

INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO



# **Dos construcciones para el Árbol Aleatorio Continuo**

TESIS

QUE PARA OBTENER EL TÍTULO DE

**LICENCIADO EN MATEMÁTICAS  
APLICADAS**

PRESENTA

**JOSÉ CARLOS CRUZ ARANDA**

ASESORA: DRA. AIRAM ASERET BLANCAS BENÍTEZ

CIUDAD DE MÉXICO

2024

«Con fundamento en los artículos 21 y 27 de la Ley Federal del Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada “**Dos construcciones para el Árbol Aleatorio Continuo**”, otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la Biblioteca Raúl Baillères Jr., la autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por tal divulgación una contraprestación.»

JOSÉ CARLOS CRUZ ARANDA

---

FECHA

---

FIRMA

# Agradecimientos

A mis padres, Emilia y Jesús, por el apoyo incondicional a lo largo de toda la carrera. A mi hermano, Gerardo.

A mis amistades, por acompañarme a lo largo de toda la carrera. Por todos aquellos momentos: los desayunos, las desveladas, las sesiones de estudio, los partidos, los intentos de escalar muros difíciles, las largas noches en facultad menor. Sin ustedes esta experiencia hubiera sido mucho más desafiante.

A mis profesores, gracias por las tareas, los exámenes y las desveladas. En especial me gustaría agradecer a mi asesora, la Dra. Airam Blancas, por haberme introducido al mundo de la probabilidad y por guiarme en el proceso de realización de este trabajo.

Al Dr. Ernesto Barrios, por ser un gran profesor, un gran jefe y sobre todo, un gran amigo.

Y sobre todo me gustaría agradecer a mi mascotas, Chiripa, Lyra, Molly, Nicole y Olenka, por acompañarme en aquellas largas noches de trabajo.

Finalmente, me gustaría agradecer a todas aquellas personas que, de manera directa o indirecta me ayudaron a poder terminar este viaje.

# Resumen

Las gráficas aleatorias han sido un tema popular los últimos 30 años. Existe una gran cantidad de preguntas interesantes en esta área, lo cual en parte es debido a la gran cantidad de posibles gráficas aleatorias. En este trabajo en particular estamos interesados en árboles aleatorios.

Incluso si acotamos las posibles gráficas, aún existen una gran cantidad de preguntas pertinentes, entre ellas encontramos cosas como *¿cuántas hojas tienen ciertos tipos de árboles?*, *¿qué tan grande es el árbol?*, pero la que nos interesa a nosotros es *¿qué pasa si hacemos que nuestro árbol crezca?*

Como es común en muchos ambientes matemáticos, el límite de objetos discretos hace que aparezcan objetos continuos, en nuestro caso, lidiaremos con Árboles Aleatorios Continuos.

Para ver la convergencia consideraremos dos modelos particulares de árboles aleatorios, **Árboles Galton-Watson** y **Árboles Aleatorios Uniformes**.

Este objeto límite fue introducido en 1991 por Aldous en [1], tomaremos mucha inspiración de ese trabajo y también en trabajos de Jean Le-Gall, principalmente en [8] para poder presentar este tema en una manera accesible para estudiantes de licenciatura.

# Abstract

Random graphs have been a popular topic among academics for around 30 years. There are a lot of interesting questions in this area, which has some correlation with the fact that there are a lot of possible random graphs. On this particular work we are interested in random trees.

Even if we narrow down the possible graphs, there are still a lot of questions worth asking, among them we can find things such as, *how many leafs does certain types of trees have?*, *how big is the tree?*, but the one that interests us is *what happens when we make our tree grow?*

As is common in many mathematical settings, the limit of discrete objects makes continuum objects appear, in our case, we will deal with Continuum Random Trees.

To see the convergence, we will focus on two particular models for random trees, **Galton-Watson trees** and **Uniform Random Trees**.

This Continuum Random Tree was introduced in 1991 by Aldous in [1], we will take great inspiration in that work and also on the works of Jean Le-Gall, principally on [8] to present these topics in such a way that a undergraduate student may be able to explore this world.

# Contents

<b>Resumen</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Introduction</b>	<b>1</b>
<b>1 What is a CRT?</b>	<b>3</b>
1.1 Discrete trees . . . . .	3
1.2 Real trees . . . . .	7
1.3 Coding real trees . . . . .	11
1.3.1 Building a metric space . . . . .	12
1.3.2 Building a real tree . . . . .	14
1.4 Brownian Motion . . . . .	24
<b>2 Construction via Galton Watson Trees</b>	<b>28</b>
2.1 Ordered trees and their encodings . . . . .	28
2.2 Defining a Galton-Watson tree . . . . .	41
2.3 Convergence to a Brownian Motion . . . . .	49
2.3.1 Results in terms of Contour functions . . . . .	56
2.4 Convergence to a CRT . . . . .	62

<b>3</b>	<b>Construction via URT</b>	<b>65</b>
3.1	What is a URT? . . . . .	65
3.2	Characterizing the URT . . . . .	68
3.2.1	The Aldous-Broder algorithm . . . . .	69
3.2.2	The Aldous algorithm . . . . .	79
3.3	The branches of a URT . . . . .	79
3.4	Construction of the CRT . . . . .	81
3.4.1	Preliminaries for the construction of the CRT . . . . .	82
3.4.2	The line-breaking construction . . . . .	85
3.5	Convergence to a CRT . . . . .	87
3.6	A Continuum Random Tree . . . . .	89
3.7	The idea behind the proof . . . . .	89
3.7.1	Convergence of set-representations . . . . .	92
3.7.2	Convergence of probability measures . . . . .	93
	<b>Conclusions</b>	<b>96</b>
<b>A</b>	<b>Codes</b>	<b>98</b>

# Introduction

Graph theory is a relatively recent area of mathematics. A particularly youthful facet within this realm is the study of random graphs. Evidence of this lies in the fact that most of the results presented in this work originate from publications that emerged towards the end of the last century. While this might seem distant, it is worth noting that calculus itself was 'discovered' at the end of the 17th century.

Among all of the possible random graphs we could dive into, we are particularly interested in random trees. Our interest in this particular type of random graphs comes from different places. First of all, the object of a tree is a convenient structure: although they have certain properties that make them simple to study, this does not mean that they are trivial, as they also have noteworthy behaviours. Actually, what motivated me to write about them is the existence of the Continuum Random Tree.

Besides, trees are used in a lot of different fields, among which we highlight combinatorics. Trees are fundamental objects in this field, and as Goldschmidt states in [9], random combinatorial structures and their scaling limits have been a very important link between probability and combinatorics.

The focus of this work revolves around the Continuum Random



Tree. The existence of this object verges on a cliché, given its association with Brownian Motion and, by extension, the Normal distribution, suggesting that with enough effort, everything tends to conform to a normal pattern.

In this work we explain the convergence of some sequence of random (discrete) trees to the Continuum Random Tree. As a first stop in our journey, we will have to define continuum trees. Our exploration delves into specific models of discrete random trees essential for observing these convergences. These models encompass Galton-Watson trees and Uniform Random trees.

We expect that the presentation of these two convergences to the Continuum Random Tree motivates the readers to dive even further in this magical world of random graphs and scaling limits.

# Chapter 1

## What is a CRT?

In order to fulfill the objective of presenting constructions for the *Continuum Random Tree* (CRT), we will first introduce this object. To make this a little bit easier we are going to start talking about discrete trees, and then we can introduce the idea of continuum trees. Finally we will attach randomness into them.

### 1.1 Discrete trees

As a first step to study trees, we have to introduce some basic concepts from Graph Theory, this area of mathematics started in the early eighteen century thanks to Leonhard Euler.

A graph is defined by the pair  $G := (V, E)$  where  $V$  is a set that contains the vertices of the graph and  $E$  is the set of edges<sup>1</sup>.

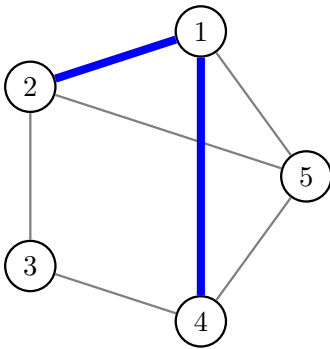
Once we have our primary object, we can talk about ways to explore them and also about attributes of the vertices. For starters, the

---

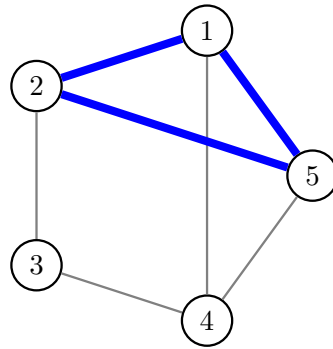
<sup>1</sup>The edges connect two vertices, they can be defined as the set that contains the two vertices that are connected, for example  $e_1 = \{v_0, v_1\}$ .

**degree** of a vertex is defined as the number of edges incident to the vertex. Define  $\#(G)$  as the number of vertices that the graph  $G$  has. A walk on a graph  $G$  between two vertices is a finite alternating sequence of adjacent vertices and edges of  $G$ . A walk that does not repeat vertices is called a **path**. We define the **length of a path** as the number of edges that are in the path. A **cycle** is a closed walk, that is, it does not repeat edges and vertices except for the first and last ones. See Figure 1.1.

**Figure 1.1.** The gray lines are the edges of the graph, and the blue lines are the edges that form the path and the cycle.



(a) Path of length 2 between vertices 2 and 4.



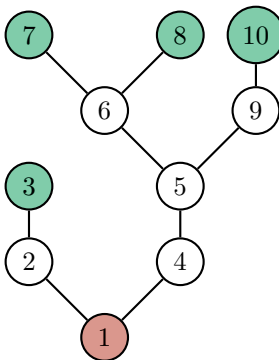
(b) Cycle on the graph  $G$ .

If for every pair of vertices on  $V$ , there exists a path between them, we say that the graph  $G$  is **connected**. If there are no cycles on the graph, we say that a graph is **acyclic**. Finally, a **tree**  $T$  is defined as a connected acyclic graph.

There are two things that will be of our interest: the **root** of a tree and the **leaves** of a tree.

The **root** of a tree is just a vertex that we decide to highlight for whatever reason; depending on the context one may have more or less reasons to select it. We can see that the root of a tree is very arbitrary. On this section, we have decided that the root of our tree is simply the vertex labeled 1. We can interpret this election as the first vertex from which our tree grows, and we choose the planar representation<sup>2</sup> of the trees to show this.

The **leaves** of a tree are those vertices that have degree 1. To make these definitions more clear see Figure 1.2.



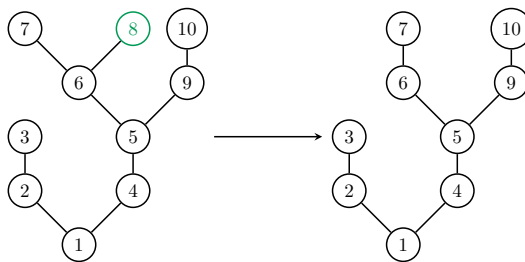
**Figure 1.2.** Tree with 10 vertices. The leaves are colored green, and the root is highlighted in red.

Let us provide an alternative definition for the degree of a vertex, which holds true in the specific case where the graph containing these vertices is a tree. For every  $v \in T$  consider the graph  $T \setminus \{v\}$  that is defined by the pair  $(V \setminus \{v\}, \hat{E})$ . Where the set  $\hat{E}$  has the same elements as the set  $E$  except for those that were incident to  $v$ .

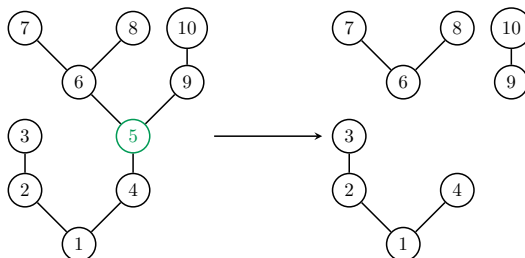
---

<sup>2</sup>The planar representation of a graph is the visual representation of the nodes and edges that form the graph, avoiding intersections of edges other than on incident vertices.

The degree of a vertex  $v$  can be defined by the number of connected components of  $T \setminus \{v\}$ . Therefore, if the graph defined by  $T \setminus \{v\}$  has only one connected component we say that  $v$  is a leaf. See Figure 1.3.



(a) On the left  $T$  and on the right  $T \setminus \{8\}$ .



(b) On the left  $T$  and on the right  $T \setminus \{5\}$ .

**Figure 1.3.** In (a), it is evident that upon removing  $\{8\}$ , a single connected component remains. In (b), it is observable that if  $v$  is not a leaf, then the removal of  $v$  from  $T$  results in more than one connected component in  $T \setminus \{v\}$ .

The last thing we will discuss regarding discrete graphs is the fact that we can see any tree  $T$  as a metric space, where the points of the space are the vertices and we assume the length of the edges to be 1. In order to properly define this as a metric space we need to attach a metric to it, so let us introduce the well known graph distance.

For any two vertices  $u, v \in T$ , we define the distance on the graph as

$$d_T(u, v) = \begin{cases} \min\{\text{length of paths between } u \text{ and } v\} \\ \infty, & \text{if there are no paths between } u \text{ and } v. \end{cases} \quad (1.1)$$

It is easy to see that  $d_T$  is indeed a metric on the tree  $T$ . The fact that  $(T, d_T)$  is a metric space will be useful on the following section. Now we have everything we need to know to begin our discussion of *real trees*.

## 1.2 Real trees

In this thesis the main objects of interest are compact real trees, for short, we will refer to them as real trees. We are particularly interested in compact real trees as their compactness is useful to prove convergences, which is the main topic of this thesis.

Note that the previous section started with the definition of a tree and ended by making it a metric space. It turns out that in order to define real trees, we need to make that journey backwards.

Let us start with a compact metric space  $(\mathcal{T}, d)$ . We want a *tree-like* behaviour. By recalling the definition of a discrete tree, we want this object to be connected and acyclic. To make this metric space connected, we require that there exists *some kind of path* between every pair of elements of  $\mathcal{T}$ . And to make it acyclic, we want those paths to be unique.

In order to define the paths, we need that for every  $a, b \in \mathcal{T}$  there exists a continuous map  $f_{a,b}$  that goes from  $[0, d(a, b)]$  into  $\mathcal{T}$ , where we define  $f(0) = a$  and  $f(d(a, b)) = b$ . Since it moves on  $\mathcal{T}$  from  $a$  to  $b$ , this map can be interpreted as the path between  $a, b$ . Moreover,  $\mathcal{T}$  is

connected because this path is defined for every pair of elements  $a, b$  on  $\mathcal{T}$ .

To make  $\mathcal{T}$  acyclic, we need to define a notion of uniqueness regarding the map  $f_{a,b}$ .

Let  $q$  be a continuous map from  $[0, 1]$  into  $\mathcal{T}$  such that  $q(0) = a$  and  $q(1) = b$ . If  $q$  and  $f_{a,b}$  are equal in all their domain, then we have that there is only one way to go from  $a$  to  $b$  on  $\mathcal{T}$  and thus we have a acyclic object.

The following definition resumes the intuition presented before.

**Definition 1.1** (Le Gall [8]). *A compact metric space  $(\mathcal{T}, d)$  is a real tree if the following two properties hold for every  $a, b \in \mathcal{T}$ .*

- (R1) *There is a unique isometric map  $f_{a,b}$  from  $[0, d(a, b)]$  into  $\mathcal{T}$  such that  $f_{a,b}(0) = a$  and  $f_{a,b}(d(a, b)) = b$ .*
- (R2) *If  $q$  is a continuous injective map from  $[0, 1]$  into  $\mathcal{T}$ , such that  $q(0) = a$  and  $q(1) = b$ , we have*

$$q([0, 1]) = f_{a,b}([0, d(a, b)]).$$

*A rooted real tree is a real tree  $(\mathcal{T}, d)$  with a distinguished vertex, denoted by  $\rho := \rho(\mathcal{T})$ , called the root.*

From our definition, we encounter the necessity to introduce some notation for the paths between elements of  $\mathcal{T}$ . Hence, the range of the mapping  $f_{a,b}$  in (R1) is denoted by  $\llbracket a, b \rrbracket$ . It represents the curve between  $a$  and  $b$  in the tree, i.e. the path between  $a$  and  $b$ .

So far we have focused on the macroscopic attributes of the trees, it is time to shift our focus to a microscopic level. Our goal now is to understand the relations between the elements of  $\mathcal{T}$ . To do so, let us present a couple of definitions.

First of all, think of  $\mathcal{T}$  as a population, the first member of this population is  $\rho$ , the root of  $\mathcal{T}$ . Following this idea, we will define the path  $\llbracket \rho, a \rrbracket$  as the **ancestral line** of vertex  $a$ .

What if we want to explore more punctual relations between the members of our tree? Following the intuition obtained from our exploration of discrete trees, there are two possibilities in this regard, either two elements are related, by one being descendant of the other, or they are not.

For the first scenario, we introduce a partial order on the tree, by setting  $a \preceq b$ ,  $a$  is an ancestor of  $b$ , if and only if  $a \in \llbracket \rho, b \rrbracket$ .

Now, if two elements do not have a direct hierarchical relationship, that is, one is not a descendant of the other or vice versa, we can look for the next best thing; e.g. a common ancestor. However, it is clear that every pair of vertices will have at least one common ancestor, the root. Therefore, if we want this ancestor to be useful, we must limit it to the most recent.

Let  $a, b \in \mathcal{T}$ , then there is a unique  $c \in \mathcal{T}$  such that  $\llbracket \rho, a \rrbracket \cap \llbracket \rho, b \rrbracket = \llbracket \rho, c \rrbracket$ . The uniqueness of  $c$  arises from the assumption that if there are at least two different common ancestors  $c$  and  $d$  (where  $c$  is not a descendant of  $d$  or vice versa), both should serve as ancestors to  $a$  and  $b$ . Consequently, there would be at least two distinct paths between the root and  $a$  and  $b$ . This would mean that  $\mathcal{T}$  does not fulfill the condition (R1). We write  $c := a \wedge b$  and call  $c$  the **most recent common ancestor** of  $a$  and  $b$ .

Finally, we can also define the **multiplicity of a vertex**  $a \in \mathcal{T}$ . It is defined as the number of connected components of  $\mathcal{T} \setminus \{a\}$ . The multiplicity gives us information about how "connected" any vertex is. In particular, vertices of  $\mathcal{T} \setminus \{\rho\}$  with multiplicity 1 are called **leaves**.

In this thesis we want to explore the convergence of trees, then we



need to define a notion of distance between trees. To determine the optimal metric to use, we must recall that real trees were defined as compact metric spaces. Therefore, we can utilize the Gromov-Hausdorff distance, a commonly employed method for measuring distances between such spaces. In order to define this metric we need to define first the Hausdorff metric.

Let  $(E, d)$  be a metric space, denote by  $d_H(X, Y)$  the usual metric between compact subsets of  $E$ :

$$d_H(X, Y) := \sup \left\{ \sup_{x \in X} d(x, Y), \sup_{y \in Y} d(X, y) \right\}, \quad (1.2)$$

where  $d(a, B) := \inf_{b \in B} d(a, b)$ .

Let  $\mathcal{T}$  and  $\mathcal{T}'$  be two rooted compact metric spaces, with respective roots  $\rho$  and  $\rho'$ . We define the distance  $d_{GH}(\mathcal{T}, \mathcal{T}')$  by

$$d_{GH}(\mathcal{T}, \mathcal{T}') = \inf \{ d_H(\varphi(\mathcal{T}), \varphi'(\mathcal{T}')) \vee d(\varphi(\rho), \varphi(\rho')) \},$$

where the infimum is over all choices of a metric space  $(E, d)$  and all isometric embeddings  $\varphi : \mathcal{T} \rightarrow E$  and  $\varphi' : \mathcal{T}' \rightarrow E$  of  $\mathcal{T}$  and  $\mathcal{T}'$  into  $(E, d)$ .

We will use an alternative definition of  $d_{GH}$ . Let  $(E, d_1)$  and  $(M, d_2)$  be two compact metric spaces. For compact metric spaces, one can define a correspondence between  $E$  and  $M$  as a subset  $\mathcal{R}$  of  $E \times M$  such that for every  $x_1 \in E$  exists at least one  $x_2 \in M$  such that  $(x_1, x_2) \in \mathcal{R}$ , similarly, for every  $y_2 \in M$  there exists at least one  $y_1 \in E$  such that  $(y_1, y_2) \in \mathcal{R}$ . The **distorsion** of the correspondence  $\mathcal{R}$  is defined by

$$\text{dis}(\mathcal{R}) = \sup \{ |d_1(x_1, y_1) - d_2(x_2, y_2)| : (x_1, x_2), (y_1, y_2) \in \mathcal{R} \}.$$

Then, if  $\mathcal{T}$  and  $\mathcal{T}'$  are two rooted compact metric spaces with respective roots  $\rho$  and  $\rho'$ , we have

$$d_{GH}(\mathcal{T}, \mathcal{T}') = \frac{1}{2} \inf \text{dis}(\mathcal{R}), \quad (1.3)$$

where the infimum is over all of the correspondences  $\mathcal{R}$  between  $\mathcal{T}$  and  $\mathcal{T}'$ , such that  $(\rho, \rho') \in \mathcal{R}$ . A proof of this fact can be found in [11].

We have covered some fundamental concepts regarding real trees, but now let us dive into the practical aspect: how exactly do we construct one? In the following section, we will explore precisely that.

### 1.3 Coding real trees

Let us think that we encounter a tree that we did not build, we do not really know anything about it. In order to be satisfied with our knowledge about this object, how much do we need to know about it? Let us start by obtaining a general idea of this unknown object.

The attribute that we will easily identify and find somewhat useful for reconstructing the tree is its general shape. To do so, we will use **contour functions**. These functions describe the silhouette that would form if we put a blanket on top of the tree and watched it from the side, see Figure 1.4.



**Figure 1.4.** On the left, we depict a discrete tree, while on the right-hand side, we illustrate how a blanket would appear if draped over the tree. The outline traced by the exterior of the blanket represents our contour function.

These functions enable us to identify specific structures and assess

the size of the tree.

Let us call the contour function  $g$ . It is continuous and there are only two points where  $g(x) = 0$ , when  $x = 0$  and on the maximum value in its domain.

Once we have understood the idea behind this type of function, let us see how we can use similar functions to codify real trees. It is important to have in mind that we are looking for two objects to define a real tree, a space and a metric.

### 1.3.1 Building a metric space

Consider a continuous function  $g : [0, \infty) \rightarrow [0, \infty)$  with compact support and such that  $g(0) = 0$ . To avoid the trivial case, assume that  $g$  is not identically zero. For every  $s, t \geq 0$ , we set:

$$m_g(s, t) = \inf_{r \in \mathcal{I}} g(r), \quad (1.4)$$

where  $\mathcal{I} = [s \wedge t, s \vee t]$ , and

$$d_g(s, t) = g(s) + g(t) - 2m_g(s, t).$$

Let us start by establishing some properties of  $d_g$ . The motivation behind the definition will be clear later on.

