Tree Morphology Classification and Analysis: Quantification of tree randomness.

Oscar Henry

15-01-2021

Abstract

The use of topological summaries, such as barcodes, over tree structures has developed multiple new possibilities in the range of tree structure classification and quantification. This report proposes a summary of the state of the art on morphology of topological trees in the context of summarizing biological neurons and generating artificial neurons. We present the mathematical fundamentals of this field, the Topological Morphology Descriptor (TMD) and the Topological Neuron Synthesis (TNS) algorithms. It allows us to review current approaches of barcodes classification and their main results as well as propose new methods of trees classification. This report is complemented by a GitHub repository.

Keywords: Tree, Persistence Barcode, Topological Data Analysis, Classification, Entropy, Algorithm.

Contents

1 Introduction

In the pas few decades, the medical advancement and the technology development made of neurological sciences one of the most active field in research because of its multidisciplinary approach and numerous impacts in other disciplines. Various fields of mathematics can be applied to biological topics, as a tool helping their comprehension or in order to develop models as replica of physical processes. Topology is frequently used in neuro-sciences because it concerns the properties of geometric objects such as brain network (graph), neuronal transmission (dynamical system), or single neuron structure (morphology). Understanding single neurons morphology is one of the most important question in neuro-sciences, but despite the research community efforts, it is still an open field on various aspects of the structure of neurons. The difficulty to find a universal descriptor for neurons morphology is inherited from their highly complex and variable structures. Our nervous system is composed by various types of neuronal cells possessing highly ramified *arborization structures* which are also called *branching structure* or *tree*. Trees structures are shared by neurons, but also plants, maps, or even in fields as architecture, fluid mechanics, etc. Tree tends to spread in multiple directions in order to accomplish or/and maximize useful processes like photosynthesis, neuro-transmission, neuro-reception, etc. This common aspect is determinant for understanding tree structure. The lack of fundamental means of standardizing their description is responsible for the diversity of methods and directions of research in this field. Even the tree morphology research follow a tree structure in a way !

Neuronal morphology has been traditionally focused on visually distinguish the shapes observed under a microscope and classify them. This motivated the idea of finding a tool using topological data analysis (TDA) which can extract a topological summary from a single neuron and be able to record the morphology of it. The *Topological morphology Descriptor (TMD)* is an algorithm encoding the branching pattern of any tree in a *persistence barcode* [1]. In order to obtain trees from persistence barcode, i.e., use the output of TMD to create new artificial trees, the *Topological Neuronal Synthesis (TNS)* algorithm was developed [2].

This text aims is to to summary and do an analysis of the state of the art in trees and barcodes encoding in order to propose guidelines about tree selection and description by translating the concept of persistent entropy to this scenario. In a first section, a review of the mathematical background that will be used in the different developments is done. We present the major items of morphology applied to trees. The next section is a summary of the TNS and TMD algorithm in order to understand how the transformation between barcodes and trees work. These steps will permit the creation of a quantified object, tree persistent entropy, allowing tree selections and measurement. This will be do in the final sections where mathematical results and their implementations through the creation of three algorithms will be presented. The aims of the algorithms is to propose an oriented selection of tree-realizations for the user in function of the tree persistent entropy of all the tree-realizations given by a single barcode.

The computational part of this project is referenced in a GitHub repository at the link: github.com/Oscar-Henry/Topological-Morphology-Algorithm-for-Tree, where a README file can be found explaining the use of the algorithms. This project is mainly inspired by three major papers in the field of tree morphology and their resulting consequences, respectively, "A topological representation of branching neuronal morphologies" [1], "Computational synthesis of cortical dendritic morphologies" [2] and "From trees to barcodes and back again: theoretical and statistical perspectives" [3].

2 Mathematical background

The first step of this report consists of presenting tree morphology bases, definitions, norms and results. Here we review the mathematical background necessary to understand the different sections and developments of the report.

2.1 Mathematical Contextualization: Trees and barcodes

This report is about *tree morphology*, which consist in analysing and processing geometric structure of trees. A *tree* is a specific type of graph, a particular geometric structure studied by the science of graph theory. In order to do manipulation, quantification and analysis over trees and create a topological summary gathering different key informations will be required. The topological summary used in this report is a *barcode*. Using *barcodes*, the description of *trees* is simplified, opening new methods of analysis.

Definition 2.1. A *graph* is a pair $G = (V, E)$ where V is a set of elements called *vertices* and E is a set of pairs of vertices called *edges*. A *path* is a subset of connected edges, two edges are connected if they share a vertex. A *directed graph* is a graph described by $G = (V, E, \phi)$ such as $\phi : E \to \{(x, y)|(x, y) \in V^2$ and $x \neq y\}$ defines a set of ordered pairs, i.e., there exists a unique direction when traveling edges. A *finite graph* is a graph $G = (V, E)$ where V and E are finite sets. An *acyclic graph* is a graph $G = (V, E)$ where a trail $(e_1, ..., e_n)$ with a vertex sequence $(v_1, ..., v_n, v_1)$ cannot exist, i.e., there is no loop in the graph. The *degree* of a vertex is a function $d : V \to N$ such as $d(v)$ equal to the number of edges connected to v with $v \in V$ [4].

Definition 2.2. A *topological tree* is an acyclic, finite, directed graph with a distinguished vertex of degree 1 called the *root*. For this report, we denote T the set of trees with $t \in T$ an element, r the root of the tree (origin), L the set of leaves (vertices of degree 1 without the root), N the set of nodes (vertices of degree 3), V the set of vertices of T, $V \equiv L \cup N$, E be the set of edges of a tree. A vertex v_i of V is a *parent* of a vertex v_j $(i \neq j)$ if there is a directed edge from v_i to v_j, the vertex v_j is then a *child* of v_i [4]. A *sub-tree* is a tree t where the root is replaced by a vertex v, it it composed by all the children vertices of v and the edges created by them.

Remark. A *geometric tree* is records the *angles* at each vertex of degree 3, the length of each branch, the diameter of each branch, whereas a topological only focus on the adjacency of the branches.

Definition 2.3. A *persistence barcode* is a finite multi-set of open intervals of the real line $\mathbb{R}$. An interval can also be called a *bar*. In this report, we denote by $B = \{(b_i, d_i)\}_{i=0,...,n}$ a single barcode and by bar_i the i^{th} bar of B, b_i and d_i are respectively the *birth* and *death* of bar_i. A barcode is called *strict* if the first bar (b_0, d_0) properly contains all the others, i.e., $b_0 < b_i$ and $d_i < d_0$ for all i, and no bars are born or die at the same time, i.e., $b_i \neq b_j$ and $d_i \neq d_j$ for all $i \neq j$. In this text, we denote by $\mathcal{B}$ the set of all barcodes, $\mathcal{B}^{st}$ will denote

the set of strict barcodes and $\mathcal{B}_n^{st}$ will denote the subset of strict barcodes with $n+1$ bars (the longest bar (b_0, d_0) is omitted in the count). Two strict barcodes B, B' are bar equivalent, denoted by $B \underset{bar}{\sim} B'$, if their deaths occur in the same order ($d_i > d_j \iff d'_i > d'_j$). The *equivalence class* of bar equivalent barcode can be encodded by the order $(i_1, ..., i_n)$, where $d_{i_k} > d_{i_{k+1}}$ for all $1 \leq k < n$ [3].

In this text, it will be assumed when manipulating barcodes, that the bars are ordered by birth value, i.e., $b_0 < b_1 < ... < b_n$.

