T_2.

Definition 2.20. ***(Proper Path Space and Proper Path)***
*Let $T_1, T_2 \in \gamma_S^w$ and Γ the geodesic in $\mathcal{T}_{|S|}$ between T_1 and T_2. Then, Γ can be represented as a path space geodesic with support $\mathcal{A} = (A_1, ..., A_k)$ of E_{T_1} and $\mathcal{B} = (B_1, ..., B_k)$ of E_{T_2} which satisfy **P1** and the following additional property:*

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

*We call a path space satisfying conditions **P1** and **P2** a **proper path space** and the associated path space geodesic a **proper path**.*

Theorem 2.5. *(From Owen et al. [4, Theorem 2.5])*
*Given $T_1, T_2 \in \gamma_S^w$ a **proper path** Γ between T_1 and T_2 with support $(\mathcal{A}, \mathcal{B})$ is a geodesic if and only if $(\mathcal{A}, \mathcal{B})$ satisfy the property:*

> ***(P3)*** *For each support pair (A_i, B_i) there is no non-trivial partitions $C_1 \cup C_2$ for A_i and $D_1 \cup D_2$ for B_i such that C_2 is compatible with D_1 and $\frac{||C_1||}{||D_1||} < \frac{||C_2||}{||D_2||}$.*

One intuitive path between any two trees that will be useful for the algorithm implementation as the starting point is the ***cone path***. This is the path that connects the two trees through two straight line segments through the origin of our space $\mathcal{T}_n$. The *cone path* will function as our starting point with support $(\mathcal{A}^0, \mathcal{B}^0)$ that vacuously satisfies (P1) and (P2). The algorithm goes as follows:

Algorithm 1. ***(Geodesic Algorithm, GTP)***

> ***Input****: $T_1, T_2 \in \gamma_S^w$;*
>
> ***Output****: The path space geodesic between T_1 and T_2.*
>
> ***Initialize****: Γ^0 = cone path between T_1 and T_2 and support $(\mathcal{A}^0, \mathcal{B}^0) = ((E_{T_1}), (E_{T_2}))$.*
>
> ***Step****: At stage l, we have proper path Γ^l with support $(\mathcal{A}^l, \mathcal{B}^l)$ satisfying conditions (P1) and (P2).*
>
> > ***if*** *$(\mathcal{A}^l, \mathcal{B}^l)$ satisfies (P3),*
> >
> > ***then*** *path Γ^l is the path space geodesic,*
> >
> > ***else*** *chose any minimum weight cover $C_1 \cup D_2$, $C_1 \subset A_i$ and $D_2 \subset B_i$ with complements C_2 and D_1, respectively, having weight $\frac{||C_1||}{||A_i||} + \frac{||D_2||}{||B_i||} < 1$. Replace A_i and B_i in $\mathcal{A}^l$ and $\mathcal{B}^l$ by the ordered pairs (C_1, C_2) and (D_1, D_2), respectively, to form a new support $(\mathcal{A}^{l+1}, \mathcal{B}^{l+1})$ with associated path Γ^{l+1}.*

Theorem 2.6. ***Geodesic Distance***
*Let $T_1, T_2 \in \gamma_S^w$ such that T_1 and T_2 have no common edges and $(\mathcal{A}^l, \mathcal{B}^l) = ((A_1, B_1), ..., (A_k, B_k))$ the resulting support of running the geodesic algorithm for T_1 and T_2. The **geodesic distance** between T_1 and T_2 is given by:*

$$d_{Geo}(T_1, T_2) = ||(||A_1||, ..., ||A_k||) + (||B_1||, ..., ||B_k||)|| \tag{7}$$

3. Implementation

3.1. General Overview

The *GTP* algorithm receives as an *input* two trees T_1 and T_2 of γ_S^w, returns the *path space geodesic* (Definition 2.19) (or its length) and consists in three parts plus one extra part: **initialization**, **guard test**, **step** (the latter two consisting of the program cycle) and **common edge handling**.

- **Initialization** consists of constructing the support $(\mathcal{A}^0, \mathcal{B}^0) = ((E_{T_1}), (E_{T_2}))$ and initializing all the variables we will need for execution.
- The **guard test** is where we verify if the support at hand $(\mathcal{A}^l, \mathcal{B}^l)$ satisfies (P3), which is equivalent to check if the **extension problem** has no solution for each pair (A_i, B_i) of this same support.
- The **step** consists of the construction of a new support $(\mathcal{A}^{l+1}, \mathcal{B}^{l+1})$ satisfying certain properties.
- **Common edge handling** replaces the last two steps for trees T_1, T_2 that share common edges. For this step, we start by identifying a common edge $e \in E_{T_1} \cap E_{T_2}$ and the split $\sigma_e = C|D$, and compute the geodesic distance between T_1^C and T_2^C and the geodesic distance between T_1^D and T_2^D to determine the geodesic distance between T_1 and T_2.