**Proposition 1.1.** *For  $s, t, u \geq 0$ ,  $d_g$  satisfies:*

- (i)  $d_g(s, t) \geq 0$ ,
- (ii)  $d_g(s, t) = d_g(t, s)$ ,
- (iii)  $d_g(s, u) \leq d_g(s, t) + d_g(t, u)$ .

*Proof.* Clearly,  $d_g(s, t) = d_g(t, s)$  and  $d_g(s, t) \geq 0$ , so we have to focus our energies to prove the third property. Let  $s, t, u \geq 0$  and note that:

$$d_g(s, t) + d_g(t, u) = g(s) + g(t) - 2m_g(s, t) + g(t) + g(u) - 2m_g(t, u).$$

We add a zero and group the terms that are of our interest,

$$\begin{aligned} d_g(s, t) + d_g(t, u) &= g(s) + g(u) - 2m_g(s, u) + 2m_g(s, u) + 2g(t) \\ &\quad - 2m_g(s, t) - 2m_g(t, u), \\ d_g(s, t) + d_g(t, u) &= d_g(s, u) + 2[m_g(s, u) + g(t) - m_g(s, t) - m_g(t, u)] \end{aligned}$$

Then we shift our attention to prove that:

$$m_g(s, u) + g(t) - m_g(s, t) - m_g(t, u) \geq 0,$$

in order to do so, without loss of generality, suppose that  $s < t < u$ . Then we can divide the interval  $[s, u]$  into the intervals  $[s, t]$  and  $[t, u]$ . Observe that they only intersect on  $t$ . It is a well known fact that if  $A \subset B$  then  $\inf(B) \leq \inf(A)$ . However in our case, it is true that

$$\inf_{r \in [s, u]} g(r) = \inf_{r \in [s, t]} g(r) \quad \text{or} \quad \inf_{r \in [s, u]} g(r) = \inf_{r \in [t, u]} g(r).$$

To see that this statement is true, note that the element  $\xi$  such that  $g(\xi) = \inf_{[s, u]} g(r)$  belongs to the interval  $[s, u]$ . As we established before,  $[s, u] = [s, t] \cup [t, u]$ . Then  $\xi$  belongs to one (or both) of the intervals. Since  $g(\xi)$  is the lowest value that  $g$  takes on the interval  $[s, u]$  it must be the lowest value it can take on any other sub interval. Let us assume that  $m_g(s, u) = m_g(s, t)$ . Then, we see that:

$$m_g(s, u) + g(t) - m_g(s, t) - m_g(t, u) = g(t) - m_g(t, u).$$

And from (1.4), we have that  $g(t) - m_g(t, u) \geq 0$ . Then we can conclude that:

$$d_g(s, t) + d_g(t, u) \geq d_g(s, u).$$

■

The properties that  $d_g$  satisfies make it a pseudometric. That is, we cannot guarantee that if  $d_g(s, t) = 0$  then  $s = t$ . Therefore, we need to change the space in which we work if we want to find our metric space.

In order to mold the space into something more useful for our purposes, we introduce an equivalence relation, where  $s \sim t$  if and only if  $d_g(s, t) = 0$ . We can see that if we were to work only with the equivalence classes of this equivalence relation, we would have a space where  $d_g$  is a metric. Therefore, let  $\mathcal{T}_g$  be the quotient space<sup>3</sup>

$$\mathcal{T}_g = [0, \infty) / \sim .$$

It is clear that  $d_g$  still satisfies the properties that we had seen before in this new space, and furthermore, if we have  $s, t \in \mathcal{T}_g$  and  $d_g(s, t) = 0$  then  $s = t$ , therefore,  $d_g$  is a metric on  $\mathcal{T}_g$ .

We finally have our metric space, but the "tree form" is still unclear.

### 1.3.2 Building a real tree

Now we have to build a real tree from our metric space, to do so, let us introduce a new function,  $p_g : [0, \infty) \rightarrow \mathcal{T}_g$ , the canonical projection<sup>4</sup>.

We set  $\rho := p_g(0)$ . Let  $\zeta > 0$  be the supremum of the support of  $g$ . We have  $p_g(t) = \rho$  for every  $t \geq \zeta$ . In particular,  $\mathcal{T}_g = p_g([0, \zeta])$  is compact under the quotient topology.

The combination of all the previously presented ingredients allows us to introduce the following theorem.

---

<sup>3</sup> The quotient space is defined by the equivalence relation, the elements of this new space are the equivalence classes.[14].

<sup>4</sup>The canonical projection is a mapping that takes an element of some space to its equivalence class under some equivalence relation [16]. In this case we will be using  $\sim$  the equivalence relation presented before.

**Theorem 1.1** (Le Gall [8]). *The metric space  $(\mathcal{T}_g, d_g)$  is a real tree. We will view  $(\mathcal{T}_g, d_g)$  as a rooted real tree with root  $\rho = p_g(0)$ .*

To improve our understanding of the previous theorem we are going to present its proof and then we can give an example of how the theorem works.

Before we begin the proof, a key ingredient in the proof of Theorem 1.1 is the following lemma.

**Lemma 1.1** (Le Gall [8]). *Let  $s_0 \in [0, \zeta)$ . For any real  $r \geq 0$ , denote by  $\bar{r}$  the unique element of  $[0, \zeta)$  such that  $r - \bar{r}$  is an integer multiple of  $\zeta$ . Set*

$$\tilde{g}(s) = g(s_0) + g(\overline{s_0 + s}) - 2m_g(s_0, \overline{s_0 + s}),$$

*for every  $s \in [0, \zeta]$ , and  $\tilde{g}(s) = 0$  for  $s > \zeta$ , and  $m_g$  is defined as in (1.4). Then, the function  $\tilde{g}$  is continuous with compact support and satisfies  $\tilde{g}(0) = 0$ , so that we can define  $\mathcal{T}_{\tilde{g}}$ .*

*Furthermore, for every  $s, t \in [0, \zeta]$ , we have*

$$d_{\tilde{g}}(s, t) = d_g(\overline{s_0 + s}, \overline{s_0 + t}),$$

*and there exists a unique isometry  $R$  from  $\mathcal{T}_{\tilde{g}}$  onto  $\mathcal{T}_g$  such that for every  $s \in [0, \zeta]$ ,*

$$R(p_{\tilde{g}}(s)) = p_g(\overline{s_0 + s}).$$

This lemma gives us a new function  $\tilde{g}$  that codifies a new tree. From Theorem 1.1, we see that the tree codified by  $\tilde{g}$  coincides with the tree  $\mathcal{T}_g$  re-rooted at  $p_g(s_0)$ . Thus the lemma provide us with the function that codes the tree re-rooted at an arbitrary vertex. We refer to [8] for the proof of Lemma 1.1.

## Proof of Theorem 1.1

We want to prove that  $(\mathcal{T}_g, d_g)$  is a real tree. According to Definition 1.1 we have to verify that  $(\mathcal{T}_g, d_g)$  is a metric space and satisfies properties (R1) and (R2). In Proposition 1.1 we already proved that  $(\mathcal{T}_g, d_g)$  is a metric space. We are left to prove that properties (R1) and (R2) are satisfied.

In order to prove properties (R1) and (R2), we need to establish some definitions. These definitions will link the characterization given by the theorem to the tree attributes that we presented in Section 1.1.

To prove (R2), we are going to use a contradiction argument that will use that certain sets are open and disjoint. So after our definitions that link tree concepts to our function  $g$  we present that fact.

After all of this preamble we proceed to the proof of properties (R1) and (R2). So, let us begin with some notation.

**Hierarchy.** For  $\sigma, \sigma' \in \mathcal{T}_g$ , we set  $\sigma \preceq \sigma'$  if and only if

$$d_g(\sigma, \sigma') = d_g(\rho, \sigma') - d_g(\rho, \sigma).$$

If  $\sigma = p_g(s)$  and  $\sigma' = p_g(t)$ , it follows from our definitions that  $\sigma \preceq \sigma'$  if and only if  $m_g(s, t) = g(s)$ .

It is immediate to verify that this defines a partial order on  $\mathcal{T}_g$ .

**Paths.** For any  $\sigma_0, \sigma \in \mathcal{T}_g$ , we set

$$\llbracket \sigma_0, \sigma \rrbracket := \{\sigma' \in \mathcal{T}_g : d_g(\sigma_0, \sigma) = d_g(\sigma_0, \sigma') + d_g(\sigma', \sigma)\}.$$

That is, we define the path between two elements as the set of points that describe the shortest path. If  $\sigma = p_g(s)$  and  $\sigma' = p_g(t)$ , then it is easy to verify that  $\llbracket \rho, \sigma \rrbracket \cap \llbracket \rho, \sigma' \rrbracket = \llbracket \rho, \gamma \rrbracket$ , where  $\gamma = p_g(r)$ , where  $r$  is any time at which the minimum of  $g$  between  $s$  and  $t$  is achieved. We then write  $\gamma = \sigma \wedge \sigma'$ .

The next lemma will be used in proving (R2). We set  $\mathcal{T}_g[\sigma] := \{\sigma' \in \mathcal{T}_g : \sigma \preceq \sigma'\}$ , that is,  $\mathcal{T}_g[\sigma]$  are all the **descendants** of  $\sigma$ .

**Lemma 1.2.** *If  $\mathcal{T}_g[\sigma] \neq \{\sigma\}$  ( $\sigma$  is not a leaf) and  $\sigma \neq \rho$ , then  $\mathcal{T}_g \setminus \mathcal{T}_g[\sigma]$  and  $\mathcal{T}_g[\sigma] \setminus \{\sigma\}$  are two nonempty disjoint open sets.*

*Proof.* They are clearly nonempty, from the limitations that we set at the construction of the sets.

To see that  $\mathcal{T}_g \setminus \mathcal{T}_g[\sigma]$  is open, let  $s$  be such that  $p_g(s) = \sigma$ , if we remember that  $p_g(t)$  is continuous and if we note that  $\mathcal{T}_g[\sigma]$  is the image under  $p_g$  of the compact set  $\mathcal{C} := \{u \in [0, \zeta] : m_g(s, u) = g(s)\}$ , then since the complement of  $\mathcal{C}$  is open we have that  $\mathcal{T}_g \setminus \mathcal{T}_g[\sigma]$  is open.

The set  $\mathcal{T}_g[\sigma] \setminus \{\sigma\}$  is open because if  $\sigma' \in \mathcal{T}_g[\sigma]$  and  $\sigma' \neq \sigma$ , it easily follows from our definitions that the open ball centered at  $\sigma'$  with radius  $d_g(\sigma, \sigma')$  is contained in  $\mathcal{T}_g[\sigma] \setminus \{\sigma\}$ , and then  $\mathcal{T}_g[\sigma] \setminus \{\sigma\}$  is open. ■

**Proof of property (R1).**

Let us fix  $\sigma_0$  and  $\sigma$  in  $\mathcal{T}_g$ . We have to prove existence and uniqueness of the mapping  $f_{\sigma_0, \sigma}$ . By using the Lemma 1.1 with  $s_0$  such that  $p_g(s_0) = \sigma_0$ , we may assume that  $\sigma_0 = \rho$ . If  $\sigma \in \mathcal{T}_g$  is fixed, we have to prove that there exists a unique isometric mapping

$$f = f_{\rho, \sigma} : [0, d_g(\rho, \sigma)] \rightarrow \mathcal{T}_g,$$

such that  $f(0) = \rho$  and  $f(d_g(\rho, \sigma)) = \sigma$ .

Let  $s \in p_g^{-1}(\{\sigma\})$ , so that  $g(s) = d_g(\rho, \sigma)$ . Now, we set for every  $a \in [0, d_g(\rho, \sigma)]$ :

$$v(a) = \inf\{r \in [0, s] : m_g(r, s) = a\}.$$



The function  $v$  gives us the smallest real such that the infimum of  $g$  in the interval  $[r, s]$  equals to the value we input to the function. Note that  $g(v(a)) = a$ .

Let us define the function  $f$  such that  $f(a) = p_g(v(a))$ . We have  $f(0) = \rho$  and  $f(d_g(\rho, \sigma)) = \sigma$ ; to see this, let us note that:

From the definition of  $v$ , we have that  $m_g(v(g(s)), s) = g(s)$ , then if we compute  $d_g(v(g(s)), s)$  we can see that

$$\begin{aligned} d_g(v(g(s)), s) &= g(v(g(s))) + g(s) - 2m_g(v(g(s)), s), \\ &= g(s) + g(s) - 2g(s), \\ d_g(v(g(s)), s) &= 0. \end{aligned}$$

Then  $v(g(s)) \sim s$  and therefore  $p_g(v(g(s))) = p_g(s) = \sigma$ .

It is also easy to verify that  $f$  is an isometry: If  $a, b \in [0, d_g(\rho, \sigma)]$  with  $a \leq b$ , and noting that  $v(a) \leq v(b)$  then we can see that  $m_g(v(a), v(b)) = a$ , and so

$$d_g(f(a), f(b)) = g(v(a)) + g(v(b)) - 2a = b - a.$$

Which proves that  $f$  is an isometry. To get uniqueness, suppose that  $\tilde{f}$  is an isometric mapping satisfying the same properties as  $f$ . Then, if  $a \in [0, d_g(\rho, \sigma)]$ ,

$$d_g(\tilde{f}(a), \sigma) = d_g(\rho, \sigma) - a = d_g(\rho, \sigma) - d_g(\rho, \tilde{f}(a)).$$

Therefore,  $\tilde{f}(a) \preccurlyeq \sigma$ . Recall that  $\sigma = p_g(s)$ , and choose  $t$  such that  $p_g(t) = \tilde{f}(a)$ . We can note that  $g(t) = d_g(\rho, p_g(t)) = a$ .

Since  $\tilde{f}(a) \preccurlyeq \sigma$  from the definitions presented at the beginning of this proof, we have that  $g(t) = m_g(t, s)$ . And from what we saw before, we know that  $a = g(v(a)) = m_g(v(a), t)$  and thus  $d_g(t, v(a)) = 0$ , then  $v(a) \sim t$  and finally,

$$\tilde{f}(a) = p_g(t) = p_g(v(a)) = f(a).$$

By presenting this argument, we establish the uniqueness of  $f$ , thereby concluding the proof of (R1).

**Proof of property (R2).**

Let  $q : [0, 1] \rightarrow \mathcal{T}_g$  be a continuous injective mapping, and remember that we aim to prove that

$$q([0, 1]) = f_{q(0), q(1)}([0, d_g(q(0), q(1))]).$$

From Lemma 1.1 we can assume that  $q(0) = \rho$ , and let us define  $\sigma = q(1)$ . Then we have that  $f_{0, \sigma}([0, d_g(\rho, \sigma)]) = \llbracket \rho, \sigma \rrbracket$ .

Let us use a contradiction argument to prove that  $\llbracket \rho, \sigma \rrbracket \subset q([0, 1])$ . Let us suppose that  $\eta \in \llbracket \rho, \sigma \rrbracket \setminus q([0, 1])$ , and in particular,  $\eta \neq \rho, \sigma$ . Then  $q([0, 1])$  is contained in the union of the disjoint open sets  $\mathcal{T}_g \setminus \mathcal{T}_g[\eta]$  and  $\mathcal{T}_g[\eta] \setminus \{\eta\}$ , since  $q(0) = \rho$ , and since  $\mathcal{T}_g[\eta]$  contains the descendants of  $\eta$  we have that  $q(0) \in \mathcal{T}_g \setminus \mathcal{T}_g[\eta]$  and  $q(1) = \sigma \in \mathcal{T}_g[\eta] \setminus \{\eta\}$ . This contradicts the fact that  $q([0, 1])$  is connected.

Then all we are left to prove is that  $q([0, 1]) \subset \llbracket \rho, \sigma \rrbracket$ . Similarly, let us suppose that there exists  $a \in (0, 1)$  such that  $q(a) \notin \llbracket \rho, \sigma \rrbracket$ . Let  $\eta = q(a)$  and define  $\gamma = \sigma \wedge \eta$ . On one hand, since  $\llbracket \rho, \eta \rrbracket$  is the ancestral line of  $\eta$  and therefore every ancestor must be contained in it, then  $\gamma \in \llbracket \rho, \eta \rrbracket$ ; on the other hand, we have that  $\gamma \in \llbracket \eta, \sigma \rrbracket$  because, from our definitions,  $\gamma = p_g(r)$  such that  $g(r) = m_g(a, s)$ . And in consequence:

$$d_g(\eta, \gamma) = g(a) + g(r) - 2m_g(a, r) = g(a) - g(r),$$

$$d_g(\gamma, \sigma) = g(r) + g(s) - 2m_g(s, r) = g(s) - g(r).$$

Then, if we remember that  $d_g(\eta, \sigma) = g(a) + g(s) - 2m_g(a, s)$ , and sum the expressions above, we have that:

$$d_g(\eta, \sigma) = d_g(\eta, \gamma) + d_g(\gamma, \sigma),$$

and therefore:  $\gamma \in \llbracket \eta, \sigma \rrbracket$ .

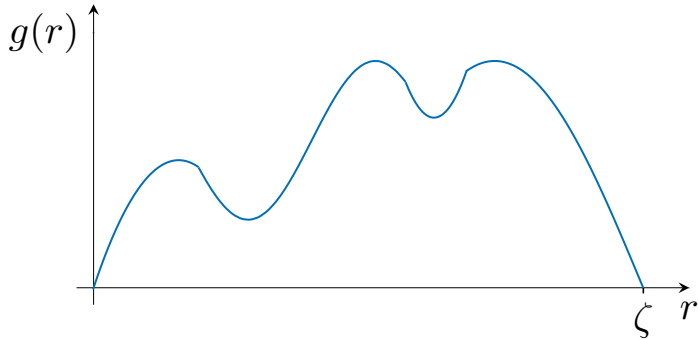
From the first part of the proof of (R2), we have that  $\gamma \in q([0, a])$ , and using Lemma 1.1 to set  $\eta$  as the root we also have that  $\gamma \in q([a, 1])$ . Since  $q$  is injective, this is only possible if  $\gamma = q(a) = \eta$ , which contradicts the fact that  $\eta \notin \llbracket \rho, \sigma \rrbracket$ . And thus finishing the proof of the theorem.

### Example

After proving Theorem 1.1 we have a way of characterising real trees. In the proof we presented how we link the concepts that we presented in Section 1.1 to attributes of the contour function  $g$ .

In the remainder of the section we will exploit all the things that we have presented so far to finally construct a real tree.

Let us have a function  $g$  as the one presented in Figure 1.5, this function is continuous.



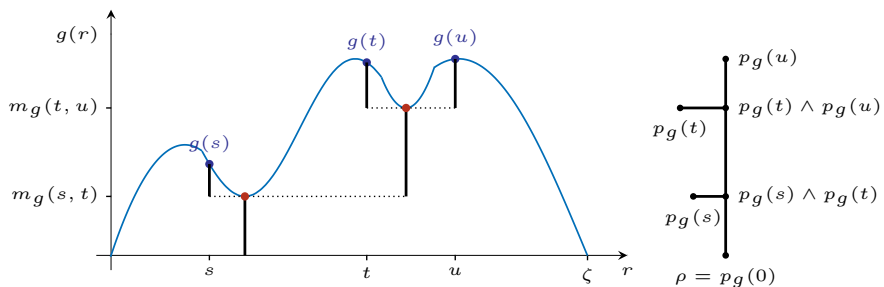
**Figure 1.5.** A contour function  $g$ .

We have to remember that we are not interested in the analytic expression of the function. Our main interest is to understand how to build a tree from it. Attempting to represent all the elements of the

tree is an impossible task because the number of nodes is uncountable, as for every  $t \in [0, \zeta]$ , we have a node on the tree. However, we can construct a subtree, which will provide us with a general understanding of the entire tree.

Let us choose three points  $s, t, u \in [0, \zeta]$ , as in Figure 1.6. One thing that we want to discover is the "tree form". In order to find it, we need to detect the most recent common ancestors of our highlighted elements,  $u, s$  and  $t$ . We will use the function  $m_g(s, t)$  (defined in (1.4)) between our elements to obtain their most recent common ancestor.

Note that our selection of elements was not arbitrary. See Figure 1.6, if we had chosen two elements situated on the same side of a "valley" in the function, one of them would be their most recent common ancestor. Consequently, this would result in redundant information at the moment of building our subtree.



**Figure 1.6.** The subtree that we build using the function  $g$ . The red dots are the most recent common ancestor between  $t$  and  $u$  and  $t$  and  $s$ . The thick lines represent the distance between every element and their ancestors.

We are interested in the distance between our elements and their most recent common ancestor. The point  $r$  where  $g(r) = m_g(s, t)$  will

be the most recent common ancestor. These points represent points where a tree branch "divides". To build the tree all we need to do is paste the branches, to see this process, look at Figure 1.6.

As we saw with the definitions that preceded the theorem, the function  $p_g$  has a vital importance in the construction of the tree, this is reflected in the labeling of the elements of the resulting tree, since the nodes on the tree are the equivalence classes, not the original elements. Remember that we introduce the notation  $u \wedge v$  to refer to the most recent common ancestor of  $u$  and  $v$ . Now the depiction of the tree is a little bit different from what we had chose before, but it is really good in representing what goes on at the moment of building the tree.

Now, it is clear that the tree that we build is not the whole tree, actually it is called the "reduced tree" with respect to  $\{s, t, u\}$  but it helped to understand how we build the trees.

**Observation.** The amount of "branches" in the tree given by the contour function  $g$  is correlated to the amount of local minimums of the contour function. See Figure 1.4

Thanks to Theorem 1.1 we have found a way to build real trees. Well, it turns out that the codification of the trees via the contour function  $g$  helps us to simplify working with the Gromov-Hausdorff distance between them.

**Lemma 1.3** (Le Gall [8]). *Let  $g$  and  $g'$  be two continuous functions with compact support from  $[0, \infty)$  into  $[0, \infty)$ , such that  $g(0) = g'(0) = 0$ . Then,*

$$d_{GH}(\mathcal{T}_g, \mathcal{T}_{g'}) \leq 2\|g - g'\|_{\infty}.$$

where  $\|\cdot\|_{\infty}$  stands for the supremum norm.

*Proof.* As we said when we introduced the Gromov-Hausdorff distance, we will use formula (1.3) to characterize it. Let us construct a

correspondence between  $\mathcal{T}_g$  and  $\mathcal{T}_{g'}$  by setting

$$\mathcal{R} = \{(\sigma, \sigma') : \sigma = p_g(t) \text{ and } \sigma' = p_{g'}(t) \text{ for some } t \geq 0\}.$$

To prove our lemma we need to bound the distortion of the correspondence.