2.2 Persistent Entropy

Persistence barcodes and topological trees statistical analysis is complex and lack of specific tests. This motivate the creation of a real number that encapsulates the information contained in the barcode. The *persistent entropy* number aim to quantify the rate of information production. The *persistent entropy of a barcode* is a stable topological statistic which is an adaptation of Shannon entropy to the persistent homology context.

Definition 2.4. The *persistent entropy* formula is derived by Shannon 3 principles of entropy [5], Shannon called this measure H and define it by :

$$H = -K \sum_{i=1}^{n} p_i \cdot \log p_i$$

Where K is a positive constant and p_i is the probability of the i^{th} event (here, equivalent to its frequency) [5].

Definition 2.5. For the *persistent entropy of barcode* we define $B = \{(b_i, d_i)\}_{i=1,...,n}$ a strict barcode, $l_i = d_i - b_i$ the lifetime of the i^th branch and $L(B) = \sum_{i=1}^{n} l_i$. Then, the *persistent entropy* of B is: [6]

$$PE(B) = \sum_{i=1}^{n} - \frac{l_i}{L(B)} log\left(\frac{l_i}{L(B)}\right)$$

.

From the previous definitions, the *persistent entropy* and the *persistent entropy for barcode* permit to describe statistical results in a particular object into a real number. Working with topological trees, it is interesting to ask our-self about the existence of an equivalent way to quantify persistent entropy in trees. If it is possible, what is understandable from the formula and what phenomenon does it describe?

This report have the purpose of trying to define a notion of persistent entropy adapted to trees by defining relevant features about it and proposing quantification formula for trees. This aim involves understanding the problematic behind tree complexity which will be the base of this reflection and will

give it sense. When having a single tree, many questions appear interesting to study in function of the final aim to quantify its entropy;

- Does the tree seem to spread hazardously of did the growth of the branches seem to be conducted by constrains?
- How different one tree is from another?

In this text, we aim to find in the particular case of artificial neuron generation, different features about topological trees issues from a single barcode:

- How these topological trees, can vary in their structure and particularly, how branches seem hazardously spread?
- Is it possible to find relevant quantified features that can optimize the generation of a spread tree?

3 TMD and TNS: from trees to barcodes and back again

Tree structures have the possibility to be summarized into a topological summary such as barcodes. This result motivate the research of an algorithm capable of returning the corresponding barcode of a given tree, this algorithm is called the *Topological Morphology Descriptor (TMD)*. Barcodes carry topological informations on trees, it is then possible to reconstruct a topological tree from the stocked information. This is the purpose of the *Topological Neuronal Synthesis (TNS)* algorithm. The two algorithms are complementary in the sense that the TMD output can be use as an input for TNS and the input of TMD and TNS output are geometrical trees.

3.1 The Topological Morphology Descriptor (TMD)

The *Topological Morphology Descriptor (TMD)* [1] algorithm is defined in different ways. We consider, for this text, the *radial distance from the root* version of the TMD. TMD can also be used with the path distance, the branch length, the branch order or any other features that allow to classify branches into a single barcode, they will not be itemized in this text. The radial distance form of TMD is equivalent to a filtration of concentric spheres of decreasing radii centered at the soma (root), i.e., the radial distance of spheres of decreasing radii, centered at the soma, will permit the classification and quantification of the tree and so the creation of the barcode, see Figure 2 for illustration.

In this section, we denote by $f : V \to \mathbb{R}$ the radial distance of v from the root. It is supposed that the root is centered so $f(r) = 0$ where r is the root. A component of T is a path from a leaf l to an internal node n. Let T_n denote the sub-tree with root at the node n and with L_n and B_n its set of respectively leaves and branch points. Let $v : N \to \mathbb{R}$, a function, defined by $v(n) = \max\{f(x)|x \in L_n\}$. The *Elder rule* stipulates that only the older, in the meaning of the highest radial distance, survives. The rule permit a classification of birth and death during the algorithm. A vertex l_i of T is *older* than a vertex l_j if there is a directed branch from both to the node n and $v(l_i) > v(l_j)$, we call the older the *parent* and the younger the *child*.

3.1.1 Algorithm concepts

The algorithm take as input a tree embedded in $\mathbb{R}^3$ and return a persistent barcode, where each bar represent a branch of the tree and are ordered by birth. The *birth* and *death* of a component with leaf $l \in L$ and node $n \in N$ occurs respectively at time ($\equiv$ radius) $f(l)$ and $f(b)$. It is possible that $f(b) \leq f(l)$. Knowing the birth and death of a component its $lifetime_i$ (length of a bar) is define by $lifetime_i = |f(n) - f(l)|$, and so $f(n) = d_i$ and $f(l) = b_i$. The final barcode will be composed by the different lifetimes of each component of T. [1][3].

3.1.2 Algorithm encoding

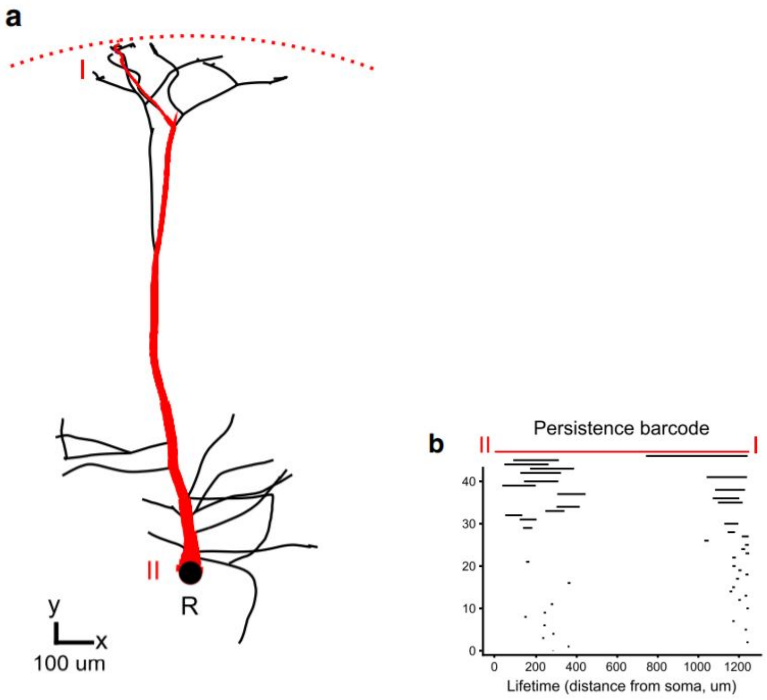
The algorithm is initialized by setting the value of $v(l) = f(l)$ for all $l \in L$. The leaves are added to a set of nodes, denoted A. Reducing the radius toward the root all but the oldest (with respect to $v : N \to \mathbb{R}$) leaves are killed, i.e., removed from A at each branch point (if they have the same value v it is equivalent to kill any one of them). For each killed component, one interval (lifetime) is added to the persistence barcode. The older branch point c_m is replaced by its parent in A and the value $v(parent)$ is set to $f(c_m)$ This operation is applied iteratively to all the nodes until the root r is reached. At this point A contains only the largest component. The last step consist only of adding the value of the interval of the largest branch $(v(r), f(r))$ to the barcode $TMD(T, f)$. Remember that $f(r) = 0$ and $v(r)$ at this point give the leaves with the highest radius from the root of the tree. [1][3].

