Root-finding Algorithms for Random Trees: Theory and Simulations

Diego A. Baptista Theuerkauf

Prof. Roberto Imbuzeiro

13/06/2018

Abstract

We study the problem of finding the first vertex in a random generated tree. This problem can be thought as finding the source of a rumor spread in a social network. We use three different models for generating these networks, namely, S.I. model, Uniform and Preferential attachment. We present different algorithms that spot the source and we use some results about the ML (Maximum Likelihood) estimator to prove some of their properties. These studied estimators are of two different types: For a given graph G, some of them output single vertices as candidates for the source v_R, and the others return sets H of $V(G)$ that satisfies that (i) with high probability v_R is in H and (ii) the size of H is independent of the size of the growing network G. For networks in which each node has at most 3 neighbors, we show that the probability that the studied estimators find the source goes above 1/4 as the graph grows. On the other hand, this probability goes to 0 for networks in which each node has at most two neighbors. We give lower bounds for the size of the output set H by analyzing the behavior of the ML estimator in the case where the studied network grows following uniform and preferential mechanisms. We do simulations to test the performance of the presented estimators in three different tasks: the first about single source finding in separate graphs, the second related to the size of the output set H and the third one about an one against all tournament.

Keywords: Root-finding Algorithm, S.I.R. model, Uniform Attachment, Preferential Attachment, Barabási-Albert Model, Maximum Likelihood Estimator, Root Estimators.

To my family.

Acknowledgements

I would first like to thank to my thesis advisor Dr. Roberto Imbuzeiro of the Instituto de Matemática Pura e Aplicada, for the continuous support during my master's study, for his constructive criticism, friendly advice, boundless knowledge, patience and motivation.

I also need to thank to the Prof. Jefferson Elbert, Prof. Rodrigo Ribeiro and Prof. Augusto Teixeira, I really appreciate your corrections.

I must express my very profound gratitude to God, to my parents, to my girlfriend and to friends for providing me with unfailing support and continuous encouragement. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

Introduction

Networks and information are a characteristic part of our society. Any relation between people, institutions and countries creates a network and each interaction between these elements gives rise to a process of information spreading. This situation can be thought as a graph which vertices are these entities and the edges between these elements represent the relations between them. For instance, financial market can be viewed as a huge directed multi-relational network, where companies, firms, financial institutions and governments play the role of nodes, and links symbolize different interactions between them. Social networks can be considered as a graph whose nodes are people and whose edges represent any social relationships like blood relation, friendship, etc; The Internet also can be seen as a graph where computers represent the nodes of the graph and links between them are its edges.

This omnipresence of networks makes us more connected beings and therefore, more dependent to others, i.e., more vulnerable. Each single change in the information that one vertex possesses may produce a cascading effect in the state of some of the other nodes of the graph creating variations on the structure of the network. For instance, if one of these networks represents the market, and suddenly one of its vertices, an institution, collapses, then this would produce a financial crisis. If one of these graphs references some Internet social network, then any misinformation given to one of its elements will spread to its neighbors leading to pernicious effects.

All these situations share a common feature: a single vertex receives information and then it sends it to its neighbors, beginning with a chain reaction that could affect the state of the other members of the set. Essentially, this situation can be thought as a *rumor* being told and then spreading over a network.

Many models of information diffusion have been introduced [5],[17]. One of the most studied models corresponds to the one who emulates viral epidemics [3]. Many different results have been established in reference to the well-known *susceptible-infected-recovered* or S.I.R model, which describes how this spreading behaves. This model allows us understanding many other phenomena not only related to diseases but also to rumor spreads. Since nodes cannot recover from rumors, it seems natural to consider a variant of the S.I.R model (namely, S.I.), for which any assumption on the recovering process is omitted, to model the rumor spreading on a network (see section 1.2).

Another well-studied situation is the one related to systems that follow some social network patterns. One of the most famous models for this scenario is the one proposed by Albert-László Barabási and Réka Albert [4] for generating random graphs under *preferential attachment mechanisms*. Preferential attachment means that the more connected a vertex is, the more likely it is to create new links. In terms of social networks, nodes represent people and edges between them represent some relationship. Strongly connected nodes represent well-known people with many relations. Clearly, for the previous case, popularity defines the growth of the network. Leaving this popularity influence aside. i.e., assuming equal popularity for the vertices, naturally leads to another well-known model called *uniform attachement*. In this case, all the vertices has the same probability of create new links.

Assuming one of these three models for the propagation of the information in the network as the ground truth, we are interested in finding the first node in this network who had the information (i.e., the *rumor source* or *vertex 0*). We consider not only the probabilistic models itself but the information about the observed structure of the network and the set of infected nodes. Essentially, we observe the graph G that is the result of the information diffusion process in some set of susceptible nodes at a given time t or for some graph size N and only with this information we try to find the vertex v of G who was the first in receiving the information. This approach was first treated by Shah and Zaman [18].

Since trying to output a single vertex as a candidate could be a difficult task, it might be easier to look at some small set of the vertices of G that are possible choices for 0. Networks are considered as random growing graphs so giving sets of choices that will be constantly growing as the same rate that the original graphs does not seem to lighten the work. For that reason, we are also interested in studying functions that output set of vertices whose size is independent of the size of the main network. These estimators are known as *root-finding algorithms*. We analyze the behavior of some specific estimators and we give lower bounds for the size of the set of choices, following the ideas and techniques developed by Bubeck, Devroye and Lugosi [6].

All the notions and basic definitions needed to understand these two problems and the remaining parts of this text are contained in Chapter 1. We dedicated separate sections to the most important concepts: the S.I. Model and the Uniform and Preferential attachment mechanisms (Section 1.2), Root-finding algorithms (Section 1.3) and Maximum Likelihood Estimator (Section 1.4).

The problems presented in the previous paragraphs about source finding are treated carefully in Chapters 2 and 3.

Namely, Chapter 2 contains the first studied function: the *SZ-estimator*. We give its definition in terms of the Maximum Likelihood Estimator and we analyze some of its properties when the underlying network has a tree structure. Using this, we present a very important notion related to this estimator: the *SZ-centrality* (originally known as *rumor centrality* [18]). We compare this measure for graphs with another classical measure: the *distance centrality*. We give alternatives to compute the SZ-centrality and we present an algorithm to calculate such measure for all the vertices of a graph using message-passing technique. Finally, we state and prove the main results in relation to single-source detection in the cases where the graph structure presents some regularity (this notion will become clear in section 2.2).

Chapter 3 gives details about root-finding algorithms. Given a graph G and an error $\varepsilon > 0$, the idea behind these algorithms is that they are capable of returning a *confidence set* $H \subset V(G)$ that satisfies the following to properties: (i) the source 0 is in H with probability greater or equal than $1 - \varepsilon$, (ii) the size of H does not depend on the size of G. We begin by presenting one of the most important tools along this study: the *simple estimator*. We prove that this is a root-finding algorithm for the uniform and the preferential attachment models. The rest of the chapter is devoted to prove the main results stated in [6] about lower bounds for the size of the sets H mentioned before.

In Chapter 4 we report several simulations that we performed to gain a better insight of the previous two chapters. We developed and used some *Python* tools to test the performance of some of the algorithms presented in the previous sections. We compare them to a couple of classic estimators. One of them based on a well-studied measure of graphs: *degree centrality*. The other one was built following a famous algorithm for ranking websites: the PageRank algorithm. We also present a novel algorithm suggested by Louigi Addario-Berry (personal communication) to look for the source in trees. The essence of this algorithm is based on the fact that once that a leaf is removed from a tree, the resulting graph is still a tree. We called the Leaves Removal Estimator. We present three different kinds of tasks and comparisons of the estimators.

Appendix A contains a brief introduction to some classical results about Pólya Urns and Dirichlet distributions.

Appendix B contains some bounds for products of exponentially distributed random variables.

Appendix C gives a upper bound for some Beta distribution.

Chapter 1

Definitions and Notation

1.1 Preliminaries

The notation $\mathbb{N}$ represents the set of natural numbers including 0. The symbol $\mathbb{N}^+$ denotes $\mathbb{N} \setminus \{0\}$. Let $a, b \in \mathbb{N}$, such that $a \leq b$, we define $[b] = \{1, 2, 3, ..., b\}$ and $[a, b] = \{a, a+1, a+2, ..., b-1, b\}$. For a set A, we denote by $\wp(A)$ the set of all the subsets of A.

A *graph* G consists of a set $V(G)$ (or V) of vertices, a set $E(G)$ (or E) of edges, and a mapping associating to each edge $e \in E$ an unordered pair $v, w \in V$. If $\{v, w\} \in E$, where $v, w \in V$, then we say that v and w are joined by and edge in V. A *loop* is an edge of the form $\{v, w\}$ where $v = w \in V$. A graph is *simple* when it has no loops and no two distinct edges have exactly the same pair of ends. In some cases, we write (u, v) instead of $\{u, v\}$ to give a particular direction to the edges of the graph.

Two vertices v and w of a graph G are called *adjacent* if they are distinct and they are joined by an edge. We use $v \sim w$ to say that v and w are adjacent. We will use $N_G(v)$ to denote the set of all vertices in G adjacent to a given vertex v; these vertices are also called the *neighbors* of v. For a given vertex v in a graph G, we define $deg(v) = |N_G(v)|$; $deg(v)$ is called the *degree of v in G* (or simply, the *degree* of v).

A *path* in a graph G is an alternating sequence

$$v_0, e_1, v_1, e_1, v_2, e_3, ..., v_{k-1}, e_k, v_k$$

of distinct vertices v_i, and edges e_i so that the ends of e_i are exactly v_{i-1} and v_i, $i = 1, 2, ..., k$. Such a path has length k. A path like this is denoted as $v_0 R v_k$. If we have such a sequence with $v_0 = v_k$ and $v_0, v_1, ..., v_{k-1}$ distinct, we say we have a *cycle* or *closed path*. The path vRv consists only of the vertex v and it has length 0.

If a path from v to w exists for every pair of distinct vertices v, w of G, then G is called *connected*. The *length* of the shortest path from v to w, if such path exists, is called the *distance* $d(v, w)$ between these vertices.

A connected graph that contains no closed paths is called a *tree*. It is well-known [14] that

A connected graph on n vertices is a tree if and only if it has $n - 1$ edges.

Therefore, for a tree G and for any pair of vertices $u, v \in V(G)$, the set uRv is uniquely defined.

A *regular tree G of degree d* is a tree which satisfies that $deg(v) = d$, for all $v \in V(G)$. Sometimes, we refer to this kind of graphs as *d-regular trees*. We call d the *degree of the tree G*. Clearly, for regular trees, $|V| = \infty$.

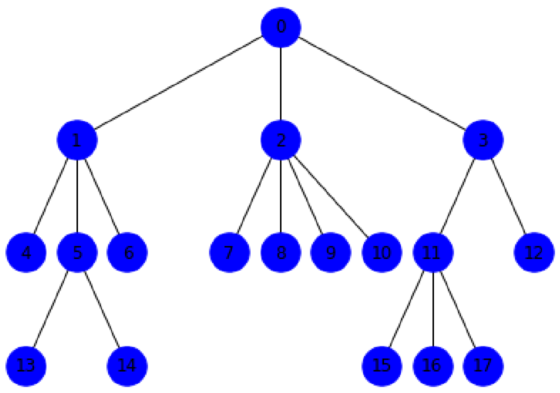
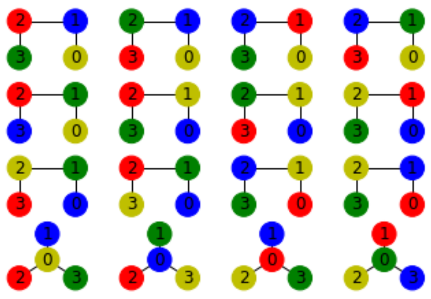
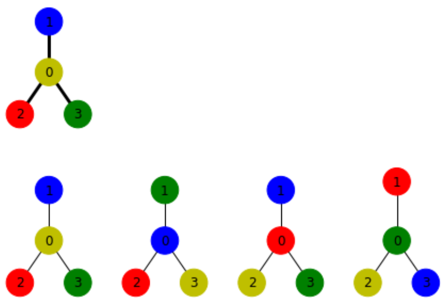


Figure 1.1: Labeled Tree



(a) Set of all labeled trees of size 4



(b) $i(T)$ for a *4-vertex* tree

Figure 1.2: Isomorphism classes

If T is a tree such that $V(T) = [0, k]$ for some positive integer k, then we say that T is a *labeled tree* (See Figure 1.1). In most situations we assume that the studied trees are labeled trees.

Let T_1 and T_2 be two trees. If there exists a bijection $\phi : V(T_1) \to V(T_2)$, such that any two vertices u and v of T_1 are adjacent in T_1 if and only if $\phi(u)$ and $\phi(v)$ are adjacent in T_2, then we say that T_1 and T_2 are *isomorphic trees* (See Figure 1.2b).

The *isomorphic class* of a tree T (denoted by $i(T)$) is the set of all the trees that are isomorphic to it.

Let T be a tree. If any path from 0 has increasing labels, then we say that T has an *increasing labeling*. A *recursive tree* is a tree labeled with an increasing labeling. Figure 1.2b also shows a recursive tree; all the paths starting at 0 and going downwards have vertices with increasing labelings.

A *rooted tree* T^v is a tree T together with a vertex v in $V(T)$. We sometimes denote a rooted tree simply as T, in which case we denote the root as $\emptyset$. In a recursive tree T it is understood that $\emptyset = 0$.

In a rooted tree T^v, the *descendants* of a vertex u are referred to those vertices w for which u belongs to the path connecting v and w. A descendant w of u is in the k^{th} *generation* if the path connectig w and u has length k. The set of vertices with no descendants (i.e., the leaves) is denoted by $\mathcal{L}$. The children of a vertex $w \neq v$, $child(w)$, in a rooted tree T^v are the descendants of w that belong to the first generation. The children of v are all the vertices adjacent to it. The children of a vertex u in an unrooted tree T are all its neighbors. The *height* of a rooted tree T^u is

$$\max_{v \in \mathcal{L}(T)} d(u, v).$$

A *rooted view* of a rooted graph T^u is simply a way of plotting the graph in a hierarchical form in which the root is placed at the top of the plot, its descendants in the 1^{st} generation are placed in a level inmediately below the root, its decendants in the 2^{nd} generation are placed in a second level inmediately below the root, and so on. Figure 1.1 shows a rooted tree plotted in rooted view.

Let G be a tree and let $u, v \in V(G)$, $u \neq v$. We define T_u^v as being the subgraph of G induced by the set

$$V(T_u^v) = \{w \in V(G) : u \in vRw\}.$$

This means that

$$E(T_u^v) = \{\{w_1, w_2\} \in E(G) : w_1, w_2 \in V(T_u^v)\}.$$

Clearly, $u \in T_u^v$. We say that v is the root of the subtree T_u^v. For $u = v$, define $T_v^v = G$. Since G is a tree, there is only one path between any two vertices in $V(G_N)$. Therefore, there is no ambiguity when asking for $u \in vRw$ in the definition of T_u^v. Note that $T_u^v = \{u\}$ if $deg(u) = 1$.

From now on, we use the symbol T_u^v to denote both T_u^v and $V(T_u^v)$. Sometimes, we denote these rooted trees simply as T_u.

The following lemma provides us many useful properties satisfied by the subgraphs T_u^v.

Lemma 1.1.1. *If $u, v \in V(G_N)$ are such that $d(u, v) = 1$, then*

1. $V(G_N) = T_u^v \cup T_v^u$
2. $T_u^v \cap T_v^u = \emptyset$,
3. $N = |T_u^v| + |T_v^u|$,
4. $T_w^v = T_w^u$, *for all* $w \in V(G_N) \setminus \{u, v\}$.

Proof.

1. Since $T_u^v \cup T_v^u \subset V(G_N)$, it is enough to prove that
$$V(G_N) \setminus T_v^u = T_u^v. \tag{1.1}$$
Let $w \in V(G_N)$, such that $x \notin T_v^u$. From the definition of T_v^u, we know that $v \notin xRv$. Therefore, since G_N is connected, it must be true that $u \in xRv$. Which means that $x \in T_u^v$. Thus, $V(G_N) \setminus T_v^u \subset T_u^v$. The reverse inclusion can be proven with the same arguments.

2. This follows from (1.1).

3. This is a consequence of the fact that $V(G_N)$ is the disjoint union of T_v^u and T_u^v.

4. It is enough to prove the result for some $w \in T_u^v$, since the arguments given for the other case are essentially the same.

   Let $z \in T_w^u$, then $w \in uRz$. We know that $uRz \subset vRz$, therefore, $w \in vRz$, and thus, $z \in T_w^v$. On the other case, if $z \in T_w^v$, then $w \in zRv$. Since $zRv = vRu \cup uRz$, and $w \notin vRu$ (otherwise, $d(u, v) \geq 2$), we have $w \in uRz$, and then, $z \in T_w^u$.

□

1.2 Graph Models

In this section we present the three types of graphs over which each one of the estimators will be analyzed: *pseudo-d-regular trees, preferential attachment trees and uniform attachment trees.* Each one of them comes from a particular real-life situation that we explained at the beginning of each subsection.

1.2.1 A S.I. Model and the Pseudo-d-regular Trees

The first kind of studied graphs comes from a simple model of an epidemic of an infectious disease in a large population. We use this same idea to think about a process of rumor spreading in a network.

To define it we consider a network thought as a d-regular tree T with $d \in \{2, 3\}$, which set of vertices V is $\mathbb{N}$, and E is its set of edges. We assume that V is countably infinite to avoid boundary effects.

We consider a variant of the well-known S.I.R model ([8],[12]) for the rumor spreading, known as the *susceptible-infected* or S.I., in which, each node v of the network has two possible and both exclusive states: or the vertex v is *susceptible*, which means that it hasn't received the rumor, or v is *infected*, which means that it has received the rumor. Once v is infected, it won't recover, i.e. v keeps the rumor forever. Once a node v has received the rumor, it is able to spread it to another susceptible node u if and only if $u \in N_T(v)$. Our principal interest is the graph G of infected nodes in the case where only one vertex v^* is the rumor source.

The process begins with v^* receiving the rumor at time 0. After this, the unique infected node passes the information to one of its susceptible neighbors (say w) according to some probabilities defined below. Once it receives the rumor, they both v^* and w have the chance of sending it to one of its susceptible neighbors. The rumor speading process continues like this at the rate of one infection per time.

Our main focus is using this S.I. model to generate finite graphs: at a certain time, we stop allowing vertices to become infected. Note that even when our finite graph G, which consists of the infected vertices until time n, satisfies that $|V(G)| = n$, we do not necessarily have $V(G) = \{0, 1, ..., n-1\}$. For this reason, we decided to relabel the vertices in G according to the following chronological rule:

- 0 is the label of v^*, the first infected vertex,
- for $i \geq 0$, i is the label of the vertex $v \in G$ if, and only if v was the first infected vertex once that we assign the label $i - 1$.

The previous process can be formalized in the following way:

Definition 1.2.1. *For $d \in \{2, 3\}$, we define a Pseudo-d-regular tree of size n, $\mathcal{R}_d(n)$, with set of vertices $[0, n]$ by induction as follows: $\mathcal{R}_d(0)$ is the tree consisting only in the vertex 0, $\mathcal{R}_d(1)$ is the unique tree on 2 vertices ($V(\mathcal{R}_d(1)) = \{0, 1\}$), and $\mathcal{R}_d(n)$ is built from $\mathcal{R}_d(n-1)$ by adding the vertex n and an edge $\{i, n\}$ where $i \in [0, n-1]$ is chosen at random with probability equal to*

$$\frac{d - deg_{\mathcal{R}_d(n-1)}(i))}{\sum_{j=0}^{n-1}(d - deg_{\mathcal{R}_d(n-1)}(j)))}.$$

Intuitively, $\mathcal{R}_d(n)$ grows over the infinite d-regular tree T: to build $\mathcal{R}_d(n+1)$ from $\mathcal{R}_d(n)$, we choose one vertex uniformly in the boundary of $\mathcal{R}_d(n)$ in T and once it is chosen, we label it as $n + 1$.

From the definition of this model, it is clear that if a vertex v is in $\mathcal{R}_d(n-1)$ and it already has d neighbors, then the edge $\{v, n\}$ has no chance of being part of $\mathcal{R}_d(n)$. Therefore, any vertex in this kind of trees has degree d, at most.

From now on, we stick to the chronological labelings when talking about this and any other probabilistic model.

1.2.2 Uniform and Preferential Attachment Models

In this section we define two other models of random growing graphs: the UA(n) graph, and the PA(n) graph, who is related to the well-known Barabási-Albert Model [4]. The first one is an artificial graph built for testing theoretical results and the second one is a random scale-free network that models several natural and human-made systems, including the Internet, the world wide web, citation networks, and some social networks. One interesting feature of this last model is that it certainly contains few nodes (called hubs) with unusually high degree as compared to the other nodes of the network.

Definition 1.2.2. *For any $\alpha \in \mathbb{R}$ we define the random tree $\mathcal{T}_\alpha(n)$ with the vertex set $[0, n]$ by induction as follows: $\mathcal{T}_\alpha(0)$ is the tree consisting on one single vertex (namely, 0), $\mathcal{T}_\alpha(1)$ is the unique tree on 2 vertices, and $\mathcal{T}_\alpha(n)$ is built from $\mathcal{T}_\alpha(n)$ by adding the vertex $n+1$ and an edge $\{i, n\}$ where $i \in [0, n-1]$ is chosen at random with probability equal to*

$$\frac{deg_{\mathcal{T}_\alpha(n-1)}(i)^\alpha}{\sum_{j=0}^{n-1} deg_{\mathcal{T}_\alpha(n-1)}(j)^\alpha}.$$

This means the edge $\{i, n\}$ is chosen with probability porportional to the degree of the vertex i in the tree $\mathcal{T}_\alpha(n-1)$ raised to the power α, which implies that vertices with a higher degree have a bigger chance of being adjacent to the new ones (if $\alpha \neq 0$).

We focus on the case $\alpha = 0$, which we alternatively denote to UA(n) for *uniform attachment model*, and on $\alpha = 1$ which we denote PA(n) for *preferential attachment model.*

In the cases where $\alpha = 0$ or $\alpha = 1$, we can respectively restate the probability of having a new edge on $\mathcal{T}_\alpha(n)$ as

$$\mathbb{P}(n \sim i | \mathcal{T}_0(n-1)) = \frac{1}{n}, \quad (n \geq 1),$$

$$\mathbb{P}(n \sim i | \mathcal{T}_1(n-1)) = \frac{deg_{\mathcal{T}(n-1)}(i)}{2(n-1)}, \quad (n \geq 1).$$

It is clear that for UA(n), each edge of the form $\{i, n\}$ for $i \in [0, n-1]$, has the same probability of being part of UA(n). On the other hand for PA(n), the vertices with higher degree have higher probability of receiving new neighbors.

Both for these models and the one defined in the previous section, the vertex 0 will be called the *source of the rumor* or simply the *root.* This label comes from the fact that this is the first vertex that appears on each graph generated from these random growing models. In some cases that the chronological labeling is not important, we will also make reference to the source as the vertex v^* or v_R.

1.3 Root-finding Algorithm

In this section we describe our main tool to solve the problem of finding the root.

Definition 1.3.1. *Given $\varepsilon \in (0,1)$ and a tree T of size n (for some $n \in \mathbb{N}$), we are interested in describing algorithms that output sets $H(T,\varepsilon) \subset V(T)$ with $|H(T,\varepsilon)| = K(\varepsilon)$, such that, with probability $1-\varepsilon$, $0 \in H(T,\varepsilon)$. Such algorithms are known as root-finding algorithms.*

The trees analyzed by such root-finding algorithms are randomly generated according to the models described above (See Section 1.2).

Note that the size of the set H must depend only on ε and not in the size of T.

Using the notation introduced before we can restate the previous definition in the following way: Let $\varepsilon \in (0,1)$ and $K \in \mathbb{N}$. We want to look for mappings H from trees to subsets of K vertices with the property that

$$\liminf_{n\to+\infty} \ \mathbb{P}\left(0 \in H(T_R(n))\right) \geq 1-\varepsilon$$

where $T_R(n) = T_\alpha(n)$ or $T_R(n) = \mathcal{R}_d(n)$. In most instances the input tree will be clear from the context, and thus we often write T instead of $T_R(n)$. The function H is also known as set estimator. We give the definition and some properties of these mappings H in Section 1.4. We also show that these algorithms don't use the information given by the labels to output their results, i.e. they are label-independent.

In the following sections we show that root-finding algorithms actually exist and we give some bounds on the size of the set H.

1.4 Maximum Likelihood Estimator

Let $n \in \mathbb{N}$ and define $\Theta_n = [0,n]$. Let Ω_n as the set of trees T such that $V(T) = \Theta_n$. Let G be a random element with $G \sim \mathbb{P}_*$, where $\mathbb{P}_*$ is one of the distributions defined on Section 1.2 (S.I model, UA and PA mechanisms). Let $T_0 \in \Omega_n$ be a realization of G, i.e., a random tree with vertices $\{0,1,2,...,n\}$ generated by one of our chosen models. Let S_n be the set of permutations of $[n]$, and for any $v \in \Theta_n$ define S_n^v as the set of all the permutations of the set $\{0,1,2,...,n\}$ that send 0 to v, i.e.,

$$S_n^v = \{f \in S_n \mid f(0) = v\}.$$

Choose one $\pi_v \in S_n^v$ uniformly at random, and define a new tree T_{π_v} such that $V(T_{\pi_v}) = \Theta_n$ and

$$E(T_{\pi_v}) \quad = \quad \{\{\pi_v(u), \pi_v(w)\} : \{u,w\} \in E(T_0)\}.$$

Let $\mathbb{P}_v$ be the distribution of T_{π_v}. This process gives us a colection of distributions $\{\mathbb{P}_u\}_{u\in\Theta_n}$.

Note that the definition of the root-finding estimator allows for an estimator that always outputs $\{0\}$. For this reason, we need a way to say that vertex labels have been "hidden". This is why we have defined T_{π_v} in terms of permutations π_v that exchange 0 for v in the original tree T_0. The idea is that π_v scrambles, not only the label for the source but also for the remaining vertices.