Let  $(\sigma, \sigma') \in \mathcal{R}$  and  $(\eta, \eta') \in \mathcal{R}$ . From our definition of  $\mathcal{R}$  we can find  $s, t \geq 0$  such that  $p_g(s) = \sigma$ ,  $p_{g'}(s) = \sigma'$  and  $p_g(t) = \eta$ ,  $p_{g'}(t) = \eta'$ . Recall that

$$\begin{aligned} d_g(\sigma, \eta) &= g(s) + g(t) - 2m_g(s, t), \\ d_{g'}(\sigma', \eta') &= g'(s) + g'(t) - 2m_{g'}(s, t). \end{aligned}$$

If we compute  $|d_g(\sigma, \eta) - d_{g'}(\sigma', \eta')|$  and apply the triangle inequality, we will have that

$$\begin{aligned} &|d_g(\sigma, \eta) - d_{g'}(\sigma', \eta')| \\ &= |g(s) - g'(s) + g(t) - g'(t) + 2[m_{g'}(s, t) - m_g(s, t)]| \\ &\leq |g(s) - g'(s)| + |g(t) - g'(t)| + 2|m_{g'}(s, t) - m_g(s, t)|. \end{aligned}$$

Recalling the definition of  $m_g(s, t)$ , presented in (1.4), and using the fact that  $|g(s) - g'(s)| \leq \|g - g'\|_\infty$ , we obtain

$$|d_g - d_{g'}| \leq 4\|g - g'\|_\infty.$$

Thus,  $\text{dis}(\mathcal{R}) \leq 4\|g - g'\|_\infty$ . Finally, using the definition of  $d_{GH}$  presented in (1.3) we obtain

$$d_{GH}(\mathcal{T}_g, \mathcal{T}_{g'}) \leq 2\|g - g'\|_\infty.$$

■

We have extensively explored techniques for investigating real trees, but to define the Continuum Random Tree, we still need to incorporate randomness. The only method that we presented to build real trees is via the contour function  $g$ . Then, in order to introduce randomness to our trees we need to use a random contour function. Thus, we will make a little deviation from what we have been talking about and make a brief exploration of Brownian Motion and Brownian Excursions.

## 1.4 Brownian Motion

Let us begin this section with the definition of a Brownian Motion (this definition was obtained in [13]).

**Definition 1.2.** *A real-valued stochastic process  $\{B(t) : t \geq 0\}$  is called a **Brownian motion** with start in  $x \in \mathbb{R}$  if the following holds:*

- $B(0) = x$ ,
- the process has **independent increments**, i.e. for all times  $0 \leq t_1 \leq t_2 \leq \dots \leq t_n$  the increments  $B(t_n) - B(t_{n-1}), B(t_{n-1}) - B(t_{n-2}), \dots, B(t_2) - B(t_1), B(t_1) - x$ , are independent random variables,
- for all  $t \geq 0$  and  $h > 0$ , the increments  $B(t+h) - B(t)$  are normally distributed with expectation zero and variance  $h$ ,
- almost surely, the function  $t \mapsto B(t)$  is continuous.

We say that  $\{B(t) : t \geq 0\}$  is a **standard Brownian motion** if  $x = 0$ .

As we said before, the only place where we could possibly use this object would be as a contour function. But we have a couple of

challenges, from what we saw in the last section, our contour function must be positive, and start on zero. Let us see why a regular Brownian Motion would not be acceptable as a codifying function.

First of all, if we do not work with standard Brownian motions we would not even be able to satisfy the initial characteristic. Now, let us talk about the behaviour of a Brownian Motion. Looking at the third property in the Definition 1.2 and if we take  $t = 0$  we can see that:

$$B(h) - B(0) = B(h) \sim \mathcal{N}(0, h).$$

This is just the "first step" of our Brownian motion, however, this is enough to see that we cannot guarantee that the image of  $B(s)$ ,  $s \geq 0$  will be positive, as the probability of  $B(h)$  being negative is  $1/2$ .

Then we have a clear problem, to work around this issue we just have to force the Brownian Motion to be positive in a certain closed domain.

To do so we are going to use something called excursion theory. The idea of an excursion is quite simple. We call an excursion  $e$  of the Brownian motion  $B$ , a process  $(e(t), t \geq 0)$  such that there exists  $L < R$  with

$$e(t) = B([L + t] \wedge R)$$

and for  $s \in [L, R]$ ,  $B(s) = 0$  if and only if  $s = L$  or  $R$ . That is,  $e$  is the piece of  $B$  between times  $L$  and  $R$ , which are two consecutive zeros of  $B$ , see [3].

The tools that we would need to make our intuition more rigorous are beyond the scope of this thesis. Let us discuss why this solves our issue with the standard Brownian motion. For once we have a closed domain where the sign of  $B(t)$  does not change, therefore in the case that  $B(t)$  is negative between  $L$  and  $R$  we can simply shift our attention to  $B'(t) := -B(t)$  this new process is also a Brownian Motion and since



$B(t) < 0$  for  $t \in [L, R]$  then  $B'(t) > 0$  in the same interval. Then without loss of generality we can assume  $e(t) > 0$  for  $t \in [0, \zeta]$  where  $\zeta = R - L$ .

Then we have the perfect contour function in  $e(t)$ . Finally, we need to define the **normalized Brownian excursion** ( $e(t), 0 \leq t \leq 1$ ). This is simply the Brownian excursion conditioned to have length 1. At last we can present the following definition.

**Definition 1.3** (Le Gall [8]). *The Continuum Random Tree (CRT) is the random real tree  $\mathcal{T}_e$  coded by the normalized Brownian excursion.*

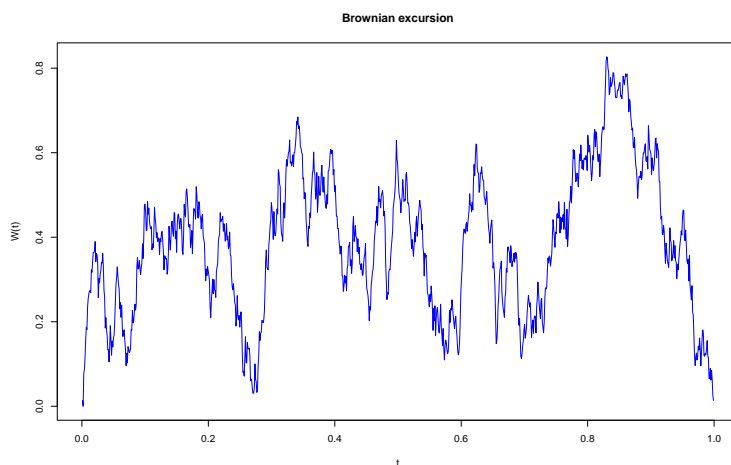
To conclude this chapter we can answer the question, **why is this particular tree exciting?** It is clear that the behaviour of  $g$  directly affects  $\mathcal{T}_g$ . However as we stated before, the amount of "branches" that our resulting tree has is directly correlated to the amount of local minimums of the function  $g$ . This is, for every "valley" on the function  $g$  the tree splits a branch.

Now to make the point clearer, consider Figure 1.7, which depicts a normalized Brownian excursion<sup>5</sup>. If we look closely at it we can see why a tree codified by this function would be interesting, there are a lot of *valleys* in this function, therefore, the form of the resulting tree will not be so simple as the one we saw in the example.

Having a look at this fact helps us to see why this object results so interesting, and we have not even talked about where else we can find the CRT. In the following chapters we will see how this kind of tree turns out to be even more important than what we can imagine at the moment.

---

<sup>5</sup>The code used to plot the excursion was based on the codes showed in [19].



**Figure 1.7.** Brownian excursion, which corresponds to the contour function of a CRT.

## Chapter 2

# Construction via Galton Watson Trees

In the last chapter we presented real trees, and gave a characterization of them. In this chapter we will present the Galton Watson process. This model is often used to describe the evolution of a discrete population of individuals. Their genealogical structure gives rise to discrete random trees that we will use to demonstrate convergence to the CRT, which is the goal of this chapter.

Before properly introducing Galton-Watson trees we have to talk about encoding discrete trees.

### 2.1 Ordered trees and their encodings

We have to set some limitations to the trees that we will use, unlike the trees that we presented at the beginning of the last chapter, we are going to be interested in the labeling of the vertices and in the position that they hold in the depiction of the tree.

The trees in which we will focus have a distinguished vertex (the root) and they are defined in such a way that the "children" of a vertex have a given planar order. This is, we will identify the children through the way they are depicted in the drawings. Indeed, we are interested in separating vertices to the left or right of any given vertex on the tree. For this we use the *Ulam-Harris* labels, which are given by the elements of

$$\mathbb{U} := \bigcup_{n=0}^{\infty} \mathbb{N}^n,$$

with the convention that  $\mathbb{N}^0 := \{\emptyset\}$ .

An element of  $\mathbb{U}$  is a sequence  $u = (u^1, \dots, u^n)$  of elements of  $\mathbb{N}$ , and we set  $|u| = n$ . We will also refer to elements of  $\mathbb{U}$  as **words**. We can define the concatenation of two words,  $u = (u^1, \dots, u^n)$  and  $v = (v^1, \dots, v^m)$  as  $uv = (u^1, \dots, u^n, v^1, \dots, v^m)$

Also, let us define the map  $\pi : \mathbb{U} \setminus \{\emptyset\} \rightarrow \mathbb{U}$  where  $\pi(u^1, \dots, u^n) = (u^1, \dots, u^{n-1})$ . We can say that  $\pi(u)$  is the "parent" of  $u$ . Now, we can properly define what is a finite rooted ordered tree.

**Definition 2.1.** *A finite rooted ordered tree  $\mathbf{t}$  is a finite subset of  $\mathbb{U}$  such that:*

- (i)  $\emptyset \in \mathbf{t}$ .
- (ii) If  $u \in \mathbf{t} \setminus \{\emptyset\}$  then  $\pi(u) \in \mathbf{t}$ .
- (iii) For every  $u \in \mathbf{t}$ , there exists an integer  $k_u(\mathbf{t}) \geq 0$  such that, for every  $j \in \mathbb{N}$ ,  $uj \in \mathbf{t}$  if and only if  $1 \leq j \leq k_u(\mathbf{t})$ .

The number  $k_u(\mathbf{t})$  is interpreted as the "number of children" of  $u$  in  $\mathbf{t}$ .

Let us understand this definition, the first condition simply states that we need an starting vertex, which will be the root of the given tree. The second condition states that for every vertex on the tree, except for

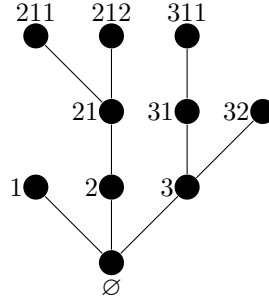
the root, the parent of that vertex must be on the tree, and in this way we ensure that the tree is connected. And finally, the last condition limits the amount of children that each vertex has, and in that way we force the tree to be locally finite and it also provides an order for its children.

Let us propose an order for the elements on the tree. The trees are defined as subsets of  $\mathbb{U}$ . We should define an order on  $\mathbb{U}$ . This order is known as the **lexicographical** order  $\triangleleft$  on  $\mathbb{U}$ , which is defined as follows:

Let  $u = u_1u_2 \dots u_k$  and  $v = v_1v_2 \dots v_l$  be two non-empty words. Define  $m = k \vee l$ , and  $\hat{u} := u0_k$  and  $\hat{v} := v0_l$ , where  $0_i = (00 \dots 0)$  is a word of length  $m - i$  with  $i \in \{k, l\}$ . Then, we will say that  $u \triangleleft v$  if and only if exists  $j \in \{1, \dots, m\}$  such that  $\hat{u}^j < \hat{v}^j$ , with the convention that  $\emptyset \triangleleft 1$ .

The interest in proposing an ordering for the elements on the tree comes from the fact that we want to find a good way to explore the tree. There are several options, and every one of them has its uses; however, for this particular work, we use the **depth-first search**. This corresponds to the exploration of the tree following the lexicographical ordering. See Figure 2.1 where the depth-first search gives the order  $(\emptyset, 1, 2, 21, 211, 212, 3, 31, 311, 32)$ .

We have defined the objects that are of interest to us; nevertheless, it is still unclear how to work with them in convenient ways. As we did in the previous chapter, we want to use functions that resume the information of the tree. We have to remark, than in comparison with the previous chapter, we will need more ways to codify trees. Without further ado let us present the codifying functions.



**Figure 2.1.** Example of a rooted tree with Ulam-Harris labels.

## Encoding trees

Working directly with graphs is not necessarily desirable in our context, that is why we need codifying functions. However, there are a lot of options from which to choose. Our criteria for choosing one will rely on how easy will it be for us to work with it and it will also be heavily influenced by the results presented in the previous chapter. So, in hopes of finding this function, let us begin our exploration of codifying functions.

Let  $\mathbf{T}$  be the set of finite rooted ordered trees. For  $\mathbf{t} \in \mathbf{T}$  with  $n$  vertices, let  $v_0, v_1, \dots, v_{n-1}$  be the vertices listed in lexicographical order, we define the **height function** of  $\mathbf{t}$  as  $(h_{\mathbf{t}}(i), 0 \leq i \leq n-1)$ , where

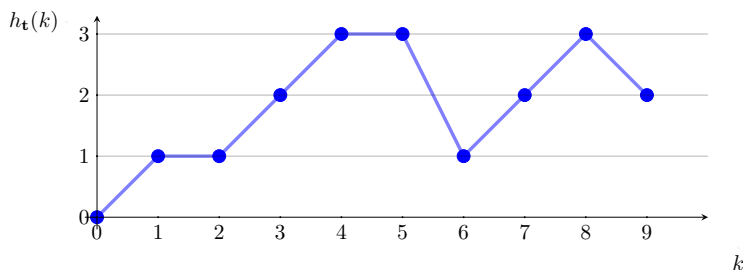
$$h_{\mathbf{t}}(i) := |v_i|, \quad 0 \leq i \leq n-1.$$

Recall that  $|\cdot|$  represents the generation of the vertex.

We can also define this function using the graph distance on  $\mathbf{t}$ ,  $d_{\mathbf{t}}$ , as seen in Chapter 1. We obtain that the equivalent expression for the height is:

$$h_{\mathbf{t}}(i) := d_{\mathbf{t}}(v_0, v_i), \quad 0 \leq i \leq n-1.$$

The height function is our first candidate to be a codifying function. In Figure 2.2 we can see how it codifies the tree shown in Figure 2.1.



**Figure 2.2.** Height function of tree in Figure 2.1, with individuals listed in lexicographical order.

We expect that the codifying function reflects in some capacity the attributes that we can see on the depiction of the tree. Let us explore how these attributes manifest on the behaviour of  $h_t$ .

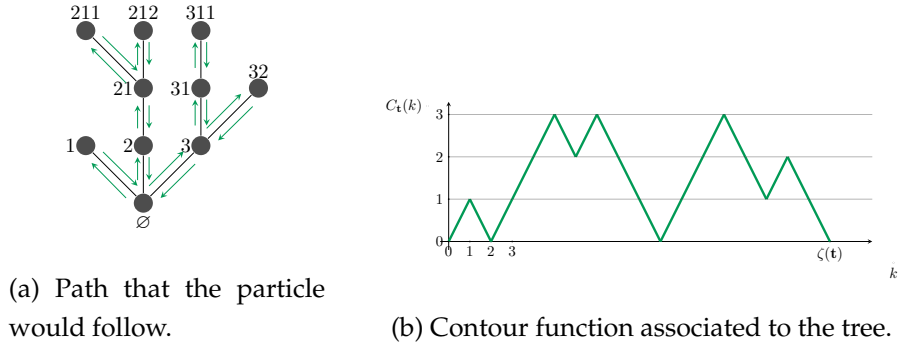
To simplify the discussion, let us introduce some terminology. We will refer to the set of all descendants of a vertex with  $h_t(i) = 1$  as the component  $i$ . This component represents the subtree formed by all descendants of vertex  $i$ . Looking at Figures 2.1 and 2.2 we can see that every time we end the exploration of a component we return to a point where  $h_t(k) = 1$ .

We can also see that when a vertex (say  $v_k$ ) does not have descendants we have two options, either we stay at the same "height" in  $k + 1$  (the next vertex is a "brother" of the root of the subtree we just visited), or we go to a lower "height" in  $k + 1$  (the height can only decrease if we do not have any vertex to explore in the remainder of the subtree in which vertex  $v_k$  is).

The height function is a good codifying function, as we will see later on this chapter. However, the function that characterizes the CRT

is the **contour function**, therefore we must present its discrete counter part. We can make the same simile that we did in the last chapter, it is as if we placed a blanket on top of the tree and watched it from the side. However we can also give more details about this function in this setting.

To understand why the function behaves in the ways that it does, imagine that we put a particle on the root of the tree at time  $t = 0$ , this particle will explore the tree from left to right, moving in a continuous fashion at unit speed, until all the edges have been explored and the particle comes back to the root. See Figure 2.3 to look at an example using the tree presented in Figure 2.1.



**Figure 2.3.** Example of the construction of the contour function.

The total time needed to explore the tree is  $\zeta(\mathbf{t}) := 2(\#(\mathbf{t}) - 1)$ . The value  $C_{\mathbf{t}}(s)$  of the contour function at time  $s \in [0, \zeta(\mathbf{t})]$  is the distance (on the tree) between the position of the particle at time  $s$  and the root. By convention  $C_{\mathbf{t}}(s) = 0$  if  $s \geq \zeta(\mathbf{t})$ . After presenting all the attributes of this function, we can conclude that it is equivalent to the function  $g$  introduced in the previous chapter. However, one might wonder: at what point do our discrete trees transition to real trees?



Recalling the definition of the CRT (defined in Definition 1.3), we characterize it using contour functions. Keeping in mind that the goal of this chapter is to unveil a convergence to this object using Galton-Watson trees, the convergence we are aiming to reveal must be expressed in terms of contour processes. Luckily, one can establish a bijection between contour and height processes. We will exploit this fact, elaborated upon in Section 2.3.1, to bypass direct manipulation of contour processes and instead focus on height processes. This approach will prove advantageous for our objective due to the discrete nature of the height process.

Even though the height process is useful, it has some downsides, which will appear once that we start working with random trees. Given that we are presenting the ingredients that we will use to make the following sections easier to read, we will present another way of codifying trees. Let us denote  $\mathcal{S}$  the set of all finite sequences of nonnegative integers  $m_1, \dots, m_p$  (with  $p \geq 1$ ) such that

- $m_1 + m_2 + \dots + m_i \geq i, \quad \forall i \in \{1, \dots, p-1\};$
- $m_1 + m_2 + \dots + m_p = p-1.$

Recall that  $v_0, v_1, \dots, v_{\#(\mathbf{t})-1}$  are the elements of  $\mathbf{t}$  listed in lexicographical order. The following proposition gives us a new way of codifying trees using these sets of sequences.

**Proposition 2.1.** *The map*

$$\Phi : \mathbf{t} \longrightarrow (k_{u_0}(\mathbf{t}), k_{u_1}(\mathbf{t}), \dots, k_{u_{\#(\mathbf{t})-1}}(\mathbf{t}))$$

*defines a bijection from  $\mathbf{T}$  onto  $\mathcal{S}$ .*

*Proof.* Suppose  $\#(\mathbf{t}) = n$ . Then, the sum  $k_{u_0}(\mathbf{t}) + k_{u_1}(\mathbf{t}) + \dots + k_{u_{n-1}}(\mathbf{t})$  counts the total number of children of all individuals in the tree and is

thus equal to  $n - 1$ , because  $\emptyset$  is not counted, as it is the only vertex that is not a child of any other vertex. Furthermore, if  $i \in \{0, 1, \dots, n - 2\}$ ,  $k_{u_0} + \dots + k_{u_i}$  is the number of children of  $u_0, \dots, u_i$  and thus greater than or equal to  $i$ , because in the lexicographical order, an individual is always visited before its children. It follows that  $\Phi$  maps  $\mathbf{T}$  into  $\mathcal{S}$ .

We have to prove that  $\Phi$  is injective. Suppose there exist two different trees,  $\mathbf{t}$  and  $\mathbf{t}'$ , such that  $\Phi(\mathbf{t}) = \Phi(\mathbf{t}')$ . Then, recalling Definition 2.1, the integers  $k_u(\mathbf{t})$  uniquely define a tree  $\mathbf{t}$ . Therefore, the fact that  $\mathbf{t}$  and  $\mathbf{t}'$  share them contradicts our assumption that they are different trees, since the integers  $k_u(\mathbf{t})$  define the relations between the vertices, and as we said in the beginning of Chapter 1, we can tell two graphs apart from the relations of their vertices, if two trees share this integer, they are the same graph. Thus proving that  $\Phi$  is injective.

Now let us see that for every sequence  $(m_1, \dots, m_p) \in \mathcal{S}$  there exists a tree  $\mathbf{t}$  on  $\mathbf{T}$  such that  $\Phi(\mathbf{t}) = (m_1, \dots, m_p)$ . To see that such tree exists let us build it.

Assume  $p = 1$ , then, define  $\mathcal{U}$  as  $\{\emptyset, 1, 2, \dots, m_1\}$ ,  $\mathcal{U}$  is a tree in which every element is a descendant of  $\emptyset$ . If  $p \geq 2$ , we have to make a more complex argument.

First, define  $\theta = \{1, 2, \dots, m_1\}$  and  $u_0 = \emptyset$ ,  $u_1 = 1$ . Then define the set  $\mathcal{U} = \{u_0, u_1\}$ .

For every  $j \in \{1, 2, \dots, m_2\}$  add  $u_1 j$  to  $\theta$ . Then order the set  $\theta \setminus \mathcal{U}$  in lexicographical order. Pick the minimum and name it  $u_2$ .

Add  $u_2$  to  $\mathcal{U}$ . For every  $j \in \{1, 2, \dots, m_3\}$  add  $u_2 j$  to  $\theta$ . Then order the set  $\theta \setminus \mathcal{U}$  in lexicographical order. Pick the minimum and name it  $u_3$ .

Performing this process in an inductive manner until the  $p - th$  step will give that  $\mathcal{U}$  is an ordered finite tree.

Let us explain how this algorithm works. The set  $\theta$  are the vertices

that we have discovered on each step, however they are not ordered yet. The set  $\mathcal{U}$  keeps track of which vertices we have "explored" in lexicographical order and finally it results to be a tree. Basically the choice of vertex in  $\theta \setminus \mathcal{U}$  corresponds to a *depth-first search* of the tree, that is, we follow the lexicographical order to walk on the tree. On each discovery of new vertices, we order the set again and decide where to move. This process will visit every vertex of the tree.

By definition,  $\emptyset \in \mathcal{U}$ . Given  $u \in \mathcal{U} \setminus \{\emptyset\}$ , we can recover  $\pi(u) \in \mathcal{U}$  from the way we build it, as every element is defined as  $u_i k$  for some  $i, k \in \mathbb{N}$ . The third property is fulfilled thanks to the fact that the number of descendants of every element  $u_j$  is  $m_{j+1}$ .