Algorithm 1 TMD algorithm

Require: T with R, B, L, $f : T \to \mathbb{R}$
Ensure: TMD(T, f), a persistence barcode obtained from a tree T and the function f

```
TMD(T, f): empty list to contain pairs of real numbers
A ← L                                   ▷ A : set of active nodes
for every l ∈ L
    v(l) = f(l)
while R ∉ A
    for l in A
        p : parent of l
        C : children of p
        if ∀n ∈ C, n ∈ A
            c_m : randomly choose one of {c | v(c) = max_{c'}(v(c')) for c' ∈ C}
            Add p to A
            for c_i in C
                Remove c_i from A
                if c_i ≠ c_m
                    Add (v(c_i), f(p)) to TMD(T, f)
            v(p) ← v(c_m)
Add (v(R), f(R)) to TMD(T, f)
Return TMD(T, f)
```

Figure 1: TMD Algorithm implementation [3]



(a) (b)

Figure 2: Realization of the TMD algorithm over a single neuron [1]: (a) Application of topological analysis to a neuronal tree showing the largest component (red) and the tree structure (black). The red dots form a circle with the root as origin and the distance between r and the farthest point as radius. (b) Persistence barcode given after TMD algorithm execution. Each horizontal line represent a branch of the neuron, marking its birth and death. Units are in microns. The largest component is again in red.

3.2 The Topological Neuronal Synthesis (TNS)

The morphological development of neurons in the brain is a complex process that depends on both genetic and environmental components. The process that contributes to neuronal growth differs between species, brain regions, and morphological types. The *Topological Neuronal Synthesis (TNS)* is an algorithm which allow to construct synthetic neurons from persistence barcodes. Taking the information of the birth and death of each bar, the algorithm will grow a dendrite, extension of a nerve cell with a branching structure, from its root and add nodes, leaves, etc, to finish with an artificial tree.

The *trunk* of an artificial neuron describes the part, after the creation of the neurite, i.e., any projection of the cell body into a tree, where the branch cannot bifurcate or terminate and have a normal direction to the soma surface. A *Galton-Watson tree* is a discrete random tree generated by the following process: at each step, a number of off springs is independently sampled from a distribution. A *segment* is defined by a pair of vertices in the tree that determine a vector length $|e|$, where e is the edges, and with a direction $D_{segment}$. The $D_{segment}$ (also called *direction vector*) is defined by three weight coefficients; the *random direction* ρ, the *targeted orientation* τ and the *memory* μ as $\rho+\tau+\mu = 1$ [2].

3.2.1 Algorithm concepts

The TNS algorithm starts from the root and create a trunk that elongates a certain time then bifurcate into branches. The branches can either bifurcate themself, terminate or continue at each step of the algorithm. It uses a persistence barcode (output from the TMD algorithm) as a descriptor to define the bifurcation and termination probabilities for the synthesis of a neuron. The bifurcation of a branch lead to a new node, resulting by the creation of two new branches that continue the algorithm. The termination of a branch create a new leaf. When not bifurcating or terminating, the branch grows in function of the $D_{segment}$ in a 3D space. A topological tree consists only of bifurcations, terminations, and continuations, so the accepted values for the number of offspring are: zero (termination), one (continuation), or two (bifurcation) [2][3].

3.2.2 Algorithm encoding

The neuronal soma (root of the tree) is generated in first place as a sphere with a radius sampled from a biological distribution with the number of neurites. The basal dendrites are initiated randomly on the soma surface and the other dendrites are added successively. Each neurites is associated with a barcodes and takes a normal direction to the soma surface initial orientation. Subsequent steps of the growth take place in a loop. Each branch of the tree is elongated step by step in function of the direction vector $D_{segment}$. At each step, the growing tip is assigned probabilities to bifurcates or terminate that depend on the path distance from the soma and by the bars of the sampled barcode. Each growing tip is assigned a bar bar_i sampled from the barcode that include a

starting path distance b_i, an ending path distance d_i and a bifurcation angle a_i. At each step the growing tip first checks the probability to bifurcate, then the probability to terminate. The probability to bifurcate and terminate increases exponentially with the growing tip getting closer to b_i. The same method is applied for the termination using d_i. The probabilities are sampled from an exponential distribution $e^{-\lambda x}$, where λ is a chosen parameter. A high value of λ results in a very steep distribution and thus will reduce the variance of the synthesized cells. Vice versa, a small value of λ results in almost random cells. It is common to choose $\lambda \approx 1$ so that the bifurcation and termination points are stochastically chosen but are strongly correlated with the input persistence barcodes. Each bar from the barcode is removed after its use (termination of the growing tip). At a bifurcation, two new branches are generated and the direction of the daughter branches depends on the bifurcation angle a_i. The tortuosity of the path during the elongation of the neurite is the some of three unit vector terms: the cumulative memory of the directions of previous segments within a branch M, a target vector T, and a random vector R, weighted by ρ, τ and μ:

$$D_{segment} = \rho R + \tau T + \mu R$$

The growth terminates when all the bars of the input barcode have been used. Finally, as an independent step, the diameters of the tree are assigned based on a diameter distribution. The algorithm starts from the tips of the tree and assigns diameters to the termination points. The tree is then traversed from the tips towards the root and the diameters are increased according to the sampled taper rate as long as it is inferior to the maximum diameter, i.e., the trunk diameter. When all the children of a section have their diameters computed, the parent section is assigned a diameter according to the *Rall ratio* $D_{rr} = \frac{3}{2}$ [2][3].

$$d_{parent} = (d_1^{D_{rr}} + d_2^{D_{rr}} + ...)^{1/D_{rr}}$$

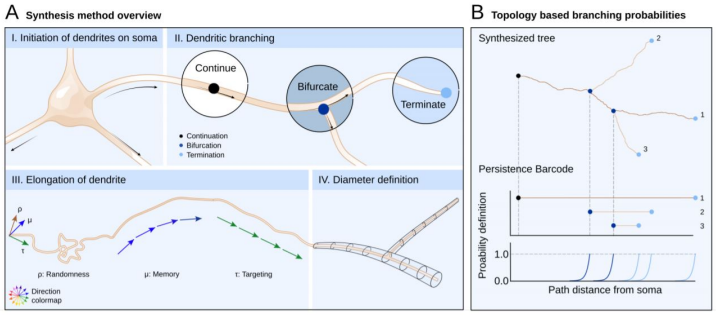


Figure 3: Method of Topological Neuron Synthesis. A. Overview of dendritic synthesis based on four stages of growth. I. Soma initiation II. Stochastic definition of bifurcation, termination, and continuation (B). III. Dendritic elongation IV. Diameter definition B. Branching based on the topological morphology descriptor (TMD) of a neuronal tree.: the probability to bifurcate, terminate, or continue derived from the TMD of a neuronal morphology. [2]

4 Tree-realization of barcodes

It is important to notice that a barcode encode the topological information of a tree and the birth and death information of it, i.e., the length of the branches. All barcodes in this section are assumed to be strict and ordered by birth

4.1 Combinatorial equivalence classes of barcodes and combinatorics of tree-realization

Manipulating barcodes and trees is not a trivial exercise, but the use of *adjacency matrix* in the different tasks permits to describe more easily the mathematical process.

Definition 4.1. A *combinatorial equivalence class* is a set of barcodes whose deaths occur in the same order, i.e., $d_i > d_j \Leftrightarrow d'_i > d'_j$. We denote the equivalence class containing a strict barcode $B = \{(b_i, d_i)\}_{i=0,...,n}$ by $(i_1, ..., i_n)$, where $d_{i_k} > d_{i_{k+1}}$ for all $1 \leq k < n$.