Finally, the parametric family of distributions built above allows us to define the ML estimator:

A function $\Psi_n : \Omega_n \to \Theta_n$ is called a *estimator*. The Maximum Likelihood estimator (denoted as *ML estimator*) is the estimator Ψ_n^* that satisfies

$$\Psi_n^*(T) = \arg\max_{v\in\Theta_n} \ \mathbb{P}_v(T).$$

Given a tree T generated with one of the distributions $\mathbb{P}_*$ previously defined, what the ML estimator does is to return the parameter $v \in \{0, 1, 2, ..., n\}$ that maximizes the probability of seeing the tree T.

On the other hand, a set estimator is a function $K_n : \Omega_n \to \wp(\Theta_n)$. For a given $p \in \mathbb{N}$, a set estimator of size p (denoted as K_n^p) is a set estimator such that $|K_n^p(T)| = p$, $\forall T \in \Omega_n$. The *Maximum Likelihood set estimator of size* p is the set estimator of size p that satisfies

$$\overline{K}_n^p(T) = \underset{S \in \wp(\Theta_n),\ |S|=p}{\arg\max} \min_{v \in S} \mathbb{P}_v(T).$$

Henceforward, we will use the label *ML estimator* to reference both ML estimator and ML set estimators.

Now, we are in conditions to state two of the simplest and most fundamental results of this study.

Theorem 1.4.1. *Let $\Psi_n : \Omega \to \Theta_n$ be an estimator. If $\Psi_n(\pi(T)) = \pi(\Psi_n(T))$, for all $\pi \in S_n$ and for all $T \in \Omega_n$, then*

$$\mathbb{P}_v \left(\Psi_n^*(T) = v \right) = \mathbb{P}_* \left(\Psi_n^*(T_0) = 0 \right).$$

1.4.1 says that if we are working with an estimator that is label-independent, then we do not have to worry about assuming that the studied trees have no longer the original labelings. In other words, computing the probability that the estimator finds the source with new labels is the same as computing it without such modifications. So, we can keep working with chronological labels when trying to get the source of the tree.

Proof. Let $\pi \in S_n$ and $T \in \Omega_n$. By hypothesis,

$$\mathbb{P}_v \left(\Psi_n(\pi_v(T_0)) = \pi_v(0) \right) = \mathbb{P}_v \left(\pi_v(\Psi_n(T_0)) = \pi_v(0) \right). \tag{1.2}$$

Since π is a bijection, we see that

$$\mathbb{P}_v \left(\pi_v(\Psi_n(T_0)) = \pi_v(0) \right) = \mathbb{P}_*(\Psi_n(T_0) = 0). \tag{1.3}$$

Combining 1.2 and 1.3, we see that

$$\mathbb{P}_v \left(\Psi_n(T) = v \right) = \mathbb{P}_* \left(\Psi_n(T_0) = 0 \right).$$

□

Theorem 1.4.1 is essentially Theorem 1.4.2 but stated for set estimators.

Theorem 1.4.2. *Let $K_n^p : \Omega \to \wp(\Theta_n)$ be a set estimator of size p. If $K_n^p(\pi(T)) = \pi(K_n^p(T))$, for all $\pi \in S_n$ and for all $T \in \Omega_n$, then*

$$\mathbb{P}_v \left(v \in K_n^p(T) \right) = \mathbb{P}_* \left(0 \in K_n^p(T_0) \right).$$

For now on, Theorems 1.4.1 and 1.4.2 will be referenced as **Property 1** and **Property 2**, respectively. The Maximum Likelihood and all other estimators we consider satisfy Properties 1 and 2.

Chapter 2

Single Rumor Source Detection on d-regular Trees

This chapter is devoted to the study of source detection when the underlying graph structure comes from regular trees [18]. We present an estimator, known as Shah-Zaman estimator, and we prove some of its properties. We give a couple of equivalent representations of such estimator and we use them to compare its performace with the well-known measure of distance centrality. We exhibit a computational less efficient way of computing this estimator that will be useful in Chapter 4. Finally, we prove a set of theorems related to the asymptotic behaviour of the estimator in relation to the problem of source detection for graphs where each vertex has at most 3 neighbors.

2.1 ML estimator and the S.I. Model

Let us suppose that a rumor begins to spread by the network G according to the S.I. model, where G is a d-regular tree for some $d \in \{2, 3\}$. Let v be the first node in G that has the rumor. Now, for some time $N \in \mathbb{N}$, let G_N be the pseudo-d-regular subtree of G resulting from the spreading of the rumor until N vertices are infected. We know that $V(G_N) = \{0, 1, ..., N-1\}$ after relabeling the vertices. By the definition of the model, it is clear that G_N is a connected subgraph of G. Our aim is to produce an estimate of v based on the observation of G_N and the structure of G.

In this framework, the ML estimator is

$$\Psi^*_N(G_N) = \underset{v \in V(G_N)}{\arg\max}\ \mathbb{P}_v(G_N).$$

2.1.1 ML for Regular Trees

Evaluating $\mathbb{P}_v(G_N)$ for each $v \in V(G_N)$ may be computationally expensive. Therefore, we propose an alternative form to compute it for regular trees.

Note that evaluating $\mathbb{P}_v(G_N)$ is to find the probability of all possible events that result in G_N after N nodes are infected starting with v as the source under the S.I. model. These such events could be thought as being permutations of the vertices of G_N: if $(v_1, v_2, ..., v_N)$ is one of such permutations, then a vertex v_i appears in the i^{th} position of this list (counted from left to right), if, and only if, v_i is the i^{th} infected vertex of the process. Therefore, given a tree, we have a way of assigning permutations to it. However, the converse is not always true. The following definition gives us a name for favorable cases.

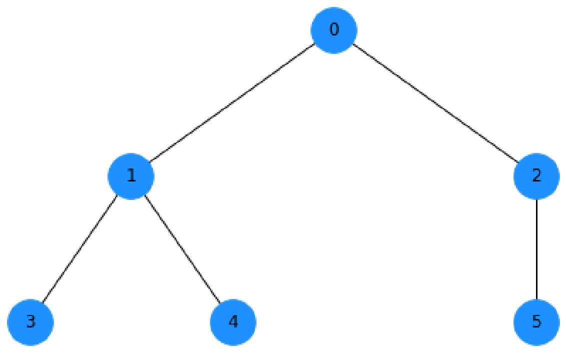


Figure 2.1: G_6 is a realization of the S.I. model with 6 vertices.

Definition 2.1.1. *Let $P(G_N)$ be the set of all the permutations of the elements of $V(G_N)$, and let $v \in V(G_N)$. Let $\sigma \in P(G_N)$ and write $\sigma = (v_1, v_2, ..., v_N)$.*

We say that σ is a ***permitted permutation starting with node** v **and resulting in rumor graph** G_N if*

1. *$v_1 = v$,*

2. *for all $i \in \{2, ..., N\}$,*
$$v_i \in \bigcup_{k=1}^{i-1} N_{G_N}(v_k).$$

Define $\Omega(v, G_N)$ as the set of all the permitted permutations starting with node v and resulting in rumor graph G_N.

For the graph G_6 (see Figure 2.1), we list the permitted permutations satarting with node 0:

$$\begin{array}{llll}
(0,1,2,3,5,4), & (0,2,1,5,3,4), & (0,2,1,3,4,5), & (0,2,1,4,3,5) \\
(0,1,3,2,5,4), & (0,1,2,3,4,5), & (0,2,1,5,4,3), & (0,2,1,4,5,3) \\
(0,1,2,5,3,4), & (0,1,3,2,4,5), & (0,1,3,4,2,5), & (0,1,4,3,2,5) \\
(0,2,5,1,3,4), & (0,1,2,5,4,3), & (0,1,2,4,3,5), & (0,1,4,2,5,3) \\
(0,2,1,3,5,4), & (0,2,5,1,4,3), & (0,1,2,4,5,3), & (0,1,4,2,3,5).
\end{array}$$

It is clear from the Figure 2.1 that a permutation like $(0, 4, 3, 1, 2, 5)$ does not generate a graph like G_6 under the hypothesis of the S.I. model.

Definition 2.1.2. *Let $\sigma \in \Omega(v, G_N)$, with $\sigma = (v_1, v_2, ..., v_N)$, and $k \in \{1, ..., N\}$.*

Define
$$V(G_k(\sigma)) = \{v_1, v_2, ..., v_k\},$$

and let $G_k(\sigma)$ be the subtree of G_N induced by $V(G_k(\sigma))$.

Let $F_k(\sigma)$ be the subset of $V(G_N)$ such that

$$F_k(\sigma) = \{w \in V(G_k(\sigma))^c : \exists u \in V(G_k(\sigma)), \quad d(w, u) = 1\},$$

and define $n_k(\sigma) := |F_k(\sigma)|$.

Given a permitted permutation σ and $k \in \{1, 2, ..., N\}$, the graph $G_k(\sigma)$ is, intuitively, the resulting graph of the same process that generated G_N but stopped once the k^{th} vertex is added. On the other hand, $F_k(\sigma)$ denoted the boundary of $G_k(\sigma)$, i.e., all the neighbors of the vertices of $G_k(\sigma)$ that are not part of $V(G_k(\sigma))$.

For the graph G_6 and the permitted permutation

$$\sigma = (0, 1, 3, 4, 2, 5),$$

we have

$$\begin{array}{lll} k = 1, & G_1(\sigma) \text{ is such that } & V(G_1(\sigma)) = \{0\} \text{ and } E(G_1(\sigma)) = \varnothing \\ k = 2, & G_2(\sigma) \text{ is such that } & V(G_2(\sigma)) = \{0, 1\} \text{ and } E(G_2(\sigma)) = \{\{0, 1\}\} \\ k = 3, & G_3(\sigma) \text{ is such that } & V(G_3(\sigma)) = \{0, 1, 2\} \text{ and } E(G_3(\sigma)) = \{\{0, 1\}, \{1, 3\}\} \end{array}$$

and so on. Also, for this permitted permutation we have

$$\begin{array}{ll} k = 1, & F_1(\sigma) = \{1, 2\}, \\ k = 2, & F_2(\sigma) = \{3, 4, 2\}, \\ k = 3 & F_3(\sigma) = \{4, 2\}, \end{array}$$

and so on.

From now on, the symbol $\{\sigma\}$ will denote the event in which the graph G_N grows in the chronological order suggested by σ. For instance, for $\sigma = (0, 1, 3, 4, 2, 5)$, we have that $\{\sigma\}$ means that in the graph G_6 (see Figure 2.1) the first infected node was the 0, the second infected node was 1, the third one was 3 and so on.

Let $k \in \{1, 2, ..., N\}$ and write $\sigma = (v_1, v_2, ..., v_N)$. If we define σ_k as $(v_1, v_2, ..., v_k)$, then, according to the previous convention, $\{\sigma_k\}$ denotes the event in which the graph $G_k(\sigma)$ grows in the chronological order proposed by the permutation σ_k. In other words, $\{\sigma_k\}$ is the event for which G_N starts to grow in the chronological order advised by σ up to step k.

The previous definitions and notation allow us to state the following proposition.

Proposition 2.1.1. *If $\sigma \in \Omega(v, G_N)$, then*

$$\mathbb{P}_v(\sigma) = \prod_{k=2}^{N} \frac{1}{n_{k-1}(\sigma)}.$$

Proof. Write $\sigma = (v_1, ..., v_N)$ with $v_1 = v$. It is clear that

$$\{\sigma\} = \bigcap_{k=1}^{N} A_k,$$

where

$$A_k = \{k^{\text{th}} \text{ infected node } = v_k\}, \quad \forall k = 1, ..., N.$$

Therefore,

$$\begin{aligned} \mathbb{P}_v(\sigma) = \quad & \mathbb{P}_v(A_1) \\ & \times \mathbb{P}_v(A_2|A_1) \\ & \times \mathbb{P}_v(A_3|A_1, A_2) \\ & \quad \cdots \\ & \times \mathbb{P}_v(A_N|A_1, A_2, \ldots, A_{N-1}), \end{aligned}$$

and thus,

$$\mathbb{P}_v(\sigma) = \prod_{k=2}^{N} \mathbb{P}_v\left(A_k | \sigma_{k-1}\right). \tag{2.1}$$

Now, note that given $G_{k-1}(\sigma)$, to build $G_k(\sigma)$ we only have to add one more vertex w to the set $V(G_{k-1}(\sigma))$. We know that we must pick this w to be a susceptible neighbor of the nodes in $V(G_{k-1}(\sigma))$, i.e., w must be chosen in $F_{k-1}(\sigma)$. Therefore, we have n_{k-1} equally likely choices for this w. In consequence,

$$\mathbb{P}_v\left(A_k | \sigma_{k-1}\right) = \frac{1}{n_{k-1}(\sigma)}. \tag{2.2}$$

Replacing (2.2) in (2.1), we obtain the desired result.

□

The following result shows us that there is an even easier way to deal with $\mathbb{P}_v(\sigma)$ because of the regularity of G.

Corollary 2.1.1. *For Pseudo-d-Regular trees,*

$$\mathbb{P}_v(\sigma) = \prod_{k=1}^{N-1} \frac{1}{(d-2)k+2}, \qquad \textit{for all } \sigma \in \Omega(v, G_N).$$

Proof. Suppose that $\sigma = (v_1, ..., v_N)$, where $v = v_1$. By Proposition (2.1.1), we only need to prove that $n_k(\sigma) = dk - 2(k-1)$.

Suppose the k^{th} node added to $G_{k-1}(\sigma)$ is v_k. Then, after adding $v_{k\sigma}$, we lost one choice from the n_{k-1} possible options we had to pick this node, but on the other hand, we gained $d-1$ new choices for picking the $k+1^{\text{th}}$ node. Therefore, $n_k(\sigma) = (n_{k-1}(\sigma) - 1) + (d-1) = n_{k-1}(\sigma) + d - 2$. Using this recursion, we can conclude

$$n_k(\sigma) \quad = \quad n_1(\sigma) + (k-1)(d-2), \quad \forall 1 \leq k \leq N-1$$

Now, replacing $n_1(\sigma) = d$ in (2.1.1), we find

$$\begin{aligned} n_k(\sigma) &= d + (k-1)(d-2) \\ &= (d-2)k + 2, \end{aligned}$$

for any $1 \leq k \leq N-1$, which completes the proof.

□

Note that Corollary 2.1.1 shows that $\mathbb{P}_v(\sigma)$ only depends on d, therefore, it is constant for every $\sigma \in \Omega(v, G_N)$ and for every v as well.

Definition 2.1.3. *Given a tree $G \in \Omega_n$ and $v \in V(G)$, we define $R(v, G) = |\Omega(v, G)|$. We say that $R(v, G)$ is the **Shah-Zaman centrality** of v on G. An element of $v_R \in V(G)$, such that*

$$R(v_R, G) = \max_{v \in G} R(v, G)$$

*will be called a **Shah-Zaman center** (or simply, **SZ-center**) of G.*

Finally, let $\Gamma_n^* : \Omega_n \to \Theta_n$ defined by

$$\Gamma_n^*(T) = \underset{v \in \Theta_n}{\arg\max}\; R(v, T).$$

Γ_n^* is the first studied estimator. Given a tree T, it outputs one SZ-center. This was proposed by Shah and Zaman [18]. It was originally known as Rumor Estimator, but we decided to call it the *Shah-Zaman estimator* (or simply, *SZ-estimator*), since for us each one of the algorithms exposed in this text is a rumor estimator.

Theorem 2.1.1. *If G_N is a pseudo-d-regular graph, then*

$$\Psi_N^*(G_N) = \Gamma_N^*(G_N).$$

That is, our maximum likelihood estimate of the rumor source coincides with SZ-center for trees generated from the S.I. model.

Proof. Note that

$$\Psi_N^*(G_N) = \underset{v \in V(G_N)}{\arg\max}\; \mathbb{P}_v(G_N) \tag{2.3}$$

by definition of the estimator. Since, G_N comes from a permitted permutation, we have

$$\underset{v \in V(G_N)}{\arg\max}\; \mathbb{P}_v(G_N) = \underset{v \in V(G_N)}{\arg\max} \sum_{\sigma \in \Omega(v, G_N)} \mathbb{P}_v(\sigma). \tag{2.4}$$

By Corollary 2.1.1,

$$\underset{v \in V(G_N)}{\arg\max} \sum_{\sigma \in \Omega(v, G_N)} \mathbb{P}_v(\sigma) = \underset{v \in V(G_N)}{\arg\max}\; |\Omega(v, G_N)|. \tag{2.5}$$

Putting 2.3, 2.4 and 2.5 together, we obtain the desired equality. □

2.1.2 Distance Centrality

In this section we compare the SZ-centrality notion previously mentioned with another well-known measure: distance centrality.

Definition 2.1.4. *Let T be a connected graph. Let $\delta_T : V(T) \to \mathbb{N}$ be defined as*

$$\delta_T(v) := \sum_{u \in T} d(v, u).$$

The quantity $\delta_T(v)$ is known as the distance centrality of v in T. We say that a vertex $v \in V(T)$ is a distance center of T if $\delta_T(v) \leq \delta_T(u)$, for all $u \in V(T)$.

SZ-Centrality vs. Distance Centrality

Proposition 2.1.2. *Let T be a tree and let $N = |V(T)|$. If v_D is the distance center of T, then, for all $v \in V(T)$ with $v \neq v_D$,*

$$|T_v^{v_D}| \leq \frac{N}{2}. \tag{2.6}$$

Proof. Consider $D : V(T) \times \wp(V(T))) \to [0, +\infty]$ defined as

$$D(v, A) = \sum_{u \in A} d(v, u), \quad \forall v \in V(T),\ A \subset V(T).$$

Let $v_1, ..., v_k \in V(T)$, be all the different children of v_D. It's enough to prove the result for some v_j, since each $T_v^{v_D}$ with $v \in T \setminus N_T(v_D)$ is contained in some $T_{v_j}^{v_D}$, and therefore, $|T_v^{v_D}| \leq |T_{v_j}^{v_D}|$. Without loss of generality, assume that $j = 1$.

Note that (2.6) is equivalent to

$$2\ |T_{v_1}^{v_D}| \leq \sum_{i=1}^{k} |T_{v_i}^{v_D}| + 1,$$

which is

$$|T_{v_1}^{v_D}| \leq \sum_{i=2}^{k} |T_{v_i}^{v_D}| + 1. \tag{2.7}$$

Let's prove the result by contradiction. Assume that (2.7) doesn't hold, i.e.,

$$\sum_{i=2}^{k} |T_{v_i}^{v_D}| - |T_{v_1}^{v_D}| + 1 < 0 \tag{2.8}$$

We'll prove that $D(v_1, T) < D(v_D, T)$. Define $S = T \setminus (T_{v_1}^{v_D} \cup \{v_D\})$.

We have

$$D(v_1, T) = \sum_{v \in T_{v_1}^{v_D}} d(v_1, v) + \sum_{v \in S} d(v_1, v) + d(v_1, v_D). \tag{2.9}$$

since $\{T_{v_1}^{v_D}, S, \{v_D\}\}$ is a partition of T.

On the other hand, $v_1 \in N_T(v_D)$, and therefore,

$$d(v_1, v) = d(v_D, v) - 1 \text{ for all } v \in T_{v_1}^{v_D} \tag{2.10}$$

$$d(v_1, v) = d(v_D, v) + 1 \text{ for all } v \in S \tag{2.11}$$

Replacing (2.10) and (2.11) in (2.9), we obtain

$$\begin{aligned} D(v_1, T) &= \sum_{v \in T_{v_1}^{v_D}} (d(v_D, v) - 1) + \sum_{v \in G} (d(v_D, v) + 1) + d(v_1, v_D) \\ &= D(v_D, T) - |T_{v_1}^{v_D}| + \sum_{v \in N_T(v_D) \setminus \{v_1\}} |T_v^{v_D}| + 1 \end{aligned}$$

Finally, the previous equality together with (2.8) implies $D(v_1, T) < D(v_D, T)$, and consequently, v_D is not the distance center, which is absurd.

$\square$

Proposition 2.1.3. *If there is a unique SZ-center on the tree, then it is also the distance center.*

Proposition 2.1.3 is a consequence of Proposition 2.1.4 that is part of the next section.

2.1.3 A simpler way of computing SZ-centrality

This section provides ways to evaluate $R(v, G_N)$ in the case when G is a regular tree.

For now on, v^* will represent the source a rumor in $V(G_N)$.

Remark. If v is such that $deg(v) = 2$, and we denote by $v_1, v_2 \in N_{G_N}(v)$, then we can count the number of permitted permutations in the following way:

Since each $\sigma \in \Omega(v, G_N)$ has exactly N vertices and the first of them fixed, we are interesting in figuring out how to pick these $N-1$ from $V(G_N)$, in a way that these elements satisfy that $|T^v_{v_1}|$ of them belong to $T^v_{v_1}$ and the other $|T^v_{v_2}|$ are from $T^v_{v_2}$. We know that there are

$$\binom{N-1}{|T^v_{v_1}|}$$

of doing this. After picking the first $|T^v_{v_1}|$, we need to consider the number of possible permutations for these elements, i.e., the number of permitted permutations in the subtree $T^v_{v_1}$. Since $N-1 = |T^v_{v_1}| + |T^v_{v_2}|$, it's clear that picking the first $|T^v_{v_1}|$ will lead us to the other $|T^v_{v_2}|$ vertices, so the last part of the process consists in counting the number of permitted permutations made with these $|T^v_{v_2}|$ elements, in the subtree $T^v_{v_2}$. For these case now becomes clear the fact that

$$\begin{aligned} R(v, G_N) &= \binom{N-1}{|T^v_{v_1}|} R(v_1, T^v_{v_1}) R(v_2, T^v_{v_2}) \\ &= (N-1)! \frac{R(v_1, T^v_{v_1})}{|T^v_{v_1}|} \frac{R(v_2, T^v_{v_2})}{|T^v_{v_2}|} \end{aligned}$$

The previous argument suggests a result for the general case:

Lemma 2.1.1.

$$R(v, G_N) = (N-1)! \prod_{u \in N_{G_N}(v)} \frac{R(u, T^v_u)}{|T^v_u|!}$$

Proof. Let $v_1, v_2, ..., v_k \in N_{G_N}(v)$ all the different children of v. Since the numbers of ways to partition $N-1$ slots for different subtrees is

$$\binom{N-1}{|T^v_{v_1}|, ..., |T^v_{v_k}|} = \frac{N-1}{|T^v_{v_1}|!...|T^v_{v_k}|!},$$

and for each choice of $N^v_{v_i}$ elements, there are $R(v_i, T^v_{v_i})$ possible ways of ordering these elements, we conclude that

$$R(v, G_N) = \frac{N-1}{|T^v_{v_1}|!...|T^v_{v_k}|!} \prod_{i=1}^{k} R(v_i, T^v_{v_i}),$$

which leads to the desired result. □

We can use the previous result to find an even much simpler way of computing the SZ-centrality of a vertex:

Lemma 2.1.2.

$$R(v, G_N) = N! \prod_{u \in V(G_N)} \frac{1}{|T^v_u|}.$$

Proof. For $u \in N_{G_N}(v)$, we know

$$R(u, T_u^v) = (|T_u^v| - 1)! \prod_{w \in N_{G_N}(u)} \frac{R(w, T_w^v)}{|T_w^v|!}$$

Therefore,

$$\begin{aligned} R(v, G_N) &= (N-1)! \prod_{u \in N_{G_N}(v)} \left(\frac{(|T_u^v - 1|)!}{|T_u^v|!} \prod_{w \in N_{G_N}(u)} \frac{R(w, T_w^v)}{|T_w^v|!} \right) \\ &= (N-1)! \prod_{u \in N_{G_N}(v)} \left(\frac{1}{|T_u^v|} \prod_{w \in N_{G_N}(u)} \frac{R(w, T_w^v)}{|T_w^v|!} \right) \end{aligned}.$$

Note that $T_l^v = \{l\}$ and $|T_l^v| = 1$ for leaf nodes l. If we repeat this last step for $x \in N_{G_N}(w)$, and so on, until we reach the leaves, we obtain

$$R(v, G_N) = (N-1)! \prod_{u \in V(G_N) \setminus \{v\}} \frac{1}{|T_u^v|}.$$

Since $|T_v^v| = N$, we conclude that

$$R(v, G_N) = N! \prod_{u \in V(G_N)} \frac{1}{|T_u^v|}.$$

□

The following proposition gives us a way to characterize the SZ-center when the graph G is a tree.

Proposition 2.1.4. *If node v^* is a SZ-center, then any subtree with v^* as the source must have the following property:*

$$|T_v^{v^*}| \leq \frac{N}{2}. \tag{2.12}$$

If there is a node u such that for all $v \neq u$

$$|T_v^u| \leq \frac{N}{2},$$

then u is a SZ-center. Furthermore, a tree can have at most 2 SZ-centers.

Proof. For a node v such that $d(v^*, v) = 1$, by Lemma 2.1.1 and Lemma 2.1.2 we find

$$\frac{R(v, T)}{R(v^*, T)} = \frac{\prod_{u \in V(G_N)} |T_u^{v^*}|}{\prod_{u \in V(G_N)} |T_u^v|} = \frac{|T_{v^*}^{v^*}||T_v^{v^*}|}{|T_{v^*}^v||T_v^v|}.$$

Note that, by definition $|T_{v^*}^{v^*}| = |T_v^v| = G_N$, and by Lemma 1.1.1, $|T_v^{v^*}| = N - |T_{v^*}^v|$, thus,

$$\frac{R(v, T)}{R(v^*, T)} = \frac{|T_v^{v^*}|}{N - |T_v^{v^*}|}. \tag{2.13}$$

Since $R(v^*, T) \geq R(v, T)$ by hypothesis, the previous equation implies,

$$\frac{|T_v^{v^*}|}{N - |T_v^{v^*}|} \leq 1,$$

so $|T_v^{v^*}| \leq \frac{N}{2}$. Since for any $w \in V(G_N)$, it's true that $|T_w^{v^*}| \leq |T_v^{v^*}| - 1$, then v^* satisfies (2.12).