Definition 3.1. ***Incompatibility Graph***
*Let $T_1, T_2 \in \gamma_S^w$ and E_1, E_2 subsets of E_{T_1} and E_{T_2} respectively. We define the **incompatibility graph** $G(E_1, E_2) = (V, E)$ such that $V = E_1 \cup E_2$ and*

$$E = \{uv : u \in E_1, v \in E_2, \ \sigma_u \text{ and } \sigma_v \text{ are incompatible splits}\}. \tag{8}$$

Lemma 3.1. *(Lemma 3.2, article [4])*
A proper path Γ with support pair $(\mathcal{A}^l, \mathcal{B}^l)$ is a geodesic if and only if the extension problem has no solution for any support pair (A_i, B_i) of $(\mathcal{A}^l, \mathcal{B}^l)$.

Definition 3.2. ***Maximum weight independent set***
*Let $G = (V, E)$ be a graph. An **independent set** $I \subseteq V$ is a set of vertices in which for all $u, v \in V$,*

$uv, vu \notin E$. *A* ***maximal independent set*** *is an independent set with maximal cardinality against every other independent set of* G. *A* ***maximum weight independent set*** *is a maximal independent set with minimal weight against every other maximal independent set.*

Definition 3.3. ***Minimum weight vertex cover*** *Let* $G = (V, E)$ *be a graph. A vertex cover* $C \subseteq V$ *is a set of vertices in which for all* $uv \in E$, *it is the case that* $u \in C$ *or* $v \in C$. *A* ***minimal vertex cover*** *is a vertex cover that has minimal cardinality against every other vertex cover of* G. *A* ***minimum weight vertex cover*** *is a minimal vertex cover with minimal weight against every other minimal vertex cover.*

Theorem 3.1. *Let* $G = (V, E)$ *with weight function* w *defined in* V. *The weight* w_I *of the maximum weight independent set and the weight* w_C *of a minimum weight vertex cover satisfy* $(\Sigma_{v \in V} w(v)) - w_I = w_C$.

Definition 3.4. ***Extension Problem*** *(final formulation)*
For all pairs (A_i, B_i) *from the support* $(\mathcal{A}^l, \mathcal{B}^l)$, *we want to check if the minimum weight vertex cover for* $G(A_i, B_i)$ *with vertex weights*

$$w_e = \begin{cases} \frac{e^2}{||A_i||^2} & \text{if } e \in A_i \\ \frac{e^2}{||B_i||^2} & \text{if } e \in B_i \end{cases} \tag{9}$$

has weight greater than 1. If that's the case for all the minimum weight covers, the extension problem is said to have no solution (hence satisfies (P3)) and we know that we can generate the ***path space geodesic*** *from the support* $(\mathcal{A}^l, \mathcal{B}^l)$.

Definition 3.5. ***Maximum flow problem***
Let $N = (V, E)$ *be a directed graph with* $s, t \in V$, *which we will designate by* ***source*** *and* ***sink*** *respectively,* $c : E \longrightarrow \mathbb{R}^+$ *a capacity function defined on the edges* N. *We designate* (N, c) *as a* ***flow network***. *A* ***flow function*** *is a mapping* $f : V \times V \longrightarrow \mathbb{R}^+$ *such that for every* $xy \in E$ *we have that* $f(xy) \leq c(xy)$, $f(yx) = -f(xy)$ *such that* $y, x \notin \{s, t\}$ *(and* yx *not necessarily in* E*) and also* $\sum_{xy \in E} f(xy) = 0$ *for any* $x \in V \backslash \{s, t\}$. *The* ***flow*** *of* N *is defined as* $|f| = \Sigma_{sx \in E} f(sx)$ *and the* ***maximum flow problem*** *consists in determining the maximum value of* $|f|$.

Definition 3.6. ***Minimum cut problem***
Let $N = (V, E)$ *be a directed graph with a weight function* $w : E \longrightarrow \mathbb{R}^+$. *We define a* ***cut*** *as a partition* $\{U_1, U_2\}$ *of* V *such that there is at least one* $xy \in E$ *such that* $x \in U_1$ *and* $y \in U_2$ *and* $s \in U_1$ *and* $t \in U_2$. *Let* $C \subseteq E$ *the subset of edges between* U_1 *and* U_2. *The* ***minimum cut problem*** *consists in finding a cut such that the weight of* C, $w(C) = \Sigma_{e \in C} w(e)$ *is minimal amongst the weight of all possible cuts.*

Theorem 3.2. ***Max-flow min-cut theorem***
In a flow network (N, c), *the maximum flow* $|f|$ *equals* $w(C)$ *where* C *is the minimum cut of* N.

Definition 3.7. ***Flow equivalent graph***
Let $G(A, B) = (A \cup B, E)$ *be the incompatibility graph of* A *and* B, *and* $w_e : A \cup B \longrightarrow \mathbb{R}^+$ *a weight function defined in the vertices of* $G(A, B)$. *We define the* ***flow equivalent graph*** $G_f(A, B) = (V_f, E_f)$, *a directed graph for our problem, as:*

$$V_f = A \cup B \cup \{s, v\}$$

$$E' = \{ab : a \in A, b \in B, ab \in E \text{ or } ba \in E\}$$

$$E_f = E' \cup \{sv\}_{v \in A} \cup \{vt\}_{v \in B}$$

and the weight function $w_f : E_f \longrightarrow \mathbb{R}^+$ *defined as:*

$$w_f(xy) = \begin{cases} w_e(y) & \text{if } x = s \\ w_e(x) & \text{if } y = t \\ \infty & \text{otherwise} \end{cases} \tag{10}$$

Theorem 3.3. *Let* $G(A, B) = (A \cup B, E)$ *be the incompatibility graph of* A *and* B, *and* $G_f = (V_f, E_f)$ *the flow equivalent graph for* $G(A, B)$. *A minimum cut in* G_f *induces a minimum weight vertex cover in* $G(A, B)$ *and vice versa.*

We are now into condition to delineate a path from the *extension problem* to its solution. In Figure 1 it is presented a schema for clearer understanding.