Definition 4.2. A topological tree T is a *tree-realization* of a barcode if TMD(T) $= B$, i.e., $T \in TMD^{-1}(B)$. For any strict barcode B, let $\tau(B)$ denote the set of *combinatorial equivalence classes* of tree-realizations of B, i.e., [3].

$$\tau(B) = TMD^{-1}(B)/\underset{\text{comb}}{\sim}$$

Remark. $TNS(B) \neq TMD^{-1}(B)$ because when using the TMD algorithm with a tree T to create a barcode $TMD(T) = B$, the use of TNS algorithm with B for input will create a artificial tree T' such as $T \neq T'$. This is due to the hazardous process involved in the TNS algorithm in the creation of the geometric tree. Adding to this that a barcode B is not necessary the result of a TMD algorithm application on a tree, it can be created directly as long as this respect the definition fixed earlier in the text.

Lemma 4.1. *If B and B', two strict barcodes with the same number of bars, then : [3]*

$$B \underset{bar}{\sim} B' \iff \tau(B) = \tau(B')$$

Definition 4.3. The *tree-realization number* of a strict barcorde B, written $TRN(B)$ or $\#\tau(B)$, is the number of combinatorial equivalence classes of tree-realizations of B. The *index*, $index_i(B)$, of a single bar $bar_i = (b_i, d_i)$ from a strict barcode $B = \{(b_i, b_i)\}_{i=0,..,n}$ is defined by the number of bar that include bar_i strictly, i.e., [3]

$$index_i(B) = \#\{j | b_j < b_i < d_i < d_j\} = \#\{j < i | d_i < d_j\}$$

.

Lemma 4.2. The tree-realization number of a strict barcode $B = \{(b_i, b_i)\}_{i=0,..,n}$ is equal to the product of the indices of its bars, i.e., [3]

$$TRN(B) = \prod_{1 \leq i \leq n} index_i(B)$$

Remark. The maximum tree-realization number for a strict barcode with $n+1$ bars is $n!$, it is obtained when all the bars are ordered by decreasing lifetimes $d_n < ... < d_1 < d_0$ (strictly ordered barcode).

Definition 4.4. A *transposition* is the action of switching the order of two consecutive bars in the barcodes. If the deaths are ordered differently, this affects the tree realization number. A *permutation* is the action of switching the deaths order in a barcode. For example, we denote by $B = \{(b_i, d_i)\}_{i=0,\dots,n}$ a strict barcode before the permutation and $B' = \{(b'_i, d'_i)\}_{i=0,\dots,n}$ a new barcode obtained by permuting the deaths d_{i_k} and $d_{i_{k+1}}$, i.e., $b_i = b'_i$ for all i and $d_i = d'_i$ for all $i \neq i_k, i_{k+1}$, while $d_{i_k} = d'_{i_{k+1}}$ and $d_{i_{k+1}} = d'_{i_k}$ [3].

Definition 4.5. Cayley Graph:

A *Cayley Graph* is a coloured directed graph, $\tau = \tau(G, S)$ where G is a group and S a generating set of G. Each element $g \in G$ is assigned a vertex and each generator $s \in S$ is assigned a colour c_s. For any $g \in G$ and $s \in S$, the vertices corresponding to the elements g and $s \cdot g$ are joined by a direct edge of colour c_s.

The Caley Graph B_3^{st} in Figure 4 shows all the death-permutations in a strict barcode with 4 bars. This graph allows to study the allowed transpositions and their effects on the barcode. The vertices of the graph represent equivalence class while the edges are colourised with blue or green to show respectively the permutation of bar_1 with bar_2 and bar_2 with bar_3 [3].

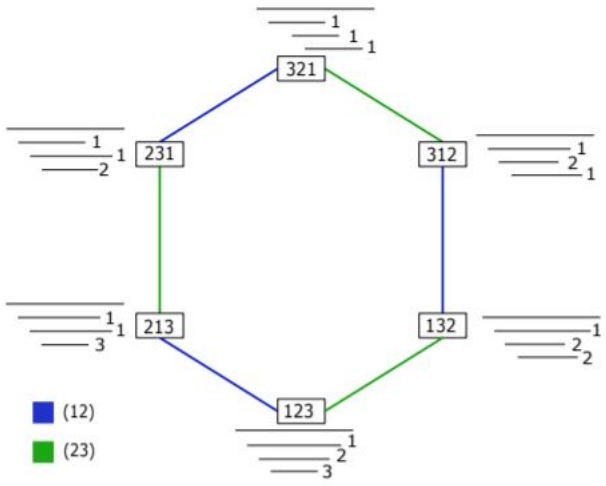


Figure 4: A representation of the space of barcodes with four bars as the Caley graph generated by the death permutation (12) and (23), respectively. The number to the right of each bar is its index [3].

4.2 From Trees and Barcodes to Adjacency Matrix

This section propose different methods for the encoding and description of the informations of a tree and its barcode using matrices.

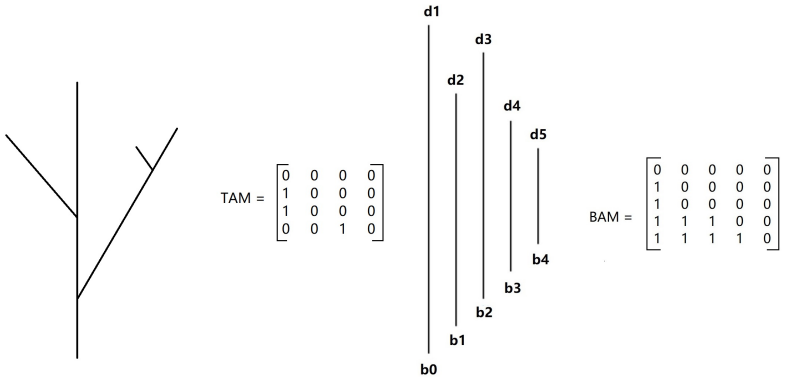
Definition 4.6. An *adjacency matrix* of graph is a square matrix $M_{n\times n}$, triangular inferior, permitting the topological description of graph. The element (i, j) of the matrix is non-zero when there exist an direct edge between the vertices v_i and v_j in the graph, i.e., the cell $u_{ij} = 1$ if v_i and v_j are connected and $i < j$, else $u_{ij} = 0$ [3].

In this report, we introduce two new notions which allow a more precise characterisation of adjacency matrix. One is for tree encoding and the other is for barcode encoding.

Definition 4.7. A *tree adjacency matrix (TAM)* is an adjacency matrix used to represent a tree. The element (i, j) of the matrix is non-zero if the i^{th} branch died by merging with the j^{th} branch.

Definition 4.8. A *barcode adjacency matrix (BAM)* is an adjacency matrix used to represent a barcode. The element (i, j) of the matrix is non-zero if the Elder Rule allows bar_i to be connected to bar_j.

Remark. The diagonal of both TAM and BAM is empty because a branch cannot attach itself. Each branch can be attached to only one other branch so each lines of a TAM contain an unique non-zero element. Nevertheless this is false for a BAM because a barcode can have multiple tree-realizations. To illustrate this situation, if a bar, of a barcode B is strictly included in three other bars with smaller birth values from the same barcode, the BAM of this barcode will have three non-zero elements in the i^th line.



(a) (b)

Figure 5: TAM and BAM examples with a given tree and a given barcode: (a) A TAM is composed by a single non-zero element for each line except the first line because the longest do not merge any other branch. (b) A BAM can have multiple non-zero elements if the barcode allow multiple tree-realizations.