For the converse, let v be a vertex such that $d(v, v^*) = 2$, and take $v_1 \in vRv^*$. Then, by (2.13)

$$\frac{R(v,T)}{R(v^*,T)} = \frac{R(v,T)R(v_1,T)}{R(v_1,T)R(v^*,T)} = \frac{|T_v^{v_1}||T_{v_1}^{v^*}|}{(N - |T_v^{v_1}|)(N - |T_{v_1}^{v^*}|)},$$

and since $|T_v^{v^*}| = |T_v^{v_1}|$, by Lemma 1.1.1, we see

$$\frac{R(v,T)}{R(v^*,T)} = \frac{|T_v^{v^*}||T_{v_1}^{v^*}|}{(N - |T_v^{v^*}|)(N - |T_{v_1}^{v^*}|)}.$$

Repeating this process by iteration over the vertices in vRv^*, we obtain that for any node v in G_N,

$$\frac{R(v,T)}{R(v^*,T)} = \prod_{v_i \in v^*Rv \setminus \{v^*\}} \frac{|T_{v_i}^{v^*}|}{(N - |T_{v_i}^{v^*}|)}, \tag{2.14}$$

Since $|T_v^{v^*}| \leq \frac{N}{2}$ for all $v \neq v^*$, we see that the ratios in (2.14) will all be less than or equal to 1. Thus, we find that

$$\frac{R(v,T)}{R(v^*,T)} \leq 1.$$

Therefore, v^* is a SZ-center, as claimed in the second part of the Proposition 2.1.4.

Finally, assume that v^* is a SZ-center. Let's consider two cases:

1. $|T_v^{v^*}| < \frac{N}{2}$, *for all* $v \in G_N, v \neq v^*$.

   The following claim guarantees that, in this case, there cannot be other SZ-centers different form v^*.

   Claim: for all $v \in G_N, v \neq v^*$, *there exists* $u \in G_N$ *such that* $|T_u^v| > \frac{N}{2}$.

   Proof of the claim: Assume that there exists $v_1 \neq v^*$ such that for every $u \neq v_1$,

   $$|T_u^{v_1}| \leq \frac{N}{2}.$$

   We know by the hypothesis of this case that $|T_v^{v^*}| < \frac{N}{2}$ for all $v \in N_{G_N}(v^*)$, $v \neq v^*$. Let v_0 be a neighbor of v^* such that $v_0 \in v_1Rv^*$. Since $T_{v_0}^{v_1} \supset T_{v^*}^{v_1} = T_{v^*}^{v_0}$, we have that $|T_{v^*}^{v_1}| \leq |T_{v_0}^{v_1}|$. Consequently,

   $$\begin{aligned} N &= |T_{v^*}^{v_0}| + |T_{v_0}^{v^*}| \\ &= |T_{v^*}^{v_1}| + |T_{v_0}^{v^*}| \\ &\leq |T_{v_0}^{v_1}| + |T_{v_0}^{v^*}| < N \end{aligned}$$

   which is absurd. □

2. *There exists* $v \in G_N$ *such that* $|T_v^{v^*}| = \frac{N}{2}$.

   Let $w \neq v$. If $w \in T_v^{v^*}$, then it is clear that $|T_w^v| \leq \frac{N}{2}$. On the other hand, if $w \notin T_v^{v^*}$, then there exists $v_0 \in N_{G_N}(v)$, such that $v_0 \in vRv^*$ and $w \in T_{v_0}^v$. Since $T_v^{v_0} = T_v^{v^*}$, we have $|T_v^{v_0}| = \frac{N}{2}$, and therefore, $|T_{v_0}^v| = \frac{N}{2}$. We are assuming that $w \in T_{v_0}^v$, so it is obvious that $T_w^v \subset T_{v_0}^v$, and thus, $|T_w^v| \leq \frac{N}{2}$. This way, $|T_w^v| \leq \frac{N}{2}$, for all $w \neq v$. This proves that v is also a SZ-center.

□

2.1.4 Message-Passing Algorithm

The simplification done in the previous section has some important theoretical implications but it is not so valuable in terms of computational costs. In this subsection we propose an alternative way of computing the SZ-centrality of each vertex using a technique known as message-passing algorithm.

Let u and v be two adjacent nodes in G_N. There is a special relation between these two and their subtrees:

$$|T_u^v| + |T_v^u| = N.$$

This relation allows us to express the SZ-centrality of u in terms of the SZ-centrality of v and the two previous subtrees mentioned before, i.e.,

$$R(u, G_N) = R(v, G_N)\frac{|T_u^v|}{N - |T_u^v|}. \tag{2.15}$$

The previous fact suggests that we can simply *pass* the SZ-centrality of a vertex to one of its neighbors to obtain its own SZ-centrality. This is the key for the message-passing algorithm.

Before explaining the algorithm, consider G_N as being $(G_N)^v$, i.e. v is the root of G_N, and let's define three important values for each node w in G_N:

- $\mathbf{T_{w\to parent(w)}}$ is the size of the subtree T_w^v, where $parent(w)$ is the unique vertex adjacent to w for which w is a first level descendant.
- $\mathbf{P_{w\to parent(w)}}$ is the cumulative product of the size of the subtrees of all nodes in T_w^v. Formally,
$$P_{w\to parent(w)} = \prod_{u\in T_w^v} T_u^v.$$
- $\mathbf{R_{w\to w'}}$ is the SZ-centrality of the vertex w, where w' is a first level descendant of w.

We want to compute $R(v, G_N)$. To do this the idea is go to the bottom of the tree, namely, to the leaves, and ask each one of them for a couple of messages. These messages are $T_{w\to parent(w)}$ and $P_{w\to parent(w)}$, where w is one of the mentioned leaves. Once this is done, we pass these two values to theirs parents, and we use them to compute both messages for each one of them. The way of doing this is simple, let's say that w' is one of the mentioned parents, then we only have to add up one and $T_{w\to parent(w)}$ for each $w \in child(w')$ to obtain $T_{w'\to parent(w')}$ and to multiply $P_{w\to parent(w)}$ for each $w \in child(w')$ together with $T_{w'\to parent(w')}$ to obtain $P_{w'\to parent(w')}$. We repeat these steps until we get to the vertex v, obtaining this way $R(v, G_N)$. This corresponds to the passing upwards part of the algorithm.

Once this is done, we use (2.15) to compute the SZ-centrality for each children of v. The desired value is equal to

$$R_{w\to v} = R(v, G_N)\frac{T_{w\to v}}{N - T_{w\to v}}.$$

Then, we continue like this until reaching the leaves of the tree again. This is the passing downwards part of the algorithm.

The following lines contain the pseudocode for the message-passing algorithm.

1. Choose a vertex v as the source node.
2. For each $u \in G_N$ do one of the following steps:
   (a) Set $T_{u\to parent(u)} = 1$ and $P_{u\to parent(u)} = 1$, if u is a leaf.
   (b) In the other case, consider one of the following two steps:
      i. if u is the root v, then $\forall v' \in child(v)$:
      $$R_{w\to w'} = \frac{N!}{N \prod_{j\in child(v)} P_{j\to v}}$$
      ii. if u is not the root, then define
      $$T_{u\to parent(u)} = \sum_{j\in child(u)} T_{j\to u} + 1$$
      $$P_{u\to parent(u)} = T_{u\to parent(u)} \prod_{j\in child(u)} P_{j\to parent(u)}$$
      $$\forall u' \in child(u) : R_{u\to u'} = R_{parent(u)\to u} \frac{T_{u\to parent(u)}}{N - T_{u\to parent(u)}}.$$

2.2 Source detection

In this section we present some results about the asymptotic behaviour of the ML estimator for pseudo-2 and pseudo-3-regular trees. The symbol $\mathbb{P}$ will denote the probability associated to the S.I. Model.

Theorem 2.2.1. *Define the event of correct rumor source detection after time n on a regular graph of degree $d = 2$ as $\mathcal{C}_n$. Then, the probability of correct detection of the ML estimator, $\mathbb{P}(\mathcal{C}_n)$, scales as*
$$\mathbb{P}(\mathcal{C}_n) = O\left(\frac{1}{\sqrt{n}}\right).$$

This theorem says that the probability of the estimator finding the source of the rumor goes to 0, as n goes to infinity. Note that a pseudo-2-regular tree is forcefully a line graph.

Proof. Let v be the source of the rumor. For this case we know that the SZ-center coincides with the result given by the ML estimator.

It follows from Proposition 2.1.4 that the SZ-center of this graph is a *middle* vertex of the graph: if we say that v_1 and v_N are the leaves of G_N, then v satisfies $d(v, v_1), d(v, v_N) \in \{N/2, N/2+1, N/2-1\}$. If N is odd, then there is only one distance center, namely, v. On the other hand, if N is even, then there exists $v^* \in N_{G_N}(v)$, such that both v and $v"*$ are distance centers. Note also that the process of rumor spreading behaves like two random variables B_n and $n - B_n$ where $B_n \sim \text{Bin}(n, 1/2)$ which correspond to the left and right expansions of the graph. Therefore, the event we are interested in can be written as
$$\mathcal{C}_n = \{n - B_n = B_n\} \cup \{n - B_n = B_n + 1\} \cup \{n - B_n = B_n - 1\}.$$

Thus,

$$\begin{aligned}\mathbb{P}(\mathcal{C}_n) &= \mathbb{P}\left(B_n = \frac{n}{2}\right) + \mathbb{P}\left(B_n = \frac{n}{2}+1\right) + \mathbb{P}\left(B_n = \frac{n}{2}-1\right) \\ &= \binom{n}{n/2}\frac{1}{2^n} + \binom{n}{n/2+1}\frac{1}{2^n} + \binom{n}{n/2-1}\frac{1}{2^n} \\ &\sim \frac{1}{\sqrt{n}},\end{aligned}$$

which concludes the proof.

□

Theorem 2.2.2. *Define the event of correct rumor source detection under the ML source estimator after time n on a regular expander tree with degree $d = 3$ as $\mathcal{C}_n$. Then*

$$\liminf_{n\to+\infty} \mathbb{P}(\mathcal{C}_n) \geq \frac{1}{4}.$$

Proof. Let v be the source node, and $v_1, v_2, v_3 \in G_N$ its children (the case where $deg(v) = 2$ is proven in a similar way). Let $N_n^{(j)}$ be the number of infected nodes in $T_{v_j}^v$ at time n, for $1 \leq j \leq 3$, and let's define the total number of infected nodes at time n as $N(n)$. Let $P_n^{(j)}$ be the number of nodes at time n that are part of the boundary of the subtree $T_{v_j}^v$ in the infinite 3-regular tree . If we define $P_n = (P_n^{(1)}, P_n^{(2)}, P_n^{(3)})^T$ for $n \in \mathbb{N}$, then P_n is a Pólya urn process with 3 colors and initial composition vector $P_0 = (1,1,1)^T$. We know by a classical result (Theorem A.0.1 in Appendix A p.(88)), that

$$\frac{1}{n}\left(P_n^{(1)}, P_n^{(2)}, P_n^{(3)}\right)$$

converges a.s. to a random vector Y such that $Y \sim \text{Dir}(1,1,1)$.

On the other hand, note that for all n and for all $i \in \{1,2,3\}$, we have that $P_n^{(i)} = N_n^{(i)} + 1$, and also,

$$\sum_{i=1}^{3} P_n^{(i)} = 3 + n. \tag{2.16}$$

Consider the event

$$\mathcal{A} = \left\{ \max_{1\leq i\leq 3} N_n^{(i)} \leq \frac{1}{2}\sum_{i=1}^{3} N_n^{(i)} \right\}.$$

Using (2.16) and the a.s. convergence of the random vector $\frac{1}{n}(P_n^{(1)}, P_n^{(2)}, P_n^{(3)})$, we see that

$$\lim_{n\to+\infty} \mathbb{P}(\mathcal{A}) = \mathbb{P}\left(\max_{1\leq i\leq 3} Y_i < \frac{1}{2}\right).$$

Now, let $U_{(1)}$ and $U_{(2)}$ be the order statistics of 2 i.i.d random variables U_1, U_2 such that $U_1 \sim \text{Unif}[0,1]$. By Remarks in Appendix A (p. 88) we know that Y satisfies

$$Y =_d (U_{(1)}, U_{(2)} - U_{(1)}, 1 - U_{(2)}).$$

Therefore,

$$\begin{aligned}\mathbb{P}\left(\max_{1\leq i\leq 3} Y_i < \frac{1}{2}\right) &= \mathbb{P}\left(\max\left\{U_{(1)}, U_{(2)} - U_{(1)}, 1 - U_{(2)}\right\} < \frac{1}{2}\right) \\ &= \mathbb{P}\left(U_1 < \frac{1}{2}, U_2 > \frac{1}{2}, U_2 - U_1 < \frac{1}{2}\right) + \mathbb{P}\left(U_2 < \frac{1}{2}, U_1 > \frac{1}{2}, U_1 - U_2 < \frac{1}{2}\right).\end{aligned}$$

Since

$$\begin{aligned}
\mathbb{P}\left(U_1 < \frac{1}{2}, U_2 > \frac{1}{2}, U_2 - U_1 < \frac{1}{2}\right) &= \int_0^{1/2}\int_{1/2}^{1} \mathbb{1}_{\{x_2 - x_1 < 1/2\}} dx_2 dx_1 \\
&= \int_0^{1/2}\int_{1/2}^{x_1+1/2} dx_2 dx_1 \\
&= \frac{1}{8},
\end{aligned}$$

and

$$\mathbb{P}\left(U_1 < \frac{1}{2}, U_2 > \frac{1}{2}, U_2 - U_1 < \frac{1}{2}\right) = \mathbb{P}\left(U_2 < \frac{1}{2}, U_1 > \frac{1}{2}, U_1 - U_2 < \frac{1}{2}\right),$$

we have that

$$\lim_{n\to+\infty} \mathbb{P}\left(\mathcal{A}\right) = \frac{1}{4}.$$

Since

$$\mathcal{A} \subset \mathcal{C}_n,$$

by Proposition 2.1.4, we conclude that

$$\liminf_{n\to+\infty} \mathbb{P}(\mathcal{C}_n) \geq \frac{1}{4}.$$

□

Chapter 3

Confidence Sets on UA and PA trees

Another approach for this source finding problem could be to ask for our estimators to output small sets for which we could guarantee, in most cases, that the source is in. Since the random growing graphs studied in this text are assumed to keep growing as time passes, it seems reasonable to hope that these small output sets grow at some similar rate. Some estimators are able to overcome this size increasing problem. They provide sets that contain the rumor source and whose size does not depend on the size of the main graph. These algorithms are known as *root-finding algorithms.* We dedicated this chapter to the study of some properties of these estimators. We analyze them in the cases where the graphs follow the Uniform and Preferential attachments models. We give lower bounds for the size of the confidence sets for any root-finding algorithm and we analyze the behaviour of some specific estimators.

For this and the remaining chapters we only consider the integer part of any of the shown ratios and square roots.

3.1 Simple Estimator

As stated in Chapter 1, given $\varepsilon \in (0,1)$ and a tree T of size n (for some $n \in \mathbb{N}$), a *root-finding algorithm* is a procedure that outputs sets $H(T,\varepsilon) \subset V(T)$ with $|H(T,\varepsilon)| = K(\varepsilon)$, such that, with probability $1-\varepsilon$ (conditioned on T, randomly generated), $0 \in H(T,\varepsilon)$. The set H is known as a *confidence set.*

In this section we present the first example of a root-finding algorithm. It will be important for analyzing the behavior of other algorithms proposed later.

Let T be a tree and consider $\psi_T : V(T) \to \mathbb{N}$, defined by

$$\psi_T(u) = \max_{v \in V(T)\setminus\{u\}} |T_v^u|.$$

Intuitively, the function $\psi_T(u)$ computes the size of biggest tree rooted at u begining at a descendant of it.

Let H_ψ the set estimator that returns the set of K vertices with smallest ψ_T values (ties are broken arbitrarily). Clearly, H_ψ satisfies Property 2. Actually, we will prove later that H_ψ is a root-finding algorithm. It is known as *simple estimator* [6].

Theorem 3.1.1. *Given $\varepsilon > 0$. In the uniform attachment model, for $K_\psi \geq 2.5\frac{\log(1/\varepsilon)}{\varepsilon}$, it is true that*

$$\liminf_{n\to\infty} \mathbb{P}(0 \in H_\psi(UA(n))) \geq 1 - \frac{4\varepsilon}{1-\varepsilon}.$$

The results related to Pólya urns and Dirichlet distributions needed for the next proof are all contained in Appendix A.

In the following proof the vertices are labeled according to order of appearance. Consider the following procedure: Let UA(n) be the uniform attachment tree with $n+1$ vertices, fix $i \in V(T)$ and let $F_i(\text{UA}(n))$ be the set of trees, subgraphs of UA(n), resulting of removing all the existing edges between vertices $\{0,1,2,...,i\}$. This graph has exactly $i+1$ components and each node in the set $\{0,1,2,...,i\}$ belongs to a different one. We reference this components using the notation $T_{i,j}$ for $j \in \{0,1,2,...,i\}$. For instance the symbol $T_{0,j}$ makes reference to the component of $F_i(\text{UA}(n))$ that contains 0.

We begin by proving some properties of the function ψ. This properties will allow us to bound the value of $\psi(0)$ which is going to be useful later.

Lemma 3.1.1. *(i) $\psi(0) \leq \max(|T_{0,1}|, |T_{1,1}|)$,*

(ii) for any $i > K$, $\psi(i) \geq \min_{1\leq l\leq K} \sum_{j=1,j\neq l}^{K} |T_{j,K}|$.

Proof. (i) Note that $T_{1,1} = T_1^0$ and $T_{0,1} = T_0^1$, and therefore, $V(T_{1,1}) \cup V(T_{0,1}) = \{0,1,2,...,n\}$. Without lost of generality, we can assume that $|T_{1,1}| \geq |T_{0,1}|$. Thus, proving the desired inequality is equivalent to show that

$$|T_v^0| \leq |T_{1,1}|, \quad \text{for all } v \in \{1,2,...,n\}. \tag{3.1}$$

On the one hand, $|T_v^0| \leq |T_{0,1}| \leq |T_{1,1}|$ for all $v \in T_{0,1}$ with $v \neq 0$. On the other, $T_w^0 \subsetneq T_1^0$, for all $w \in T_1^0$, with $w \neq 1$. Therefore, $|T_v^0| \leq |T_{1,1}|$ for all $v \in T_{0,1}$ with $v \neq 0$ and $|T_w^0| \leq |T_1^0| = |T_{1,1}|$, for all $w \in T_1^0$. These two arguments show that (3.1) is true.

(ii) Since $i > K$, then there exists $l \in \{0,1,..,K\}$ such that $i \in T_{l,K}$. Let v_l be the neighbor of i in the unique path iRl. Clearly,

$$\psi(i) \geq |T_{v_l}^i|. \tag{3.2}$$

Claim 1: Any $w \in (T_{l,K})^c$ belongs to $T_{v_l}^i$.

Proof of the Claim 1. The claim 1 is equivalent to the claim that if $w \in \{0,1,2,...,n\}$ and $w \notin T_{l,K}$, then the unique path iRw in the tree $UA(n)$ contains v_l.

In fact we will show that the path iRw goes through l. Note that all edges in UA(n) belong to one of the following kinds:

(a) Edges internal to $T_{j,K}$ $(j = 0,...,K)$,

(b) Edges between $j_1, j_2 \in 0,...,K$.

Remember that we are assuming that i belongs to $T_{l,K}$ but w does not. Therefore, there must be an index t such that

$$i = v_1 \sim v_2 \sim ... \sim v_t \sim v_{t+1} \sim ... \sim v_r = w$$

where $v_1,...,v_t \in T_{l,K}$ and $v_{t+1} \notin T_{l,K}$. Thus, $\{v_t, v_{t+1}\}$ is not of type (a), which implies $v_t, v_{t+1} \in \{0,1,...,K\}$. This way, $v_t \in \{0,1,...,K\} \cap T_{l,K}$, and thus, $v_t = l$, and the proof is complete. □

From the claim we see that

$$V(T^i_{v_l}) \supset \bigcup_{j\neq i, 0\leq j\leq K} V(T_{j,K}),$$

and since $V(T_{j,K}) \cap V(T_{k,K}) = \emptyset$, for $j \neq k$, we have that

$$|T^i_{v_l}| \geq \sum_{j\neq i, 0\leq j\leq K} |T_{j,K}|.$$

This last inequality combined with (3.2) completes the proof of (ii).

$\square$

Proof of the Theorem 3.1.1. Note that the vector $(|T_{0,K}|, ..., |T_{K,K}|)$ follows a standard Pólya urn with $K+1$ colors, and thus, using a classical result (Theorem A.0.1 in Appendix A p.88),

$$\frac{1}{n}(|T_{0,K}|, ..., |T_{K,K}|)$$

converges, in distribution, to a Dirichlet distribution with parameters $(1)_{i=1}^{K+1}$.

On the other hand, note that, by definition of H_ψ,

$$\mathbb{P}(0 \notin H_\psi) \leq \mathbb{P}(\exists i > K : \psi(i) \leq \psi(0)).$$

Moreover,

$$\begin{aligned}\mathbb{P}(\exists i > K : \psi(i) \leq \psi(0)) \quad &= \quad \mathbb{P}(\exists i > K : \psi(i) \leq \psi(0) < (1-\varepsilon)n) \\ &\quad\; + \mathbb{P}((1-\varepsilon)n \leq \psi(0))\end{aligned}$$

and

$$\mathbb{P}(\exists i > K : \psi(i) \leq \psi(0) < (1-\varepsilon)n) \leq \mathbb{P}(\exists i > K : \psi(i) \leq (1-\varepsilon)n)$$

thus

$$\mathbb{P}(0 \notin H_\psi) \leq \mathbb{P}(\exists i > K : \psi(i) \leq (1-\varepsilon)n) + \mathbb{P}((1-\varepsilon)n \leq \psi(0)).$$

By Lemma 3.1.1(i),

$$\psi(0) \leq \max(|T_{0,1}|, |T_{1,1}|),$$

therefore,

$$\mathbb{P}(\psi(0) \geq (1-\varepsilon)n) \leq 2\mathbb{P}(|T_{0,1}| \geq (1-\varepsilon)n).$$

Since $|T_{0,1}|/n$ and $|T_{1,1}|/n$ are identically distributed and converge in distribution to a uniform random variable in $[0,1]$ by Corollary A.0.2, we see

$$\limsup_{n\to+\infty} \mathbb{P}(\psi(0) \geq (1-\varepsilon)n) \leq 2 \limsup_{n\to+\infty} \mathbb{P}(|T_{0,1}| \geq (1-\varepsilon)n) = 2\varepsilon$$

.

On the other hand, by Lemma 3.1.1(ii), for any $i > K$,

$$\psi(i) \geq \min_{0\leq l\leq K} \sum_{j=0, j\neq l}^{K} |T_{j,K}|$$

and therefore,

$$\{\exists i > K : \psi(i) \leq (1-\varepsilon)n\} \subset \{\exists\ 0 \leq k \leq K : \sum_{j=0, j\neq k}^{K} |T_{j,K}| \leq (1-\varepsilon)n\}$$

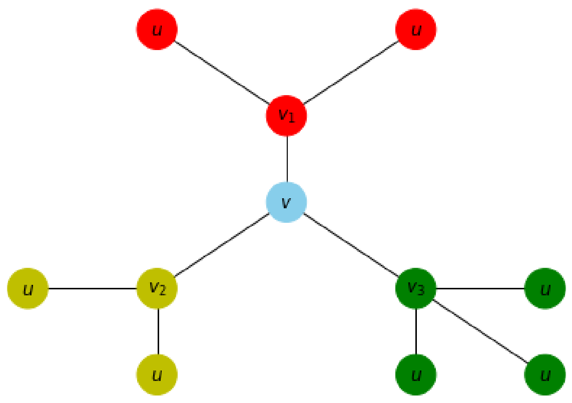


Figure 3.1: T_1 in red, T_2 in yellow, T_3 in green

Using this last inclusion and the fact that $\frac{1}{n}\sum_{j=1,j\neq k}^{K} |T_j, K|$ converges, in distribution, to the Beta$(K,1)$ distribution, we see

$$\begin{aligned}\limsup_{n\to+\infty} \mathbb{P}(\exists i > K : \psi(i) \leq (1-\varepsilon)n) &\leq \lim_{n\to+\infty} \mathbb{P}\left(\exists\, 0 \leq l \leq K : \textstyle\sum_{j=1,j\neq l}^{K} |T_{j,K}| \leq (1-\varepsilon)n\right) \\ &\leq (K+1)(1-\varepsilon)^K.\end{aligned}$$

This way, we showed that

$$\limsup_{n\to+\infty} \mathbb{P}(0 \notin H_\psi) \leq 2\varepsilon + (K+1)(1-\varepsilon)^K,$$

and K must satisfy $K \geq 2.5\log(1/\varepsilon)/\varepsilon$ in order to get

$$\limsup_{n\to+\infty} \mathbb{P}(0 \notin H_\psi) \leq \frac{4\varepsilon}{1-\varepsilon}.$$

□

3.2 Lower Bound for UA(n)

As we mentioned in the previous section, a root-finding algorithm is an estimator that returns a set that contains the source with high probability, whose size does not depend on the size of the main graph. In this section we give a lower bound for the size of the output set of *any* root-finding algorithm. This means that if K_n^p is a root-finding algorithm, then p must be greater that some constant that we will exhibit later.

We begin by proving a proposition on the number of recursive trees.

For a rooted tree T and a vertex v, we define $\mathbf{Aut}(v,T)$ as follows. Let $T_1,...,T_k$ be the subtrees of T rooted at the children of v (in particular $k \in \{deg(v)-1, deg(v)\}$, where the cases correspond to either $v \neq 0$ or $v = 0$). Let $\mathcal{S}_1,...,\mathcal{S}_L$ be the different isomorphism classes realized by these rooted subtrees. For $i \in [L]$, let $l_i = |\{j \in [k] : i(T_j) = \mathcal{S}_i\}|$. Finally, we let $\mathbf{Aut}(v,T) := \prod_{i=1}^{L} l_i!$.

Note that for the graph T and $v \in V(T)$ (both shown in Figure 3.1) we have that $l_1 = 2$ and $l_2 = 1$. Therefore, for $\mathbf{Aut}(v,T) = (2!)(1!) = 2$.