Therefore, for every element of  $\mathcal{S}$  there exists a tree on  $\mathbf{T}$ , thus proving that  $\Phi$  defines a bijection from  $\mathbf{T}$  onto  $\mathcal{S}$ . ■

We now have a codification using the amount of children of every vertex, but instead of this sequence, we will find it convenient to use a different sequence. To introduce it, let  $\mathbf{t} \in \mathbf{T}$ ,  $n = \#(\mathbf{t})$  and  $\Phi(\mathbf{t}) = (k_1, \dots, k_n)$ . We define:

$$x_i := \sum_{j=0}^{i-1} (k_j - 1), \quad 0 \leq i \leq n$$

which satisfies the following properties

- $x_0 = 0$  and  $x_n = -1$ .
- $x_i \geq 0$  for every  $0 \leq i \leq n - 1$ .
- $x_i - x_{i-1} \geq -1$  for every  $1 \leq i \leq n$ .

This sequence is known as a *Lukasiewicz path*. From the definition, we can observe a strong connection between these paths and the

sequences presented in Proposition 2.1. This codification differs from the one that is given by  $\Phi$  due to the transformation that the entries of the sequences undergo to construct the Łukasiewicz paths. Thanks to the strong relation between the sequences presented in the last proposition and the Łukasiewicz paths, the map  $\Phi$  induces a bijection between trees in  $\mathbf{T}$  and Łukasiewicz paths.

This sequence is useful because its particular behaviour. It keeps track of the number of known vertices left to visit. We can observe this behaviour by looking at its definition with more caution. On every step  $k$ , to obtain the  $k$ -th value of the Łukasiewicz path when we add the number of children that the vertex  $k$  had and we subtract one unit. This subtraction takes into account the fact that we have visited a new vertex, vertex  $k$ , and therefore that vertex is not longer unexplored.

A very important result is that there exists a simple relation between the Łukasiewicz path of a tree and its height function; such relation is given by the following proposition.

**Proposition 2.2** (Le Gall, Le Jan [12]). *For  $0 \leq k \leq n - 1$*

$$h_{\mathbf{t}}(k) = \# \left\{ 0 \leq i \leq k - 1 : x_i = \inf_{i \leq j \leq k} x_j \right\}. \quad (2.1)$$

After the proof of this theorem we present an example to understand how to use the formula given by the proposition.

*Proof.* We can think that this proof is divided in two steps. First we explore the behaviour of the depth-first walk as it visits different parts of the tree. On the second step we want to see how we can link subtrees of the original tree with the height function. And finally we conclude.

We want to see how the Łukasiewicz path behaves as it visits subtrees. Let us begin by working with the simplest kind of subtree, a node  $v_l$  with  $r$  children, where the children do not have descendants.

Suppose that we arrive at  $v_l$  in the  $m + 1$  step of our walk. Assume that  $x_m = \eta_0$ , with  $\eta_0 \geq 0$ , then  $x_{m+1} = \eta_0 + r - 1$ , by definition. On the next steps, we will have the following behaviour:

$$x_{m+1+j} = \eta_0 + r - 1 - j, \quad \text{for } 1 < j \leq r.$$

Then, when we arrive at step  $m + r + 1$ , that is, the moment we finish exploring the subtree, we have that the Łukasiewicz path takes the value  $\eta_0 - 1$ . Note that the new value is one less than the one that we had when we were at the root of this subtree.

Note that if we let go our assumption that the descendants of vertex  $v_l$  did not have children the value that the Łukasiewicz path would take when it finishes exploring the subtree would be the value that it had at vertex  $v_l$  minus one. This is because, for every node with children we would enter another subtree and therefore we would leave it with our Łukasiewicz path having the value that it had at this new starting vertex minus one. Therefore it would be as if we only visited a vertex without descendants.

We are now interested in exploring the behavior of the Łukasiewicz path when traversing subtrees. Our aim is to identify occurrences within the subtrees where the Łukasiewicz path repeats a value. In the earlier part of the proof, we observed that the Łukasiewicz path repeats a value upon entering and exiting a subtree. Consequently, at least, the Łukasiewicz path repeats its value at the root of a subtree. Thus, the Łukasiewicz path reaches a local minimum within an interval as many times as we reach the root of a subtree.

For any vertex  $v_k$ , its height will be the number of subtrees that we have begun exploring, but not exhausted at the step before we reach vertex  $v_k$ . Indeed for every subtree that we have not finished exploring we would have entered a new generation of the tree, then  $h_t$  would

increase its value.

As we said before, the Łukasiewicz path reaches its minimum value at every root of a subtree, therefore,

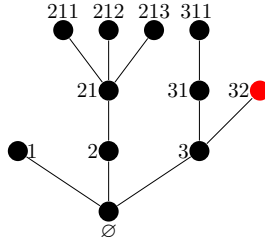
$$h_t(k) = \# \left\{ 0 \leq i \leq k-1 : x_i = \inf_{i \leq j \leq k} x_j \right\}.$$

■

Thanks to this result, if we want to show convergence of height functions we can look at Łukasiewicz paths of the trees.

To end this section about codifications, we would like to present a concrete example. We aim to show different behaviours that appear in the codification functions that only appear when the structure of the tree is more complicated.

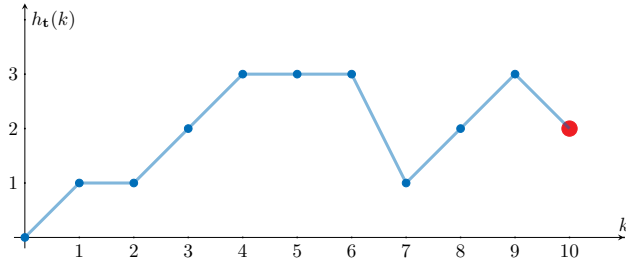
First, let us introduce our example tree in Figure 2.4, this tree is a modified version of the one that we presented before.



**Figure 2.4.** Tree with labels given by the lexicographical order. Highlighted, vertex vertex 32; is the 10th vertex in lexicographical order.

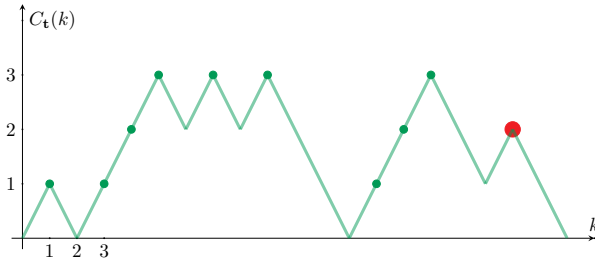
To begin exploring the different codifying functions, let us present the height function of the tree presented in Figure 2.4.

One attribute that is quite significant about the height function is that it does not 'return' to zero when we finish exploring the tree. Also,



**Figure 2.5.** Height function of the tree presented in Figure 2.4, the highlighted vertex is vertex 32 of the tree. Observe that  $h_t(10) = 2$ .

it is important to highlight the change in value that we observe when moving from the 6th to the 7th vertex (in lexicographical order), we have a drop of more than 1 unit, something that does not happen in any of the other codifying functions. The next codifying function that we present is the contour function. See Figure 2.6.

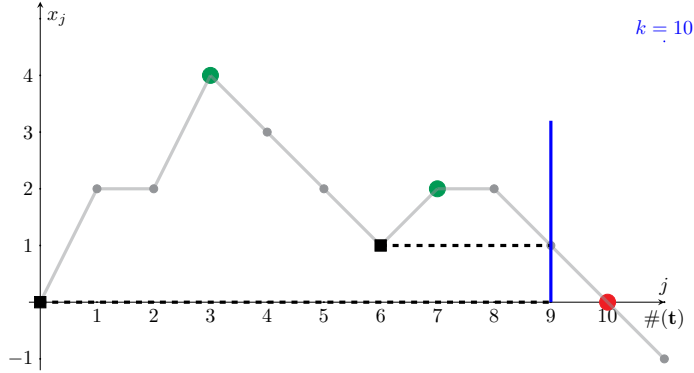


**Figure 2.6.** The contour function of the tree presented in Figure 2.4. The highlighted vertex once again corresponds to vertex 32.

On the contour function we can observe that the values do not change in more than one unit between units of time, also, it is important to note that this function does return to zero once we finish

exploring the tree.

Finally, we present in Figure 2.7 the Łukasiewicz path associated to the tree presented in Figure 2.4. Also, it provides some cues to understand the formula given by Proposition 2.2.



**Figure 2.7.** Łukasiewicz path associated with the tree.

**Figure 2.8.** We aim to determine the height of the 10th vertex (in lexicographical order), as shown in Figure 2.4. In Figure 2.7, the blue vertical line represents the upper limit of the interval that is used to compute the formula given in (2.1). The two instances in which the condition given by the formula is fulfilled are marked by the black squares. Also, observe that if you take any other two moments of the walk, let us say 3 and 7, we do not count them because there is a smaller value in the remaining interval up to 10. Consequently, we corroborate that  $h_t(10) = 2$ .

## 2.2 Defining a Galton-Watson tree

After our little dive into the codification of ordered trees, we present the definition of a Galton-Watson tree. In this type of random tree, the



randomness comes from the amount of children each vertex begets. The number of children depends on the law of reproduction of the individual, which is shared by all members of the population. This is often called the *offspring distribution*.

Let  $\mu$  be a probability measure on  $\mathbb{Z}_+$  such that

$$\sum_{k=0}^{\infty} k\mu(k) \leq 1.$$

We say that  $\mu$  is a critical offspring distribution if the equality happens in the previous inequality or subcritical offspring distribution if this is not the case. The criticality has an effect on the size of the resulting trees.

Let us present the definition of a Galton-Watson tree.

Let  $(K_u, u \in \mathbb{U})$  be a collection of independent random variables with law  $\mu$ , indexed by the label set  $\mathbb{U}$ . We denote by  $\theta$  the random subset of  $\mathbb{U}$  defined by

$$\theta := \{u = u^1 \dots u^n \in \mathbb{U} : u^j \leq K_{u^1 \dots u^{j-1}} \text{ for every } 1 \leq j \leq n\}. \quad (2.2)$$

**Proposition 2.3** (Le Gall [8]).  *$\theta$  is a.s. a tree. Moreover if*

$$Z_n = \#\{u \in \theta : |u| = n\}$$

*$(Z_n, n \geq 0)$  is a Galton-Watson process with offspring distribution  $\mu$  and initial value  $Z_0 = 1$ .*

*Proof.* We aim to demonstrate that  $\theta$  is a.s. a tree. Then, we want to prove that it satisfies Definition 2.1.

It is evident that  $\emptyset \in \theta$ . To establish that for every  $u \in \theta \setminus \{\emptyset\}$   $\pi(u) \in \theta$  we only need to reference the definition of  $\theta$ , as each  $u$  is defined as the concatenation of a natural number with the parent of the individual.

Finally, we observe that  $k_u(\theta) = K_u$  satisfies that for all  $u^j \in \mathbb{N}, uu^j \in \theta$  if and only if  $1 \leq u^j \leq K_u$ .

Therefore,  $\theta$  is a.s. a finite ordered tree. ■

**Remark 2.1.** Note  $k_u(\theta) = K_u$  for every  $u \in \theta$ .

To avoid leaving loose ends, we present the **classical definition of a Galton-Watson process**.

Let  $(Z_n, n \geq 0)$  be a Markov chain defined recursively as

$$Z_0 = 1 \quad \text{and} \quad Z_{n+1} = \sum_{i=1}^{Z_n} \xi_i^{(n)}, \quad n \geq 0$$

where  $Z_n$  is the population size at generation  $n$ ;  $\xi_i^{(n)}$  denotes the offspring of the  $i$ -th individual living at generation  $n$  with distribution  $\mu$ . We assume that the variables  $(\xi_i^{(n)}, i \geq 1, n \geq 1)$  are all independent.

The tree  $\theta$ , or any random tree with the same distribution, is called a Galton-Watson tree with offspring distribution  $\mu$ , or in short a  $\mu$ -Galton-Watson tree. We also write  $\Pi_\mu$  for the distribution of  $\theta$  on  $\mathbf{T}$ .

If  $\mathbf{t}$  is a tree and  $1 \leq j \leq k_\emptyset(\mathbf{t})$ , we write  $T_j \mathbf{t}$  for the tree shifted at  $j$ :

$$T_j \mathbf{t} = \{u \in \mathbb{U} : ju \in \mathbf{t}\}.$$

Then  $\Pi_\mu$  may be characterized by the following two properties:

- (i)  $\Pi_\mu(k_\emptyset = j) = \mu(j), j \in \mathbb{Z}_+$ .
- (ii) For every  $j \geq 1$  with  $\mu(j) > 0$ , the shifted trees  $T_1 \mathbf{t}, \dots, T_j \mathbf{t}$  are independent under the conditional probability  $\Pi_\mu(\cdot | k_\emptyset = j)$  and their conditional distribution is  $\Pi_\mu$ .

Property (ii) is often called the *branching property* of the Galton-Watson tree.

The explicit formula for  $\Pi_\mu$  is given in the next proposition.

**Proposition 2.4** (Le Gall [8]). *For every  $\mathbf{t} \in \mathbf{T}$ ,*

$$\Pi_\mu(\mathbf{t}) = \prod_{u \in \mathbf{t}} \mu(k_u(\mathbf{t})). \quad (2.3)$$

*Proof.* The proof is rather easy, in order to get the same tree, we need to match the number of children that every vertex has,

$$\{\theta = \mathbf{t}\} = \bigcap_{u \in \mathbf{t}} \{K_u = k_u(\mathbf{t})\}.$$

Then, if we compute the probability of this event:

$$\Pi_\mu(\mathbf{t}) = \mathbb{P}(\theta = \mathbf{t}) = \mathbb{P}\left(\bigcap_{u \in \mathbf{t}} \{K_u = k_u(\mathbf{t})\}\right),$$

recalling the fact that  $(K_u, u \in \mathbb{U})$  are a collection on random independent variables, we have:

$$\Pi_\mu(\mathbf{t}) = \prod_{u \in \mathbf{t}} \mathbb{P}(K_u = k_u(\mathbf{t})) = \prod_{u \in \mathbf{t}} \mu(k_u(\mathbf{t})).$$

■

Recall the definition of the map  $\Phi$ , given in Proposition 2.1. As we said when it was introduced, the height process has its downsides, one of which is that it is non-Markovian, which is not something that we enjoy. That is why we cannot use it to directly codify our trees. Nevertheless, thanks to Proposition 2.2, we can indirectly use the height process if we work with the Łukasiewicz paths of Galton-Watson trees. In that spirit, let us explore the stochastic behaviour of the map  $\Phi$ .

**Proposition 2.5** (Le Gall [8]). *Let  $\theta$  be a  $\mu$ -Galton-Watson tree. Then*

$$\Phi(\theta) \stackrel{(\mathcal{L})}{=} (M_1, M_2, \dots, M_T),$$

where the random variables  $M_1, M_2, \dots$  are independent with distribution  $\mu$ , and

$$T = \inf\{n \geq 1 : M_1 + \dots + M_n < n\}.$$

At first we may think that this proposition is redundant since we stated that  $(K_u, u \in \mathbb{U})$  is a collection of independent random variables with law  $\mu$ . Although, the random variables  $M_1, M_2, \dots$  correspond to a collection of those in  $(K_u)_{u \in \mathbb{U}}$ , the construction of  $\theta$  in (2.2) involves a reordering of such variables (into  $M_1, M_2, \dots$ ). It is not straightforward that this procedure may not induce some dependencies. Indeed, Proposition 2.5 states that the independence and distribution are preserved through that rearrangement process.

*Proof.* We may assume that  $\theta$  is given by the construction given in (2.2). Write  $U_0 = \emptyset, U_1 = 1, \dots, U_{\#(\theta)-1}$  for the elements of  $\theta$  listed in lexicographical order, in such a way that

$$\Phi(\theta) = (K_{U_0}, K_{U_1}, \dots, K_{U_{\#(\theta)-1}}).$$

From how the sequences on  $\mathcal{S}$  were previously defined, we know that  $K_{U_0} + \dots + K_{U_p} \geq p + 1$  for every  $p \in \{0, 1, \dots, \#(\theta) - 2\}$ , and also they satisfy  $K_{U_0} + \dots + K_{U_{\#(\theta)-1}} = \#(\theta) - 1$ .

We will find convenient to also define  $U_p$  for  $p \geq \#(\theta)$ , for instance by setting

$$U_p = U_{\#(\theta)-1} 1 \dots 1$$

where in the right-hand side we added the word of length  $p - \#(\theta) + 1$  that only contains 1.

From these definitions, the proof comes down to checking that, for every  $n \geq 0$ ,  $K_{U_0}, \dots, K_{U_n}$  are independent with distribution  $\mu$ . Note that the labels  $U_j$  are random and depend on the collection  $(K_u, u \in \mathbb{U})$  therefore we cannot use the fact that the variables  $K_u, u \in \mathbb{U}$  are i.i.d. with distribution  $\mu$ .

Our argument will be made using induction over  $n$ . For  $n = 0$  or  $n = 1$ , the result is obvious since  $U_0 = \emptyset$  and  $U_1 = 1$  are deterministic. Thus we have our base step for induction.

Fix  $n \geq 2$  and let us assume that for  $n - 1$  we have that.

$$K_{U_0}, K_{U_1}, \dots, K_{U_{n-1}} \quad \text{are i.i.d. with law } \mu.$$

Recall the notation introduced when we talked about the lexicographical order on  $\mathbb{U}$ . We want to see that, for every fixed  $u \in \mathbb{U}$ , the random set

$$\theta \cap \{v \in \mathbb{U} : v \trianglelefteq u\}$$

is measurable with respect to the  $\sigma$ -field  $\mathcal{F}_u := \sigma(K_v, v \triangleleft u)$ . We can see that this is true, since  $\mathcal{F}_u$  is the  $\sigma$ -algebra generated by the set of the number of children ( $K_v$ ) of every vertex smaller (in lexicographical order) to  $u$ . Since  $\theta$  is defined by the collection of  $(K_u, u \in \mathbb{U})$  then it is clear that the intersection of  $\theta$  and  $\{v \in \mathbb{U} : v \trianglelefteq u\}$  is constructed by the set that defines the  $\sigma$ -algebra.

As a consequence, the events

$$\{U_n = u\} \cap \{\#(\theta) > n\} \quad \text{and} \quad \{U_n = u\} \cap \{\#(\theta) \leq n\}$$

are measurable with respect to  $\sigma(K_v, v \triangleleft u)$ . Hence  $\{U_n = u\}$  is measurable with respect to  $\sigma(K_v, v \triangleleft u)$ .

Finally, let  $g_0, g_1, \dots, g_n$  be nonnegative functions on  $\{0, 1, \dots\}$ . A very common argument used in these types of proofs is to show that

$\mathbb{E}[\prod_{i=0}^n g_i(K_{U_i})] = \prod_{i=1}^n \mathbb{E}[g_i(K_{U_i})]$ . This because if we use as function  $g(K_{U_i}) = \mathbb{1}_{K_{U_i}}$ , where  $\mathbb{1}$  is the indicator function, we obtain that  $\mathbb{P}(K_{U_0}, K_{U_1}, \dots, K_{U_n}) = \prod_{i=1}^n \mathbb{P}(K_{U_i})$  which is the definition of independence. We will use this argument to see the independence of our random variables.

Let us begin by using the tower property,

$$\mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_n(K_{U_n})] = \mathbb{E}[\mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_n(K_{U_n})|\mathcal{F}_{u_n}]].$$

Recalling that the random set  $\theta \cap \{v \in \mathbb{U} : v \trianglelefteq u_n\}$  is measurable with respect to  $\mathcal{F}_{u_n}$ . We have:

$$\begin{aligned} & \mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_n(K_{U_n})] \\ &= \mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_{n-1}(K_{U_{n-1}})\mathbb{E}[g_n(K_{U_n})|\mathcal{F}_{u_n}]]. \end{aligned}$$

Let us condition the last expectation with respect to the event

$\mathcal{U} := \{U_0 = u_0, U_1 = u_1, \dots, U_n = u_n\}$ . Then:

$$\begin{aligned} & \mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_n(K_{U_n})] \\ &= \mathbb{E}[\mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_{n-1}(K_{U_{n-1}})\mathbb{E}[g_n(K_{U_n})|\mathcal{F}_{u_n}]]|\mathcal{U}]] \\ &= \sum_{\mathbf{u}=u_0 \triangleleft u_1 \triangleleft \cdots \triangleleft u_n} \mathbb{E}[g_0(K_{U_0}) \cdots g_{n-1}(K_{U_{n-1}})] \mathbb{E}[g_n(K_{U_n})|\mathcal{F}_{u_n}] \mathbb{P}(\mathcal{U} = \mathbf{u}) \\ &= \sum_{u_0 \triangleleft u_1 \triangleleft \cdots \triangleleft u_n} \mathbb{E}[1_{\{U_0=u_0, \dots, U_n=u_n\}} g_0(K_{U_0}) \cdots g_{n-1}(K_{U_{n-1}})] \mathbb{E}[g_n(K_{U_n})|\mathcal{F}_{u_n}]. \end{aligned}$$

Since  $K_{u_n}$  is independent of  $\mathcal{F}_{u_n}$  we have that:

$$\begin{aligned} & \mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_n(K_{U_n})] \\ &= \sum_{u_0 \triangleleft u_1 \triangleleft \cdots \triangleleft u_n} \mathbb{E}[1_{\{U_0=u_0, \dots, U_n=u_n\}} g_0(K_{U_0}) \cdots g_{n-1}(K_{U_{n-1}})] \mathbb{E}[g_n(K_{U_n})] \\ &= \mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_{n-1}(K_{U_{n-1}})] \mathbb{E}[g_n(K_{U_n})]. \end{aligned}$$

We apply the induction assumption and we have that

$$\mathbb{E}[g_0(K_{U_0})g_1(K_{U_1}) \cdots g_n(K_{U_n})] = \mathbb{E}[g_0(K_{U_0})] \mathbb{E}[g_1(K_{U_1})] \cdots \mathbb{E}[g_n(K_{U_n})].$$

Thus completing the proof. ■

**Corollary 2.1.** *Let  $(S_n, n \geq 0)$  be a random walk on  $\mathbb{Z}$  with initial value  $S_0$  and jump distribution  $\nu(k) = \mu(k+1)$  for every  $k \geq -1$ . Set*

$$T = \inf\{n \geq 1 : S_n = -1\}.$$

*Then the Łukasiewicz path of a  $\mu$ -Galton-Watson tree  $\theta$  has the same distribution as  $(S_0, S_1, \dots, S_T)$ . In particular  $\#(\theta)$  and  $T$  have the same distribution.*

*Proof.* Recall the definition of the Łukasiewicz path. Applying it to a  $\mu$ -Galton-Watson tree gives us:

$$X(i) := \sum_{j=0}^{i-1} (K_j - 1), \quad 0 \leq i \leq n$$

Where  $\Phi(\theta) = (K_0, K_1, \dots, K_{n-1})$ . Observe the change in notation given that now our trees are random. Thanks to the preceding proposition:

$$X(i) \stackrel{(\mathcal{L})}{=} \sum_{j=0}^{i-1} (M_j - 1), \quad 0 \leq i \leq n$$

Let us explore the event  $\{M_j - 1 = k\}$  for fixed  $j$  and  $k \in \mathbb{Z}$ .

$$\mathbb{P}(M_j - 1 = k) = \mathbb{P}(M_j = k + 1) = \mu(k + 1) = \nu(k).$$

Thus,  $(X(i), i \geq 0)$  has the same distribution as a random walk with jump distribution  $\nu(k)$ . ■

So far, we have explored the stochastic characteristics of the codifications of a  $\mu$ -Galton-Watson tree. However, our interest is not on Galton-Watson trees, but instead on establishing their convergence to the CRT.