In our cases, the death and birth informations are decisive for the characteristics of our tree unlike a classical topological tree. This brought us to create a particular form of adjacency matrix where the non-zero element carry this information. A method to encode additional informations in adjacency matrix by using the non-zero elements to carry quantitative features is presented in the paper: "A Structural Average of Labeled Merge Trees for Uncertainty Visualization" [7]. The information carried at the case (i, j) is the result of a function taking for argument the vertices v_i and v_j.

Definition 4.9. A *description matrix* is an adjacency matrix (TAM or BAM) in which the lifetime description are encoded. The non-zero element at position (i, j) of the adjacency matrix take for value the branch i^{th} death time and the diagonal (i, i) contain the birth time of the i^{th} branch. (For deeper information view [7]). For example, the cell $u_{ij} = d_i$ if v_i and v_j are connected, if else $i = j$ $u_{ii} = b_i$, else $u_{ij} = 0$.

Remark. The trunk has its birth at the position (0, 0) in the matrix and so, there is no $j \in \mathbb{N}_+$ as $j < i$ to stock the death information. It is then impossible to conserve the death and birth of the root. Nevertheless, in tree-realizations, it is often considered that the birth of the root is at time $t_0 = 0$ so death can be conserved if presupposing this assumption. If it is not the case, it will be necessary to create an variable conserving the tree lifetime information, or death time to asses ether the first one or the other.

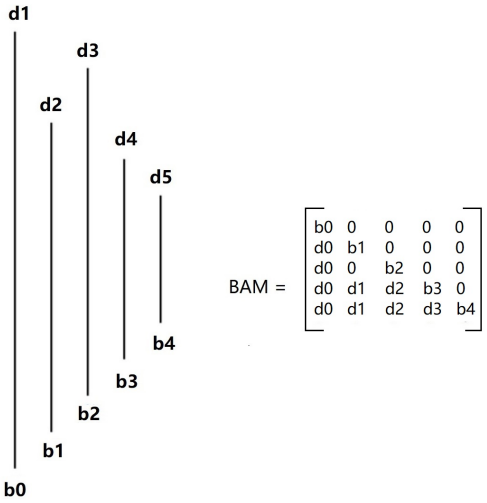


Figure 6: Example of a description matrix using the BAM of the Figure 5 (b).

4.3 Classification of trees

Tree structures carry an important amount of information and finding a universal number which encompass all of them is impossible. This section reviews methods of quantification of different aspects of tree structures. Some of these methods are inspired by previous scientific publications and other were developed for this project.

4.3.1 Principal structures

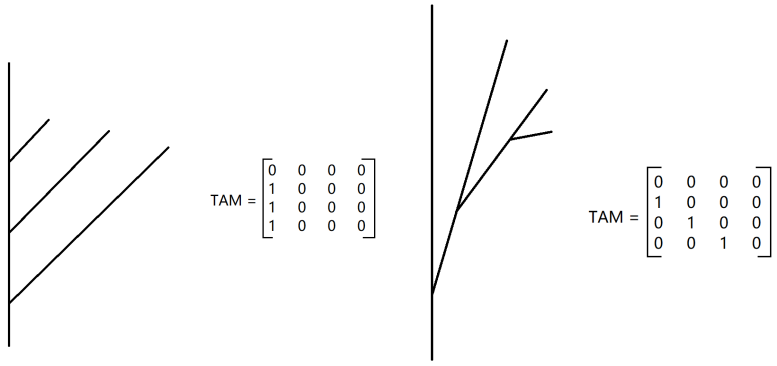
Before diving into these methods, we describe some specific barcodes and tree structure that great to illustrate the method process. The two most important tree structures are the *fir tree* and the *wheat tree* and the barcode we will refer to is the *Russian-doll* barcode.

Definition 4.10. The *Russian doll* barcode is a strictly ordered barcode $B = \{(b_i, d_i)\}_{i=0,...,n}$ where all deaths are ordered in the opposite order of births, i.e., $b_0 < ... < b_n$ and $d_0 > ... > d_n$, B equivalence class is then in the form $(1...n)$.

For the next definitions we will consider a Russian doll barcode to illustrate them.

Definition 4.11. The *fir tree* name is directly inherited from its structure. This particular tree consists of a tree where all branches are merged directly with the longest branch. The TAM of the fir tree is composed by a first column where all elements are non-zero. All the other elements of the matrix are then equal to zero.

Definition 4.12. The *wheat tree* is a tree where the branch structure gives an ear of wheat like form. All branches of this tree always merged with the youngest branch that strictly contain them. The TAM of this tree is composed by a diagonal -1, which is the cases $(i+1, i)$ for all i in the matrix, where all elements are non-zero All the other elements of the matrix are then equal to zero.



(a) (b)

Figure 7: Example of (a) a fir tree and (b) a wheat tree and their associated TAM matrices.

4.3.2 Quantification features

Trees have many features that can be quantified, and in order to do this, various methods exist. These methods allow to classify trees in different sub-groups. This part is mainly inspired by the work proposed in the paper "Tree Topological Features for Unlexicalized Parsing" [8].

Definition 4.13. The *focus parser* is a classifier function which allows to quantify informations of tree structures into a real number. The *focus parser* is a function $f : N \to \mathbb{N}$, is defined by $f(n)$, called the *focus index of* v, equal to 1 if n is a node between the longest branch and another branch or to $f(n_i) + 1$ where $f(n_i)$ is the *focus index* of the first node encountered when taking the shortest pathway to a node of *focus index* equal to 1, i.e., the pathway to the closer node of the longest branch.

This classifier was developed for the project. When using it on a fir tree, all the *focus indices* will be equal to 1 because all the branches are merged to the trunk, but for a wheat tree the *focus indices* will give the strictly positive integers because every nodes of this tree are on the same path when following the first node of the trunk to the node of the smallest branch.

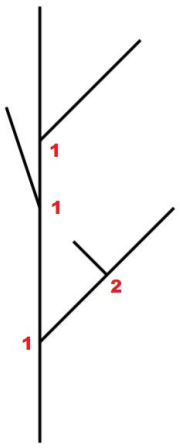


Figure 8: Example of a topological tree with the focus index annotated beside each nodes of the tree in *red*.

Definition 4.14. The *centrifugal parser* is similar to the focus parser conceptually, it is a function $c : N \to \mathbb{N}$ defined by $c(n)$, called the *centrifugal index*, equal to 0 if n is the first bifurcation of the longest branch when starting from the root, or $c(n)$ is equal to $c(n_i) + 1$ where $c(n_i)$ is the *centrifugal index* of the first node encountered when taking the shortest pathway to the root. [9]

This classifier was presented in the publication "Measures for quantifying dendritic arborizations." [9].

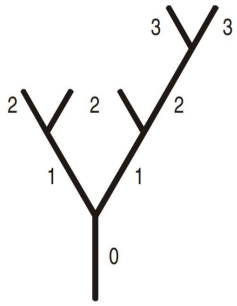


Figure 9: Classification of nodes according to their topological distance from the root, i.e., centrifugal index [9]

In our search of classifier, an important point to notice is that in the case of trees and barcodes, an interval for each branch is given, i.e., additional information is given from a classical topological tree. We are going to use this difference to create a new classifier that takes this information into account.