Proposition 3.2.1. *Let T be a rooted tree, then*

$$|\{t \text{ recursive tree } : t \in i(T)\}| = \frac{|T|!}{\prod\limits_{v \in V(T)\setminus\mathcal{L}(T)} (|T_v| \cdot \mathbf{Aut}(v,T))}. \tag{3.3}$$

Proof. We prove this proposition by induction on the number of vertices n of T.

Let's verify that the result is true for $n = 2$: Since T_1 is itself its isomorphism class and we have no other isomorphism class (in this case there is only one subtree rooted because v has only one child, say v_1), we have that $l_1 = 1$ and therefore, $\mathbf{Aut}(v,T) = 1$. We also know that $|T| = 2$ and $|T_v| = 2$. This way

$$\frac{|T|!}{|T_v| \cdot \mathbf{Aut}(v,T)} = 1$$

and since there is only one recursive tree of 2 vertices, we see that (3.3) is true for $n = 2$.

Let $T_1, ..., T_k$ be the subtrees rooted at the children of 0 (in particular $k = deg(0)$). Let's construct an increasing labeling for T in the following way: First, partition $\{1, ..., n-1\}$ into subsets L_i of sizes $|T_i|$, for each $i \in [k]$ and then choose each T_i and use the labels in L_i to name each element of T_i in an increasing way. We know that the number of possibilities for choosing this partition is

$$\binom{n-1}{|T_1|, ..., |T_k|} = \frac{(n-1)!}{\prod_{i=1}^{k}(|T_i|!)}$$

.

Note that all the labelings can be obtained this way. Now, fix a partition $\{L_1, ..., L_k\}$ of $\{1, ..., n-1\}$. Fix an element of these partion, say $L_i = \{a_1, a_2, ..., a_k\}$, where $a_j < a_{j+1}$ for all $1 \leq j < k$. The number of right labelings that can be assigned to the subtree T_i using this set L_i is equal to the number of recursive trees t such that t is isomorphic to T_i. Consequently,

$$\prod_{i=1}^{k} |\{t \text{ recursive tree } : t \in i(T_i)\}|$$

should be equal to the number of right labelings for a given partition. In this calculation we have counted several times right labelings for isomorphic subtrees. This way, we need to remove these repeated labelings, which is equivalent to removing the number of isomorphic subtrees of T. By definition, we have that this number is $\mathbf{Aut}(0,T)$. Therefore,

$$|\{t \text{ recursive tree } : t \in i(T)\}| = \frac{(n-1)!}{\mathbf{Aut}(0,T)\prod_{i=1}^{k}|T_i|!} \prod_{i=1}^{k} |\{t \text{ recursive tree } : t \in i(T_i)\}|.$$

By the induction hypothesis we know that

$$|\{t \text{ recursive tree } : t \in i(T_i)\}| = \frac{|T_i|!}{\prod\limits_{v \in V(T_i)\setminus\mathcal{L}(T_i)} |(T_i)_v| \cdot \mathbf{Aut}(v,T_i)},$$

and since

$$\prod_{i} \prod_{v \in V(T_i)\setminus\mathcal{L}(T_i)} (|(T_i)_v| \cdot \mathbf{Aut}(v,T_i)) = \prod_{v \in V(T)\setminus\mathcal{L}(T)\cup\{0\}} (|T_v| \cdot \mathbf{Aut}(v,T))$$

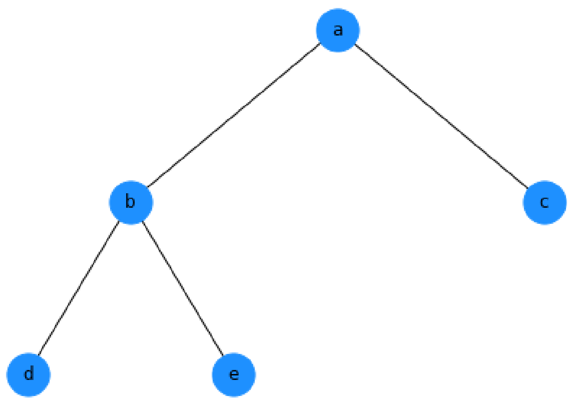
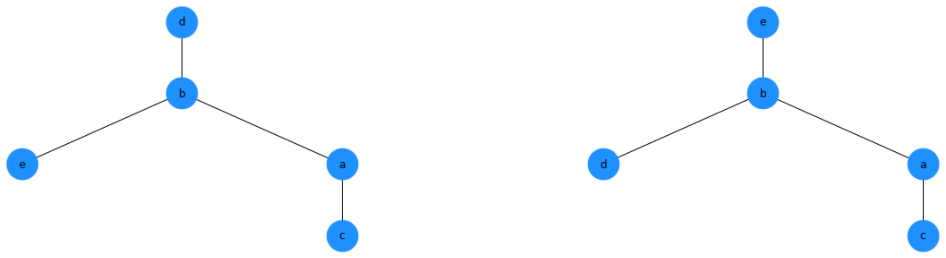


Figure 3.2: Graph T



(a) Rooted view (b) Rooted view

Figure 3.3: Ilustrating $\overline{\text{Aut}}(u, T)$ for T Figure 3.2

we can conclude

$$|\{t \text{ recursive tree} : t \in i(T)\}| = \frac{|T|!}{\prod_{v \in V(T)\setminus\mathcal{L}(T)} (|T_v| \cdot \mathbf{Aut}(v, T))}.$$

$\square$

Let T be an unrooted tree. Define $\overline{\mathbf{Aut}}(u, T)$ as the number of vertices v such that T^v is isomorphic to T^u. Intuitively, $\overline{\mathbf{Aut}}(u, T)$ counts the number of vertices v for which T^u and T^v have the same shape in the rooted view. For instace, if T is the tree shown in Figure 3.2, then $\overline{\mathbf{Aut}}(i, T) = 2$ for $i \in \{d, e\}$, while $\overline{\mathbf{Aut}}(i, T) = 1$ for $i \in \{a, b, c\}$.

Note that, in the UA mechanism, there exists $C > 0$, such that

$$\begin{aligned}\mathbb{P}_v(T) &= \frac{\mathbb{P}_v(T = T^v)}{\overline{\mathbf{Aut}}(v, T)} \\ &= C\, \frac{|\{t \text{ recursive tree} : t \in i(T^v)\}|}{\overline{\mathbf{Aut}}(v, T)}\end{aligned}$$

and Proposition 3.2.1 implies that

$$\mathbb{P}_v(T) = C\, \frac{|T|!}{\overline{\mathbf{Aut}}(v, T) \prod_{u \in V(T)\setminus\mathcal{L}(T)} (|T_u^v| \cdot \mathbf{Aut}(u, (T^v)))}.$$

Consequently, the maximum likelihood estimator for the root in the uniform attachment model is the vertex v minimizing the function

$$\zeta_T(v) = \overline{\mathbf{Aut}}(v,T) \prod_{u \in V(T)\setminus\mathcal{L}(T)} (|T_u^v| \cdot \mathbf{Aut}(u, T^v)).$$

Lemma 3.2.1. *Let X be a random variable on the set $[n]$. Let $p_i = \mathbb{P}(X = i)$ for all $i \in [n]$ and suppose that*

$$p_1 \geq p_2 \geq \ldots \geq p_n.$$

Fix $k \in [n]$. If $\mathcal{S}$ is a subset of $[n]$ of size k that also satisfies

$$\mathbb{P}(X \in \mathcal{S}) = \max_{|S|=k} \mathbb{P}(X \in S),$$

then $\mathcal{S} = [k]$.

Proof. This is a consequence of the fact that

$$\mathbb{P}(X \in S) = \sum_{i \in S} p_i \leq \sum_{i=1}^{k} p_i,$$

for any subset S of $[n]$. □

A application of the Lemma 3.2.1, shows us that since

$$\mathbb{P}_v(T) = \frac{C_T}{\zeta_T(v)} \quad \text{for } v \in V(T) \quad \text{(where } C_T \text{ is a normalization constant)},$$

then the optimal strategy to output a set of K vertices is choosing those with the smallest values of ζ.

We denote the mapping corresponding to this optimal strategy by H_ζ. Using this representation for the optimal procedure we prove now the main result of this section.

Let H be an algorithm for outputting vertices as candidates for the source. For a given p, denote H_p be a set estimator of size p built by the algorithm H, i.e., given UA(n), then $H_p(\text{UA}(n))$ is a subset of $V(\text{UA}(n))$ for candidates to the source obtained using the algorithm H and $|H_p(\text{UA}(n))| = p$. For $\varepsilon > 0$, define

$$K_H(\varepsilon) = \min \{p \in \mathbb{N} : \mathbb{P}(0 \in H_p(\text{UA}(n))) \geq 1 - \varepsilon\}.$$

Theorem 3.2.1. *There exists $\varepsilon_0 \in (0,1)$, such that for any positive $\varepsilon < \varepsilon_0$ and any set estimator H of size p,*

$$K_H(\varepsilon) \geq \exp\left(\sqrt{\frac{1}{30} \log \frac{1}{2\varepsilon}}\right).$$

Proof. For simplicity we will write K for $K_H(\varepsilon)$.

Let $K = \exp\left(\sqrt{\frac{1}{30}\log\frac{1}{2\varepsilon}}\right)$, which we assume to be large enough (that is ε is small enough, see below).

Since any algorithm that outputs a set of candidates for trees of size $n+1$ can be simulated given a tree of size n, we have that $\mathbb{P}(0 \in H_\zeta(UA(n))) \geq \mathbb{P}(0 \in H_\zeta(UA(n+1)))$. Thus, we only have to show that $\mathbb{P}(0 \notin H_\zeta(UA(K))) > \varepsilon$ to prove the theorem. In order to do this, we are going to define a collection of trees $\mathcal{T}$ and then using the fact that

$$\begin{aligned} \{0 \notin H_\zeta(\text{UA}(K))\} &\supset \bigcup_{T \in \mathcal{T}} \{\text{UA}(K+1) = T\} \\ &= \{\text{UA}(K+1) \in \mathcal{T}\} \end{aligned}$$

we will reduce the problem of bounding $\mathbb{P}(0 \notin H_\zeta(\mathrm{UA}(K)))$ to get a bound for

$$\mathbb{P}\left(\mathrm{UA}(K+1) \in \mathcal{T}\right).$$

Define

(i) $\mathcal{T}_1$ as the set of recursive trees T such that the vertices $\{0, ..., 10\log(K)\}$ form a path with 0 being an endpoint,

(ii) $\mathcal{T}_2$ as the set of recursive trees such that all vertices in $\{10\log(K)+1, ..., K+1\}$ are descendants of $10\log(K)$,

(iii) $\mathcal{T}_3$ as the set of recursive trees T such that the $h(T^{10\log(K)}) \leq 4\log(K)$, where the mapping h gives the height of the graph.

And finally, let $\mathcal{T} = \mathcal{T}_1 \cap \mathcal{T}_2 \cap \mathcal{T}_3$. We will verify that

$$\mathbb{P}\left(T \in \mathcal{T}\right) \geq \frac{1}{2}\exp(-30\log^2(K)) = \varepsilon.$$

To do this, we will analyze each condition on the definition of $\mathcal{T}$ separately.

(i) Clearly,

$$\mathbb{P}(T \in \mathcal{T}_1) = \prod_{i=2}^{10\log K} \frac{1}{i-1} = \frac{1}{(10\log(K)-1)\,!},$$

and since

$$\frac{1}{(10\log(K)-1)\,!} \geq \exp(-10\log^2(K)),$$

we have

$$\mathbb{P}(T \in \mathcal{T}_1) \geq \exp(-10\log^2(K)).$$

(ii) On the other hand,

$$\mathbb{P}(T \in \mathcal{T}_2 | T \in \mathcal{T}_1) = \prod_{i=10\log(K)+1}^{K} \left(1 - \frac{10\log(K)-1}{i}\right),$$

since conditioned on $T \in \mathcal{T}_1$, we must choose $j \in \{10\log(K)+1, 10\log(K)+2, ..., K\}$ and connect it to $i+1$ in order to get $T(i+1)$ from $T(i)$, for some $i \geq 10\log(K)+1$. Using

$$\prod_{i=10\log(K)+1}^{K} \left(1 - \frac{10\log(K)-1}{i}\right) \geq \exp(-20\log^2(K)),$$

we see

$$\mathbb{P}(T \in \mathcal{T}_2 | T \in \mathcal{T}_1) \geq \exp(-20\log^2(K))$$

(iii) Let $T \in \mathcal{T}_1 \cap \mathcal{T}_2$. Note that conditioned on $T \in \mathcal{T}_1$, the tree $T^{10\log K}$ which is the subtree of T rooted at $10\log K$, satisfies

$$T^{10\log K} =_d \mathrm{UA}(K - 10\log K). \tag{3.4}$$

To prove this we need to show that at a given time $1 \leq i \leq K - 10\log K$ and for any $j \leq i$,

$$\mathbb{P}(i+1+10\log K \sim j + 10\log K | T^{10\log K}(i)) = \frac{1}{i+1}.$$

We know that conditioned on $T \in \mathcal{T}_1$,

$$\mathbb{P}(i+1+10\log K \sim j | T(i+10\log K)) = 0,$$

by the definition of $\mathcal{T}_1$. Therefore, to go from $T(i+10\log K)$ to $T(i+1+10\log K)$ we only can add the vertex $i+1+10\log K$ to some vertex chosen uniformly from $T^{10\log K}(i)$. Conditioned on $T \in \mathcal{T}_1$, getting $T(i+10\log K)$ to $T(i+1+10\log K)$ is exactly as getting $T^{10\log K}(i+1)$ to $T^{10\log K}(i)$, we see that (3.4) is true.

Equality (3.4) implies that

$$\mathbb{P}(T \in \mathcal{T}_3 | T \in \mathcal{T}_1 \cap \mathcal{T}_2) = \mathbb{P}(h(\mathrm{UA}(K-10\log K)) \leq 4\log K),$$

and since [10]

$$\mathbb{P}\left(\left|\frac{h(\mathrm{UA}(n))}{n} - e\right| > \varepsilon\right) \underset{n\to+\infty}{\longrightarrow} 0, \quad \forall \varepsilon > 0.$$

we have that

$$\mathbb{P}(T \in \mathcal{T}_3 | T \in \mathcal{T}_1 \cap \mathcal{T}_2) = 1 - o(1),$$

Combining items (i), (ii) and (iii), we obtain

$$\begin{aligned} \mathbb{P}(T \in \mathcal{T}) &= \mathbb{P}(T \in \mathcal{T}_3 | T \in \mathcal{T}_1 \cap \mathcal{T}_2)\ \mathbb{P}(T \in \mathcal{T}_2 | T \in \mathcal{T}_1)\ \mathbb{P}(T \in \mathcal{T}_1) \\ &\geq \frac{1}{2}\exp(-30\log^2(K)) \end{aligned}$$

Now, to conclude the proof we show that for trees in $\mathcal{T}$, the root 0 has the largest ζ value, and therefore, it is not in the output set given by H_ζ.

First observe that, in general for any tree T and vertices u, v, with $(u_1, ..., u_k)$ being the unique path with $u_1 = u$ and $u_k = v$,

$$\begin{gathered} \zeta_T(u) > \zeta_T(v) \\ \Longleftrightarrow \\ \overline{\mathbf{Aut}}(u,T)\prod_{i=1}^k \left(|T^u_{u_i}| \cdot \mathbf{Aut}(u_i, T^u)\right) > \overline{\mathbf{Aut}}(v,T)\prod_{i=1}^k \left(|T^v_{u_i}| \cdot \mathbf{Aut}(u_i, T^v)\right), \end{gathered} \tag{3.5}$$

because $T^u_w = T^v_w$, for all w that is not part of the path that connects u and v. We will use this equivalence to prove that $\zeta(0)$ is maximum.

We also know that computation of $\mathbf{Aut}(u_i, T^v)$ is based on a list of subtrees $T_1, ..., T_k$ and only one of those subtrees is modified for computing $\mathbf{Aut}(u_i, T^u)$ (or possibly a subtree is added if $i = 1$). Thus, by definition of Aut, the ratio $\mathbf{Aut}(u_i, T^v)/\mathbf{Aut}(u_i, T^u)$ is bounded by the number of subtrees $k + 1 \leq |T^v_{u_i}|$, therefore

$$\mathbf{Aut}(u_i, T^v) \leq |T^v_{u_i}| \cdot \mathbf{Aut}(u_i, T^u). \tag{3.6}$$

Now, using the bound $1 \leq \overline{\mathbf{Aut}}(u,T) \leq |T|$ and the inequality (3.6), we have the following chain of implications

$$\begin{aligned} & \prod_{i=1}^k |T^u_{u_i}| &&> |T|\prod_{i=1}^k |T^v_{u_i}|^2 \\ \Rightarrow\ & \overline{\mathbf{Aut}}(u,T)\prod_{i=1}^k |T^u_{u_i}| &&> \overline{\mathbf{Aut}}(v,T)\prod_{i=1}^k |T^v_{u_i}|^2 \\ \Rightarrow\ & \overline{\mathbf{Aut}}(u,T)\prod_{i=1}^k |T^u_{u_i}|\mathbf{Aut}(u_i,T^u) &&> \overline{\mathbf{Aut}}(v,T)\prod_{i=1}^k |T^v_{u_i}|^2\mathbf{Aut}(u_i,T^u) \\ \Rightarrow\ & \overline{\mathbf{Aut}}(u,T)\prod_{i=1}^k |T^u_{u_i}|\mathbf{Aut}(u_i,T^u) &&> \overline{\mathbf{Aut}}(v,T)\prod_{i=1}^k |T^v_{u_i}|\mathbf{Aut}(u_i,T^v), \end{aligned}$$

and thus, we obtain

$$\prod_{i=1}^{k} |T^u_{u_i}| > |T| \prod_{i=1}^{k} |T^v_{u_i}|^2 \Rightarrow \zeta_T(u) > \zeta_T(v). \tag{3.7}$$

Fix $T \in \mathcal{T}$, and pick any $v \in \{10\log(K)+1, ..., K+1\}$. Let $u_1 \sim ... \sim u_k$ be the unique path with $u_1 = 0$, $u_k = v$ and let j be the unique index such that $u_j = 10\log(K)$ for some $j < k$. Note that

$$\prod_{i=1}^{k} |T^v_{u_i}| = (10\log(K))! \prod_{i=j+1}^{k} |T^v_{u_i}| \le (10\log(K))!(K+1)^{4\log(K)}$$

and since

$$(10\log(K))! \le (10\log(K))^{10\log(K)},$$

we see

$$\prod_{i=1}^{k} |T^v_{u_i}| \le (K+1)^{4\log(K)} (10\log(K))^{10\log(K)}$$

On the other hand,

$$\prod_{i=1}^{k} |T^u_{u_i}| \ge (K+1-10\log(K))^{10\log(K)}.$$

Thus using (3.7) and the fact that for the value of K fixed initially,

$$(K+1-10\log(K))^{10\log(K)} \ge (K+1)^{4\log(K)} (10\log(K))^{10\log(K)},$$

we have that $\zeta_T(0) > \zeta_T(v)$. Furthermore it is obvious that $\zeta_T(0) > \zeta_T(v)$ for $v \in \{1, ..., 10\log(K)\}$. This way, 0 maximizes ζ_T and, therefore, $0 \notin H_\zeta(UA(K+1))$ which concludes the proof. □

3.2.1 An upper bound for K

In this section we present another source estimator and we prove an upper bound for the size of the output set it returns.

Let T be a tree and consider $\phi_T : V(T) \to \mathbb{N}$, defined by

$$\phi_T(u) = \prod_{v \in V(T)\setminus\{u\}} |T^u_v|.$$

Let H_ϕ the set estimator of size K that returns the set of K vertices with smallest ϕ_T values (ties are broken arbitrarily). It can be proved that the value

$$\frac{1}{\phi_T(u)}$$

coincides with the SZ-centrality of u mentioned in the Definition 2.1.3. Note that the estimator H_ϕ also satisfies Property 2.

To begin let's prove the following result on ϕ_T.

Lemma 3.2.2. *If u and v are vertices of T, then, for any vertex $w \in uRv$, it is true that*

$$\phi(w) \le \max(\phi(u), \phi(v)). \tag{3.8}$$

Proof. Assume first that $uRv = (u, w, v)$ and suppose that $\phi(u) \leq \phi(w)$. Note that we can rewrite $\phi(w)$ and $\phi(u)$ as

$$\begin{aligned} \phi(w) &= \left(\prod_{j \in T_w^u} T_j^w\right)\left(\prod_{j \in T_w^v} T_j^w\right)\left(\prod_{j \in C} T_j^w\right) \\ \phi(u) &= \left(\prod_{j \in T_u^w} T_j^u\right)\left(\prod_{j \in T_w^u} T_j^w\right), \end{aligned} \tag{3.9}$$

where $C = (T_w^u \cup T_w^v \cup \{w\})^c$. Then, from $\phi(u) \leq \phi(w)$ and (3.9), we have

$$\prod_{j \in T_w^u} T_j^w \leq \left(\prod_{j \in T_v^w} T_j^w\right)\left(\prod_{j \in C} T_j^w\right),$$

which implies

$$\left(\prod_{j \in T_w^u} T_j^u\right)\left(\prod_{j \in T_u^w} T_j^w\right) \leq \left(\prod_{j \in T_v^w} T_j^w\right)\left(\prod_{j \in C} T_j^w\right)\left(\prod_{j \in T_u^w} T_j^w\right).$$

Note that

$$\begin{gathered} \left(\prod_{j \in T_w^u} T_j^u\right)\left(\prod_{j \in T_u^w} T_j^w\right) = \left(\prod_{j \in C} T_j^u\right)\left(\prod_{j \in T_v^w} T_j^u\right)\left(\prod_{j \in T_u^w} T_j^w\right) = \phi(w), \\ \left(\prod_{j \in T_v^w} T_j^w\right)\left(\prod_{j \in C} T_j^w\right)\left(\prod_{j \in T_u^w} T_j^w\right) = \phi(v), \end{gathered}$$

and therefore, $\phi(w) \leq \phi(v)$, which proves that

$$\phi(w) \leq \max(\phi(u), \phi(v)). \tag{3.10}$$

Now, for the general case, write the unique path between u and v as

$$u \sim w_1 \sim w_2 \sim \ldots \sim w_{i-1} \sim w_i \sim w_{i+1} \sim \ldots \sim w_l \sim v,$$

where $w = w_i$ for some $i \in [l]$. Assume that $\phi(w_i) \leq \phi(w_{i+1})$. Since

$$\phi(w_{i+1}) \leq \max(\phi(w_i), \phi(w_{i+2}))$$

by (3.10) we necessarily have that $\phi(w_{i+1}) \leq \phi(w_{i+2})$. Repeating this same argument but this time for $\phi(w_{i+2})$, we obtain

$$\phi(w_i) \leq \phi(w_{i+1}) \leq \phi(w_{i+2}) \leq \phi(w_{i+3}).$$

Iterating this technique, we are going to get

$$\phi(w_i) \leq \phi(v),$$

which proves (3.8). Assume now that $\phi(w_{i+1}) \leq \phi(w_i)$. Once again by (3.10), we have

$$\phi(w_i) \leq \phi(w_{i-1}).$$

Therefore,

$$\phi(w_{i+1}) \leq \phi(w_i) \leq \phi(w_{i-1}).$$

This argument, applied repeatedly, leads to

$$\phi(w_i) \leq \phi(u),$$

which implies (3.8).

□

Now, we are in conditions to state and prove the main result of this section:

Theorem 3.2.2. *Let $\varepsilon > 0$. There exist $a, b > 0$ such that if*

$$K_\phi \leq a \exp\left(b \frac{\log(1/\varepsilon)}{\log\log(1/\varepsilon)}\right),$$

then

$$\liminf_{n\to+\infty} \mathbb{P}(0 \in H_\phi(UA(n))) \geq 1 - \varepsilon.$$

Before proving the previous theorem, we exhibit the proof of another useful result. The definition of the Gamma distribution is contained in Appendix A.

Lemma 3.2.3. *Let $a_1, ..., a_l \in \mathbb{N}$ and $s = \sum_{k=1}^{l} ka_k$. For $k \in [l]$, let $X_k \sim Ga(a_k, k)$, with $X_1, ..., X_l$ being independent. Then for any $t \in (0, s)$,*

$$\mathbb{P}\left(\sum_{k=1}^{l} X_k < t\right) \leq \exp\left(-\sqrt{\frac{s}{2}} \log\left(\frac{s}{et}\right)\right).$$

Proof. Let $\lambda \geq 0$. Note that

$$\begin{aligned} \mathbb{P}\left(\textstyle\sum_{k=1}^{l} X_k < t\right) &= \mathbb{P}\left(\exp\left(\lambda t - \textstyle\sum_{k=1}^{l}\right) > 1\right) \\ &\leq \exp(\lambda t)\mathbb{E}\exp\left(-\lambda \textstyle\sum_{k=1}^{l} X_k\right). \end{aligned} \tag{3.11}$$

Since

$$\mathbb{E}_{X\sim\mathrm{Ga}(a,b)} \exp(-\lambda X) = \frac{1}{(1+\lambda b)^a}, \tag{3.12}$$

we have that

$$\begin{aligned} \mathbb{E}\exp\left(-\lambda \textstyle\sum_{k=1}^{l} X_k\right) &= \prod_{k=1}^{l} \mathbb{E}\exp(-\lambda X_k) \\ &= \prod_{k=1}^{l} \left(\frac{1}{(1+\lambda k)^{a_k}}\right) \\ &= \exp\left(\sum_{k=1}^{l} (-a_k)\log(1+\lambda k)\right) \end{aligned}$$

Combining (3.11),(3.12) and using the fact that the map $x \mapsto \frac{log(1+x)}{x}$ is non-increasing we can see

$$\begin{aligned} \mathbb{P}\left(\textstyle\sum_{k=1}^{l} X_k < t\right) &\leq \exp\left(\lambda t - \sum_{k=1}^{l} \lambda k a_k \frac{\log(1+\lambda l)}{\lambda l}\right) \\ &= \exp\left(\lambda t - \frac{s}{l}\log(1+\lambda l)\right). \end{aligned}$$

Taking $\lambda = (s/t - 1)/l > 0$ yields

$$\mathbb{P}\left(\sum_{k=1}^{l} X_k < t\right) \leq \exp\left(-\frac{s\log\left(\frac{s}{et}\right)}{l}\right).$$

Finally, since $l^2/2 \leq s$ we have that

$$\exp\left(-\frac{s\log\left(\frac{s}{et}\right)}{l}\right) \leq \exp\left(-\sqrt{\frac{s}{2}}\log\left(\frac{s}{et}\right)\right)$$

which concludes the proof. $\square$

Another result that must be stated is Hardy-Ramanujan formula on the number of partitions of an integer:

Lemma 3.2.4 (Erdős (1942))**.**

$$\left|\left\{(a_1, ..., a_l) \in \mathbb{N}^l : l \in \mathbb{N}, \sum_{k=1}^{l} k a_k = s\right\}\right| \leq \exp\left(\pi\sqrt{\frac{2}{3}s}\right).$$

Proof of the Theorem 3.2.2. Let $T = UA(n)$, and let $S \geq 1$ be a value chosen later. This proof is decomposed into 5 steps:

Step 1. We introduce a different labeling of the vertices of UA(n): Let $(a_1, ..., a_l) \in \mathbb{N}^l$. A vertex v in $V(\text{UA}(n))$ is labeled as $(a_1, ..., a_l)$ if, and only if, v is the a_l^{th} descendant (in birth order) of the node $(a_1, ..., a_{l-1})$. This gives a labeling for the elements of UA(n) built with the elements of $\mathbb{N}^* = \cup_{l=0}^{\infty}\mathbb{N}^l$. Now for $i \in \mathbb{N}$, consider the mapping L that gives, for $v \in \mathbb{N}^*$, the depth of this vertex. In other words, $L(v) = k$ if, and only if, $v \in \mathbb{N}^k$. Also, consider the mapping $s(v) = \sum_{k=1}^{L(v)}(L(v) + 1 - k)a_k$, for $v \in \mathbb{N}^*$. For $v = 0$, we define $s(v) = 0$.