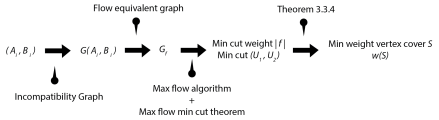

Figure 1: A diagram for the path from the support set until its respective minimum weight vertex cover and its weight

Putting it all together, the bigger slice of complexity is indeed in the Edmonds Karp algorithm, which is capsuled inside the main program cycle, which the number of runs is, as mentioned before, bounded by the minimum number of internal edges for one of the input trees. Each internal edge in the input trees corresponds to a vertex in the incompatibility graph, **therefore, we can conclude that the worst case time complexity of the *GTP Algorithm* is** $O(|E|^2|V|^2)$. More specifically, and putting it in terms of the input T_1 and T_2 instead of vertices and edges of the incompatibility graphs,

we have that $|V|$ equals the number internal edges of T_1 and T_2 summed and $|E|$ equals the number of incompatible splits to which the internal edges of T_1 and T_2 are associated with. Then, let t_1 and t_2 be the number of internal edges of T_1 and T_2 respectively and k the number of incompatible splits between T_1 and T_2. The complexity of the *GTP Algorithm* is $O(k^2 min(t_1, t_2)(t_1 + t_2))$.

4. Results and Conclusions

As you will witness, our work was divided in two core parts: an expositive text regarding thirteen of the most popular or promising methods to compare tree-like structures; specification of details for the *Geodesic Distance*, implementation of the *GTP Algorithm* and an worst-case time complexity analysis. We will discuss conclusions for these two topics separatly.

4.1. Distance synopsis

After all the formalization and property lifting of all the metrics we can say that we managed to compile information about all these with the same notation, providing a valuable resource for someone who looks for an introduction for comparative metrics. Some original work was developed (as stated in Introduction) and that helped us gluing some gaps in our understanding of the distances we've approached.

In our full work we present two tables compiling the information regarding the complexity and discriminative power of all thirteen the metrics. Since d_H and d_{SPR} don't have yet feasible algorithms to be calculated, it's presented the conjectured problem hardness of their computation. The worst case time complexity of the geodesic distance is also revised by our work.

4.2. SplitToTree and GTP Algorithm Implementation

In regards to our achievements in Implementation, we can say that the representation of trees in an edge weight vector made it fairly straightforward to handle trees in the *GTP Algorithm*, and representing trees in this way is extremely practical once we have access to a sketch. Those vectors are generated by, first, checking the splits (induced partitions on the set of labels S for each e internal edge, checking which labels belong to each connected component on the graph upon removal of e), check their position according to a lexicographical order, and placing the weight of each e in a vector of zeroes of size $(2n-3)!!$ ($|S| = n+1$).

In regards to our implementation, it matches the worst case time complexity reported by Owen et al in [4]. This comes from the fact that the maximum flow problem is solved with an *Edmonds Karp* algorithm, which has $O(|V||E|^2)$ complexity. The data structure we chose for input led us to some compromise in terms of memory, and for big label sets, it might be a good idea to rethink the way compatible splits are checked. Also, its worth to note that the fact we chose *Wolfram Mathematica* as a coding language was also a drawback when it came to analyze complexity, since it is proprietary code and function specification does not cover complexity, neither we have way to check its implementation.

Acknowledgements

I am very grateful to professor Alexandre Francisco and professor Pedro Martins Rodrigues for all patience and availability to answer my questions and address my curiosity in regards to comparison metrics, such as the courses they taught during my academic journey in Técnico Lisboa.

References

[1] L. Billera, S. Holmes, and K. Vogtmann. Geometry of the space of phylogenetic trees. *Adv. in Appl. Math.*, 27:733–767, 2001.

[2] W. H. E. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2:7–28, 1985.

[3] J. Gottlieb, G. Raidl, B. Julstrom, and F. Rothlauf. Prüfer numbers: A poor representation of spanning trees for evolutionary search. *GECCO-2001*, pages 343–350, 2006.

[4] M. Owen and J. Provan. A fast algorithm for computing geodesic distances in tree space. *arXiv:0907.3942v2 [q-bio.PE]*, 2009.

[5] D. Robinson and L. Foulds. Comparison of weighted labeled trees. *Lect. Note. Math.*, 748:119–126, 1979.

[6] E. Schröder. Vier combinatorische probleme. *Zeit. für. Math. Phys.*, 15:361–376, 1870.

[7] C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.