Definition 4.15. The *weight parser* is a classifier based on the lifetime information of each branch. The *weight parser* is a function $w : N \to \mathbb{R}$ defined by $w(n)$, called the *weight index* of node n, equal to the sum of the lifetimes generated in the sub-tree T_n.

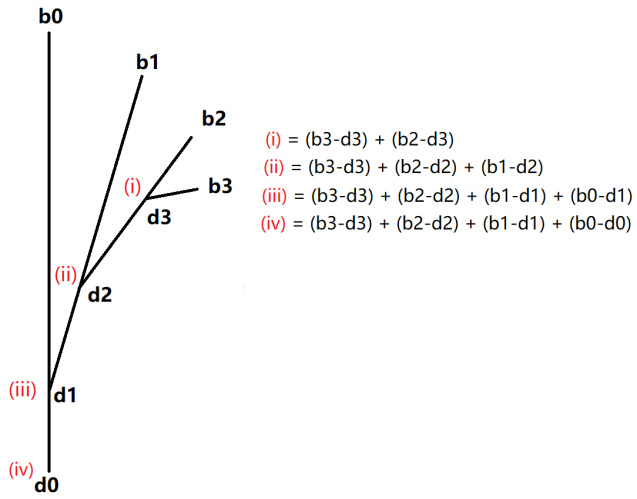


Figure 10: Weight classifier on a wheat tree. It is important to notice that the creation of a sub-tree T_{n_j} have for consequence the cutout of the branch j if branch i is merged with branch j, i.e., $d_j = d_i$ and $lifetime_j = (b_j - d_i)$

4.4 Tree persistent entropy

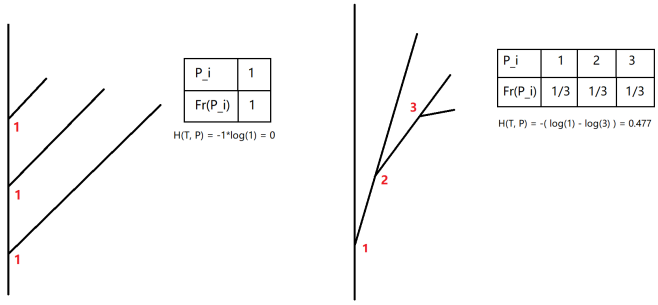
In the previous sections, we defined parsing functions that allow the quantification of tree structure characteristics. For this section, we denote by P the classifier function. The use of a classifier function over all the variable of a tree T gives us a vector $V_P = \{(\text{input}_i, P(\text{input}_i))\}_{i=1,..,n}$, where input_i depends on the chosen classifier function. We denote by F_P a second vector. The size of F_p is equal to the number of different outputs in V_P. $F_P = \{(P_i, \text{frequency}(P_i))\}_{i=0,...,m}$ where P_i is the i^th possible output and $\text{frequency}(P_i)$ is a function which take P_i and return its frequency, i.e., the number of apparition of P_i in the tree, normalized with the total number of output. To summarize, V_P stocks all the results when applying the parser function over all the possible elements of a tree and F_P stocks the frequency of the different results. The vector F_P contain m elements and their frequency, it is then possible then to apply the principles of Shannon entropy to this scenario.

Lemma 4.3. Let T be a topological tree, P a classifier function, P_i a result of this function and $p_i = frequency(P_i)$. The *tree persistent entropy* of T and P, written $H(T, P)$ is defined by:

$$H(T, P) = -\sum_{i=1}^{n} p_i \cdot \log p_i$$

Remark. The more different outputs the classifier function lead to, the bigger H(T, P) will be, and, conversely, the less varied outputs the classifier function lead to, the smaller the entropy will be.

It is then possible to use this definition of tree entropy with various classifier functions. New classifiers can be designed and applied to this formula in order to calculate the entropy of the tree.



(a) (b)

Figure 11: Example of entropy calculation using (a) the fir tree and (b) the wheat tree. The tables represent the F_p used for the calculation. The entropy of the fir tree is 0 because only 1 possible output exist so its frequency is 1 and $\log(1) = 0$.

5 Computational Realization and Tree Analysis

The main goal of this project was to build a useful quantitative method to describe tree generation in a topological way. For the mathematical properties described in the previous sections, an important objective was to be able to implement them. The different algorithms of this section are implemented using Python programming language on an JupyterLab environment. Adding to Python basic elements, few libraries as Numpy, Random or Mathplotlib were used. A GitHub repository of containing all the algorithm with a README explaining their utilisation is available under this link: github.com/Oscar-Henry/Topological-Morphology-Algorithm-for-Tree. For this section, basis knowledge is expected in Python and the adjacent libraries for a good understanding.

5.1 Representation of trees and barcodes

The different algorithms have similarity in manipulating trees and barcodes, thus they will share some part of their codes. Working with trees and barcodes one needs to firstly characterize them rigorously in our algorithm: this step is critical for a healthy algorithm behavior. A barcode will always be represented by a vector of pair, each pair containing respectively the birth and death of a bar. This structure assures to be able to represent barcodes as barcode adjacency matrices (BAM) with lifetime description. The solution of tree adjacency matrix (TAM) with lifetime description in order to represent topological tree was chosen. Each lines of the matrix represent the bars in the same order than the barcode. Only strict ordered barcodes were used in the different soft-wares. The diagonal element is equal to the birth of the branch. The other non-zero element are in the triangular inferior part of the matrix representing the death and so nodes of branches or possible nodes in BAM.

The algorithms are designed to be used with various barcodes, that can be chosen by the user. The implementation of tests is needed to avoid unwanted inputs. Three tests are implemented in order to assure that:

- All births need to be smaller than their corresponding death.
- All births need to be ordered (strict barcode).
- The trunk must contain all the bars without exception.

The past paragraph draw a common beginning of all algorithms by the successive steps of:

- The importation of the different needed libraries.
- The initialization of the chosen barcode.
- The test of the barcode with the set of tests.

5.2 The Combinatorics of Tree-Realizations Algorithm

The *Combinatorics of tree-realization algorithm* takes for input a single barcode and have for output the equivalence class of the barcode and its tree-realization number. This algorithm's main advantage is to reduce drastically calculation time, a lower calculation time unlocks the possibility to calculate tree-realization number or find the equivalence class number of much larger barcode.

There exists several methods to approach the construction of the algorithm but one easy way to think of this problem is by looking at each of the n bar of a given barcode except the longest one (b_0, d_0) and compare the different deaths of the bars. If the death of bar_i is smaller than the death of a previous bar, this previous bar contain strictly bar_i. This solution gives a complexity in $O(n^2)$ because the two loop, $i \in [1, n]$ and $j \in [i, n]$, give a number of instructions proportional to the sum of all positive integers. Each index can be added in a n dimensions vector (there is $n + 1$ bars but the longest bar do not have an index). When we have all the bar indices we can apply the *Lemma 4.2* formula to find the tree-realization number of the barcode. This can be done in the loop mentioned before by multiplying a variable initialised with the value of 1 by the found index.

Finding the equivalence class of a barcode is more complex even if the process is similar to the previous algorithm. This algorithm goes through the respective steps:

- Creating a new vector equal to the difference between the vector of index and the position of each bar. This already gives which bars are contained by all the previous ones because $\text{Index}_{bar_i} - \text{position}(bar_i) = 0$ if bar_i is strictly contained by bar_k with $k < i$.
- Running a loop $n - 1$ times which add the number of already passed loop to the smallest value.
- Running an internal loop inside the previous loop to find the actual smallest value in the vector.