## 2.3 Convergence to a Brownian Motion

In this section, we set some important precedent to discover a convergence to the CRT. The goal of this section is to show that height functions of Galton-Watson trees converge in distribution, modulo a suitable re-scaling, to Brownian excursions. Also, at the end of the section we make the translation from the convergence of the height functions to contour functions.

To discover this convergence, we must limit our Galton-Watson trees to those that have a critical offspring distribution  $\mu$  with finite variance  $\sigma^2 > 0$ . Criticality means that we now have

$$\sum_{k=1}^{\infty} k\mu(k) = 1.$$

In this case, the random walk that we identified with the Łukasiewicz path is recurrent; in particular the mean of the jump distribution is zero and the tree is a.s. finite. However, the total size of the tree, has infinite mean.

To study convergences, we have to explore the behaviour of sequences of trees. From now on, we shift our focus to random forests instead of trees. To do so, we need to define a function to explore this environment.

Let  $\theta_1, \theta_2, \dots$  be a sequence of independent  $\mu$ -Galton-Watson trees. For each  $\theta_i$  define  $n_i = \#(\theta_i)$ . To each  $\theta_i$  we can associate its height function  $(h_{\theta_i}(j), 0 \leq j \leq n_i - 1)$ . We then define the height process  $(H(k), k \geq 0)$  of the forest by concatenating the functions  $h_{\theta_1}, h_{\theta_2}, \dots$ :

$$H(k) = h_{\theta_i}(k - (n_1 + \dots + n_{i-1}))$$

if  $n_1 + \dots + n_{i-1} \leq k \leq n_1 + \dots + n_i$ . The function  $(H(k), k \geq 0)$  determines the sequences of trees since the values of  $H$  between its  $j$ -th



zero and the next one are the same as the values of the height function of the  $j$ -th tree in the sequence.

**Proposition 2.6** (Le Gall [8]). *We have for every  $k \geq 0$*

$$H(k) = \# \left\{ j \in \{0, 1, \dots, k-1\} : S_j = \inf_{j \leq i \leq k} S_i \right\} \quad (2.4)$$

where  $(S_k, k \geq 0)$  is a random walk with the distribution described in Corollary 2.1.

*Proof.* From Proposition 2.2 we have that:

$$h_{\mathbf{t}}(k) = \# \left\{ 0 \leq i \leq k-1 : X(i) = \inf_{i \leq j \leq k} X(j) \right\}.$$

And from Corollary 2.1 we know that:

$$(X(j), 0 \leq j \leq T) \stackrel{(\mathcal{L})}{=} (S_0, S_1, \dots, S_T).$$

Thus,

$$h_{\mathbf{t}}(k) = \# \left\{ 0 \leq i \leq k-1 : S_i = \inf_{i \leq j \leq k} S_j \right\}.$$

Finally, since  $H(k)$  is a concatenation of height functions, the proposition is satisfied. ■

Our convergence happens when the amount of vertices on the tree grows. For Galton-Watson trees we specify the offspring distribution, however, we do not have the ability to force them to be of a certain size, as the size of the tree will be random.

Since we are in a probabilistic setting, we can go around this issue conditioning the trees on the amount of vertices they have. In this way we can ensure a given size for trees. Thus, let us define these objects

and how to characterize them. For the next results we will assume that  $\mu$  has a small exponential moment.

For every  $n \geq 1$  we denote by  $\theta^{(n)}$  a  $\mu$ -Galton-Watson tree conditioned to have  $\#(\theta) = n$ . If we want this to make sense, we need that  $\mathbb{P}(\#(\theta) = n) > 0$  for every  $n \geq 1$ , which holds if  $\mu(1) > 0$ .

We denote by  $(H^{(n)}(k), 0 \leq k \leq n)$  the height process of  $\theta^{(n)}$ , with the convention,  $H^{(n)}(n) = 0$ .

After introducing conditioned Galton-Watson trees and the height process for this object, we present a fundamental result for our purposes.

**Theorem 2.1** (Le Gall [8]). *Let  $\theta_1, \theta_2, \dots$  be a sequence of independent  $\mu$ -Galton-Watson trees. We have*

$$\left( n^{-1/2} H^{(n)}(\lfloor nt \rfloor), 0 \leq t \leq 1 \right) \xrightarrow[n \rightarrow \infty]{(\mathcal{L})} (2\sigma^{-1} \mathbf{e}(t), 0 \leq t \leq 1).$$

Where  $\mathbf{e}(t)$  is the standardized brownian excursion presented in Chapter 1.

*Proof.* We assume that  $H$  is given in terms of the random walk  $S = (S_k, k \geq 0)$  as in (2.4).

We denote by  $T_1$  the number of vertices of the first tree in the sequence, given the definition of our Height process, we have that  $T_1$  is given by

$$T_1 = \inf\{n \geq 1 : H(n) = 0\} = \inf\{n \geq 0 : S_n = -1\}.$$

We want to compute  $\mathbb{P}(T_1 = n)$ , that is, we want to know what is the probability for the first tree to be of size  $n$ . To obtain this probability, we will present a combinatorial argument that is strongly inspired by the presentation made by Bennes and Kersting in [2].

This combinatorial argument uses circular permutations. Let us have a tree  $\mathbf{t}$  of size  $N$ , codified by a random walk  $S$ . Let

$M \in \{1, \dots, N\}$  be the label of some vertex of  $\mathbf{t}$ , we mark that vertex. Call the pair  $(\mathbf{t}, M)$  a **marked tree**. To this new tree we associate the path  $S' = (S'_0, \dots, S'_N)$  given by

$$S'_0 = 0, \quad S'_i - S'_{i-1} = K'_i - 1, \quad i = 1, \dots, N$$

where  $K'_i = K_{\pi(i)}$  and  $\pi(i)$  is the label with

$$\pi(i) \equiv i + (M - 1) \pmod{N}$$

The sequence  $S'$  results from  $S$  by making the rotation

$$S'_i = \begin{cases} S_{i+M-1} - S_{M-1}, & 0 \leq i \leq N - M + 1 \\ S_{i+M-N-1} - S_{M-1} - 1, & N - M + 1 < i \leq N \end{cases}$$

This means that we took the path  $S$  and we cut it at point  $M - 1$  into two pieces, which we interchange and then we paste them together again.

Assume that  $\mathbf{t}$  is a Galton-Watson tree and  $M$  is chosen uniformly from  $\{1, \dots, N\}$ . For every excursion of length  $n$  using this process we can rearrange them in  $n$  different bridges of equal probability, thus we obtain:

$$\mathbb{P}(N = n) = \frac{1}{n} \mathbb{P}(S_n = -1).$$

We can then apply this result to our context and then we have that

$$\mathbb{P}(T_1 = n) = \frac{1}{n} \mathbb{P}(S_n = -1).$$

Note that the jump distribution  $\nu$  associated to the random walk has mean 0 and finite variance  $\sigma^2$ , thus it is recurrent.

On the other hand classical results for random walk (see e.g. P9 in Chapter II of F. Spitzer. [17]) give

$$\lim_{n \rightarrow \infty} \sqrt{n} \mathbb{P}(S_n = -1) = \frac{1}{\sigma \sqrt{2\pi}},$$

and it follows that

$$\mathbb{P}(T_1 = n) \underset{n \rightarrow \infty}{\sim} \frac{1}{\sigma \sqrt{2\pi n^3}} \quad (2.5)$$

To continue with the next part of our proof, we have to introduce some notation.

$$M_n = \sup_{0 \leq k \leq n} S_k \quad I_n = \inf_{0 \leq k \leq n} S_k$$

For every  $n \geq 0$  introduce the time-reversed random walk  $\widehat{S}^n$  defined by

$$\widehat{S}_k^n = S_n - S_{(n-k)^+}$$

and note that  $(\widehat{S}_k^n, 0 \leq k \leq n)$  has the same distribution as  $(S_k, 0 \leq k \leq n)$ . Also, let us introduce the map  $\Phi_n$  for discrete trajectories  $\omega = (\omega(0), \omega(1), \dots)$  defined as:

$$\Phi_n(\omega) = \#\{k \in \{1, \dots, n\} : \omega(k) = \sup_{0 \leq j \leq k} \omega(j)\}.$$

Note that,

$$\begin{aligned} \Phi_n(\widehat{S}^n) &= \#\{k \in \{1, \dots, n\} : \widehat{S}_k^n = \sup_{0 \leq j \leq n} \widehat{S}_j^n\} \\ &= \#\{k \in \{1, \dots, n\} : S_n - S_{(n-k)^+} = \sup_{0 \leq j \leq n} S_n - S_{(n-j)^+}\} \\ &= \#\{k \in \{1, \dots, n\} : -S_{(n-k)^+} = -\inf_{0 \leq j \leq n} S_{(n-j)^+}\} \\ &= \#\{k \in \{1, \dots, n\} : S_{(n-k)^+} = \inf_{0 \leq j \leq n} S_{(n-j)^+}\} \\ &= \#\{k \in \{1, \dots, n\} : S_k = \inf_{k \leq j \leq n} S_j\}. \end{aligned}$$

Thus,  $\Phi_n(\widehat{S}^n) = H(n)$  from Proposition 2.2. Also, set

$$K_n = \Phi_n(S) = \#\{k \in \{1, \dots, n\} : S_k = M_k\}.$$

To finish this proof, let us state a useful Lemma.

**Lemma 2.1.** *Let  $\varepsilon \in (0, \frac{1}{4})$ . We can find  $\varepsilon' > 0$  and an integer  $N \geq 1$  such that, for every  $n \geq N$  and  $\ell \in \{0, 1, \dots, n\}$ ,*

$$\mathbb{P} \left[ \left| M_\ell - \frac{\sigma^2}{2} K_\ell \right| > n^{\frac{1}{4} + \varepsilon} \right] < \exp(-n^{\varepsilon'}).$$

In order to apply Lemma 2.1 we need to see that the pair  $(M_n, K_n) \stackrel{(\mathcal{L})}{=} (S_n - I_n, H(n))$ , we can see this fact replacing  $S$  with  $\widehat{S}^n$ .

On one hand, the equality in distribution of  $K_n$  and  $H(n)$  is clear from the fact that  $K_n = \Phi_n(S)$  and  $H(n) = \Phi_n(\widehat{S}^n)$ .

On the other hand, we can see that  $M_n \stackrel{(\mathcal{L})}{=} S_n - I_n$  because, when we replace in  $M_n$  the reversed-time walk  $\widehat{S}^n$  we get:

$$\begin{aligned} M_n &= \sup_{0 \leq j \leq k} S_j \\ &= \sup_{k \leq j \leq n} \widehat{S}_j^n \\ &= \sup_{k \leq j \leq n} S_n - S_{(n-j)^+} \\ &= S_n - \inf_{k \leq j \leq n} S_{(n-j)^+} \\ &= S_n - I_n. \end{aligned}$$

Thus, from the Lemma 2.1 and our last observation, taking  $\varepsilon = 1/8$  we get

$$\mathbb{P} \left[ \left| S_\ell - I_\ell - \frac{\sigma^2}{2} H(\ell) \right| > n^{\frac{3}{8}} \right] < \exp(-n^{\varepsilon'}).$$

Hence

$$\mathbb{P} \left[ \sup_{0 \leq \ell \leq n} \left| S_\ell - I_\ell - \frac{\sigma^2}{2} H(\ell) \right| > n^{\frac{3}{8}} \right] < n \exp(-n^{\varepsilon'}).$$

From the preceding bound we can deduce that for  $n$  large enough,

$$\mathbb{P} \left[ \sup_{0 \leq t \leq 1} \left| S_{\lfloor nt \rfloor} - I_{\lfloor nt \rfloor} - \frac{\sigma^2}{2} H(\lfloor nt \rfloor) \right| > n^{\frac{3}{8}} \right] < n \exp(-n^{\varepsilon'}).$$

Now, doing some algebra inside the argument of the supremum in the preceding inequality, we obtain, that we can find some  $\varepsilon > 0$  for  $n$  large enough

$$\mathbb{P} \left[ \sup_{0 \leq t \leq 1} \left| \frac{H(\lfloor nt \rfloor)}{\sqrt{n}} - \frac{2}{\sigma^2} \frac{S_{\lfloor nt \rfloor} - I_{\lfloor nt \rfloor}}{\sqrt{n}} \right| > \frac{2}{\sigma^2} n^{-1/8} \right] < n \exp(-n^\varepsilon).$$

By comparing with equation (2.5), we see that when we condition on the event  $\{T_1 = n\}$  our probability will not explode, then we have that for  $n$  large

$$\mathbb{P} \left[ \sup_{0 \leq t \leq 1} \left| \frac{H(\lfloor nt \rfloor)}{\sqrt{n}} - \frac{2}{\sigma^2} \frac{S_{\lfloor nt \rfloor} - I_{\lfloor nt \rfloor}}{\sqrt{n}} \right| > \frac{2}{\sigma^2} n^{-1/8} \middle| T_1 = n \right] < n \exp(-n^{\varepsilon'}),$$

for any  $\varepsilon' < \varepsilon$ . Since  $I_k = 0$  for  $0 \leq k < T_1$ , we have also for  $n$  large

$$\mathbb{P} \left[ \sup_{0 \leq t \leq 1} \left| \frac{H(\lfloor nt \rfloor)}{\sqrt{n}} - \frac{2}{\sigma^2} \frac{S_{\lfloor nt \rfloor}}{\sqrt{n}} \right| > \frac{2}{\sigma^2} n^{-1/8} \middle| T_1 = n \right] < n \exp(-n^{\varepsilon'}).$$

Now, under  $\mathbb{P}(\cdot | T_1 = n)$ ,  $(H_k^{(n)}, 0 \leq k \leq n)$  has the same distribution as  $(H_k, 0 \leq k \leq n)$ . Therefore Theorem 2.1 is a consequence of the last bound and the following lemma which relates the normalized Brownian excursion to the random walk excursion with a fixed long duration.

**Lemma 2.2.** *The distribution of the process  $(\frac{1}{\sigma\sqrt{n}} S_{\lfloor nt \rfloor}, 0 \leq t \leq 1)$  under the conditional probability  $\mathbb{P}(\cdot | T_1 = n)$  converges as  $n$  tends to  $\infty$  to the law of the normalized Brownian excursion.*

The proof of this lemma will not be presented in this work; see for example [10]. ■

We have now stated our main result in terms of the height function, it is time to translate this result to contour functions.

### 2.3.1 Results in terms of Contour functions

On this section we are going to discuss how to translate our results that were presented in terms of height functions to contour functions.

The main result of the last few sections is Theorem 2.1, which is stated in terms of the height process of a conditioned Galton-Watson tree, however, before we talk about the conditioned contour process let us first translate the *unconditioned* height process and afterwards we can translate the conditioned version

Consider a sequence  $\theta_1, \theta_2, \dots$  of independent  $\mu$ -Galton-Watson trees. In a similar fashion to what we did when we presented the height function of a random forest, now we have to define the concatenation of contour functions.

Recalling the definition of the contour function, presented in Section 2.1. It was defined on the time interval  $[0, \zeta(\theta)]$ , where  $\zeta(\theta) := 2(\#(\theta) - 1)$ . This definition has the consequence that the contour function of a tree that only consists of a root is trivial. To get around this inconvenience Le Gall makes the artificial assumption in [8] to define the contour function of a tree  $\theta$ ,  $C_\theta(t)$  for  $0 \leq t \leq \xi(\theta) = 2\#(\theta) - 1$ , by taking  $C_\theta(t) = 0$  if  $\zeta(\theta) \leq t \leq \xi(\theta)$ . Then we can define  $(C(t), t \geq 0)$  as the concatenation of the functions  $(C_{\theta_1}(t), 0 \leq t \leq \xi(\theta_1))$ ,  $(C_{\theta_2}(t), 0 \leq t \leq \xi(\theta_2))$ , etc.

To make our connection between codifications, we have to see that the contour process can be recovered from the height process in a rather easy way. Note that we are going to work on the contour process defined on the interval  $[0, \zeta(\theta)]$ , and we will not mention the conventions mentioned earlier, that is, because the convention only adds values to the function after  $\zeta(\theta)$ , thus we can work on the initial interval and then we can add the clarification that the value of the

contour is zero on the remainder of the interval  $[0, \xi(\theta)]$ .

First, set  $b_k = 2k - H(k)$ , for  $0 \leq k < \#(\theta)$  and  $b_{\#(\theta)} = 2(\#(\theta) - 1)$ . Let us see that  $\{b_k\}$  is an strictly increasing sequence.

Let  $k \in [0, \#(\theta)]$ . We want to see that  $b_{k+1} > b_k$ . To see this, first let us note that

$$H(k+1) - H(k) < 2.$$

Recall that we are exploring the tree in a lexicographical order. Then we have three cases for the height of two consecutive elements on the tree.

$$(i) H(k+1) < H(k) \quad (ii) H(k+1) = H(k) \quad (iii) H(k+1) > H(k)$$

It is clear that on cases (i) and (ii) that the condition is satisfied.

For case (iii) we have to realize that the only case in which that can happen is the one in which the vertex  $k+1$  is a direct descendant from vertex  $k$ , thus,  $H(k+1) = H(k) + 1$ . Therefore,  $H(k+1) - H(k) = 1 < 2$ . Then we have:

$$\begin{aligned} H(k+1) - H(k) &< 2 \\ \Leftrightarrow 2k - H(k) &< 2k + 2 - H(k+1) \\ \Leftrightarrow b_k &< b_{k+1}. \end{aligned}$$

Thus the sequence  $\{b_k\}$  is strictly increasing. Furthermore,  $b_k \geq k$  for every  $k \in [0, \zeta(\theta)]$ .

This derives from the fact that since we are working with lexicographical order, the generation of the individual  $k$  cannot be greater than the amount of nodes that have been visited, because on lexicographical order we visit the *parents* of the vertices before than the vertices. Thus  $H(k) < k$  and therefore  $b_k \geq k$  for every  $k \in [0, \zeta(\theta)]$ .



Then, we observe that

$$0 = b_0 < b_1 < \dots < b_{\#(\theta)-1} < b_{\#(\theta)} = 2(\#(\theta) - 1).$$

Finally, we claim that for  $t \in [b_n, b_{n+1}]$ , the contour process goes from individual  $n$  to individual  $n + 1$ .

To proof this claim, we need to make an induction argument over  $n$ . Before starting the proof, recall that the contour process is like placing a particle at the root at time zero and letting it explore the tree going in lexicographical order moving through the edges of the tree.

For  $n = 0$ , it is satisfied, since  $b_0 = 0$ , and noting that the height of the first element on a tree is always 1, we see that  $b_1 = 1$ . Thus it is satisfied.

We assume that for  $n$  the interval  $[b_n, b_{n+1}]$  describes the time interval in which the contour process goes from individual  $n$  to individual  $n + 1$ .

We have to proof that this is true for  $n + 1$ .

To proof our claim, let us note that  $b_{n+1} = 2n + 2 - H(n + 1)$  and  $b_{n+2} = 2n + 4 - H(n + 2)$ . From the induction assumption we have that on time  $b_{n+1}$  the contour process is at vertex  $n + 1$ . Depending on the relation between vertices  $n + 1$  and  $n + 2$ , the time that will take the contour process to go from one to the other will vary. Once again, we have cases.

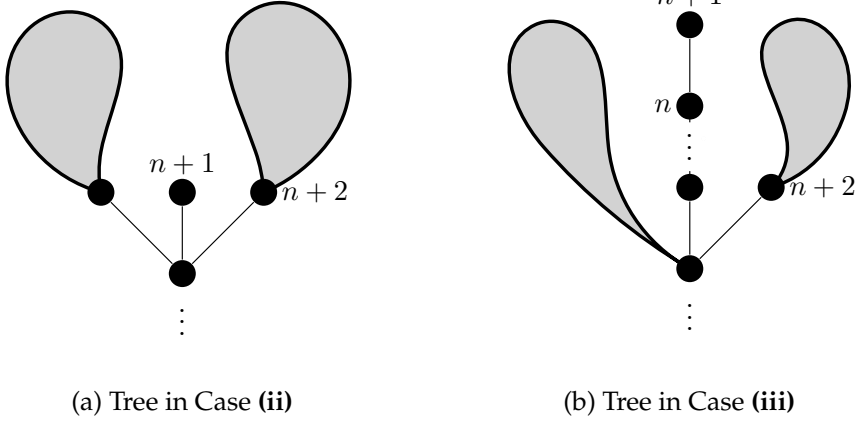
Case (i)  $H(n + 2) > H(n + 1)$ .

From the proof of the behaviour of the sequence  $\{b_n\}$ , we know that  $H(n + 2) = H(n + 1) + 1$ , thus

$$b_{n+2} = 2n + 4 - H(n + 2) = 2n + 3 - H(n + 1) = b_{n+1} + 1.$$

Which supports our claim, since in this case, the  $n + 2$  element of the tree would be linked through an edge to the  $n + 1$  element and therefore

it would only take 1 unit of time for the Contour process to go from one to the other.



**Figure 2.9.** Illustrations of Cases (ii) and (iii).

Case (ii)  $H(n+2) < H(n+1)$ .

In this case we have a behaviour as the one shown in Figure 2.9a. Therefore the contour process would take 2 units of time to go from individual  $n+1$  to individual  $n+2$ . Let us see what value  $b_{n+2}$  takes on this case:

$$b_{n+2} = 2n + 4 - H(n+2) = 2n + 4 - H(n+1) = b_{n+1} + 2.$$

Thus supporting our claim.

Case (iii)  $H(n+2) = H(n+1)$ .

In this case we have behaviour as the one shown in Figure 2.9b. We can see that the time that will take the contour function to go from the  $(n+1)$ -th individual to the  $(n+2)$ -th is given by their height difference and 2 units more. Equivalently, we want

$$b_{n+2} = b_{n+1} + H(n+1) - H(n+2) + 2.$$

Which we can show using the oldest trick in the book, adding a zero,

$$\begin{aligned}
b_{n+2} &= 2n + 4 - H(n + 2) \\
&= 2n + 4 - H(n + 1) + H(n + 1) - H(n + 2) \\
&= b_{n+1} + H(n + 1) - H(n + 2) + 2.
\end{aligned}$$

Thus completing the proof of the fact that on the interval  $[b_n, b_{n+1}]$  the contour process goes from individual  $n$  to individual  $n + 1$ . Once that we have proven this fact, we can write the contour process in terms of height processes.