First, we prove the following important property for $s(v)$:

Lemma 3.2.5. *If v is such that $s(v) > 3S$, then either*

(i) there exists u such that $s(u) \in (S, 3S]$ and $v \in T_u^0$, or

(ii) there exists u such that $s(u) \leq S$ and $v \in T_{(u,j)}^0$ for some $j > S$.

Proof. We prove this lemma by induction on the depth of v, i.e., on $L(v)$. For $L(v) = 1$, we have that taking $u = 0$, then $s(u) = 0$ and therefore (ii) is true. Assume that $L(v) > 1$ and take $u = parent(v)$. There are three cases on $s(u)$:

1. $s(u) > 3S$: In this case, we can apply the induction hypothesis on u since $L(u) = L(v) - 1$.

2. $S < s(u) \leq 3S$: In this case, condition (i) is true.

3. $s(u) \leq S$: Let $h = L(v)$. Therefore, $u = (a_1, ..., a_{h-1})$. Since

$$s(u) = \sum_{k=1}^{h-1}(h-k)a_k, \quad \text{and}$$

$$\begin{aligned} s(v) &= \sum_{k=1}^{h}(h+1-k)a_k \\ &= \sum_{k=1}^{h-1}(h-k)a_k + \sum_{k=1}^{h-1} a_k + a_h, \end{aligned}$$

we have that

$$\begin{aligned} s(v) &= \sum_{k=1}^{h} a_k + s(u) \\ &< a_h + \sum_{k=1}^{h-1}(h-k)a_k + s(u) \\ &\leq 2s(u) + a_h. \end{aligned}$$

Combining this last inequality with the fact that $s(u) \leq S$ and $s(v) \geq 3S$, we see that $a_h > s(v) - 2s(u) \geq S$. Thus, $v = (u, j)$, for $j = a_k > S$.

□

Step 2. From Lemma 3.2.2, we know that

$$\phi(w) \leq \max(\phi(v), \phi(u)),$$

and using this and Step 1, we see

$$\begin{aligned} \mathbb{P}(\exists\ v : s(v) > 3S \text{ and } \phi(v) \leq \phi(0)) = {} & \mathbb{P}(\exists\ u : s(u) \in (S, 3S] \text{ and } \phi(u) \leq \phi(0)) \quad (3.13) \\ & + \mathbb{P}(\exists\ u, j : s(u) \leq S \text{ and } \phi((u,j)) \leq \phi(0)) \quad (3.14) \end{aligned}$$

Step 3. On the other hand, observe that for $v = (a_1, a_2, ..., a_l)$, one has

$$\begin{aligned} \phi(v) \leq \phi(0) \quad &\iff \quad \prod_{u \in V(T)\setminus\{v\}} T_u^v \leq \prod_{u \in V(T)\setminus\{0\}} T_u^0 \\ &\iff \quad \left(\prod_{u\in P} T_u^v\right)\left(\prod_{u\in C} T_u^v\right)(T_0^v) \leq \left(\prod_{u\in P} T_u^0\right)\left(\prod_{u\in C} T_u^0\right)(T_v^0), \quad (3.15) \end{aligned}$$

where $P = 0Rv \setminus \{0, v\}$ and $C = P^c \setminus \{v, 0\}$, and since $T_u^0 = T_u^v$ for all $u \in C$, we have that $\phi(v) \leq \phi(0)$ is equivalent to

$$\left(\prod_{u\in P} T_u^v\right)(T_0^v) \leq \left(\prod_{u\in P} T_u^0\right)(T_v^0),$$

and we rewrite as

$$\prod_{i=0}^{L(v)-1} T^v_{(a_0,\ldots,a_i)} \leq \prod_{i=1}^{L(v)} T^0_{(a_1,\ldots,a_i)}.$$

This is equivalent to

$$\prod_{i=1}^{L(v)} \left(n - T^0_{(a_1,\ldots,a_i)}\right) \leq \prod_{i=1}^{L(v)} T^0_{(a_1,\ldots,a_i)} \quad (3.16)$$

This shows

$$\phi((v,j)) \leq \phi(0), \text{ for some } j > S$$
$$\Rightarrow \prod_{i=1}^{L(v)} \left(n - T^0_{(a_1,\dots,a_i)}\right) \left(n - \sum_{j=S+1}^{\infty} T^0_{(v,j)}\right) \leq \left(\prod_{i=1}^{L(v)} T^0_{(a_1,\dots,a_i)}\right) \left(\sum_{j=S+1}^{\infty} T^0_{(v,j)}\right).$$

Since the random variables

$$\sum_{j=S+1}^{\infty} T^0_{(v,j)}$$

are stochastically dominated by $T^0_{(v,S)}$, we have that

$$\mathbb{P}(\exists j > S : \phi((v,j)) \leq \phi(0)) \leq \mathbb{P}((v,S) \leq \phi(0)).$$

Thus, the term in (3.14) is bounded by the right part of (3.13). Coming back to (3.13) and using a union bound, we conclude that

$$\mathbb{P}(\exists\ v : s(v) > 3S \text{ and } \phi(v) \leq \phi(0)) \leq 2 \sum_{v:s(v)\in(S,3S]} \mathbb{P}(\phi(v) \leq \phi(0)). \tag{3.17}$$

Step 4. Here we prove that for any vertex v with $s(v) \geq 10^{10}$, it is true that

$$\limsup_{n\to+\infty} \mathbb{P}(\phi(v) \leq \phi(0)) \leq 7\exp\left(-0.21\sqrt{s(v)}\log(s(v))\right). \tag{3.18}$$

Note that the random vector

$$\left(\frac{1}{n}\left|T^0_{(a_1,a_2,\dots,a_i)}\right|\right)_{i=1,\dots,l(v)}$$

evolves according to a random vector of standard Pólya urns.

Therefore, we have that

$$\left(\frac{1}{n}\left|T^0_{(a_1,a_2,\dots,a_i)}\right|\right)_{i=1,\dots,L(v)}$$

converges in distribution to the random vector $\left(\prod_{k=1}^{i} U_{a_k,k}\right)$, where $U_{a_1,1}, U_{a_2,2},\dots,$ are independent products of a_k independent uniform random variables in $[0,1]$. Therefore, by (3.15) and (3.16), we have

$$\begin{aligned}
\limsup_{n\to+\infty} \mathbb{P}(\phi(v) \leq \phi(0)) &= \limsup_{n\to+\infty} \mathbb{P}\left(\prod_{i=1}^{L(v)} \frac{\left|T^0_{(a_1,\dots,a_i)}\right|}{n} \geq \prod_{i=1}^{L(v)} 1 - \frac{\left|T^0_{(a_1,\dots,a_i)}\right|}{n}\right) \\
&= \mathbb{P}\left(\prod_{i=1}^{L(v)} \prod_{k=1}^{i} U_{a_k,k} \geq \prod_{i=1}^{L(v)} \left(1 - \prod_{k=1}^{i} U_{a_k,k}\right)\right) \\
&= \mathbb{P}\left(\prod_{i=1}^{L(v)} U_{j_i,i}^{L(v)+1-i} \geq \prod_{i=1}^{L(v)} \left(1 - \prod_{k=1}^{i} U_{a_k,k}\right)\right) \\
&\leq \mathbb{P}\left(\prod_{i=1}^{L(v)} U_{j_i,i}^{L(v)+1-i} \geq \exp\left(-\frac{s(v)}{R}\right)\right) \\
&\quad + \mathbb{P}\left(\prod_{i=1}^{L(v)} \left(1 - \prod_{k=1}^{i} U_{a_k,k}\right) \leq \exp\left(-\frac{s(v)}{R}\right)\right)
\end{aligned}$$

when $R > 0$ will be chosen later.

Using Lemma 3.2.3 one obtain

$$\begin{aligned}\mathbb{P}\left(\prod_{i=1}^{L(v)} U_{j_i,i}^{L(v)+1-i} \geq \exp\left(-\frac{s(v)}{R}\right)\right) &= \mathbb{P}\left(\sum_{i=1}^{L(v)} -\log U_{j_i,i}^{L(v)+1-i} \leq \frac{s(v)}{R}\right) \\ &\leq \exp\left(-\sqrt{\frac{s(v)}{2}}\log\left(\frac{R}{e}\right)\right).\end{aligned}$$

On the other hand, since $1-\exp(-x) \geq \frac{1}{2}\min(x,1)$ for any $x \geq 0$, we see that for each $i \in \{1, ..., l(v)\}$

$$\begin{aligned}1-\prod_{k=1}^{i} U_{a_k,k} &\geq \frac{1}{2}\min\left(-\log\left(\prod_{k=1}^{i} U_{a_k,k}\right),1\right) \\ &= \frac{1}{2}\min\left(\sum_{k=1}^{i}\log\left(U_{a_k,k}^{-1}\right),1\right).\end{aligned}$$

Therefore,

$$\prod_{i=1}^{L(v)}\left(1-\prod_{k=1}^{i} U_{a_k,k}\right) \geq \frac{1}{2^{l(v)}}\prod_{i=1}^{L(v)}\min\left(\sum_{k=1}^{i}\log\left(U_{a_k,k}^{-1}\right),1\right),$$

and if X is equal in distribution to

$$\min\left(\sum_{k=1}^{i}\log\left(U_{a_k,k}^{-1}\right),1\right),$$

by the previous inequality we have

$$\prod_{i=1}^{L(v)}\left(1-\prod_{k=1}^{i} U_{a_k,k}\right) \geq \frac{1}{2^{l(v)}}X.$$

By Lemma B.0.2 (see Appendix B) and the last enquality one gets,

$$\begin{aligned}\mathbb{P}\left(\prod_{i=1}^{L(v)}\left(1-\prod_{k=1}^{i} U_{a_k,k}\right) \leq \exp\left(-\frac{s(v)}{R}\right)\right) &= \mathbb{P}\left(\frac{1}{2^{L(v)}}X \leq \exp\left(-\frac{s(v)}{R}\right)\right) \\ &\leq 6 \cdot 2^{\frac{L(v)}{4}}\exp\left(-\frac{s(v)}{4R}\right) \\ &\leq 6\exp\left(\frac{\sqrt{s(v)}}{4}-\frac{s(v)}{4R}\right),\end{aligned}$$

where the last inequality comes from

$$\begin{aligned} 2^{\frac{L(v)}{4}} &\leq \exp\left(\frac{\sqrt{s(v)}}{4}\right) \\ \iff \log 2\,\frac{L(v)}{4} &\leq \frac{\sqrt{s(v)}}{4} \\ \iff \frac{L(v)}{2} &\leq s(v).\end{aligned}$$

Taking $R = e \cdot s(v)^{0.3}$ we obtain

$$\begin{aligned}\limsup_{n\to+\infty} \mathbb{P}(\phi(v) \leq \phi(0)) &\leq \exp\left(-\sqrt{\tfrac{s(v)}{2}}\log\left(-s(v)^{0.3}\right)\right) + \exp\left(\tfrac{\sqrt{s(v)}}{4} - \tfrac{s(v)}{4e\cdot s(v)^{0.3}}\right) \\ &\leq 7\exp(-0.21\sqrt{s(v)}\log(s(v)))\end{aligned}.$$

Step 5. Combining (3.17), (3.18) and the fact $S \geq 10^{10}$, we have

$$\limsup_{n\to+\infty} \mathbb{P}(\exists v : s(v) > 3S \wedge \phi(v) \leq \phi(0)) \leq 14 \cdot |\{v : s(v) \leq 3S\}| \cdot \exp\left(-0.21\sqrt{S}\log(S)\right).$$

Using the Hardy-Ramanujan formula we see

$$|\{v : s(v) \leq 3S\}| \leq 3S \exp \pi\sqrt{2S},$$

and finally, for $S \geq 10^{10}$, one can check that

$$\limsup_{n\to+\infty} \mathbb{P}(\exists v : s(v) > 3S \wedge \phi(v) \leq \phi(0)) \leq \exp\left(-\frac{1}{100}\sqrt{S}\log(S)\right),$$

which concludes the proof.

□

3.3 Simple root-finding algorithm for PA(n)

Finally, we close this chapter by proving a lower bound for the simple root-finding algorithm in the Barabási-Albert model.

Once again, let H_ψ the mapping that returns the set of K vertices with smallest ψ_T values (ties are broken arbitrarily).

Theorem 3.3.1. *Let $\varepsilon > 0$. In the preferential attachment model, there exists K such that*

$$\liminf_{n\to+\infty} \mathbb{P}(0 \in H_\psi(PA(n))) \geq 1 - \varepsilon.$$

Proof. From now on, we use $deg(i)$ to make reference to $deg_{\mathrm{PA}(K)}(i)$ for any $0 \leq i \leq n$. We keep using the same notation that we defined for Theorem 3.1.1. In this case, the vector

$$2(T_{0,K-1}, T_{1,K-1}, ..., T_{K-1,K-1})$$

follows a Pólya urn with K colors and replacement matrix $2I_K$ where I_K is the identity matrix of size K, and thus, using the classical result in the appendix (Theorem A.0.1), we have that conditionally on the first $K-1$ steps of PA$(K-1)$,

$$\frac{1}{n}\left(|T_{0,K-1}|, |T_{1,K-1}|, ..., |T_{K-1,K-1}|\right) \underset{n\to+\infty}{\longrightarrow} \mathrm{Dir}\left(\frac{deg(0)}{2}, ..., \frac{deg(K-1)}{2}\right).$$

Let $\eta \in (0,1)$. We know that

$$\mathbb{P}(0 \notin H_\psi) \leq \mathbb{P}(\exists i \geq K : \psi(i) \leq \psi(0))$$

and then, since,

$$\mathbb{P}(\exists i \geq K : \psi(i) \leq \psi(0)) \leq \mathbb{P}(\psi(0) \geq (1-\eta)n) + \mathbb{P}(\exists i \geq K : \psi(i) \leq (1-\eta)n).$$

we have

$$\mathbb{P}(0 \notin H_\psi) \leq \mathbb{P}(\psi(0) \geq (1-\eta)n) + \mathbb{P}(\exists i \geq K : \psi(i) \leq (1-\eta)n). \tag{3.19}$$

Using that $\psi(0) \leq \max(|T_{0,1}|, |T_{1,1}|)$, and the fact that $|T_{0,1}|/n$ and $|T_{1,1}|/n$ are identically distributed and converge in distribution to a $\text{Beta}(1/2, 1/2)$, we can see

$$\begin{aligned}\limsup_{n\to+\infty} \mathbb{P}(\psi(0) \geq (1-\eta)n) &\leq 2 \lim_{n\to+\infty} \mathbb{P}(|T_{0,1}| \geq (1-\eta)n) \\ &= \frac{2}{\pi}\arcsin(\sqrt{\eta}) \\ &\leq \sqrt{\eta}. \end{aligned}\tag{3.20}$$

On the other hand, by Lemma 3.1.1(ii) we know that, for any $i \geq K$,

$$\psi(i) \geq \min_{1\leq k\leq K-1} \sum_{j=0, j\neq k}^{K-1} |T_{j,K-1}|$$

and also, that

$$\frac{1}{n}\sum_{j=0,j\neq k}^{K-1} |T_{j,K-1}|$$

is stochastically lower bounded by $\frac{1}{n}\sum_{j=0}^{K-1} |T_{j,K-1}|$ which converges in distribution to a

$$\text{Beta}\left(K - \frac{deg(0)}{2}, \frac{deg(0)}{2}\right).$$

Therefore,

$$\begin{aligned}\limsup_{n\to+\infty} \mathbb{P}\left(\exists i \geq K : \psi(i) \leq (1-\eta)n\right) &\leq \limsup_{n\to+\infty} \mathbb{P}\left(\exists 0 \leq k < K : \sum_{j=0,j\neq k}^{K-1} |T_{j,K-1}| \leq (1-\eta)n\right) \\ &\leq \limsup_{n\to+\infty} \mathbb{P}\left(\exists 0 \leq k < K : \frac{1}{n}\sum_{j=0}^{K-1} |T_{j,K-1}| \leq 1-\eta\right) \\ &\leq K\,\mathbb{P}\left(\text{Beta}\left(K - \frac{deg(0)}{2}, \frac{deg(0)}{2}\right) \leq 1-\eta\right)\end{aligned}\tag{3.21}$$

Finally, from Theorem C.0.1 in the appendix, we know that there exists $K = K(\eta)$ such that

$$\mathbb{P}\left(\text{Beta}\left(K - \frac{deg(0)}{2}, \frac{deg(0)}{2}\right) \leq 1-\eta\right) \leq \eta. \tag{3.22}$$

Combining (3.19), (3.20), (3.21) and (3.22) and taking a proper value of η , we have

$$\limsup_{n\to+\infty} \mathbb{P}(0 \notin H_\psi) \leq \varepsilon,$$

which concludes the proof. $\square$

Last but not least, we have to mention that the maximum likelihood estimator in the preferential attachment model is obtained by minimizing the function,

$$\xi_T(u) = \frac{\overline{\mathbf{Aut}}(u,T)}{d_T(u)} \prod_{v \in V(T) \setminus \mathcal{L}(T^u)} (|T_v^u| \cdot \mathbf{Aut}(v,(T,u))),$$

by a similar approach.

In the following chapter we will show by simulations how well these and some other algorithms perform in regards to the root-finding algorithm property.

Chapter 4

Simulation Results

4.1 Some other estimators

In this section we present three estimators that together with Simple, Distance and SZ-estimator will be our main tools for trying to solve the rumor source detection problem.

4.1.1 Degree Estimator

We begin by presenting one that is based on a well-studied graph measure: degree centrality.

Let T be a tree and consider $\sigma_T : V(T) \to \mathbb{N}$, defined by

$$\sigma_T(v) = deg(v).$$

The value $\sigma_T(v)$ is known as the *degree centrality* of v.

The *degree estimator* receives a tree T and outputs a vertex v such that

$$v \in \arg\max_{u \in V(T)} \sigma_T(u)$$

chosen uniformly at random. Such v is called *degree center* of the graph T.

4.1.2 Pagerank Estimator

The estimator exposed here uses the well-known PageRank algorithm to output a vertex v. At a first glance, what PageRank algorithm does is based on the following idea of ranking. Given an directed graph G, define, for each u in $V(G)$,

$$F_u = \{w : (u, w) \in E(G)\},$$
$$B_u = \{w : (w, u) \in E(G)\},$$

and

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{|F_v|}, \tag{4.1}$$

where c is a normalization constant. The value $R(u)$ is known as the *rank* of u. The equation (4.1) is recursive but it could be computed by starting with any set of ranks and iterating the computaton until it converges. This last part is not entirely true: in the case where the graph has a cycle over two vertices u and v in G for which there is only one incoming edge either to u or v, this version gives a diverging increasing sequence of ranks for both u and v. This situation is known as a *rank sink*. PageRank is a modified version of the previous definition of rank for which rank sinks are overcomed:

Definition 4.1.1. *Let $E : V(G) \to \mathbb{R}$. Then, the PageRank of a set $V(G)$ is a function, $R' : V(G) \to \mathbb{R}$, that satisfies*

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{|F_v|} + cE(u), \text{ for all } u \in V(G), \tag{4.2}$$

such that c is maximized and $||R'|| = 1$, where $||R'||$ denotes the L_1 norm of R'.

Since PageRank algorithm was originally developed to sort Websites, the function $E(u)$ was usually defined as the probability that a random surfer finds the site u.

For a detailed explanation of PageRank algorithm see [16].

4.1.3 Leaves Removal Estimator

This section contains the basic information about an estimator that is not as well-known as the previous two. This estimator was suggested to us by Louigi Addario-Berry (personal communication). We call it the *Leaves Removal Estimator.*

Let T be a tree and consider the following algorithm:

(i) Compute $\mathcal{L}(T)$ the set of leaves of T.

(ii) Choose $v \in \mathcal{L}(T)$ uniformly at random.

(iii) Construct a new graph T^* by removing v and its incident edges from T. Formally, T^* is the graph induced by $V(T) \setminus \{v\}$.

(iv) If $|V(T^*)| > 1$, then repeat (i) but this time with T^* instead of T.

Clearly, the resulting graph after executing for the first time the steps $(i), (ii)$ and (iii) is a tree of size $|T| - 1$. The previous algorithm stops when $|V(T^*)| = 1$, i.e. when the resulting graph consists only of one vertex v. The leaves removal estimator uses this algorithm and in the last step it returns such v.

4.2 Set-up

In this section we present the basic framework for testing the estimators.

The main programming tools used here to do such tests and visualize the results are the following ones:

- **Python**: a interpreted, object-oriented programming language that emphasizes code readability. `https://www.python.org/`
- **Jupyter Notebooks App**: a program that produces documents which contains both computer code and rich text elements. `http://jupyter.org/`
- **NetworkX**: a python library for studying graphs and networks. `https://networkx.github.io/`
- **Matplotlib**: a python library which produces publication quality figures. `https://matplotlib.org/`
- **Numpy**: a python package that provides array data structures tools. `http://www.numpy.org/`

- **Plotly**: a python package that provides online graphing, analytics and statistics tools. `https://plot.ly/`

We compare such algorithms in three different scenarios:

- **Explicitly finding the source:** In this section we test each algorithm to see how well (according to quality measures) they are trying to output the source.
- **Confidence sets:** In this section we ask for the algorithms to return some sets such that they contain the root with high probability.
- **Tournament test:** This test consists of running the estimators over the same particular graph and giving awards for those who return the exact value of the root.

For each of the previous scenarios we manage the simulations on three kinds of random-growing graphs (see Section 1.2):

- **Pseudo-3-regular Trees.**
- **Uniform Attachment Trees.**
- **Preferential Attachment Trees (Barabási-Albert Model).**

All the graphs generated according to these three models are labeled trees. The labels assigned to each vertex correspond to its apperance order. This means, the root or first vertex of the graph has the label 0 and in general, a vertex receives the label i if, and only if, it was added to the graph right after the vertex $i-1$.

All the functions used in this section are well-described in a GitHub's repository: `https://github.com/diegoabt/master-research` .

4.3 First Test: Finding the source

In this section, a simulation consists in generating a tree of a predefined size under one of the growing models mentioned before. We use **networkx** tools to create such trees.

The task, as we said before, is to ask for each estimator on each simulation to output its estimate of the source.

For this and the remaining tests, we analyze the behaviour of the estimators by separating them into groups. The first group consists of the *fastest* ones (**Leaves Removal**, **Degree** and **PageRank** estimators) and the second group of the *slowest* ones (**Simple**, **Distance** and **SZ-estimator**). This section will show us why they deserve these labels.

Let T be a random tree generated using one of the known models. When asking to output a single vertex, the estimators based Shah-Zaman, Simple, Degree, Distance and PageRank algorithms compute $val(u)$ for all $u \in T$ (being $val(u)$ the value associated to each one of the mentioned estimators), store these values in a list and pick the vertex v that has the biggest value in the list. They break ties by choosing one of the involved vertices at random. Clearly, Leaves Removal estimator does not need to do this since it does not assign a value to each vertex.

For the fastest ones, we did 500 simulations in UA(N), PA(N) and P3R(N) graphs for $N \in \{100, 200, 300, 400, 500, 1000, 1500, 2000, 2500, 5000\}$. Figures 4.1 (p.64)-4.9 (p.66) show the simple and cumulative histograms of the results for each one of these estimators for $N = 5000$.

For the slowest ones, we did 50 simulations in UA(N), PA(N) and P3R(N) graphs for $N \in \{50, 75, 100\}$. Figures 4.10-4.18 (pp.66-70) show the simple and cummulative histograms of the results for each one of these estimators for $N = 100$.

Leaves Removal estimator found the source more than 80 times for PA graphs in 500 attempts. Its performance in reference to successfully outputting the source was better in this sense, for this graph model than for UA and P3R models. On the other hand, from the 5000 possible outcomes, it picked the 400 first vertices, at least once, for this kind of graph. It had a smaller range of candidates for the source in the case when the random graph followed the P3R model.

PageRank estimator had its best performance in relation to candidate's range size and amount of successful outputs for the PA model. For the UA model, it only found the source around 30 times. For the P3R model, the estimator output almost all the possible 5000 vertices at least once. The real source was not even close to being one of the most returned vertices.

With regards to candidate's range size and amount of successful outputs, Degree estimator did its best execution for the PA model. Once again, the P3R graphs represent a hard case for this estimator.

Figures 4.19-4.21 (pp. 70-71) show time performance for each one of the fastest algorithms in the 500 simulations for each of the graphs sizes mentioned at the begining of this section. They are shown in groups separated by the random growing graph model.