This method assures to give the equivalence class of the barcode by ranking each bar by death from the oldest one to the youngest.

```
The tree-realization number of this barcode is :  24
The equivalent group of this barcode is : ( [1. 2. 3. 4.] ).
```

Figure 12: Output example of the algorithm with respectively the value of the tree-realization number and then the equivalence class written in a vector.

5.3 The Random Tree Generator Algorithm

It is possible to generate different trees from a single barcode. The *Random Tree Generator algorithm* aim to generate all the possible tree-realizations of a barcode. Knowing that a barcode can be represented with a BAM, this algorithm

take the BAM of a barcode given as the input and returns a tree-realization example using a realization TAM. The algorithm can process this realisation a large number of times and the created tree-realization is chosen by a random process. Each tree-realization have the same probability to appear. The algorithm verifies this statement by comparing the frequency of each tree-realizations when a large number of trees are created, the frequency should converge to the probability when the number of created trees approaches infinity. The uniformly distributed probability between trees is an arbitrary decision and it can be discussed, for example it can be interesting to imagine constraint in the generation of tree, where certain structure may be more selected.

What does the sentence "each different tree-realizations have the same probability to appear" mean? When having the BAM of the given barcode, each branch except the trunk can merge with at least one branch. The probability that a branch merges with any branch from its set of possibilities is equal for each branch of the set. The probability is uniformly distributed among the branches of the set. Let call $p_j(i)$ the probability that the branch j merge branch i, then:

$$p_j(i) = \frac{1}{\sum\limits_{k=0;\ k<j}^{j} I(j,k)} \quad I(x,y) = 1 \text{ if } (x,y) \neq 0, \quad I(x,y) = 0 \text{ else.}$$

. Using this formula, it is possible to create a *probability matrix* which contain $p_j(i)\ \forall i,j$; this is equivalent to normalizing each line.

The next part of the algorithm is to find a way to pick which branch will be merged in function of the probability written in the probability matrix. To resolve this problem, a common method is to pick a random number $random \in [0,1]$. The algorithm will iterate on each branch. Because of the normalized line, the sum of all probability in a single line equals 1. This assures us that each branch will be connected to only one different branch. Let p_{j_i} be a number as if $random \leq p_{j_i}$ then the branch j will merged the branch i and stop looking for other branches,

$$p_{j_i} = \sum_{k=0;\ k\leq i}^{j} p_j(k).$$

In order to verify the results, i.e, the process did generated all tree-realizations uniformly, we create a frequency matrix where,

$$\text{cell}(i,j) = \frac{\sum\limits_{k=0}^{n} I(M_k(i,j))}{n}.$$

The function $I(M_k(i,j)$ return 1 if the cell (i,j) of the k^{th} TAM is non-zero and n is the total number of generated TAM. Then, with a large number of tree, these cells should approach the value of the probability matrix.

We can illustrate this with the case where a branch j can merge the 3 previous branches, i.e., bar_j is strictly included in them, in a barcode of 4 bars. The

branch j can attach to the three other branches with the respective probabilities $[1/3, 1/3, 1/3]$. Calculating $p_{j_i}, i \in [0, 2]$ for each cell, a new vector is obtained $p_j = [1/3, 2/3, 1]$. By taking a random number, for example $random = 0.5$ the condition will be valid for $i = 1$ and thus the algorithm will stop, merging the j^th branch with the i^th branch.

```
All possible matrices are in the form:
 [[0. 0. 0. 0. 0.]
 [9. 1. 0. 0. 0.]
 [8. 8. 2. 0. 0.]
 [7. 7. 7. 3. 0.]
 [6. 6. 6. 6. 4.]]

The last matrix realised was:
 [[0. 0. 0. 0. 0.]
 [9. 1. 0. 0. 0.]
 [0. 8. 2. 0. 0.]
 [0. 0. 7. 3. 0.]
 [6. 0. 0. 0. 4.]]

The Probability matrix is:
 [[0.         0.         0.         0.         0.        ]
 [1.         0.         0.         0.         0.        ]
 [0.5        0.5        0.         0.         0.        ]
 [0.33333333 0.33333333 0.33333333 0.         0.        ]
 [0.25       0.25       0.25       0.25       0.        ]]

The Frequency matrix over 1000 realisations is:
 [[0.         0.         0.         0.         0.        ]
 [1.         0.         0.         0.         0.        ]
 [0.480005   0.519995   0.         0.         0.        ]
 [0.33797403 0.36098202 0.30104396 0.         0.        ]
 [0.27396004 0.268001   0.23001998 0.22801898 0.        ]]
```

Figure 13: Output example with firstly the BAM matrix which show the multiple tree-realization possibilities, secondly the last realization matrix created by the algorithm to show what a tree can look like with this given barcode, then the probability matrix followed by the frequency matrix to compare the result and assure the well being of the algorithm.

5.4 The Tree Persistent Entropy Algorithm

The *Tree Persistent Entropy Algorithm* aims to use the results developed in this project to guide tree selection. This algorithm takes for input a barcode and return multiple output: the algorithm generates random tree-realizations using the *Random Tree generator* algorithm, calculates for each tree using classifier function, in our case the *focus parser*, their indices and the resulting tree entropy. The algorithm conserve the different entropy of each tree-realization in a vector and can show a density function of the entropy for the barcode. The aim of this algorithm is to propose an oriented selection of tree-realizations for the user in function of the tree persistent entropy of all tree-realizations of a barcode.

As said in the previous paragraph, this algorithm performs a random tree generation in the first place and returns the output of the *Random Tree Generator* algorithm, see Figure 13. When a single tree is generated and represented in the algorithm by a TAM, the algorithm computes the *focus parser* function on it. The choice of the *focus parser* is arbitrary and the objective is to convince the users to adapt the algorithm to their own functions. In order to calculate the *focus index* of each nodes of the tree, the algorithm goes through different steps:

- The creation of an empty vector which will contain all the nodes and their associated *focus index*;
- In order to compute the parser function, we can use the TAM and its properties to find the *focus index* of each vertex in a tree-realization. A branch merged with the trunk will have its non-zero element in the first column. We can go through the first column and then set for all $(i, 0) \neq 0$, $P_{focus}(\text{vertex}_i) = 1$. Then, for each cell $(i, j) \neq 0$ with $i > j > 0$ the corresponding focus index is:

  $$P_{focus}(\text{vertex}_i) = P_{focus}(\text{vertex}_j) + 1,$$

  where $P_{focus}(vertex_i)$ is the focus index of the i^th vertex. This formula is well defined because $i > j > 0$ and the algorithm go through each branch from 1 to n, we can remark that the first branch is inevitably ligated to the trunk.
- Now, We need to count how many different values of *focus index* we have and how many times they appear. In order to do this we can simply create a vector where we check if the i^th *focus index* value is already in, if yes we can increase by 1 its number of appearances and if not we can add a new pair containing the new value with a number of appearance of 1. At the end, we can normalize the number of appearances by the total number of vertices to have the frequencies.
- The final step of this process consist to calculate, for each tree-realization, its entropy using the formula of *Lemma 4.3* and stock this entropy in a vector. The vector containing the entropy values encodes also the number

of tree-realizations that give this entropy by using the same method than the method used to obtain the frequencies of *focus index*. A graph summarizing the different entropy values and the density of tree-realizations is presented at the end of the algorithm as well as the maximal and minimal entropy and there associated TAM.