For  $k < \#(\theta) - 1$  and  $t \in [b_k, b_{k+1}]$

$$C(t) = \begin{cases} H(k) - (t - b_k), & \text{if } t \in [b_k, b_{k+1} - 1) \\ H(k + 1) + t - b_{k+1}, & \text{if } t \in [b_{k+1} - 1, b_{k+1}] \end{cases} \quad (2.6)$$

and

$$C(t) = H(\#(\theta) - 1) - (t - b_{\#(\theta)-1}), \quad \text{if } t \in [b_{\#(\theta)-1}, b_{\#(\theta)}).$$

This identity between both functions is easy to see if you recall that the contour function takes the values of the height function at its *peaks* and everywhere else has a slope equal to  $\pm 1$  uniting those points.

From this identity, we get the following inequality

$$\sup_{t \in [J(n), J(n+1)]} |C(t) - H(n)| \leq |H(n + 1) - H(n)| + 1.$$

Define a random function  $\varphi : \mathbb{R}_+ \rightarrow \{0, 1, \dots\}$  by setting  $\varphi(t) = n$  iff  $t \in [b_n, b_{n+1})$ . From the previous bound, we get for every integer  $m \geq 1$ ,

$$\sup_{t \in [0, m]} |C(t) - H(\varphi(t))| \leq \sup_{t \in [0, b_m]} |C(t) - H(\varphi(t))| \leq 1 + \sup_{n \leq m} |H(n+1) - H(n)|. \quad (2.7)$$

Similarly, it follows from the definition of  $b_n$  that

$$\sup_{t \in [0, m]} \left| \varphi(t) - \frac{t}{2} \right| \leq \sup_{t \in [0, b_m]} \left| \varphi(t) - \frac{t}{2} \right| \leq \frac{1}{2} \sup_{n \leq m} H(n) + 1. \quad (2.8)$$

To present the following result, we introduce the notation  $C^{(n)}(t)$  which is the contour function of a tree conditioned to be size  $n$ .

**Theorem 2.2.** *We have*

$$\left( n^{-1/2} C^{(n)}(\lfloor 2nt \rfloor), 0 \leq t \leq 1 \right) \xrightarrow[n \rightarrow \infty]{(\mathcal{L})} (2\sigma^{-1} \mathbf{e}(t), 0 \leq t \leq 1).$$

Where  $\mathbf{e}(t)$  is the standarized brownian excursion presented in Chapter 1.

*Proof.* For every  $n \geq 1$ , set  $\varphi_n(t) = n^{-1} \varphi(nt)$ , we are scaling our original function  $\varphi$ . By (2.7), we have that for every  $n \geq 1$ :

$$\begin{aligned} \sup_{t \leq m} \left| \frac{1}{\sqrt{n}} C^{(n)}(2nt) - \frac{1}{\sqrt{n}} H^{(n)}(n\varphi_n(2t)) \right| \\ \leq \frac{1}{\sqrt{n}} + \frac{1}{\sqrt{n}} \sup_{t \leq 2m} \left| H^{(n)}(\lfloor nt \rfloor + 1) - H^{(n)}(\lfloor nt \rfloor) \right|. \end{aligned} \quad (2.9)$$

We have the following convergence.

$$\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{n}} \left| H^{(n)}(\lfloor nt \rfloor + 1) - H^{(n)}(\lfloor nt \rfloor) \right| \xrightarrow[n \rightarrow \infty]{} 0.$$

Which tells us that the contour process and the height process converge when  $n$  grows. Also, from (2.8) we get

$$\sup_{t \leq m} |\varphi_n(2t) - t| \leq \frac{1}{n} \sup_{k \leq 2mn} H^{(n)}(k) + \frac{2}{n}; \quad (2.10)$$

Where, the right hand side of the inequality, satisfies

$$\frac{1}{n} \sup_{k \leq 2mn} H^{(n)}(k) + \frac{2}{n} \xrightarrow[n \rightarrow \infty]{} 0.$$

The statement of the theorem is fulfilled thanks to Theorem 2.1 and equations (2.9) and (2.10). ■

## 2.4 Convergence to a CRT

Before presenting the theorem that states the convergence of certain Galton-Watson trees to the CRT, let us define  $\mathcal{R}$  as the set of all real trees. Also recall from Section 2.1 the notation  $\mathbf{T}$  as the set for all finite rooted ordered trees. Finally, denote by  $\mathbf{T}_n$  the subset of  $\mathbf{T}$  consisting of trees with  $n$  vertices.

To make the proof of the last theorem of this chapter easier to follow, we present the following proposition.

**Proposition 2.7.** *Let  $\theta$  be a Galton-Watson tree with geometric offspring distribution of parameter  $1/2$ , that is,  $\mu(k) = \left(\frac{1}{2}\right)^{k+1}$  for  $k \geq 0$ . Then  $\theta$  conditioned on having  $n$  edges has a uniform distribution over  $\mathbf{T}_n$ .*

The following proof is inspired by the one shown in [6] by Nicolas Curien.

*Proof.* Let  $\mathbf{t}_0$  be a tree with  $n$  edges. We want to compute its distribution, and thanks to Proposition 2.4 we have:

$$\begin{aligned} \Pi_\mu(\mathbf{t}_0) &= \prod_{u \in \mathbf{t}_0} \mu(k_u(\mathbf{t}_0)) \\ &= \prod_{u \in \mathbf{t}_0} \left(\frac{1}{2}\right)^{k_u(\mathbf{t}_0)+1} \\ &= \left(\frac{1}{2}\right)^{n+1+\sum_{u \in \mathbf{t}_0} k_u(\mathbf{t}_0)}. \end{aligned}$$

Note that the amount of nodes on the  $\mathbf{t}_0$  is  $n+1$ . Recall from Proposition 2.1 that  $\sum_{u \in \mathbf{t}_0} k_u(\mathbf{t}_0) = |\mathbf{t}_0| - 1 = n$ . Then, we have that:

$$\Pi_\mu(\mathbf{t}_0) = \left(\frac{1}{2}\right)^{2n+1}.$$

This probability does not depend on  $\mathbf{t}_0$  as long as it has  $n$  edges. Hence the conditional law of  $\theta$  on  $\mathbf{T}_n$  is the uniform law. ■

Let us make some observations, by looking at the proof of Proposition 2.7, we can see that the parameter of the geometric distribution did not affect the result, therefore, we could have used any non trivial parameter. However, we said at the beginning of this section that the offspring distribution had to be critical, hence, using a geometric distribution with parameter  $1/2$  we satisfy that assumption.

Let  $\mathbf{t} \in \mathbf{T}$ , if  $(C(t), t \geq 0)$  is the contour function of the tree, then we can identify  $\mathbf{t} = \mathcal{T}_C$ . For any  $\lambda > 0$  and a tree  $\mathcal{T} \in \mathcal{R}$ , the tree  $\lambda\mathcal{T}$  is the "same" tree with all distances multiplied by the factor  $\lambda$ . After presenting these facts, we can see the following theorem.

**Theorem 2.3** (Le Gall [8]). *For every  $n \geq 1$ , let  $\mathcal{T}_{(n)}$  be a random tree distributed uniformly over  $\mathbf{T}_n$ . Then  $(2n)^{-1/2}\mathcal{T}_{(n)}$  converges in distribution to the CRT  $\mathcal{T}_e$ , in the space  $\mathcal{R}$ .*

*Proof.* Let  $\theta$  be a Galton-Watson tree with geometric offspring distribution  $\mu(k) = 2^{-k-1}$ , and for every  $n \geq 1$ , let  $\theta_n$  be distributed as  $\theta$  conditioned to have  $n$  vertices, then, from Proposition 2.7, we know that  $\theta_n$  has the same distribution as  $\mathcal{T}_{(n)}$ . On one hand, let  $(C^n(t), t \geq 0)$  be the contour function of  $\theta_n$ , and let

$$\tilde{C}^n(t) = (2n)^{-1/2}C^n_{2nt}, \quad t \geq 0.$$

From Theorem 2.2, we have

$$(\tilde{C}^n(t), t \geq 0) \xrightarrow[n \rightarrow \infty]{(\mathcal{L})} (\mathbf{e}_t, t \geq 0)$$

On the other hand, from the observations preceding the theorem, and the fact that  $\theta_n$  has the same distribution as  $\mathcal{T}_{(n)}$ , it is immediate that the tree  $\mathcal{T}_{\tilde{C}^n}$  coded by  $\tilde{C}^n$  has the same distribution as  $(2n)^{-1/2}\mathcal{T}_{(n)}$ . Since there exists a convergence in the contour functions of both trees, from Lemma 1.3 we can conclude that the statement of the theorem is true. ■

This theorem gives us a new form of obtaining the CRT. As we saw in the last theorem, not every Galton-Watson tree converges to the CRT, only the Uniformly distributed one. These trees appear in combinatorial problems, but can be studied with the tools that surround Galton-Watson processes.

However, since Uniform Random Trees do not only appear when talking about Galton-Watson trees, there are different ways of working with them. On the following chapter we explore the same convergence but using tools that do not involve Galton-Watson processes.

## Chapter 3

# Construction via URT

In the previous chapters we defined the CRT and we showed a convergence using Galton-Watson trees. Nevertheless, we saw that the convergence happens for a "random tree that distributes uniformly over  $\mathbf{T}_n$ ".

On this chapter we will show the same convergence using different tools. For starters we have to introduce the concept of Uniform Random Trees (for short URT) and how to characterize them.

To see the convergence using these new objects and characterizations we will have to introduce a different definition of the CRT. This new definition will be based on the so-called *line-breaking construction*.

### 3.1 What is a URT?

Uniform Random Trees are the simplest model to create a random tree. As its name suggest, we can obtain a URT by performing a uniform sampling from a particular set of trees. Let us precisely define

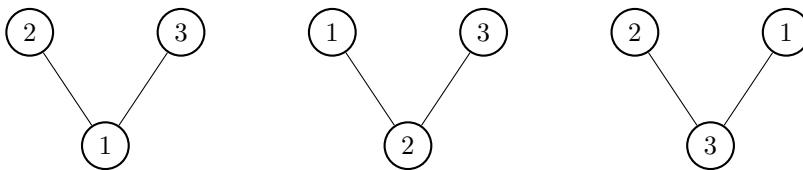


the ingredients that come into place to build our random trees.

The set of trees from which we sample are the set of (unordered) trees<sup>1</sup> with labels given by  $[n] := \{1, 2, \dots, n\}$ . We will call this set of trees  $\mathbb{T}_n$ . These trees have  $n$  vertices and no root.

To get more information about this sampling we can use Cayley's formula<sup>2</sup>, to obtain the amount of trees in  $\mathbb{T}_n$ . The formula states that the amount of different labeled trees that can be constructed from  $n$  vertices is  $n^{n-2}$ . Therefore, the cardinality of  $\mathbb{T}_n$  is  $n^{n-2}$ . Thus we can see that the probability of choosing any tree  $t$  has to be  $n^{2-n}$ .

To better understand the sets that interest us in this chapter, we present in [Figure 3.1](#) all the elements of  $\mathbb{T}_3$ .



**Figure 3.1.** All the elements of  $\mathbb{T}_3$ .

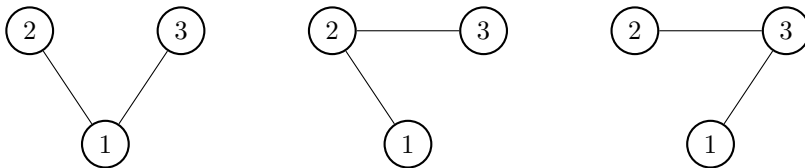
Now, the reader may be wondering, how can this be all the elements of  $\mathbb{T}_3$  if they all look the same. It is true that they all look similar, however we can deduce from what was said at the beginning of [Chapter 1](#) that the attribute that distinguishes a graph from another with the same amount of nodes are their edges. To see more clearly how these trees differ from one another, on the [Figure 3.2](#) we have

---

<sup>1</sup>Unlike the previous chapter, where the uniform tree was over the set of ordered trees.

<sup>2</sup>See [\[5\]](#) for a proof of this result.

arranged the nodes in a way that fixes their position with respect to the others throughout the illustrations.

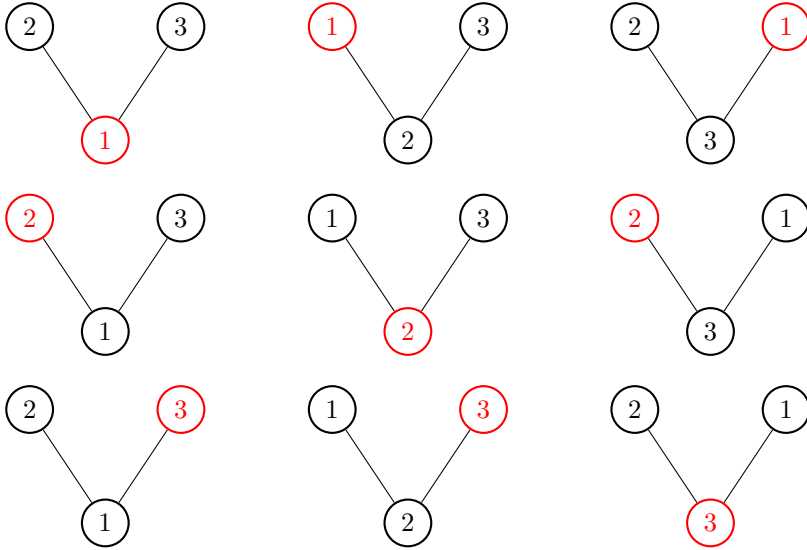


**Figure 3.2.** All the elements of  $\mathbb{T}_3$  with the nodes in the same position.

Thanks to this illustration of the trees with the same amount of nodes, we can confirm that in order to distinguish trees with the same amount of nodes, we only have to look at the edges of the trees.

Following the introduction of this chapter, we want to prove the same convergence that we saw in the previous chapter, thus we need to work with rooted trees. Then, we define the set  $\mathbb{T}_n^*$ , as the elements of  $\mathbb{T}_n$  with a distinguished vertex, the root. We have changed the space in which we are going to define our uniform distribution, therefore we need to discuss the cardinality of this new set.

Since we let the root of the trees be any vertex, we can compute the cardinality of  $\mathbb{T}_n^*$  using a simple counting argument. We recall that the cardinality of  $\mathbb{T}_n$  is  $n^{n-2}$ . Noting that there are  $n$  ways of choosing a root in each of this trees, we can see that the number of possible trees changes to  $n * n^{n-2}$ . Therefore the cardinality of  $\mathbb{T}_n^*$  is  $n^{n-1}$ . One can easily see that the amount of trees augmented considerably, since for every element of  $\mathbb{T}_n$  on  $\mathbb{T}_n^*$  we now have  $n$  different elements with the same configuration of edges. In [Figure 3.3](#) we include all the elements of  $\mathbb{T}_3^*$ . To see how radical the change is, compare it with [Figure 3.1](#).



**Figure 3.3.** All the elements of  $\mathbb{T}_3^*$  with the root of each one marked in red.

At this point, we have defined the URT and talked about the space in which we find them, but something that we have not said yet it is how to characterize them. Without a characterization we cannot do much with these objects. The next section focuses on providing a characterization of these objects.

## 3.2 Characterizing the URT

Even when we know what a URT is, we do not have a simple way to work with them. We have to find tools to differentiate the possible URTs that we encounter. One form to characterize them will be using

algorithms. This characterization has its caveats as an algorithm does not necessarily provide a concise representation of the trees. However, an algorithm is useful in helping to visualize the behaviour of the trees as the amount of vertices increases.

### 3.2.1 The Aldous-Broder algorithm

This algorithm is presented as the **Aldous-Broder algorithm** by Goldschmidt in [9], and it generates a uniform random tree on  $\mathbb{T}_n^*$ .

The idea behind this particular algorithm is to walk on a complete graph and in a random way choose edges to construct a random tree. The fundamental object that this algorithm uses to build URTs is a **random walk**, for short RW.

For sake of completeness, based on [18] we include in the following, the properties for RW required for our development.

A **random walk**  $(S_n, n \geq 0)$ , with state space<sup>3</sup>  $\mathcal{S}$ , defined by

$$S_n = S_0 + \sum_{r=1}^n X_r, \quad \forall S_0 \in \mathcal{S}.$$

where  $(X_n, n \geq 1)$  is a sequence of independent identically distributed random variables. And  $S_0$  is independent from the  $X_i$ . Random walks are discrete time Markov chains, with **transition probabilities matrix**  $P$  defined by

$$(P)_{ij} = \mathbb{P}(S_n = j | S_{n-1} = i), \quad i, j \in \mathcal{S}.$$

The entries of the matrix  $P$  represent the behaviour of the chain in the short term, as they only tell you what could happen in the next step based on what happened in the step before. We can also be interested

---

<sup>3</sup>The state space is usually a suitable subset of integers.

in what happens in more than one step, let us say  $n$  steps. To see how our chain behaves, thanks to the Chapman-Kolmogorov equations, we can take the  $n$  product of the matrix  $P$ , that is  $P^n$ . Then, we denote the  $n$ -step probabilities by

$$p_{ij}^{(n)} = \mathbb{P}(X_{m+n} = j | X_m = i) = (P^n)_{ij}, \quad i, j \in \mathcal{S}.$$

We say that a random walk is **irreducible** if all the states are communicated, this is, for every  $i, j \in \mathcal{S}$  there exists  $0 \leq m < \infty$  such that  $p_{ij}^{(m)} > 0$ .

Furthermore, the chain is said to be **regular** if for some  $n_0 < \infty$  we have

$$p_{jk}^{(n_0)} > 0, \quad \text{for all } j \text{ and } k.$$

Also, we say that the **period**  $d(j)$  of a state  $j$  is  $d(j) = \gcd\{n : p_{jj}^{(n)} > 0\}$ . If  $d(j) = 1$  then  $j$  is **aperiodic**.

We say that a transition matrix is **doubly stochastic**, if it satisfies that  $\sum_{j \in \mathcal{S}} p_{jk} = 1$ , that is the sum along the columns is also 1.

In addition, we refer to an irreducible chain  $X$  with a stationary distribution  $\pi$  that satisfies the following conditions as **reversible**:

$$\pi_j p_{jk} = \pi_k p_{kj}. \quad (3.1)$$

Once that we have talked about these definitions, we can introduce the algorithm.

The idea goes as follows: Starting from the complete graph with  $n$  vertices. First, we choose a vertex uniformly from the  $n$  vertices, this vertex will be the root of our tree. Once we have a root, we run a random walk on the graph, such that its values are determined by choosing uniformly at random a vertex from the  $n$  vertices in the graph. When the random walk arrives at a vertex that has not been

visited before, we keep both the vertex and the edge along which it was reached. We stop when all vertices have been visited; this happens almost surely thanks to the attributes of the RW. We see this in more detail on the next pages.

The idea of the algorithm is clear thanks to the previous paragraph, however we would like to program the algorithm in order to give a concise example.

To make the codification of the algorithm easier, we next present a more concise form to write the algorithm. It is important to remark that the algorithm focuses on the edges of the tree, not on the vertices, so if we want to program the algorithm it is only necessary to return the edges once the algorithm ends.

In what follows, let us denote by  $dunif(n,a,b)$  to the function that simulates  $n$  values from a random variable that follows discrete uniform distribution over the set  $\{a, a + 1, \dots, b\}$ . Now the pseudo-code is presented as **Algorithm 1**.

---

**Algorithm 1** Aldous-Broder algorithm

---

**Require:**  $n \in \mathbb{N}$ : number of vertices on Graph.

$root \leftarrow duni f(1, 1, n)$ ,  $S_0 \leftarrow root$ ,  $N \leftarrow \{\}$ ,  $i \leftarrow 1$ ,  $E \leftarrow \{\}$ .

**while**  $i < n$  **do**

$S_1 \leftarrow duni f(1, 1, n)$ .

**if**  $S_1 \notin N$  **then**

        Add  $S_1$  to  $N$ .

        Add  $(S_0, S_1)$  to  $E$ .

$i \leftarrow i + 1$

**end if**

$S_0 \leftarrow S_1$

**end while**

**return**  $E$

**Ensure:** the resulting rooted tree is uniform on  $\mathbb{T}_n^*$ .

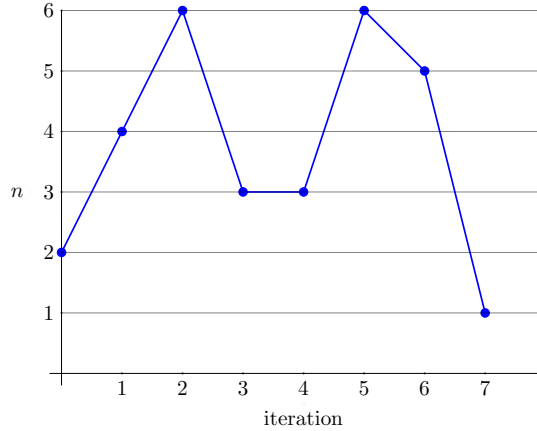
---

**Remark 3.1.** *Even though, the algorithm only returns the set  $E$ , the root of the resulting tree can be obtained by looking at the first component of the first pair  $(S_0, S_1)$ .*

To make this algorithm easier to understand let us look at an illustration of how it works.

### Illustration of the algorithm

For this example, we will use  $n = 6$ . As we said before, the fundamental object of the algorithm is the random walk. Therefore, as a first step, we show the random walk that was used to build the URT. The sample of the walk corresponds, in Algorithm 1, to the numbers  $duni f(1, 1, n)$ . This random walk was obtained using code that you can find in [Appendix A](#). See [Figure 3.4](#).



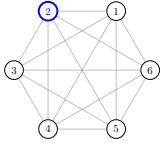
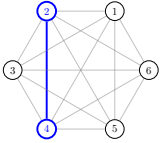
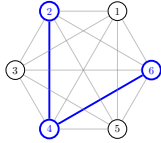
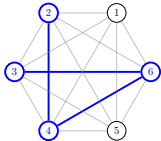
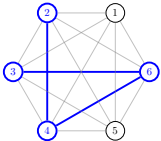
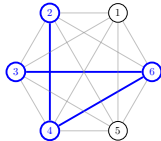
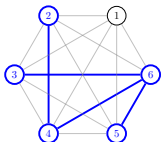
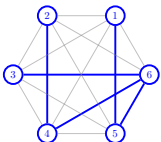
**Figure 3.4.** Random Walk of Algorithm 1.

After looking at [Figure 3.4](#) and *performing* what the algorithm would do, we can build [Table 3.1](#).

Let us discuss the example shown in [Table 3.1](#). For the first three iterations we added new vertices to our random tree on every step, because the RW had not visited any of them. In the following two iterations the tree did not grow, the reason was that the RW visited the state 3 two times in a row and then visited the state 6, which had already been visited, so the tree did not grow. Finally, during the last two iterations, the tree grew, and on the seventh iteration, as we had already visited all of the nodes, the algorithm stops.