Leaves Removal Estimator is considerably slower than the other two estimators for the biggest graphs sizes. It needed around 6 thousand seconds to do the 500 simulations, while the PageRank and the Degree estimators did the job is less than one thousand seconds. Note that PageRank and Degree centrality have highly optimized implementations in Python, whereas Leaves Removal does not.

On the other hand, for the slowest ones we can see that they had a similar behaviour: they successfully output the root between 10 and 20 times for each graph model, they showed a bigger range of wrong outputs for the P3R model, the first three vertices that appeared in the UA and PA graph were the most returned ones (the source and vertices 1 and 2).

For our 50 attempts, the Simple Estimator successfully output the source around 50% of times for the PA model. This was the best performance of the three slowest estimators in terms of finding the source.

Simple and SZ-estimator needed at least more than 6 thousand seconds to do 50 simulations on a graph of size 100 meanwhile the fastest ones could solve the task for graphs of size 5000 in at most the same amount of time.

The fastest algorithm of this group is the Distance Estimator, but despite the results shown in Figures 4.19-4.21 (pp. 70-71), it still is considerably slower than Degree, PageRank and Leaves Removal estimators.

The previous evaluation suggests that

- In regards to successfully outputting the source for graphs that followed the P3R model, Leaves Removal is significantly better than the other two fastest estimators.

- The output set of possible candidates for the source that is returned by PageRank and Degree estimator is around 1% of the total possible choices for the PA model.

- Leaves Removal estimator has a more consistent performance when looking at the three possible graphs scenarios. In the three situations it successfully found the source between 60 and 90 times of the 500 attempts, and the real source was the most outputed vertex.

- For the three graph models, PageRank needs more time to estimate the source in the cases where the graphs come from the UA and PA models. The fastest time performance of the P3R model does not compensate its poor finding-source execution.

- The slowest three have a better rate of success than the fastest ones.

4.4 Second Test: Confidence sets

In this test we consider a variation of the previous procedure. We are no longer interested in single source finding methods but in trying to analyze the size of some output sets. Each algorithm implicitly ranks the vertices from 0 to n-1. In this case we analyse the rank attriuted to the true root by each procedure. The smallest the rank the better the algorithm.

Before presenting the results for the different types of random graphs, we are going to give a idea of how each estimator works for this particular task.

- **PageRank** (also for **Degree** and **Shah-Zaman**):
  nx.pagerank(G) (also **nx.degree_ centrality(G)** and **sz(G)**) gives a list **L** with the pagerank value (degree and SZ-centrality) of each vertex on **G** so to compute the confidence set we just sort the list on increasing-order, look for the label 0 in the sorted list and choose all the vertices with higher or equal pagerank values (degree and SZ-centrality) than the source to be the output set.

- **Distance** (also for **Simple**): this is similar to the previous case except for the fact that we pick the vertices with lower distance centrality (anticentrality). To get the distance centralities of a set of vertices of a graph **G** we simply use the built-in function **nx.closeness_ centrality(G)** (and the own-created function **simple(G)**).

- **Leaves Removal**: to obtain a possible confidence set, we execute the algorithm until it explicitly finds the 0 labeled vertex and then we simply output the remaining vertices of the graph.

As we explained before for test 1, it is possible to have multiple vertices with the same value; estimators break this kind of ties at random.

It is worth noting that so far we have only proven that Simple and SZ-estimator are the ones that behave as root-finding algorithms, for some particular random growing trees. The core of this section is to look for root-finding behavior for the remaining estimators. Once again, a root-finding procedure is a method that given an ε and a graph T, it returns a set f vertices, subset of $V(T)$ (independent of the size of T) such that the probability that 0 belongs to this set is at least 1-ε. We aim to see this kind of independence in the set returned by the studied algorithms. The output set itself is not relevant, it is the size of such set which we want to see.

Figures 4.25-4.33 (pp. 72-74) show the results for the fastest estimators. We did 500 simulations in UA(N), PA(N) and P3R(N) graphs for $N \in \{100, 200, 300, 500, 1000, 1500, 2000\}$. Each figure contains the results for a particular graph model. Figures 4.34-4.36 (pp. 74-75) show the results for the slowest ones. We did 50 simulations in UA(N), PA(N) and P3R(N) graphs for $N \in \{50, 75, 100\}$. We grouped the results by graph model and not by estimator. The column named *Probability* denotes the probability of success, for different values of ε.

Figures 4.28-4.33 (pp. 73-74) show that in most cases, the graph size returned by PageRank and Degree estimators increases for fixed values of ε as the size of the graph grows.

For the Leaves Removal estimator the situation is similar for the case where the graph followed the PA model (See Figure 4.26 (p. 72). For the other two graphs models (Figures 4.25 p. 72 and 4.27 p. 72), we can appreciate that the output size graph has a more stable behavoir, especially for the case of the P3R model.

For the slowest ones, even when we know that two of them are root-finding algorithms, it is difficult to detect some pattern for a particular graph model.

In summary, the results suggest that it might be the case that Leaves Removal estimator behaves like a root-finding algorithm for the UA and P3R cases. On the other hand, it also seems that PageRank and Degree estimator are far from having that property. Nothing relevant can be said for the slowest ones.

4.5 Third Test: The tournament

This test is a variation of the first test. We decided to measure the quality of the estimators by generating a random graph G under some previous defined model and asking for each one of them to look for the source vertex in this particular graph. Each time that an estimator succesefully found the root, we counted this as a victory and we rewarded it with a point. When a tie ocurred, each one the involved estimators received a point. We tested the 6 estimators in UA(N), PA(N) and P3R(N) for small values of N and the fastest ones for the some other bigger values. We exhibit the results in pie charts (See Figures 4.37-4.42 pp. 75-80). Blue tones correspond to the fastest group and green tones to the slowest one. Once again. we did 50 simulations in the case where the slowest ones were involved and 500 in the case where we were only testing the fastest ones.

We can see that for PA(N) with $N = 25$ there is a regular behaviour for 5 of the 6 estimators (Figure 4.37a p. 75). The least amount of victories in this case returned by the Leaves Removal Estimator was considerably lower than for the other algorithms. Figures 4.37b (p. 75) and 4.37c (p. 75) show a better performance for the slowest ones compared to the fastest ones. For PA(N) with $N = 75$, the slowest estimators had almost 60% of the total amount of victories (as we said before ties were counted repeatedly). In these three scenarios, Leaves Removal estimator was the one with least successfull finds.

The situation for the UA model was not so different compared to the previous case: best performance was once again, by the slowest ones. Actually, they got almost the same number of victories than for PA graphs. Leaves Removal estimator provided better numbers in this cases than for the PA model.

For the third kind of graph, the one generated by the P3R model, we still can see a better performance of the slowest ones. On the other hand, this task started to be much harder to solve for the fastest ones. Although Leaves Removal had a more satisfactory performance, results for PageRank and Degree were definitely less accurate than for the other two graphs models. For the 3 graph models, PageRank only got a victory out of 150 attempts.

The rest of the figures shows the results of tournament test but only for the fastest ones.

There is a similar amount of victories for PageRank and Degree estimators on each of the 4 graph sizes for the PA model. Leaves Removal offered the worst results (see Figures 4.40 p.78) .

On the other hand, the number fo successfull finds was reduced for each estimator in the UA model (See Figure 4.41 p.79). In this case, Degree Estimator had a considerably better performance compared to PageRank. In fact, PageRank was not only worse than the Degree Estimator but also than the Leaves Removal.

Finally, for the P3R model (Figure 4.42 p.80), there is a near tie for 2 of the 3 estimators as was remarked before. PageRank and Degree algorithms offered poor results compared to the ones exhibited for the previous two scenarios. There was not any victory for Pagerank and only 14 out of 2000 attempts (500 simulations for any of the 4 graph sizes) for the estimator based on degree centrality.

Since the previous results showed similar performances for the slowest algorithms, we have decided to watch closer the work done by them in regards to the output vertices. Figures 4.43-4.44 (pp. 81-82) contain the vertices returned not only for the slowest estimators but also for the fastest ones in the first 25 simulations for PA(75), UA(75) and P3R(75).

The first thing that must be observed is that for each one of the graph models, the Simple, Distance and SZ-estimator output the same vetices on each iteration.

Another fact that must be remarked is that even when the slowest estimators did not have a victory, their estimate of the source was not far away from the real value. Some behavior like this was observed in the results of the Test 2. On the other hand, for PageRank and Degree estimators, results are not so satisfactory in these terms: some of the estimates provided by them were not only wrong but very far from the rumor source.

Note that the performance of the Leaves Removal Estimator was better for BA than for the other two kind of graphs. This might come from the fact that for this kind of graphs the first nodes that apper have a bigger probability of having more neighbors, therefore, it takes some time to the algorithm to find them and remove them (leaves are removed first). The other two have a more regular structure in terms of degree.

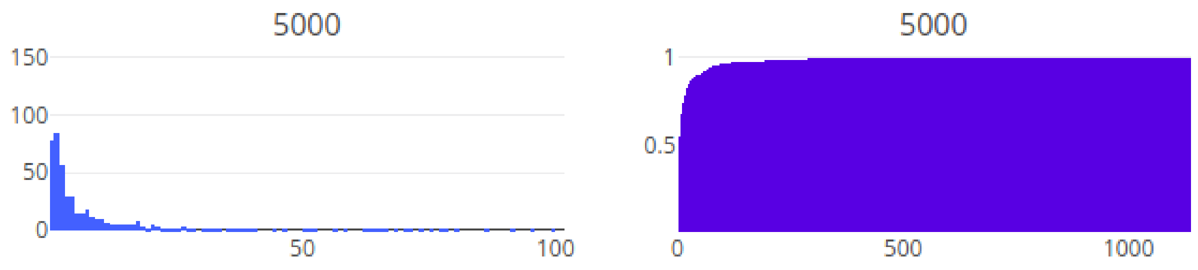


Figure 4.1: Leaves Removal in BA(5000): histogram for output vertices (left) and its cummulative version (right).

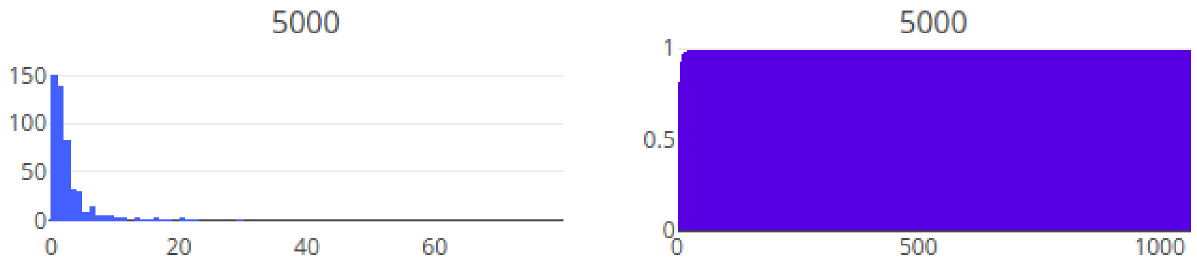


Figure 4.2: Pagerank in BA(5000): histogram for output vertices (left) and its cummulative version (right).

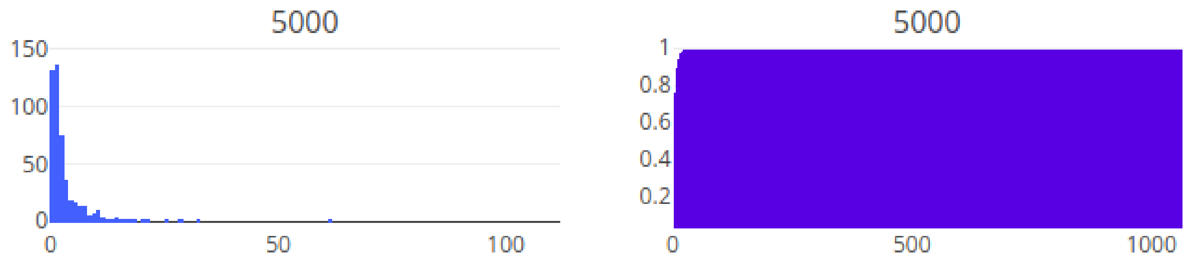


Figure 4.3: Degree in BA(5000): histogram for output vertices (left) and its cummulative version (right).

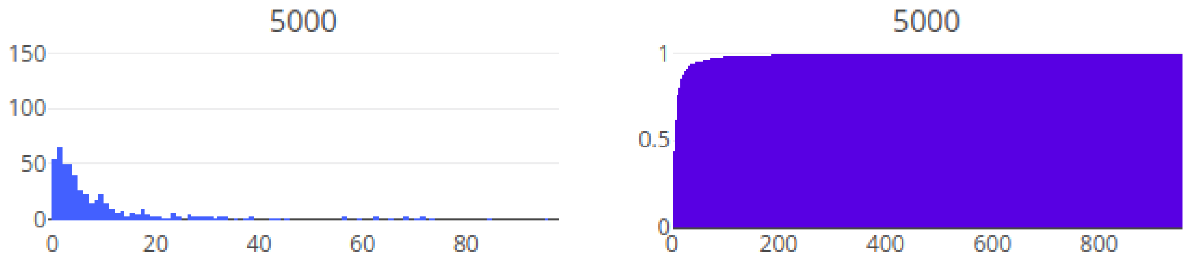


Figure 4.4: Leaves Removal in UA(5000): histogram for output vertices (left) and its cummulative version (right).

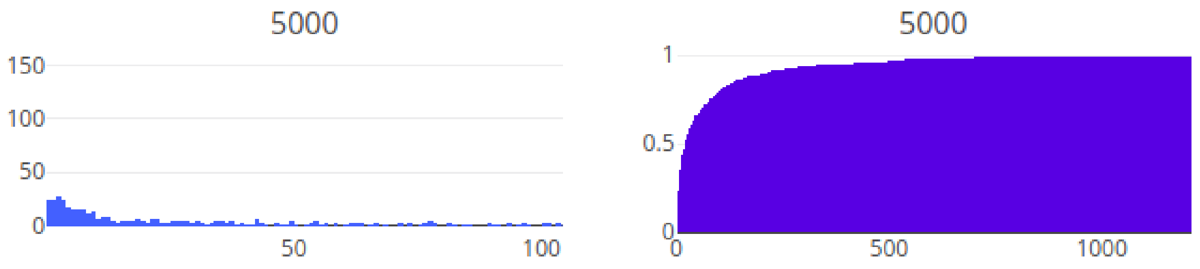


Figure 4.5: Pagerank in UA(5000): histogram for output vertices (left) and its cummulative version (right).

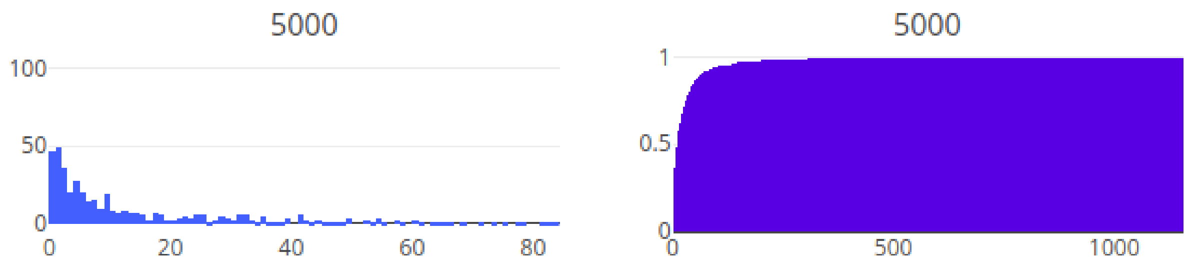


Figure 4.6: Degree in UA(5000): histogram for output vertices (left) and its cummulative version (right).

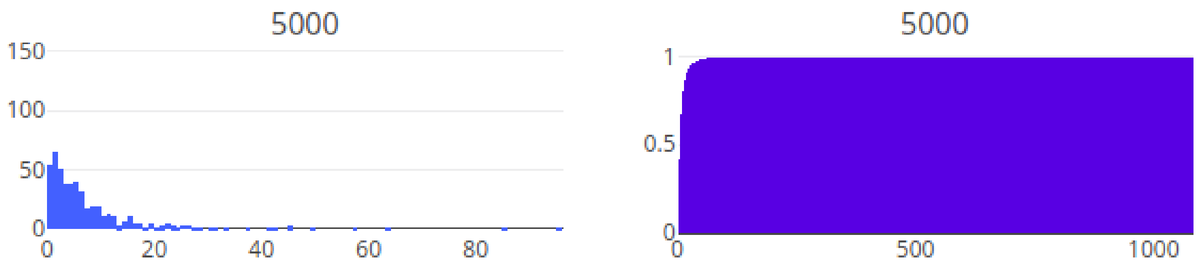


Figure 4.7: Leaves Removal in P3R(5000): histogram for output vertices (left) and its cummulative version (right).

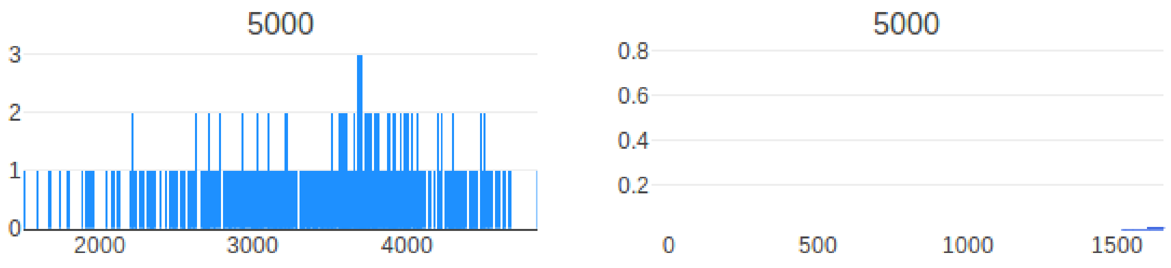


Figure 4.8: Pagerank in P3R(5000): histogram for output vertices (left) and its cummulative version (right).

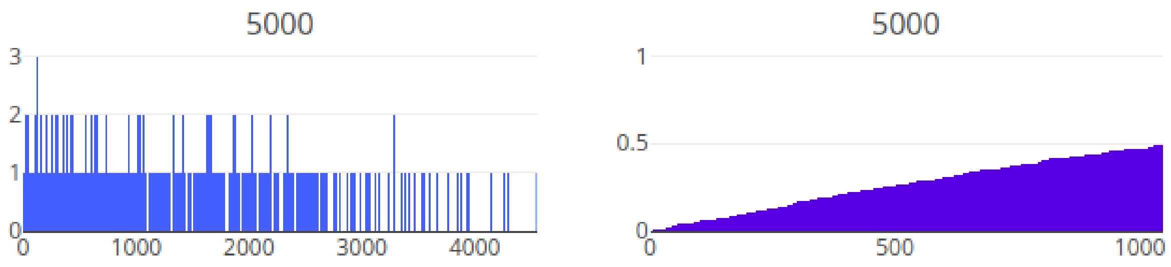


Figure 4.9: Degree in P3R(5000): histogram for output vertices (left) and its cummulative version (right).

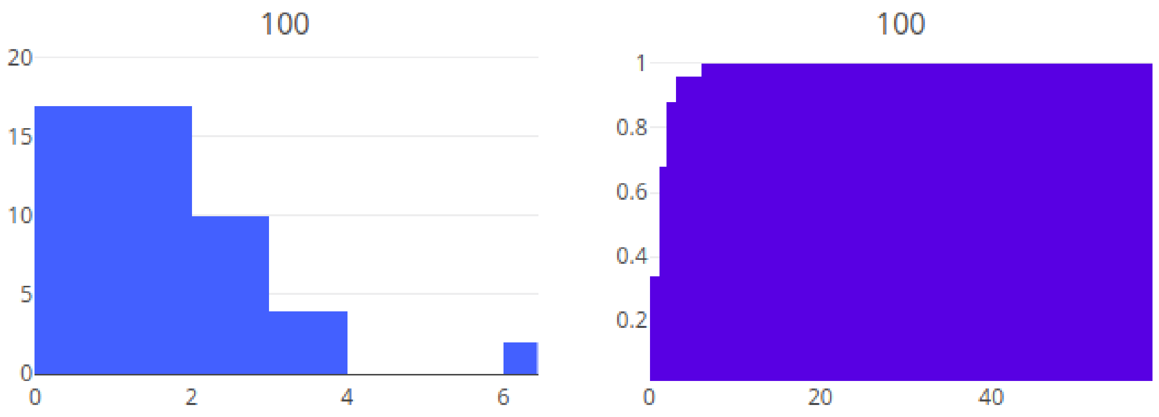


Figure 4.10: SZ-estimator in BA(100): histogram for output vertices (left) and its cummulative version (right).

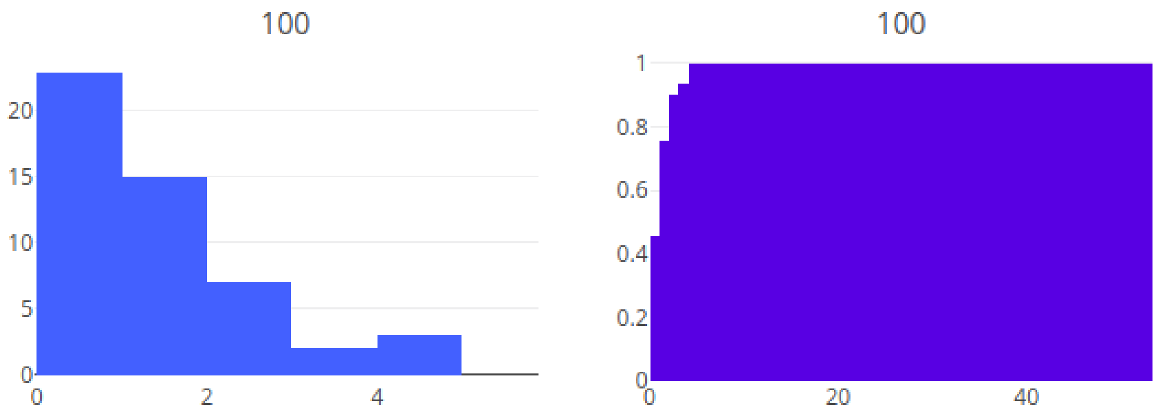


Figure 4.11: Simple in BA(100): histogram for output vertices (left) and its cummulative version (right).

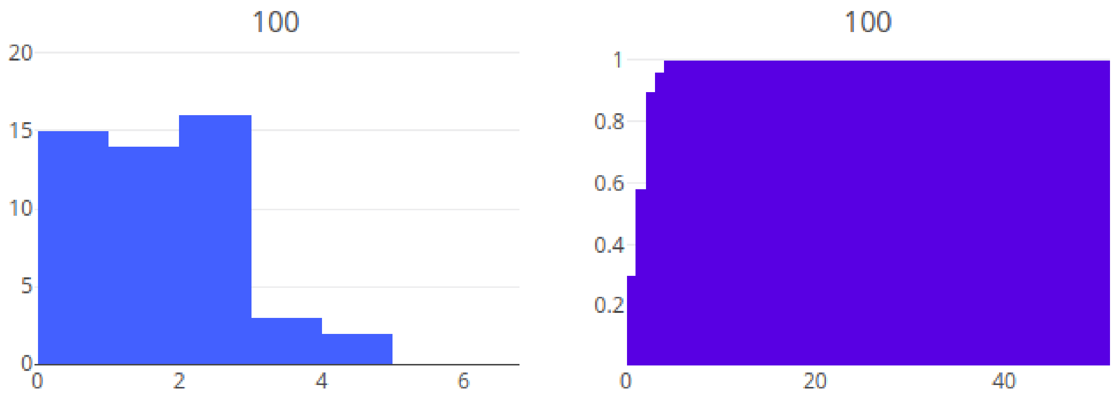


Figure 4.12: Distance in BA(100): histogram for output vertices (left) and its cummulative version (right).

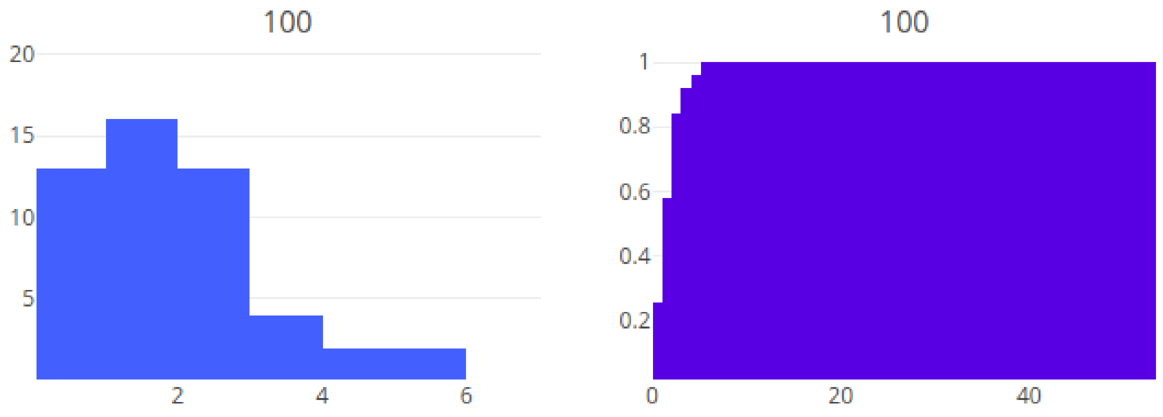


Figure 4.13: SZ-estimator in UA(100): histogram for output vertices (left) and its cummulative version (right).