This algorithm reposes on a fundamental principle of learning machine: comparing the different values of a quantified characteristics in order to compare and pick a chosen value which fit the best the objectives. This algorithm was also performed with the *weight parser*. This version is included in the GitHub repository.

```
The frequency of each tree persistent entropy of 1000 tree-realizations is :

[[0.0, 0.009], [0.217, 0.094], [0.292, 0.32], [0.413, 0.136], [0.458, 0.327], [0.579, 0.105], [0.699, 0.009]]
```

Figure 14: The first output of the algorithm is F_P, a vector containing the different entropy value and their associated frequency. The frequency represent the probability of creating randomly a tree with the input barcode which lead to this entropy using the parser function chosen.

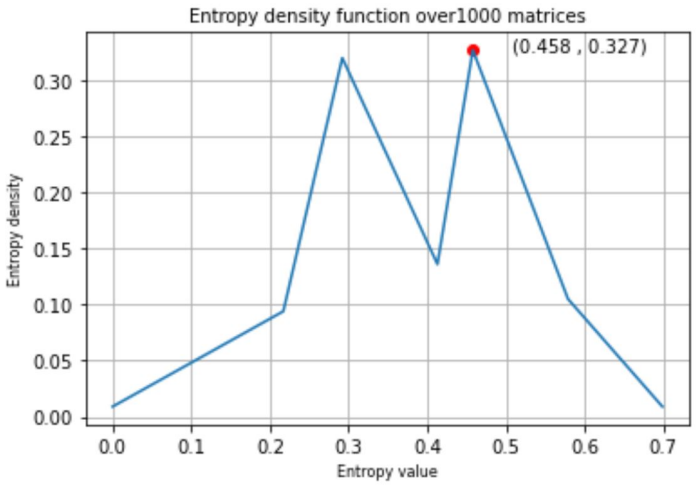
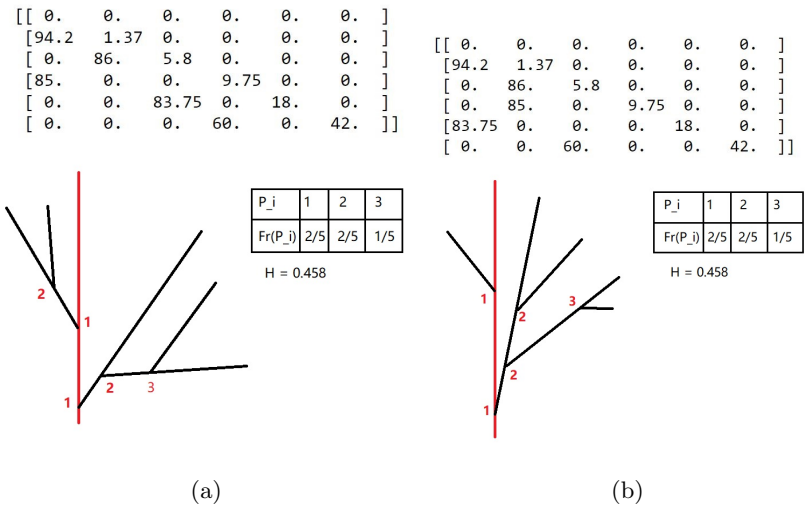


Figure 15: This graph represents the density probability function of different entropy values over the random generation of 1000 tree-realizations from a single barcode. The *red* point represent the maximum possible frequency with the associated entropy value.

The presented results are obtained using the algorithm with the focus parser and a barcode B such as $B \in (12345)$, with (12345) the equivalence class. Using this result to show tree-realizations TAM in order to look a the different structures resulting in this entropy, we obtain:



(a) (b)

Figure 16: Tree-realizations of the barcode B with the longest branch in red, the focus index next to each vertices in red, the frequency vector and the associated entropy.

6 Conclusion

6.1 Summary

The main goal of this project was to propose a first introduction to topology applied in a biological problematic. The presentation of the fundamental tools and rules of this field where presented in the first section. A summary of TNS and TMD algorithm allow to understand how using topological summaries such as barcodes, can allow the conservation of topological information and the creation of new topological structures, here in the context of new artificial neurons generation. Representing branching structures into barcodes unlock multiple mathematical manipulations described in the the fourth section of this report. New results are then possible by using the properties of barcodes and new methods of classification can be developed. This motivates to research new ways to describe topological trees and how this information can be stocked and used in order to optimize various aspect of artificial tree generation. In this report, the notion of persistent entropy for trees was introduced and various computational results where presented in order to facilitate the future research of this field. A GitHub repository was implemented and available a this link: github.com/Oscar-Henry/Morphological-Algorithms, all the source codes is open-source allowing modifications by the community. Even though this project reached its principal goals, multiple problems remain unsolved and new research tracks have been opened.

6.2 Discussion

The complex structure of topological trees makes impossible the idea of finding a universal quantification function. The method presented in this project have the property to be adaptable to various scenarios and techniques of quantification. This approach can be extended to multiple mathematical results in the tree morphology context. When the *parser* can bring to a quantification of features in tree structures, similar methods can be developed in order to quantify tree-realizations directly and so apply the persistent entropy formula as described in this report. More than quantifying tree-realizations, we can also find parser functions that describe barcodes characteristics and calculate the entropy of the different equivalent classes, etc. The strength of this method is its versatility over the given input.

This approach still has many defaults. The necessity of finding multiple relevant parser functions in order to have relevant results leads to many calculation which sometimes can be difficult even for a computer. This method is then restraint to small barcodes because the comparison of large barcodes leads to too large amounts of calculations, this is due to the exponential complexity of these algorithms. The tree persistent entropy is limited to giving indications over small tree structures and does not deconstruct entirely the complex structure of them so additional method need to be performed in parallel to consolidate the results.

6.3 Acknowledgement

I thank Ms K. Hess for taking me into my first research experience in the Laboratory for Topology and Neuroscience in the context of my bachelor curriculum at EPFL. I thank also Adélie Garin for guiding me through the development of this project. I am very grateful for all they did for me and I send them my best regards.

References

[1] K. Lida, P. Dłotko, M. Scolamiero, R. Levi, J. Shillcock, K. Hess, and H. Markram, "A topological representation of branching neuronal morphologies," *Neuroinformatics*, vol. 16, 10 2017.

[2] K. Lida, H. Dictus, A. Chalimourda, W. Geit, B. Coste, J. Shillcock, K. Hess, and H. Markram, "Computational synthesis of cortical dendritic morphologies," *SSRN Electronic Journal*, 01 2020.

[3] K. Lida, G. Adélie, and K. Hess, "From trees to barcodes and back again: theoretical and statistical perspectives," *none*, 09 2020.

[4] B. Edward A. and W. S. Gill, "Lists, decisions and graphs," *University of California*, p. 147., 2010.

[5] S. C. E., "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423,623–656, 10 1948.

[6] N. Atienza, L. M. Escudero, M. J. Jimenez, and M. Soriano-Trigueros, "Persistent entropy: a scale-invariant topological statistic for analyzing cell arrangements," *none*, 05 2019.

[7] Y. Lin, W. Yusu, M. Elizabeth, G. Ellen, and W. Bei, "A structural average of labeled merge trees for uncertainty visualization," *IEEE Transactions on Visualization and Computer Graphics*, 10 2019.

[8] C. Samuel, C. Lawrence, and C. Mickey, "Tree topological features for unlexicalized parsing," *Coling 2010: Poster Volume*, pp. 117–125, 09 2010.

[9] H. Uylings and J. Pelt, "Measures for quantifying dendritic arborizations.," *Network (Bristol, England)*, vol. 13, pp. 397–414, 09 2002.