Let us remark that the tree we obtained is rooted at 2, even though our planar representation of the tree does not clearly reflect this, as we have not highlighted the root in any way. Up next we will depict a planar representation of the tree that clearly states the root of the tree. To understand [Figure 3.5](#), let us remark that the  $y$  axis represents the



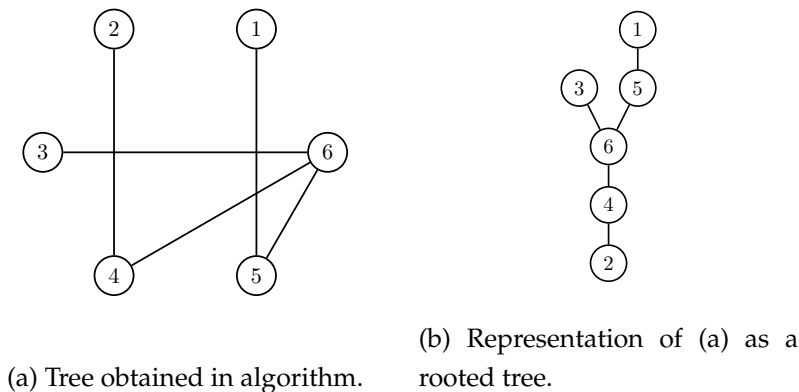
Graph			
$k$	0	1	2
$S_k$	2	4	6
Graph			
$k$	3	4	5
$S_k$	3	3	6
Graph			
$k$	6	7	
$S_k$	5	1	

**Table 3.1.** Illustration of the growth of the tree with  $S_k$ .

*generation* of the node, and we consider our root as the first member of the genealogy.

This URT in particular has an interesting form, we got unlucky when we ran the RW so our resulting tree looks like a line that goes through the nodes, but the 6th node had a couple of "descendants", so it shows a more general idea of how rooted trees could look like. See [Figure 3.5](#).

From what we saw in this example, there are some questionable choices that the previous algorithm does. For instance, the fact that



**Figure 3.5.** Planar representations of the tree obtained by performing the algorithm with  $n = 6$ .

there are steps on which we do not add new edges can be problematic. Actually, the steps at which we add new edges are separated by a geometrically-distributed number of steps on which we add no edges. So if  $n$  is large, the algorithm is slow. We could see that with a small  $n = 6$  we had 3 iterations where the tree did not grow. Also, we can note that nothing changes if we permute all of the vertex-labels uniformly, i.e. we still have a uniform random tree, so we could also remove the labelling from the algorithm and do it at the end. These deficiencies in the algorithm give us a valid excuse to look for an alternative. This alternative was presented by **Aldous** in [1] and we explore it after proving that the Aldous-Broder algorithm actually does what we said it does.

We have to prove that indeed this tree is uniform on  $\mathbb{T}_n^*$ . In the proof we will first examine the random walk used in the algorithm, once we see that it satisfies certain properties we extend these to the trees that it generates.

**Proposition 3.1.** *The tree given by Algorithm 1 is uniform on  $\mathbb{T}_n^*$ .*

The following proof was inspired by the sketch in [9]. The details of the proof were completed thanks to personal communication with Christina Goldschmidt.

*Proof.* In the algorithm the random walk  $\mathbf{S} := (S_k, k \geq 0)$  is of vital importance, so we start this proof by examining the walk as we said earlier. Let us note that the state space of the walk is  $\mathcal{S} = [n]$ .

The transition matrix  $P$  associated to  $\mathbf{S}$  has the following form:

$$(P)_{ij} = \frac{1}{n} \quad \text{for } i, j \in \mathcal{S}.$$

As a first step, we want to show that  $\mathbf{S}$  has a uniform stationary distribution and is reversible.

From the transition matrix we can see that it is irreducible because  $\frac{1}{n} > 0$ . Furthermore, we can see that for every  $j \in \mathcal{S}$  the period of  $j$  is 1. Since  $1 \in \{n \geq 1 : p_{ij}^{(n)} > 0\}$ ,  $\forall i, j$  we have that  $d(i) = 1$  for all  $i \in \mathcal{S}$ . Therefore, the random walk is aperiodic. Then  $P$  is irreducible and aperiodic therefore using Prop. 3.20 of [15], we can conclude that  $P$  is regular.

Turns out  $P$  has another important characteristic, it is doubly stochastic. It is easy to see that our transition matrix satisfies this property since:

$$\sum_{j \in \mathcal{S}} p_{jk} = \sum_{j=1}^n \frac{1}{n} = \frac{n}{n} = 1.$$

Finally we can see that because  $P$  is finite, regular and doubly stochastic, using Prop. 3.21 of [15] we conclude that it has a uniform stationary distribution.

Up next, we want to see that it is reversible. As our stationary distribution is uniform, we have that  $\pi_j = \frac{1}{n}$  for every  $j \in \mathcal{S}$ , and by

remembering that every entry of our matrix  $P$  is  $\frac{1}{n}$ , then we have that for every  $k, j \in \mathcal{S}$

$$\pi_j = \pi_k \quad \text{and} \quad p_{jk} = p_{kj},$$

then  $\mathbf{S}$  satisfies equation (3.1), and therefore it is reversible. Hence we have  $(S_k, -\infty < k < \infty)$ .

However, we want to talk about the trees not the walk that builds them. Recall  $\mathbb{T}_n^*$  the set of rooted spanning trees in  $G$ , where  $G$  is the complete graph of  $n$  vertices, and define  $X_j$  as the tree constructed by  $(S_j, S_{j+1}, \dots)$ . That is, the tree rooted at  $S_j$  and with edges given by

$$(S_{T_v^j-1}, S_{T_v^j}); \text{ for } v \neq S_j,$$

where  $T_v^j = \min\{k \geq j : S_k = v\}$ .

Let us translate the properties that the walk has to the trees that it creates, to do so, we have to think of the algorithm as an stochastic process. To understand process  $X_j$ , think about the table that was shown in the illustration of the algorithm (Table 3.1). That tree is the resulting tree of the walk started at time 0, and depending on the time  $j$  that we pick to perform the algorithm (with the sequence  $(S_j, S_{j+1}, \dots)$ ), the resulting tree will change.

Consider the transition matrix  $Q$  for the chain in reversed time:

$$Q(\theta, \theta') = \mathbb{P}(X_{-1} = \theta' | X_0 = \theta),$$

where  $\theta, \theta' \in \mathbb{T}_n^*$ .

We can see that the process  $(X_j; -\infty < j < \infty)$  is irreducible. To do so, we have to show that for every tree  $\theta \in \mathbb{T}_n^*$  there exists a finite path  $(x_0, x_1, \dots, x_C)$  which visits every vertex and is such that the tree constructed is  $\theta$ . This is not so hard to see. Consider the "depth first

search" of the tree  $\theta$ , as we defined it in Chapter 2 therefore  $X_j$  is irreducible.

Also,  $X_j$  is stationary since we had that  $S_j$  was stationary. Then, we make two affirmations:

- (i) Given  $\theta$  there are exactly  $n$  trees  $\theta'$  such that  $Q(\theta, \theta') = \frac{1}{n}$ .
- (ii) Given  $\theta'$  there are exactly  $n$  trees  $\theta$  such that  $Q(\theta, \theta') = \frac{1}{n}$ .

So let us see the veracity of this affirmations, first, given  $S_0 = \theta$ , with  $\text{root}(\theta) = v$ , say, the  $S_{-1}$  has chance  $\frac{1}{n}$  to choose each of the  $n$  neighbours of  $v$ , and for each of them,  $X_{-1}$  is a different tree  $\theta'$ . On the other hand, given  $X_{-1} = \theta'$ , the first vertex of  $X_0$  is not necessarily the root of  $X_{-1}$ , thus, the tree needs to grow to the right of the last vertex that constructed  $X_{-1}$ . Suppose that the last edge of  $X_{-1}$  is obtained in time  $\zeta$ , then  $S_{\zeta+1}$  has chance  $\frac{1}{n}$  to choose each of the  $n$  neighbours of that last vertex, and for each of them,  $X_0$  is a different tree  $\theta$ .

Now, (i) and (ii) gives us that  $Q$  is doubly stochastic, and since it is also aperiodic then  $Q$  has a uniform stationary distribution on  $\mathbb{T}_n^*$ .

■

After proving the algorithm indeed, provides a tree uniform on  $\mathbb{T}_n^*$ , we can give some closing thoughts. As we said before, the algorithm is not efficient. Recalling that one of our interests in algorithms is to see the behaviour of the trees when  $n$  grows, we need to have a more efficient algorithm.

### 3.2.2 The Aldous algorithm

First, we start from a single vertex labeled 1. Then, for  $2 \leq i \leq n$ , we connect vertex  $i$  to vertex  $V_i$  by an edge, where

$$V_i := \begin{cases} i-1 & \text{with probability } 1 - \frac{i-2}{n-1}, \\ k & \text{with probability } \frac{1}{n-1} \text{ for } 1 \leq k \leq i-2. \end{cases} \quad (3.2)$$

And finally, we uniformly permute the vertex labels.

Unlike the first algorithm, this one "grows" the tree, in the sense that we do not have an underlying graph on which we move. We can think that the algorithm grows "branches" of the tree as paths of consecutive vertex-labels with a random length. The paths end whenever they reach a vertex  $V_i \neq i-1$ . The first path that "breaks" has length

$$C_1^n := \inf\{i \geq 2 : V_i \neq i-1\}. \quad (3.3)$$

Then  $C_1^n$  describes the first branch of the URT. Since we are interested in the asymptotic behaviour of URTs now we study the length of the branches of the tree as the amount of vertices grows.

### 3.3 The branches of a URT

Let us say that  $T_n$  is a URT given by the last algorithm. We can get a glimpse to the scaling behaviour of  $T_n$  by looking at the next proposition which will help us understand better how the tree behaves as it grows.

**Proposition 3.2** (Goldschmidt [9]). *We have the following convergence in distribution*

$$\frac{C_1^n}{\sqrt{n}} \xrightarrow[n \rightarrow \infty]{(\mathcal{L})} C_1,$$

where  $\mathbb{P}(C_1 > x) = \exp \left\{ -\frac{1}{2}x^2 \right\}$ ,  $x \geq 0$ .

*Proof.* Since  $C_1^n$  is a discrete random variable,

$$\mathbb{P}(C_1^n > x\sqrt{n}) = \mathbb{P}(C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1).$$

Hence, we will analyze the event  $\{C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1\}$  to get the limit as  $n \rightarrow \infty$ . According to the definition of  $C_1^n$  given on (3.3) we have that

$$V_i = i - 1 \quad \text{for } i = 2, \dots, \lfloor x\sqrt{n} \rfloor.$$

This implies, together with the definition of  $V_i$  given in (3.2):

$$\mathbb{P}(C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1) = \prod_{i=2}^{\lfloor x\sqrt{n} \rfloor} \left( 1 - \frac{i-2}{n-1} \right).$$

By performing the change of variable  $j = i - 2$  we have:

$$\mathbb{P}(C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1) = \prod_{j=0}^{\lfloor x\sqrt{n} \rfloor - 2} \left( 1 - \frac{j}{n-1} \right).$$

Also, we can see that when  $j = 0$  the argument of the product equals to 1, so we can exclude the first term of the product:

$$\mathbb{P}(C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1) = \prod_{j=1}^{\lfloor x\sqrt{n} \rfloor - 2} \left( 1 - \frac{j}{n-1} \right).$$

Taking logarithms and using the Taylor expansion, we have:

$$\begin{aligned}
\log \mathbb{P}(C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1) &= \log \left[ \prod_{j=1}^{\lfloor x\sqrt{n} \rfloor - 2} \left( 1 - \frac{j}{n-1} \right) \right] \\
&= \sum_{j=1}^{\lfloor x\sqrt{n} \rfloor - 2} \log \left( 1 - \frac{j}{n-1} \right) \\
&= \sum_{j=1}^{\lfloor x\sqrt{n} \rfloor - 2} -\frac{j}{n-1} + o(1) \\
\log \mathbb{P}(C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1) &= -\frac{(\lfloor x\sqrt{n} \rfloor - 2)(\lfloor x\sqrt{n} \rfloor - 1)}{2(n-1)} + o(1).
\end{aligned}$$

Finally,

$$\log \mathbb{P}(C_1^n \geq \lfloor x\sqrt{n} \rfloor + 1) \xrightarrow{n \rightarrow \infty} -\frac{x^2}{2}.$$

Since the exponential function is continuous we can conclude that

$$\mathbb{P}(n^{-1/2}C_1^n > x) \xrightarrow{n \rightarrow \infty} \exp \left\{ -\frac{1}{2}x^2 \right\}$$

This is exactly what we wanted to prove. ■

As we said, we introduced these URTs to see a convergence to the CRT, however, we have not yet presented the CRT. On the following section we will present a new characterization of the CRT.

### 3.4 Construction of the CRT

From what we have seen until this point, we can develop the intuition that there are at least two ways of characterizing trees, in the first chapter we characterize the CRT via a formal construction, based on contour functions that helped us to see the relation with Galton



Watson Trees. However on this chapter we have seen that it is not as simple to do this kind of construction on all types of random trees. The fact that we are talking about another construction of the CRT may hint that we can also define that object via algorithms, now the natural question is, how do we do so?

Let us describe in a verbal manner the process that will build our continuum tree, this process is known as the *line-breaking construction*.

This process is indexed on time. We take the half-line  $[0, \infty)$ . Consider  $C_1, C_2, \dots$  the jump times of a non-homogeneous Poisson process with rate  $r(t) = t$ . We are going to cut the half-line into intervals  $[C_i, C_{i+1})$ . To build the tree, we start with the line segment  $[0, C_1)$ , we grow the tree inductively by pasting  $[C_i, C_{i+1})$  as a branch attached to a random point  $B_i$ , chosen uniformly over the existing tree. The tree will be the closure of the tree at time infinity.

This process shows us how we build the continuum tree, however if we want to make it more rigorous we have to define some things first.

### 3.4.1 Preliminaries for the construction of the CRT

We want to make our construction rigorous, and also we are interested in showing a convergence to the object that we will define. If you recall what we saw in Theorem 2.3 to show convergence to the CRT we needed to re-scale trees. Scaling an abstract object as a tree does not make sense, so, in order to be able to re-scale these objects, we have to embed them into a metric space.

We still want them to have the properties they have in their abstract setting, that is, we want the embedded trees to not have cycles. Therefore, we do not want intersections. To avoid intersections we can

look for a infinite dimensional linear space. A suitable candidate is the space  $(\ell_1, \|\cdot\|)$ . That is the Banach space of absolutely additive real sequences  $x = (x_1, x_2, \dots)$  with norm  $\|x\| = \sum |x_i|$ .

Let  $z_i = (0, \dots, 0, 1, 0, \dots)$  be the  $i$ th unit vector in the natural "basis" of  $\ell_1$ . Even though we are in a Banach space and we do not have a inner product, for simplicity we call the vectors  $z_i$  orthogonal.

This space serves multiple purposes: firstly, it allows us to establish simple rules for constructing trees within it. Additionally, within this space, we can identify a projection of the tree where the distances between vertices preserve the values obtained with the graph distance on the original tree.

To represent the trees that we build, we need to introduce the **set representation** of a tree  $\mathbf{t}$ . This is a labelling of the vertices as a subset  $S = \{v, w, \dots\} \subset \ell_1$  in such a way that  $d_{\mathbf{t}}(v, w) \equiv \|v - w\|$ . That is, we look for subsets of  $\ell_1$  such that there is an isomorphism between the embeded tree and the abstract tree.

One can see that this representation exists if we pick a vertex of  $\mathbf{t}$  and label it  $\emptyset$ . Then we order the edges arbitrarily as  $e_1, \dots, e_{n-1}$ . And we use a map  $\varphi : \mathbf{t} \rightarrow \ell_1$  to embed the tree in  $\ell_1$ :

$$\varphi(v) = \sum_i z_i J(v, e_i), \quad v \in \mathbf{t},$$

where  $J(v, e) = 1$  if  $e$  is in the path from  $v$  to  $\emptyset$ , and  $J(v, e) = 0$  in other case. That is,  $J(v, e)$  indicates if an edge is part of the path between a vertex and the root<sup>4</sup>. Then

$$\|v - w\| = \sum_i |J(v, e_i) - J(w, e_i)|,$$

---

<sup>4</sup>Recall that since we are working with trees, we only have one possible path between two given vertices.

which agrees with the graph distance  $d_{\mathbf{t}}(v, w)$  defined in (1.1).

We can see that this is true because, recalling the definitions made in Chapter 1.  $J(v, e)$  indicates which edges are part of the ancestral line of the vertex  $v$ . Then the sum of them gives us the distance between  $v$  and the root. When we take two vertices  $v$  and  $w$ , and see in which edges both  $J(v, e)$  and  $J(w, e)$  are equal to one, then we have that they share that edge in their ancestral line. However, on the edges where they differ, corresponds precisely to the path between the vertices. Therefore, the term  $|J(v, e_i) - J(w, e_i)|$  tells us if the edge  $e_i$  is in the path between both vertices or if the edge is on their shared ancestral line. Summing over  $i$ , one obtains the distance on the graph between both vertices.

The "structure" in which we embed the tree is given by  $S$ , however we do not know anything about the labels of the tree. Therefore  $S$  determines  $\mathbf{t}$  as an unlabelled tree, and if we are given a tree  $\mathbf{t}$  we have many different possible set-representations.

A measure-representation  $\mu$  of  $\mathbf{t}$  is a probability measure on  $\ell_1$  which is uniform on some set representation  $S$  of  $\mathbf{t}$ .

These two representations will define our trees. As we saw in the previous chapter, we are interested in convergence. Then, we need to define convergence of closed subsets of  $\ell_1$  (for the set-representations) and we need convergence of probability measures.

For closed subsets of  $\ell_1$ , convergence shall mean convergence in the Hausdorff metric, defined in Chapter 1. For probability measures on  $\ell_1$ , convergence shall mean weak convergence (or convergence in distribution). It is important to denote that neither implies the other.

However, we recall that we are talking about random objects, therefore we need to make a clarification. For random trees  $\mathcal{J}_n$  there exists random set-representations  $S_n$  and corresponding random

measure-representations  $\mu_n$ , and so we can talk about convergence in distribution,

$$\begin{aligned} n^{-1/2} S_n &\xrightarrow[n \rightarrow \infty]{(\mathcal{L})} S \\ n^{-1/2} \mu_n &\xrightarrow[n \rightarrow \infty]{(\mathcal{L})} \mu \end{aligned}$$

where now the limits are random sets and random measures. All our spaces are Polish, that is, metric, complete and separable, so in place of "convergence in distribution" we could say "there exist versions which converge a.s."

After this preamble, we can finally present the rigorous construction of the continuum tree using the *line-breaking construction*.

### 3.4.2 The line-breaking construction

Let  $C_0 = B_0 = 0$ , recall that  $C_i$  are the times of a non-homogeneous Poisson process with rate  $r(t) = t$  and  $B_i$  are the points on which we paste the next branch on the existing tree. Now to define  $B_i$  we have to remember that it chooses a point uniformly over the already constructed tree, therefore, we can define  $B_i$  as  $\xi_i C_i$ , where  $\xi_i \sim \text{Unif}(0, 1)$  and is independent of the  $C_i$ 's. We lack a way to go from  $[0, \infty)$  to  $\ell_1$ , then let us define  $\rho : [0, \infty) \rightarrow \ell_1$  as  $\rho(0) = 0$ , and

$$\rho(x) = \rho(B_i) + (x - C_i)z_{i+1}, \quad \text{for } C_i < x \leq C_{i+1}, \quad i \geq 0$$

where  $z_i$  are the unit vectors of the natural basis of  $\ell_1$ .

To better understand  $\rho$ , we have to see that it is a projection, from the abstract tree to the space  $\ell_1$ . This mapping "pastes" the branch to the tree on  $\ell_1$ , so first,  $\rho(B_i)$  places you on the point chosen uniformly of the existing tree on  $\ell_1$ , and then, you paste the branch on  $z_{i+1}$  until

you reach the next cutting time. Write

$$\begin{aligned}\mathcal{J}_t &= \rho([0, t]), \\ \mathcal{J}_\infty &= \rho((0, \infty)), \\ \mathcal{J} &= \overline{\mathcal{J}_\infty}.\end{aligned}$$

The fundamental result in which we support the rest of this chapter is that  $\mathcal{J}$  is almost surely compact. This immediately implies that  $\mathcal{J}_t \xrightarrow{a.s.} \mathcal{J}$  in the Hausdorff metric. Think of  $\mathcal{J}$  as the set-representation of some (Platonic) continuum random tree. More information is contained in its measure-representation  $\mu$  specified by the following theorem. The uniform probability distribution on  $[0, t]$  induces a distribution on  $\mathcal{J}_t$  and hence on  $\ell_1$ ; call this random probability measure  $\mu_t$ .

**Theorem 3.1** (Aldous [1]). *With probability 1,*

- (i)  $\mathcal{J}$  is compact,
- (ii)  $\mu_t \rightarrow \mu$  as  $t \rightarrow \infty$ , for a certain  $\mu$ ,
- (iii)  $\mathcal{J}$  is the support of  $\mu$ ,
- (iv)  $\mu(\mathcal{J}_\infty) = 0$ .

The Proof of Theorem 3.1 can be found in [1].

We can see that  $\mathcal{J}$  is "tree-like" in several senses. Call  $\mathcal{J}_\infty \setminus \cup_{i=0}^\infty \{\rho(C_i)\}$  the "skeleton" of  $\mathcal{J}$ , and call the remainder of  $\mathcal{J}$  the "leaves".

The "skeleton" is all the tree formed by the process except for the end points of every branch that we pasted on the tree. We can clearly see why the remainder of  $\mathcal{J}$  are called "leaves".

For each pair of points  $x, y$  in  $\mathcal{J}$  there is a unique path in  $\mathcal{J}$  from  $x$  to  $y$ , and this path lies within the skeleton except at  $x$  and  $y$ . Removing

a point  $x$  in the skeleton will disconnect  $\mathcal{J}$ , whereas removing a leaf  $x$  will not.

We have characterized the continuum random tree in a different manner and now we can finally observe how the convergence works.

### 3.5 Convergence to a CRT

We are looking for a finite set-representation that converges to the (Platonic) set representation of a CRT. The claim that the remainder of this chapter focuses on proving is that the Aldous algorithm gives us such finite representation.

As a first step we find helpful to rewrite the random variable  $V_i$  (defined in Aldous algorithm) as follows:

$$V_i = \min\{U_i, i - 1\}; \quad (3.4)$$

where  $U_2, \dots, U_n$  are independent random variables with distribution uniform on  $[n]$ . The argument to see that these two forms of defining the  $V_i$ 's are equivalent is rather simple. Let us begin by analyzing the probability that  $V_i = U_i$ :

$$\begin{aligned} \mathbb{P}(V_i = U_i) &= \mathbb{P}[\min(U_i, i - 1) = U_i] = \mathbb{P}(U_i < i - 1) \\ &= \mathbb{P}(U_i \leq i - 2) = \sum_{k=1}^{i-2} \mathbb{P}(U_i = k) = \sum_{k=1}^{i-2} \frac{1}{n-1} \\ &= \frac{i-2}{n-1}. \end{aligned}$$

Then, it is clear that  $\mathbb{P}(V_i = i - 1) = 1 - \frac{i-2}{n-1}$ . Also note that  $U_i$  can be any value between 1 and  $i - 2$ . Therefore is clear that the definitions (3.2) and (3.4) are equivalent.