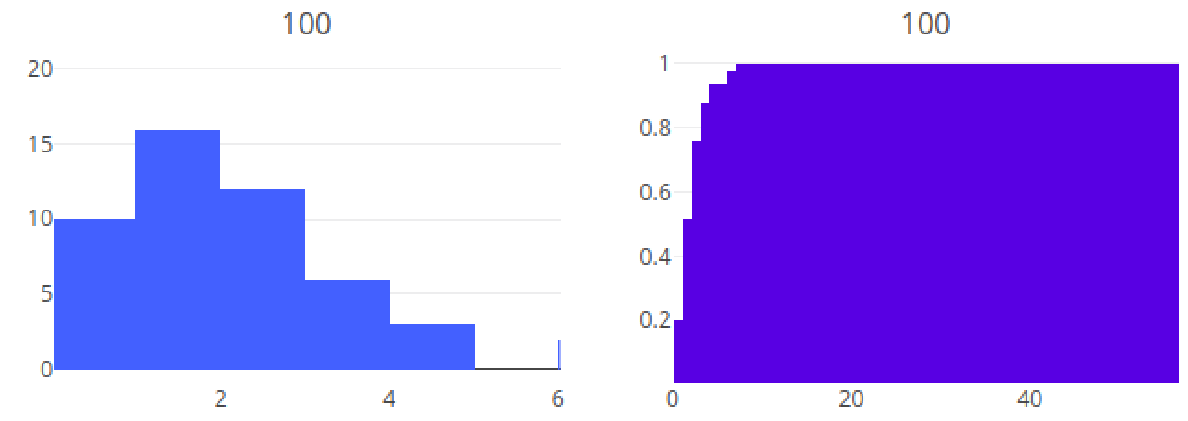


Figure 4.14: Simple in UA(100): histogram for output vertices (left) and its cummulative version (right).

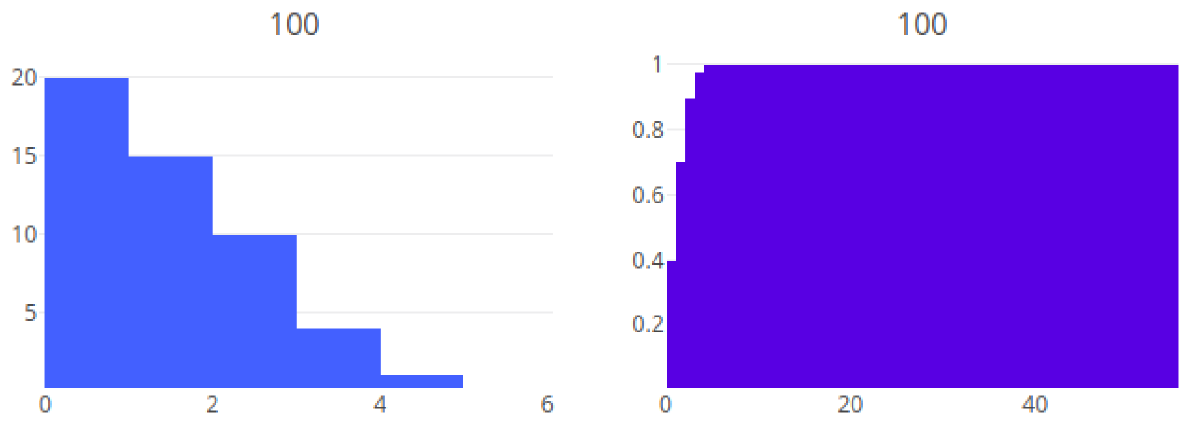


Figure 4.15: Distance in UA(100): histogram for output vertices (left) and its cummulative version (right).

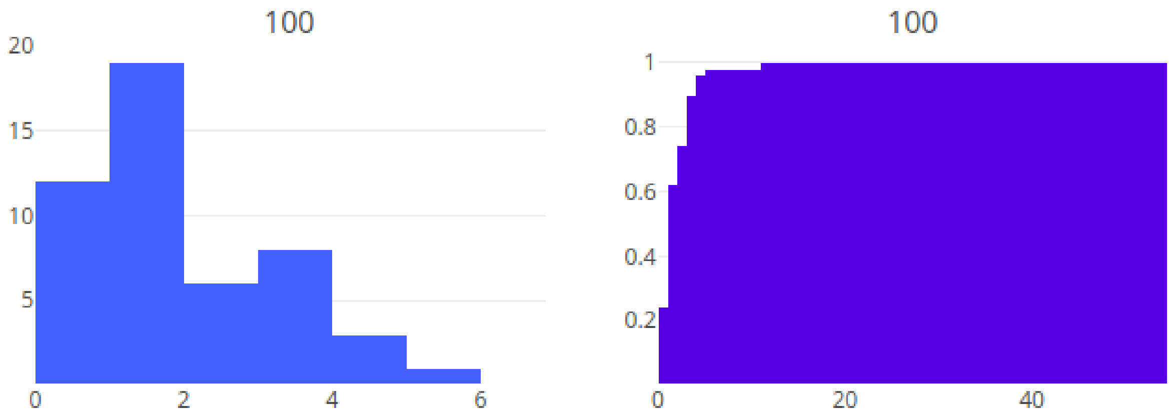


Figure 4.16: SZ-estimator in P3R(100): histogram for output vertices (left) and its cummulative version (right).

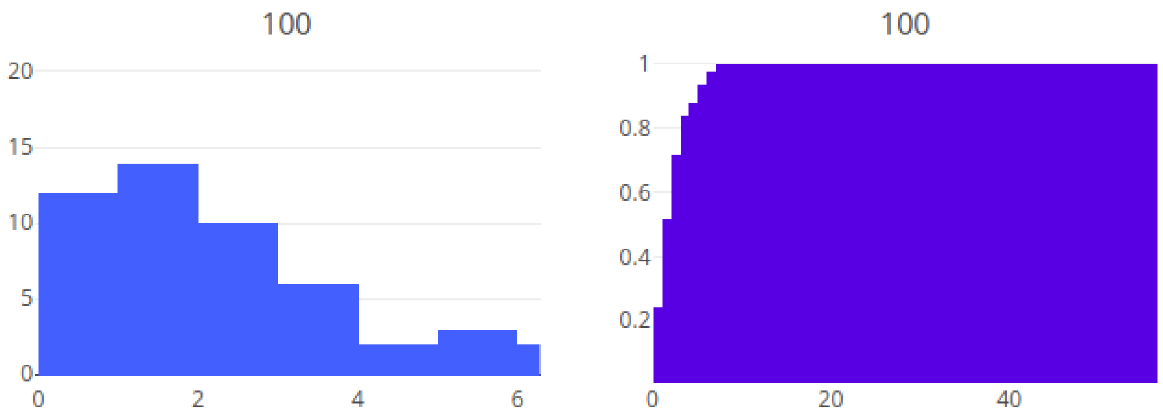


Figure 4.17: Simple in P3R(100): histogram for output vertices (left) and its cummulative version (right).

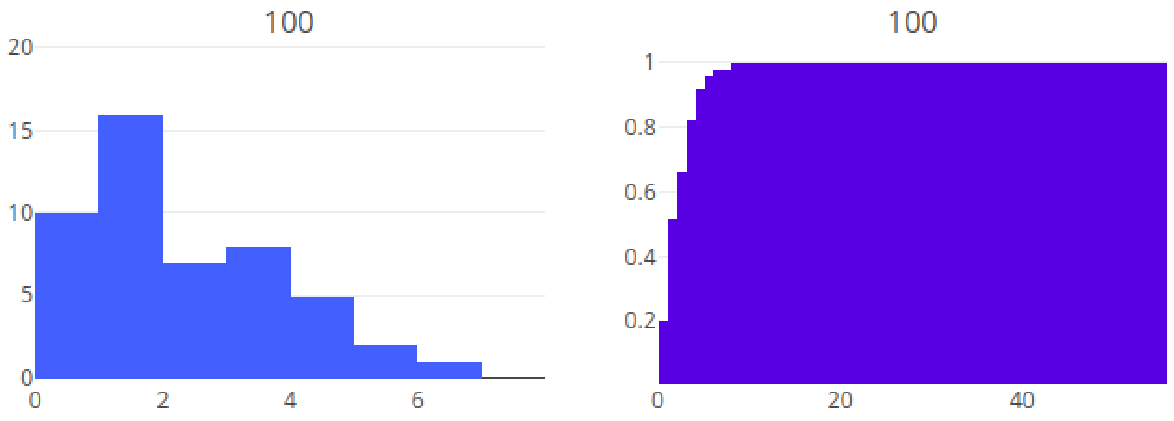


Figure 4.18: Distance in P3R(100): histogram for output vertices (left) and its cummulative version (right).

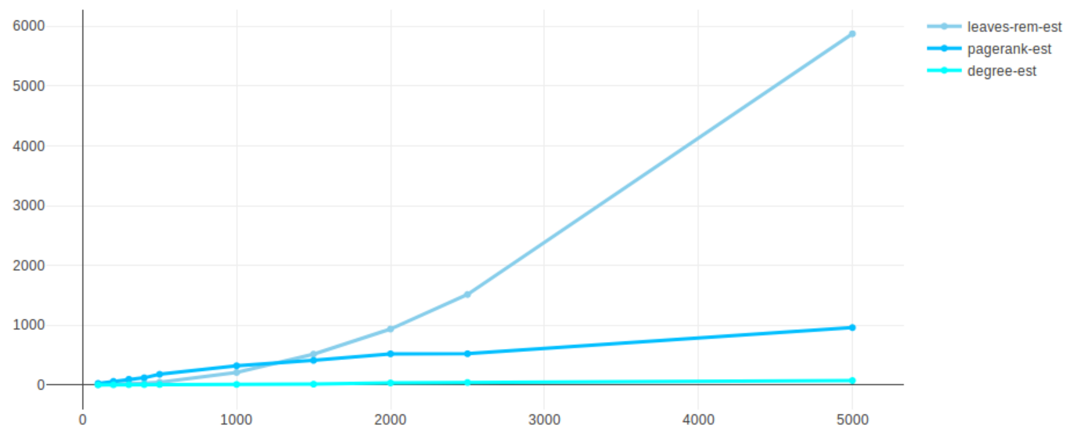


Figure 4.19: Finding the source: time performance for the 3 fastest estimators in BA

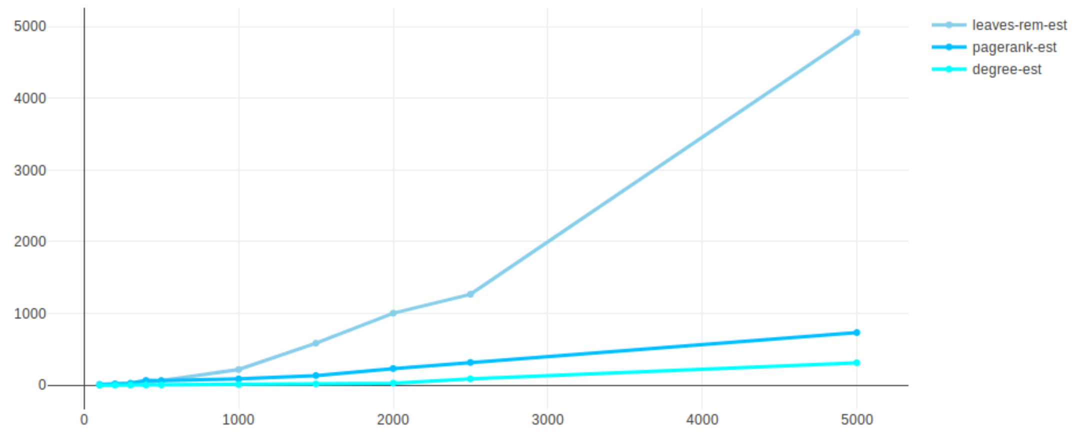


Figure 4.20: Finding the source: time performance for the 3 fastest estimators in UA

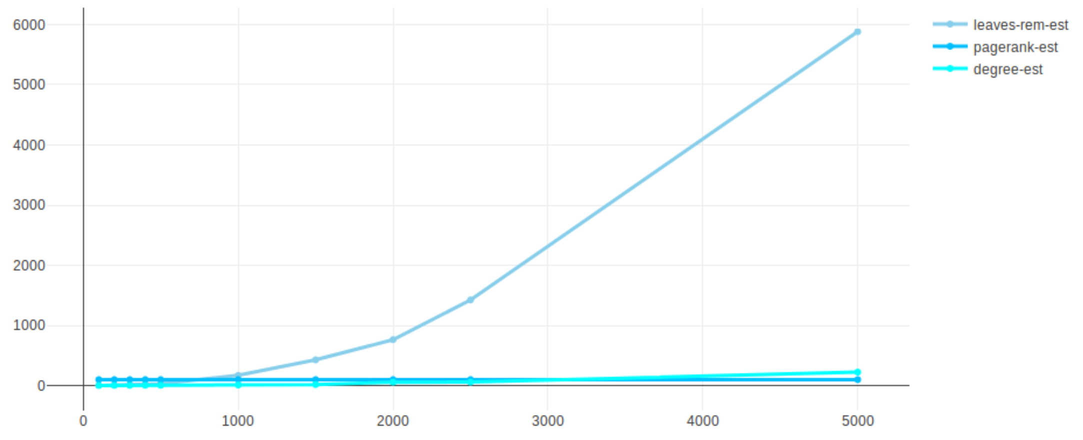


Figure 4.21: Finding the source: time performance for the 3 fastest estimators in P3R

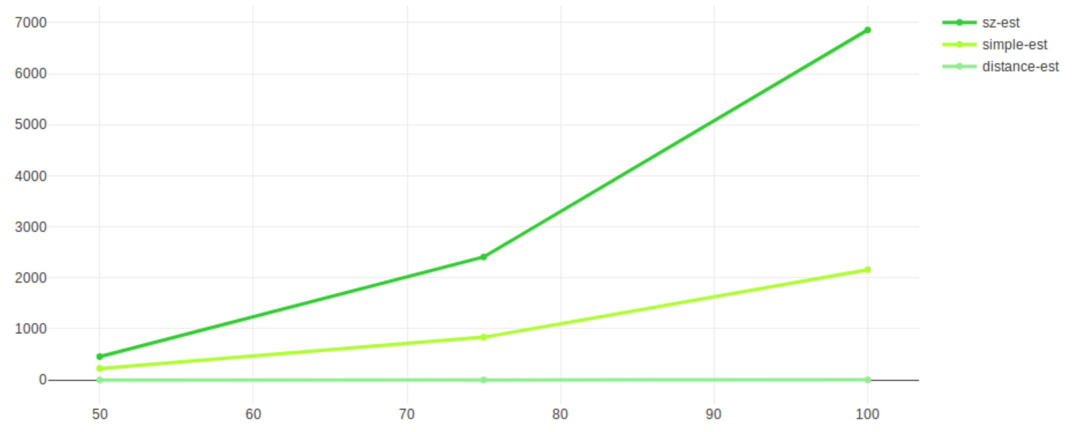


Figure 4.22: Finding the source: time performance for the 3 slowest estimators in BA

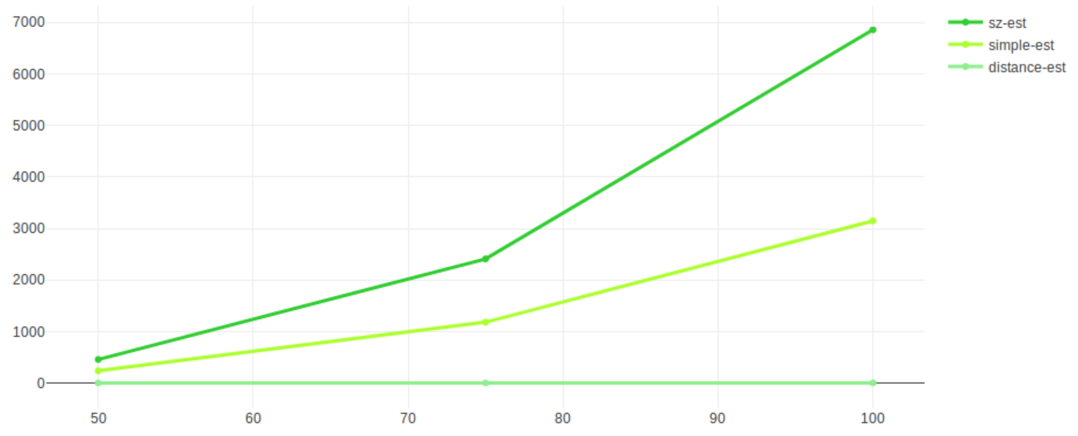


Figure 4.23: Finding the source: time performance for the 3 slowest estimators in UA

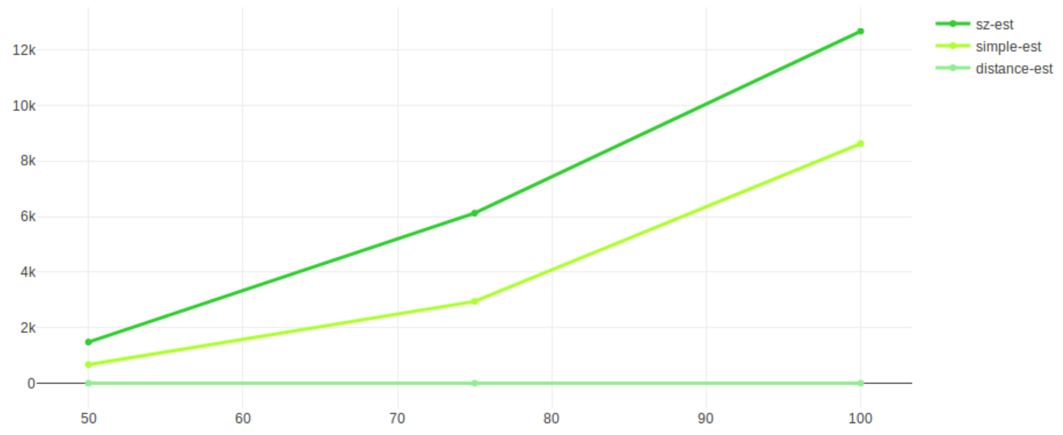
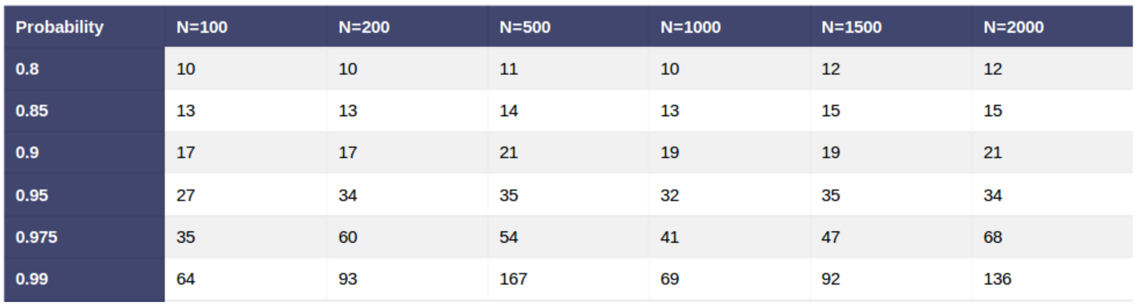
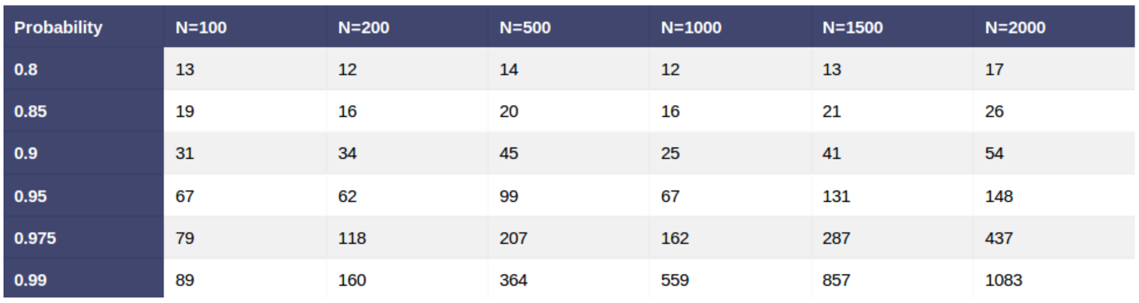


Figure 4.24: Finding the source: time performance for the 3 slowest estimators in P3R



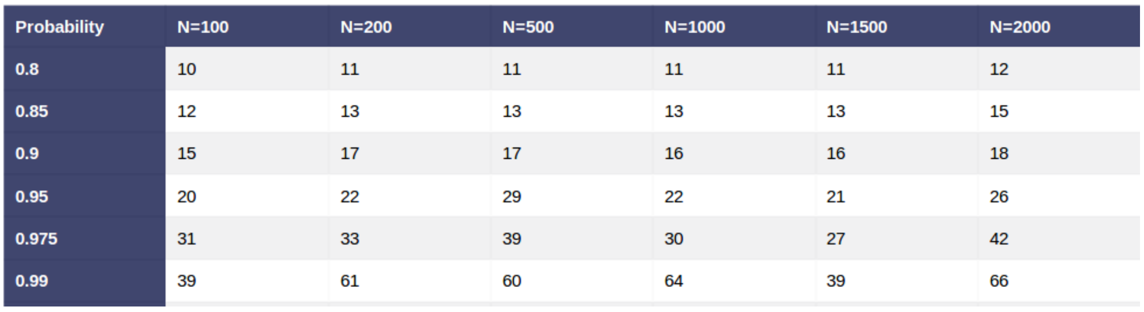
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	10	10	11	10	12	12
0.85	13	13	14	13	15	15
0.9	17	17	21	19	19	21
0.95	27	34	35	32	35	34
0.975	35	60	54	41	47	68
0.99	64	93	167	69	92	136

Figure 4.25: Output size set for Leaves Removal (UA)



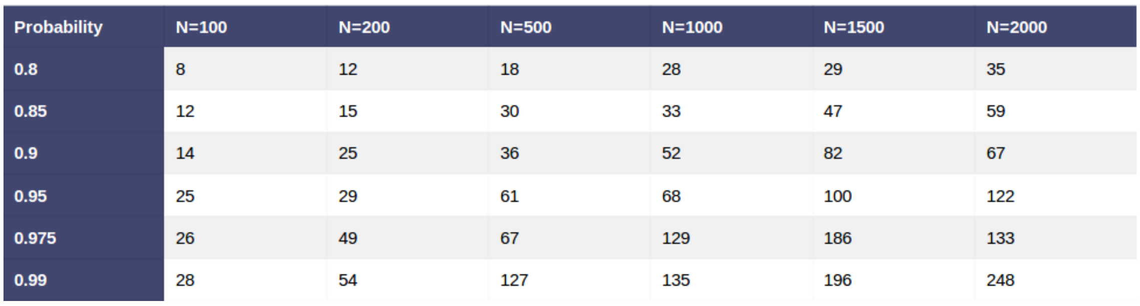
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	13	12	14	12	13	17
0.85	19	16	20	16	21	26
0.9	31	34	45	25	41	54
0.95	67	62	99	67	131	148
0.975	79	118	207	162	287	437
0.99	89	160	364	559	857	1083

Figure 4.26: Output size set for Leaves Removal (BA)



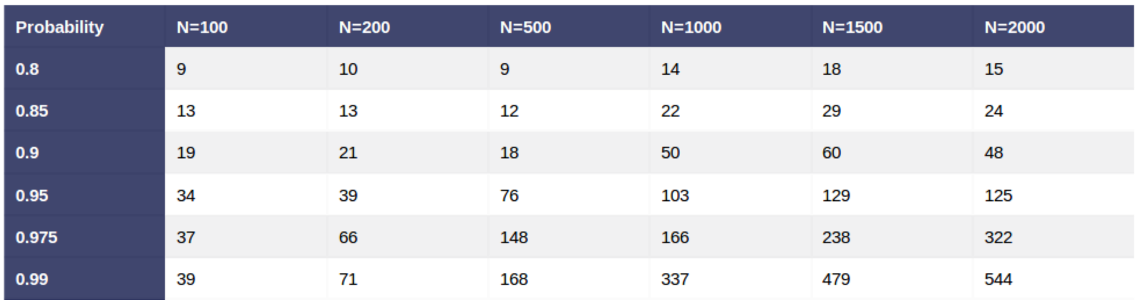
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	10	11	11	11	11	12
0.85	12	13	13	13	13	15
0.9	15	17	17	16	16	18
0.95	20	22	29	22	21	26
0.975	31	33	39	30	27	42
0.99	39	61	60	64	39	66

Figure 4.27: Output size set for Leaves Removal (P3R)



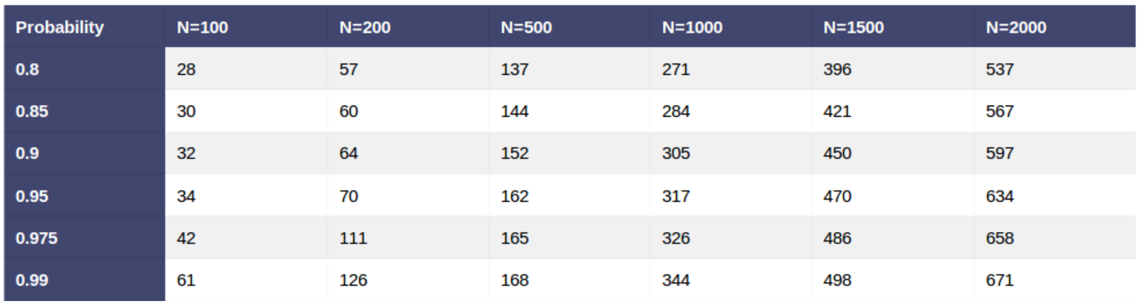
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	8	12	18	28	29	35
0.85	12	15	30	33	47	59
0.9	14	25	36	52	82	67
0.95	25	29	61	68	100	122
0.975	26	49	67	129	186	133
0.99	28	54	127	135	196	248

Figure 4.28: Output size set for Degree Estimator (UA)



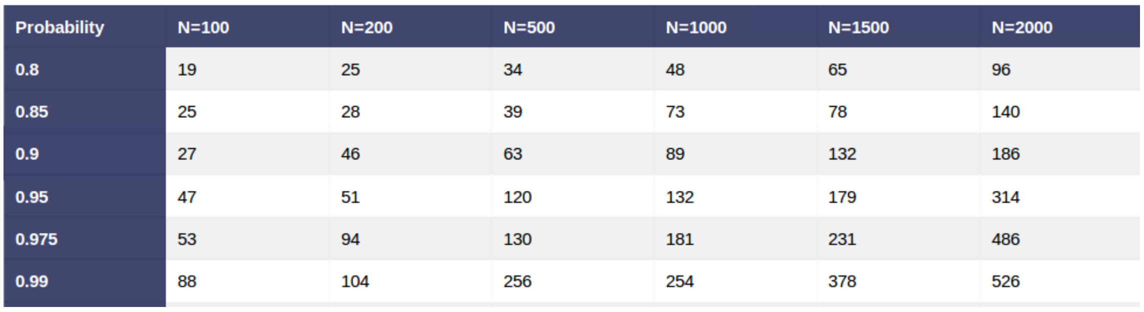
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	9	10	9	14	18	15
0.85	13	13	12	22	29	24
0.9	19	21	18	50	60	48
0.95	34	39	76	103	129	125
0.975	37	66	148	166	238	322
0.99	39	71	168	337	479	544