This reformulation helps us see a fundamental similarity between the processes. On the Aldous algorithm, at each step we grow a branch, and at certain times these branches are cut and we start a new growth. From the way we rewrite  $V_i$ , presented on (3.4), the place where we begin growing a new branch-growth is uniformly chosen from the vertices on the already existing tree. So we can see why this algorithm could give us a form of constructing the CRT.

Let us call the times  $i$ , where  $U_i < i - 1$  **cut-times**. In a similar fashion to what we did when we build the CRT, we construct the tree by adding unit edges in the same direction between cut-times, forming a branch; and at a cut-time, we start a new branch by adding edges in a new, orthogonal direction from a random vertex already constructed.

Formally, recall that  $(z_i; i \geq 1)$  is the natural unit vector basis of  $\ell_1$ . Fix  $n$  and let  $(U_i; 2 \leq i \leq n)$  be independent uniform on  $[n]$ . Define  $J_1 = 1$ ,

$$J_i = \begin{cases} J_{i-1} + 1 & \text{if } U_i < i - 1, \\ J_{i-1} & \text{if } U_i \geq i - 1. \end{cases} \quad (3.5)$$

Note that  $J_i$  counts every time that we make a cut. Therefore, we can think of it as keeping track of the amount of branches that we have at any moment.

Then, following the algorithm we can construct a set representation  $\mathcal{J}_n = \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$  of  $T_n$  that can be defined as follows:  $\mathcal{V}_1 = 0$ ,

$$\mathcal{V}_i = \mathcal{V}_{\min(U_i, i-1)} + z_{J_i}, \quad i > 1.$$

Let  $\mu_n$  be the corresponding measure-representation. For  $t \leq n$  we can consider the **partial tree**

$$\mathcal{J}_n(t) = \{\mathcal{V}_1, \dots, \mathcal{V}_{\lfloor t \rfloor}\}. \quad (3.6)$$

Let  $\mu_n(t)$  be the uniform probability distribution on  $\mathcal{J}_n(t)$ . Once we have defined these concepts, we can present the following theorem.

**Theorem 3.2** (Aldous [1]). *Let  $\mathcal{J}, \mu$  be the set- and measure- representations of the continuum tree, as in Theorem 3.1. Then as  $n \rightarrow \infty$ ,*

$$\left(n^{-1/2} \mathcal{J}_n, n^{-1/2} \mu_n\right) \xrightarrow{(\mathcal{L})} (\mathcal{J}, \mu).$$

The proof of this theorem requires domain of skills that surpass the undergraduate level. Therefore, we present the main ideas behind the proof so our readers can get a glimpse of the key arguments. But before we do that, we can use the theorem to visualize a CRT.

## 3.6 A Continuum Random Tree

The theorem presented in the last section gives us the chance to visualize how this convergence works thanks to a codification of Aldous algorithm.

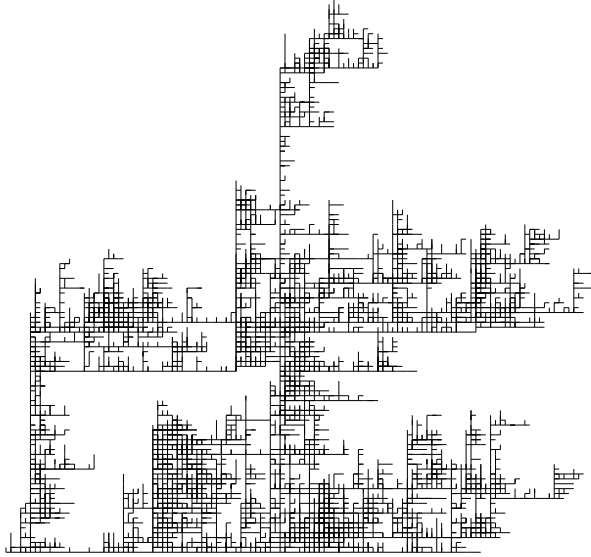
One thing that we must highlight is that the object presented before exists in infinite-dimensional vector spaces. Since we have not yet discover how to make plots in  $R^\infty$ , the illustration will lack the tree form, as we were left without enough directions to draw the tree without intersections.

Once that we have depicted a CRT in Figure 3.6, we can present the ideas behind the proof of the theorem.

## 3.7 The idea behind the proof

To prove the theorem, it is enough to show the convergence of the partial trees to  $(\mathcal{J}_t, \mu_t)$ , for fixed  $t \in (0, \infty)$ . This is enough thanks to





**Figure 3.6.** A CRT generated using Aldous algorithm.

the fact that Theorem 3.1 establishes that  $(\mathcal{J}_t, \mu_t) \xrightarrow{a.s.} (\mathcal{J}, \mu)$ . Therefore if we show the convergence of the partial trees for fixed  $t$ , we can assure the convergence stated in the theorem.

For each  $n$ , construct  $(C_j^n, B_j^n)$ ,  $j \geq 1$  as follows:  
Let  $C_j^n$  be the  $j$ th cut-time, that is, the  $j$ th element of  $\{i : U_i < i - 1\}$ , and  $B_j^n$  is the corresponding value of  $U_i$ , that is  $B_j^n = U_{C_j^n}$

As a first step, we would have to show that

$$\begin{aligned} ((n^{-1/2}C_1^n, n^{-1/2}B_1^n), (n^{-1/2}C_2^n, n^{-1/2}B_2^n), \dots) \\ \xrightarrow{(\mathcal{L})} ((C_1, B_1), (C_2, B_2), \dots) \end{aligned} \quad (3.7)$$

for  $(B_i, C_i)$  as in Section 3.4.2.

Observe that  $B_i \sim \text{Unif}(0, C_i)$  and  $B_i^n \sim \text{Unif}(0, C_i^n)$ , then  $n^{-1/2}B_i^n \sim \text{Unif}(0, n^{-1/2}C_i^n)$ , therefore we only need to see the convergence of  $n^{-1/2}C_i^n$  and as a consequence we would have the convergence of the  $B_i^n$ 's.

In fact, we saw in Section 3.3 that  $n^{-1/2}C_1^n \xrightarrow{(\mathcal{L})} C_1$ , we will not present the rest of the proof of the convergence, but it seems natural that the joint convergence holds as well.

Then (3.7) implies the convergence of the partial trees defined in (3.6). Now our focus changes to proving that for fixed  $0 < t < \infty$ ,

$$(n^{-1/2}\mathcal{J}_n(tn^{1/2}), n^{-1/2}\mu_n(tn^{1/2})) \xrightarrow{(\mathcal{L})} (\mathcal{J}_t, \mu_t); \quad (3.8)$$

where  $\mathcal{J}_t, \mu_t$  are as in the construction of Theorem 3.1.

As we said when we introduced the set and measure representations, the convergence of each is independent from the other. We would proof (3.8) by a routine weak convergence lemma ([4], Theorem 4.2); by using this result, it suffices to prove

$$\lim_{t \rightarrow \infty} \limsup_{n \rightarrow \infty} \mathbb{P}(d_H(n^{-1/2}\mathcal{J}_n(tn^{1/2}), n^{-1/2}\mathcal{J}_n) > \varepsilon) = 0, \quad (3.9)$$

$$\lim_{t \rightarrow \infty} \limsup_{n \rightarrow \infty} \mathbb{P}(d_W(n^{-1/2}\mu_n(tn^{1/2}), n^{-1/2}\mu_n) > \varepsilon) = 0; \quad (3.10)$$

where  $d_H$  is the Hausdorff metric on compact sets and  $d_W$  is a metrization of weak convergence of probability measures.

To highlight the differences between the two arguments, we divide the remanding sketch into two sections.

### 3.7.1 Convergence of set-representations

The objective that we pursue in this part of the proof is to show that

$$\lim_{t \rightarrow \infty} \limsup_{n \rightarrow \infty} \mathbb{P}(d_H(n^{-1/2} \mathcal{J}_n(tn^{1/2}), n^{-1/2} \mathcal{J}_n) > \varepsilon) = 0,$$

which corresponds to equation (3.9).

Define

$$D_n(i, j) := \min_{1 \leq u \leq i} \|\mathcal{V}_j - \mathcal{V}_u\|.$$

This gives us the distance on the tree between the point labelled  $j$  and the partial tree  $\mathcal{J}_n(i)$ . Since (3.7) shows that the partial trees converge in distribution to  $\mathcal{J}_t$  then we can rephrase (3.9). Recall the definition of the Hausdorff metric shown in Chapter 1, equation (1.2). Then (3.9) can be rephrased as,

$$\lim_{t \rightarrow \infty} \limsup_{n \rightarrow \infty} \mathbb{P} \left( \sup_j D_n(\lfloor tn^{1/2} \rfloor, j) > \varepsilon n^{1/2} \right) = 0. \quad (3.11)$$

This is because in (3.7) we showed that the partial trees converge to  $\mathcal{J}_t$ . Therefore we only need to make comparisons within partial trees.

The following lemma will be useful to prove the last fact.

**Lemma 3.1** (Aldous [1]). *For integers  $1 \leq i < j \leq n$ ,  $b \geq 1$ ,*

$$\mathbb{P}(D_n(i, j) > b) \leq \left(1 - \frac{i}{n}\right)^b.$$

This lemma tells us that the distances  $D_n$  are bounded by a geometric distribution.

Thanks to that fact, following a similar argument to the one shown in Lemma 6 of [1], we can establish the following inequality.

There exist  $k_0 < \infty$  and  $A < \infty$  such that for all  $k \geq k_0$  and all  $n$ ,

$$\mathbb{P}(\max_{u \leq n} D_n(\lfloor n^{1/2} e^k \rfloor, u) \geq 6ke^{-k} n^{1/2}) \leq Ae^{-k}.$$

This last bounds yields (3.11) because the term in the right hand side converges to zero when  $n$  grows.

### 3.7.2 Convergence of probability measures

The first step to see how this convergence can be proven is introducing the metric  $d_W$ . To do so, we need to recall some analytic facts:

Given a metric space  $(E, d)$ , recall that for a real-valued function  $f$  on  $E$ , the Lipschitz seminorm is defined by

$$\|f\|_L := \sup_{x \neq y} \frac{|f(x) - f(y)|}{d(x, y)}.$$

Then  $\|f\|_L = 0$  if and only if  $f$  is a constant function (hence the term "seminorm"). Call the supremum norm  $\|f\|_\infty := \sup_x |f(x)|$ . Let  $\|f\|_{BL} := \|f\|_L + \|f\|_\infty$ . Here "BL" stands for "bounded Lipschitz".<sup>5</sup>

We introduce Lipschitz functions because, on general metric spaces, the most natural form of smoothness comes from the conditions that define them.

To show convergence we need a metric, and thanks to our little deviation towards analysis we can define it. Let  $d_W$  be defined as

$$d_W(\theta_1, \theta_2) = \sup \left\{ \left| \int f d\theta_1 - \int f d\theta_2 \right| : \|f\|_{BL} \leq 1 \right\}.$$

One can prove that  $d_W$  is a metric on the set of all laws on  $\ell_1$ . See Section 11.3 from [7].

The following fact helps us simplify the arguments needed to proof the convergence.

---

<sup>5</sup>All of the definitions given in this paragraph were obtained from [7].

**Lemma 3.2.** *Let  $\theta_1, \theta_2$  be probability measures. Let  $K_1, \dots, K_N$  be disjoint sets, of diameter at most  $d_0$ , which cover the support of  $\theta_1$ . Then,*

$$d_W(\theta_1, \theta_2) \leq d_0 + \sum_i |\theta_1(K_i) - \theta_2(K_i)|.$$

This last lemma gives us a upper bound to the  $d_W$  metric that we just introduced.

We have to introduce more definitions to work with the measures. Define  $\ell_1^k$  as the vectors formed by the first  $k$  elements of the sequences in  $\ell_1$ . Let  $\pi_k : \ell_1 \rightarrow \ell_1^k$  be the projection  $\pi_k(x) = (x_1, \dots, x_k)$ . Then  $\pi_k$  acts on measures in the natural way.

Let us fix  $n, M$  and condition on the partial tree  $\mathcal{J}_n(C_M^n)$ , that is the tree consisting of only the first  $M$  branches. Let  $K$  be a subset of  $\ell_1^M$ . For each  $j \geq C_M^n$ ,  $\pi_M(\mu_n(j))$  is a probability measure on  $\ell_1^M$ , and  $\pi_M \mu_n(j)(K)$  denotes the mass it assigns to  $K$ .

Now, using Lemma 3.2 we can prove the convergence using this following bound.

$$\begin{aligned} & \max_K \mathbb{P} \left( \max_{j \geq C_M^n} |\pi_M \mu_n(j)(K) - \mu_n(C_M^n)(K)| > \delta \middle| \mathcal{J}_n(C_M^n) \right) \quad (3.12) \\ & \leq \delta^{-2} A(n, M), \end{aligned}$$

where  $A(n, M)$  are random variables such that

$$\lim_{M \rightarrow \infty} \limsup_{n \rightarrow \infty} EA(n, M) = 0. \quad (3.13)$$

Let us see that we can model the growth of the tree as a urn model. The Urn model usually consists of two random variables, let us say  $U_m$  and  $V_m$ , this variables denote the number of black and white balls in an urn at time  $m$ . We choose a color by touching a uniform random ball in the urn; then we add  $\Delta_m$  balls of that color, where  $\Delta_m$  is random with

arbitrary distribution that varies with  $m$ , but it is independent from the choice of the color.

To apply this to our trees, color black the vertices of the tree at time  $C_M^n - 1$  which are in  $K$  and the other vertices white. Recalling the algorithm, it will grow the tree in segments  $[C_m^n, C_{m+1}^n - 1]$ . Color each new vertex according to the color of the vertex it is attached to. Then, we can see that if we see the tree only at times  $C_m^n - 1$ , then the proportions of black vertices

$$\pi_M \mu_n(C_m^n - 1)(K), \quad m \geq M,$$

form a martingale.

This fact gives that we can find the said random variable  $A(n, m)$ .

To prove convergence, we have to find the partition that allows us to use the results presented before and also we have to justify how the bound (3.12) can be constructed.

# Conclusions

In this thesis we explored one of the most beautiful aspects mathematics has to offer, the ability to find the same answer using methods that appear to be different, but they are essentially not.

Even if the tools and complexity differ greatly between both approaches, we cannot ignore the fact that we observed the same phenomenon in both cases, the convergence of Uniform Random Trees to the Continuum Random Tree. On one of them we used a technical approach that helped diminish the amount of calculus needed to see the nature of the convergence. On the other approach we got our hands dirty trying to grasp the nature of the objects under study.

This presentation of the convergence to the CRT has the purpose of making it accessible on a undergraduate level. Although the topics touched upon here are vastly above the undergraduate level. For the curious reader that wants to keep exploring this world of random graphs, there exist a lot of possible paths to take.

For once, on the same article that introduces the CRT, Aldous also presents  $\alpha$ -stable tree and a self-similar continuum tree. Furthermore, the other questions presented on the abstract of this thesis do not have simple answers, and there are a lot of more questions that one can ask

about random trees.

All of these alternatives to study can get increasingly difficult. As we saw, this "simple" topic made us reach to realms far beyond our understanding. One may argue that the best thing about mathematics is the fact that you do not know everything.



# Appendix A

## Codes

This chapter contains the codes that were used to build the random trees that we presented in Chapter 3.

First we present the code of the Aldous-Broder algorithm.

```
1 #Function that generates pseudo-random values from a discrete
   uniform distribution
2 disunif <- function(a,b){
3   n <- b-a+1
4   u <- runif(1)
5   step <- 1/n
6   p <- 1/n
7   i <- a
8   flag <- TRUE
9   while(flag){
10     if(u<=p){flag <- F}
11     else{
12       p <- p + step
13       i <- i + 1
14     }
15   }
16   return(i)
```

```

17 }
18 #The following function codifies the Aldous Broder algorithm,
    it returns only "E" that is, it returns the matrix of
    adjacent edges described in the pseudo-code presented in
    Chapter 2.
19 AldousBroder <- function(n) {
20   root <- disunif(1,n)
21   S0 <- root
22   N <- rep(0,n)
23   N[1] <- root
24   E <- matrix(rep(0,2*(n-1)),n-1,2)
25   i <- 1
26   while (i<n) {
27     S1 <- disunif(1,n)
28     flag <- S1 %in% N
29     if(!flag) {
30       i <- i+1
31       N[i] <- S1
32       E[i-1,] <- c(S0,S1)
33     }
34     S0 <- S1
35   }
36   return(E)
37 }

```

For the example presented in Figure 3.4, we codified different functions that would show step by step the Aldous-Broder algorithm. To make the images of the Random Walk we used the following code

```

1 randomWalking <- function(n, file) {
2   sink(file,append = TRUE)
3   cat("\n\\begin{center}\\n\\begin{tikzpicture}\\n")
4   root <- disunif(1,n)
5   S0 <- root
6   N <- rep(0,n)
7   N[1] <- root

```

```

8   States <- c(S0)
9   E <- matrix(rep(0,2*(n-1)),n-1,2)
10  i <- 1
11  j <- 1 #Caminata aleatoria
12  while (i<n) {
13    S1 <- disunif(1,n)
14    States[j+1] <- S1
15    flag <- S1 %in% N
16    cat("\\draw[blue, thick]",paste0("(",j-1,"",S0, ")", "
--", paste0("(",j,"",S1, ")", ";", "\n")
17    cat("\\filldraw[blue]",paste0("(",j,"",S1, ")", "circle
(2pt) ;\n")
18    if(!flag){
19      i <- i+1
20      N[i] <- S1
21      E[i-1,] <- c(S0,S1)
22    }
23    S0 <- S1
24    j <- j+1
25  }
26  cat("\\draw[] (-0.2,0)--",paste0("(",j,"",0)", "; \n")
27  cat("\\draw[] (0,-0.2)--",paste0("(0", "",n,")", "; \n")
28  for(i in 1:n){
29    cat("\\filldraw[] ",paste0("(0",",",i,")", " circle(0.2pt) "
,paste0("node[anchor=east]{",i,"};\n"))
30    cat("\\draw[draw opacity=0.5] ",paste0("(-0.2",",",i,")--")
,paste0("(",j,"",",",i,")", "; \n")
31  }
32  cat("\\end{tikzpicture}\n\\end{center}\n")
33  cat("States:", States, "\n")
34  sink()
35  return(E)
36 }

```

Finally to draw the random trees in Table 3.1 that result from the random walk obtained we use the following function

```

1 #Auxiliary functions to draw trees
2 #Auxiliar function that draws nodes in a circular manner
3 Nodewrite <- function(i,a,r){
4   x <- round(r*cos(a), digits = 3)
5   y <- round(r*sin(a), digits = 3)
6   cat("\node",paste0("(",i, ")"), "at",paste0("(",x, ",",y, ") "
7     ),paste0("{",i,"};\n"))
8 }
9 #Function that draws all of the nodes that form a tree
10 NodeDraw <- function(n,file){
11   sink(file,append = TRUE)
12   cat("\n\\begin{scope}[every node/.style={circle,thick,draw}]
13     ", "\n")
14   step <- pi*(2/n)
15   lins <- seq(from = 0, to = 2*pi, by = step)
16   r <- n*(1.5/4) #arbitrary constant that made nice trees
17   for(i in 1:n){
18     a <- lins[i+1]
19     Nodewrite(i,a,r)
20   }
21   cat("\n\\end{scope}", "\n")
22   sink()
23 }
24 TreeFRW <- function(n,S,file){
25   #S is a vector formed by the states of the walk
26   sink(file,append = TRUE)
27   cat("\n\\begin{center}\n\\begin{tikzpicture}\n")
28   sink()
29   NodeDraw(n,file)
30   N <- c(S[1])
31   j <- 2
32   m <- length(S)
33   sink(file,append = TRUE)
34   cat("\n\\begin{scope}[every edge/.style={blue, ultra thick,
35     draw}] ", "\n")

```

```

33   for(i in 1:(m-1)){
34       flag <- S[i+1] %in% N
35       if(!flag){
36           N[j] <- S[i+1]
37           cat("\\path [-]",paste0("(",S[i], ")"), "edge node {}",
38               paste0("(",N[j], ")"), ";", "\\n")
39           j <- j+1
40       }
41       cat("\\end{scope}", "\\n")
42       cat("\\end{tikzpicture}\\n\\end{center}\\n")
43       sink()
44   }

```

# Bibliography

- [1] D. ALDOUS, *The continuum random tree. i*, The Annals of Probability, 19 (1991), pp. 1–28.
- [2] J. BENNIES AND G. KERSTING, *A random walk approach to galton–watson trees*, Journal of theoretical probability, 13 (2000), pp. 777–803.
- [3] N. BERESTYCKI, *Recent progress in coalescent theory*, arXiv preprint arXiv:0909.3985, (2009).
- [4] P. BILLINGSLEY, *Convergence of probability measures*, John Wiley & Sons, 2013.
- [5] J. A. BONDY, *Graph theory with applications*, Macmillan New York, 1982.
- [6] N. CURIEN, *A random walk among random graphs*, October 2024.
- [7] R. M. DUDLEY, *Real analysis and probability*, CRC Press, 2018.
- [8] J.-F. L. GALL, *Random trees and applications*, Probability Surveys, 2 (2005), pp. 245 – 311.
- [9] C. GOLDSCHMIDT, *Scaling limits of random trees and random graphs*, in Random Graphs, Phase Transitions, and the Gaussian Free

Field: PIMS-CRM Summer School in Probability, Vancouver, Canada, June 5–30, 2017, Springer, 2019, pp. 1–33.

- [10] W. D. KAIGH, *An invariance principle for random walk conditioned by a late return to zero*, The Annals of Probability, (1976), pp. 115–121.
- [11] N. J. KALTON AND M. I. OSTROVSKII, *Distances between banach spaces*, Forum Math, (1999), pp. 17–48.
- [12] J.-F. LE GALL AND Y. LE JAN, *Branching processes in lévy processes: the exploration process*, The Annals of Probability, 26 (1998), pp. 213–252.
- [13] P. MÖRTERS AND Y. PERES, *Brownian motion*, vol. 30, Cambridge University Press, 2010.
- [14] J. MUNKRES, *Topology*, Pearson, 2014.
- [15] L. RINCÓN, *Introducción a los procesos estocásticos*, UNAM, Facultad de Ciencias, 2012.
- [16] H. L. ROYDEN AND P. FITZPATRICK, *Real analysis*, vol. 2, Macmillan New York, 1968.
- [17] F. SPITZER, *Principles of random walk*, vol. 34, Springer Science & Business Media, 2013.
- [18] D. STIRZAKER, *Stochastic processes and models*, OUP Oxford, 2005.
- [19] Z. YANG AND D. ALDOUS, *Brownian motion simulation project in r*. <https://www.stat.berkeley.edu/~aldous/Research/Ugrad/ZY1.pdf>, 2023. Accessed July, 2023.

*Dos construcciones para el Árbol Aleatorio*

*Continuo*, escrito por José Carlos Cruz

Aranda, se terminó de imprimir ....