Figure 4.29: Output size set for Degree Estimator (BA)



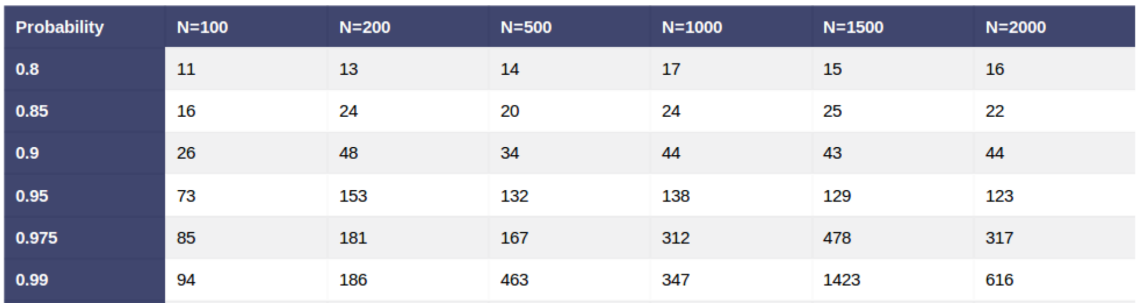
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	28	57	137	271	396	537
0.85	30	60	144	284	421	567
0.9	32	64	152	305	450	597
0.95	34	70	162	317	470	634
0.975	42	111	165	326	486	658
0.99	61	126	168	344	498	671

Figure 4.30: Output size set for Degree Estimator (P3R)



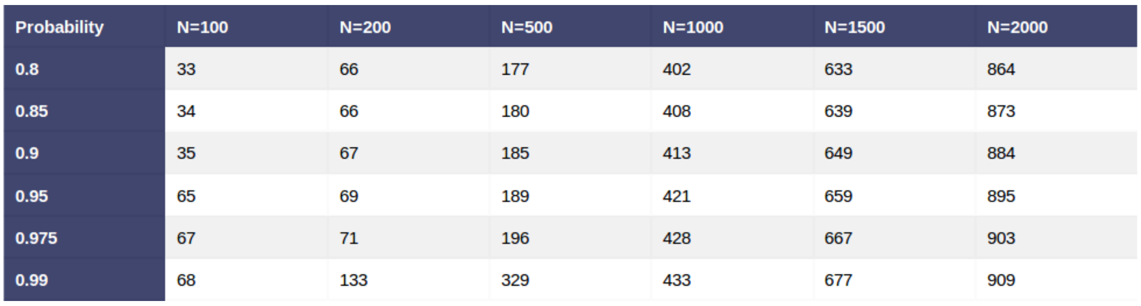
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	19	25	34	48	65	96
0.85	25	28	39	73	78	140
0.9	27	46	63	89	132	186
0.95	47	51	120	132	179	314
0.975	53	94	130	181	231	486
0.99	88	104	256	254	378	526

Figure 4.31: Output size set for PageRank Estimator (UA)



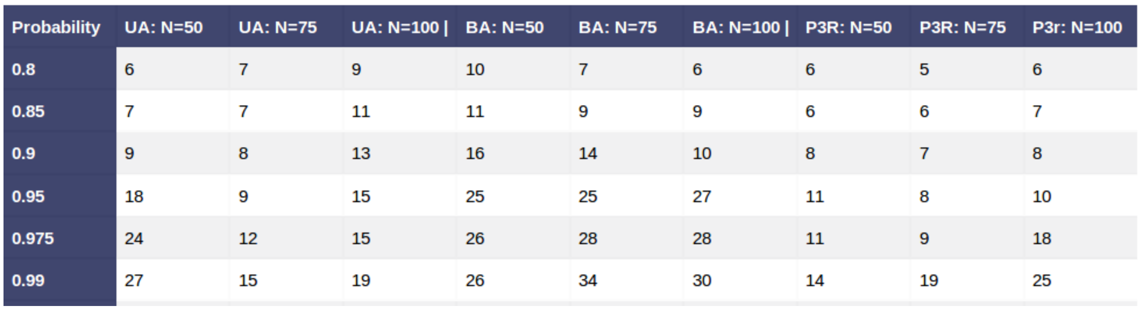
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	11	13	14	17	15	16
0.85	16	24	20	24	25	22
0.9	26	48	34	44	43	44
0.95	73	153	132	138	129	123
0.975	85	181	167	312	478	317
0.99	94	186	463	347	1423	616

Figure 4.32: Output size set for PageRank Estimator (BA)



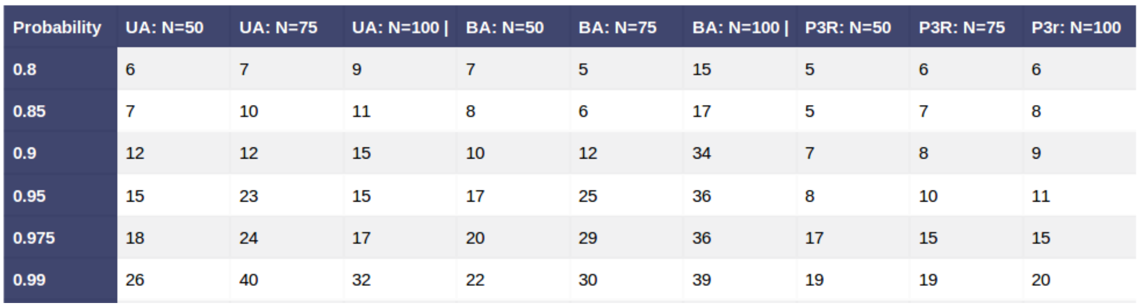
Probability	N=100	N=200	N=500	N=1000	N=1500	N=2000
0.8	33	66	177	402	633	864
0.85	34	66	180	408	639	873
0.9	35	67	185	413	649	884
0.95	65	69	189	421	659	895
0.975	67	71	196	428	667	903
0.99	68	133	329	433	677	909

Figure 4.33: Output size set for PageRank Estimator (P3R)



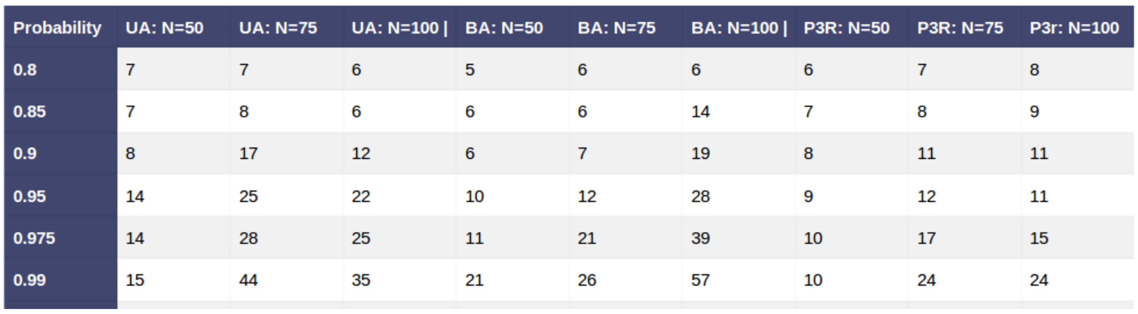
Probability	UA: N=50	UA: N=75	UA: N=100 |	BA: N=50	BA: N=75	BA: N=100 |	P3R: N=50	P3R: N=75	P3r: N=100
0.8	6	7	9	10	7	6	6	5	6
0.85	7	7	11	11	9	9	6	6	7
0.9	9	8	13	16	14	10	8	7	8
0.95	18	9	15	25	25	27	11	8	10
0.975	24	12	15	26	28	28	11	9	18
0.99	27	15	19	26	34	30	14	19	25

Figure 4.34: Output size set for SZ-estimator



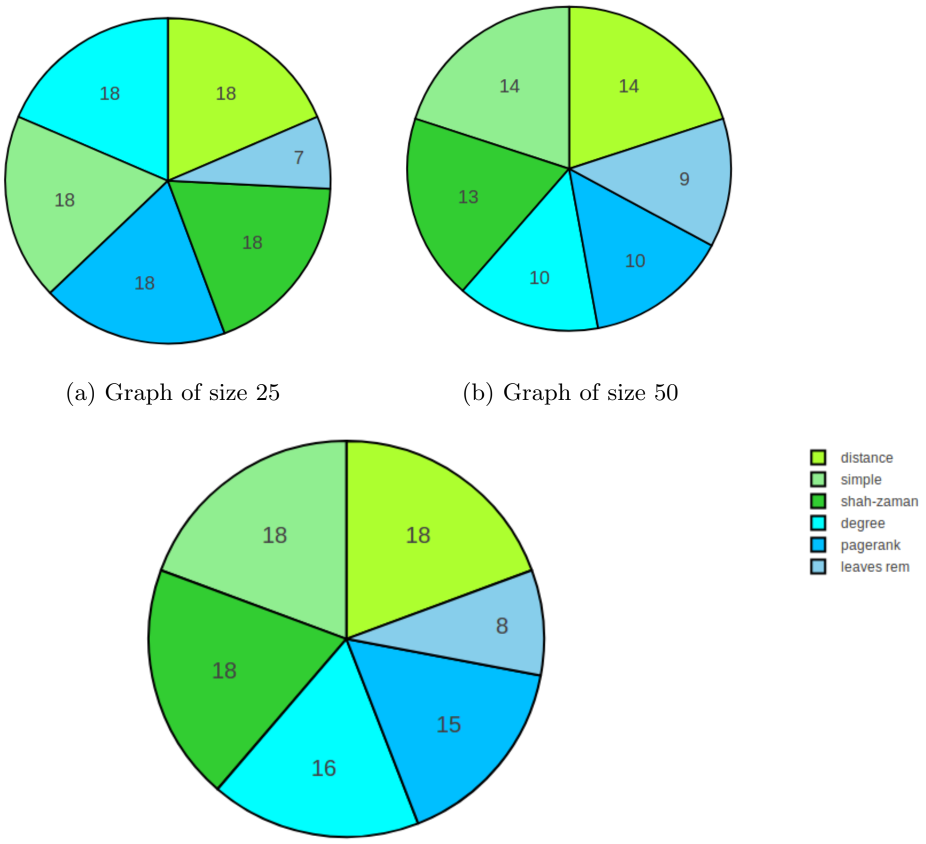
Probability	UA: N=50	UA: N=75	UA: N=100 |	BA: N=50	BA: N=75	BA: N=100 |	P3R: N=50	P3R: N=75	P3r: N=100
0.8	6	7	9	7	5	15	5	6	6
0.85	7	10	11	8	6	17	5	7	8
0.9	12	12	15	10	12	34	7	8	9
0.95	15	23	15	17	25	36	8	10	11
0.975	18	24	17	20	29	36	17	15	15
0.99	26	40	32	22	30	39	19	19	20

Figure 4.35: Output size set for Simple Estimator



Probability	UA: N=50	UA: N=75	UA: N=100 |	BA: N=50	BA: N=75	BA: N=100 |	P3R: N=50	P3R: N=75	P3r: N=100
0.8	7	7	6	5	6	6	6	7	8
0.85	7	8	6	6	6	14	7	8	9
0.9	8	17	12	6	7	19	8	11	11
0.95	14	25	22	10	12	28	9	12	11
0.975	14	28	25	11	21	39	10	17	15
0.99	15	44	35	21	26	57	10	24	24

Figure 4.36: Output size set for Distance Estimator

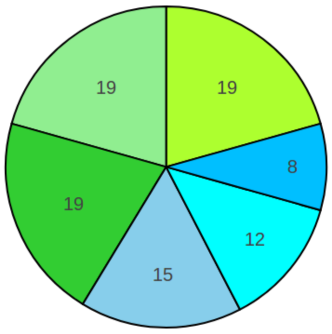


(a) Graph of size 25

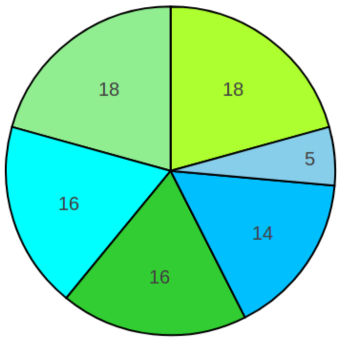
(b) Graph of size 50

(c) Graph of size 75

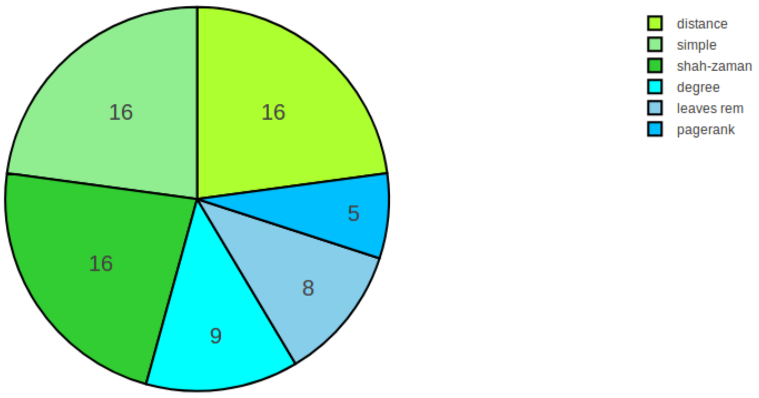
Figure 4.37: Tournament results for BA



(a) Graph of size 25

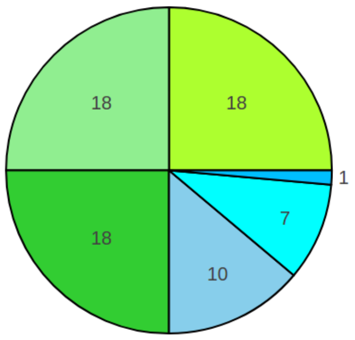


(b) Graph of size 50

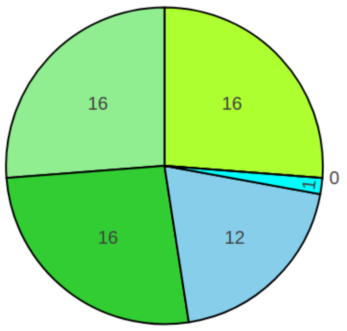


(c) Graph of size 75

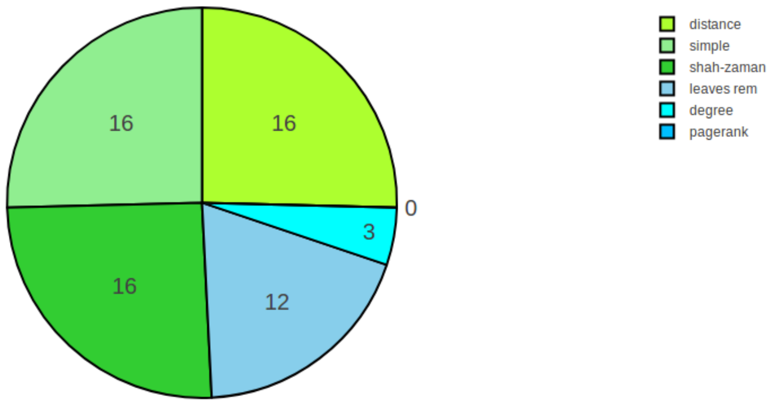
Figure 4.38: Tournament results for UA



(a) Graph of size 25

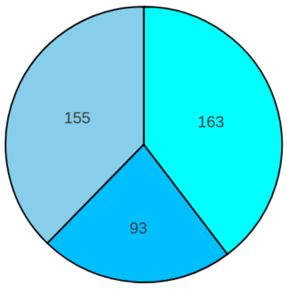


(b) Graph of size 50

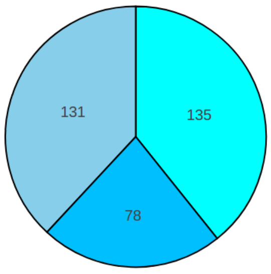


(c) Graph of size 75

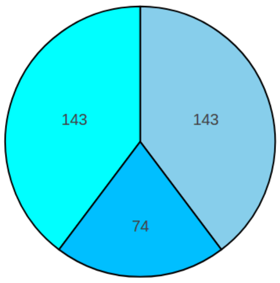
Figure 4.39: Tournament results for P3R



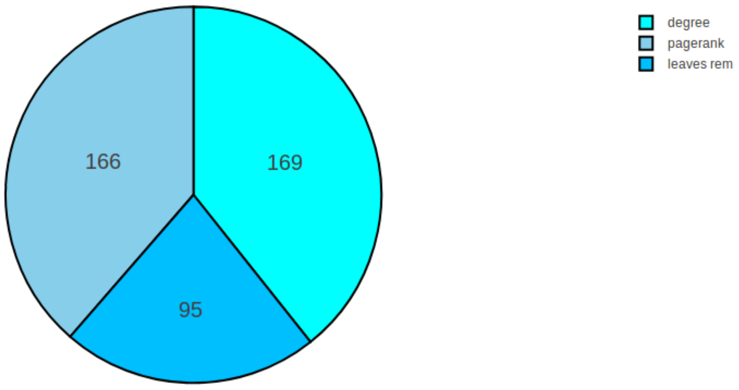
(a) Graph of size 300



(b) Graph of size 400

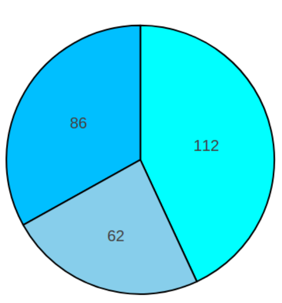


(c) Graph of size 500

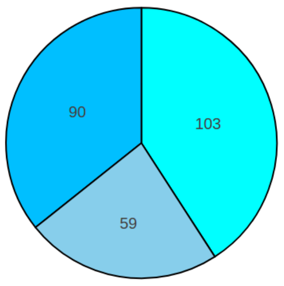


(d) Graph of size 1000

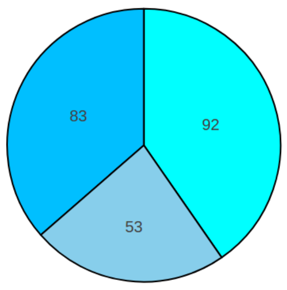
Figure 4.40: Tournament results for the 3 fastest estimators in PA graphs



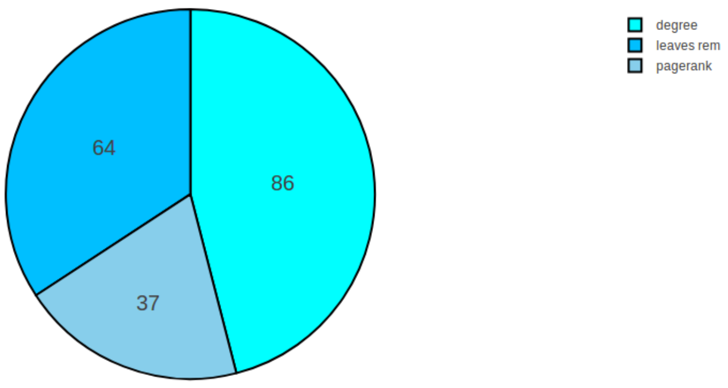
(a) Graph of size 300



(b) Graph of size 400



(c) Graph of size 500



(d) Graph of size 1000

Figure 4.41: Tournament results for the 3 fastest estimators in UA graphs

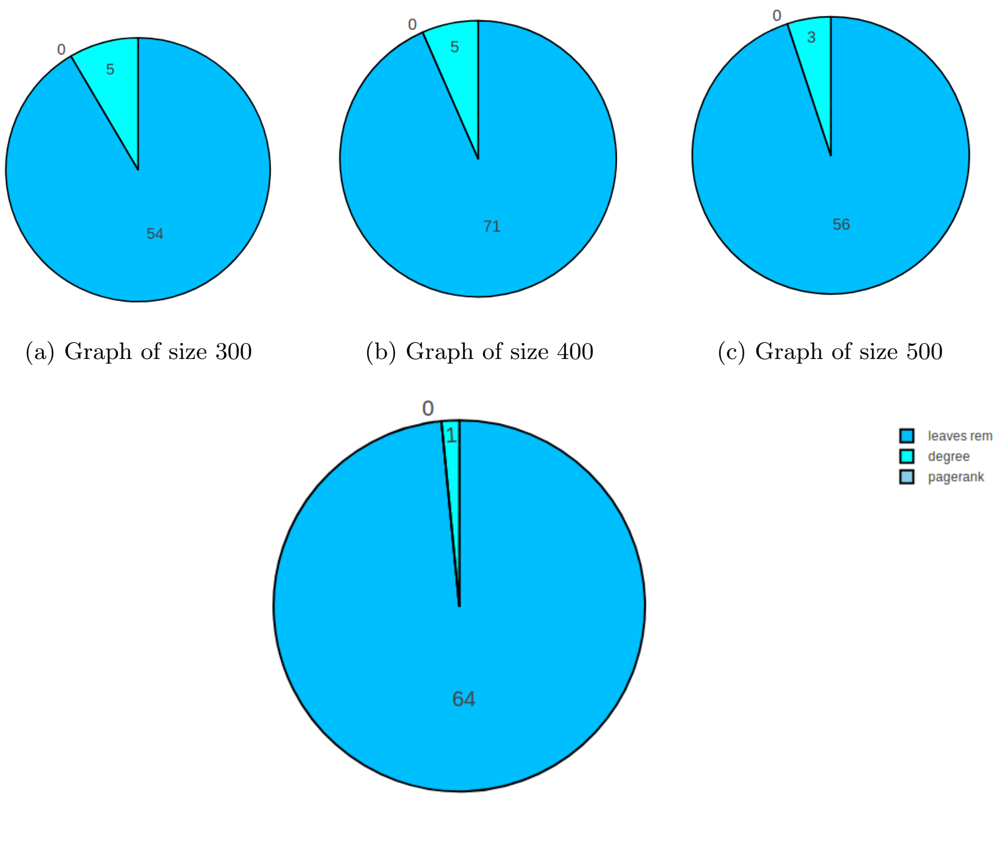
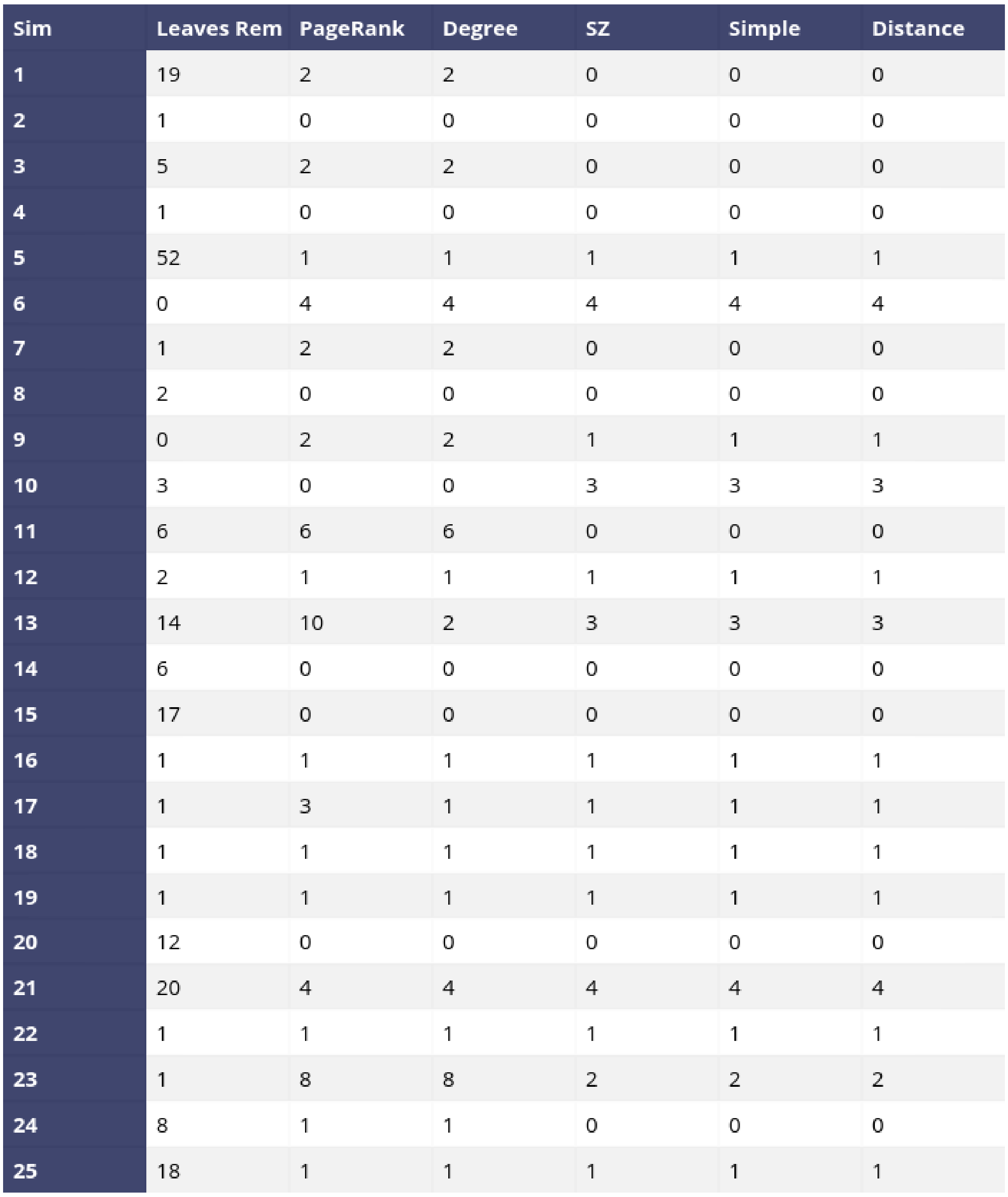
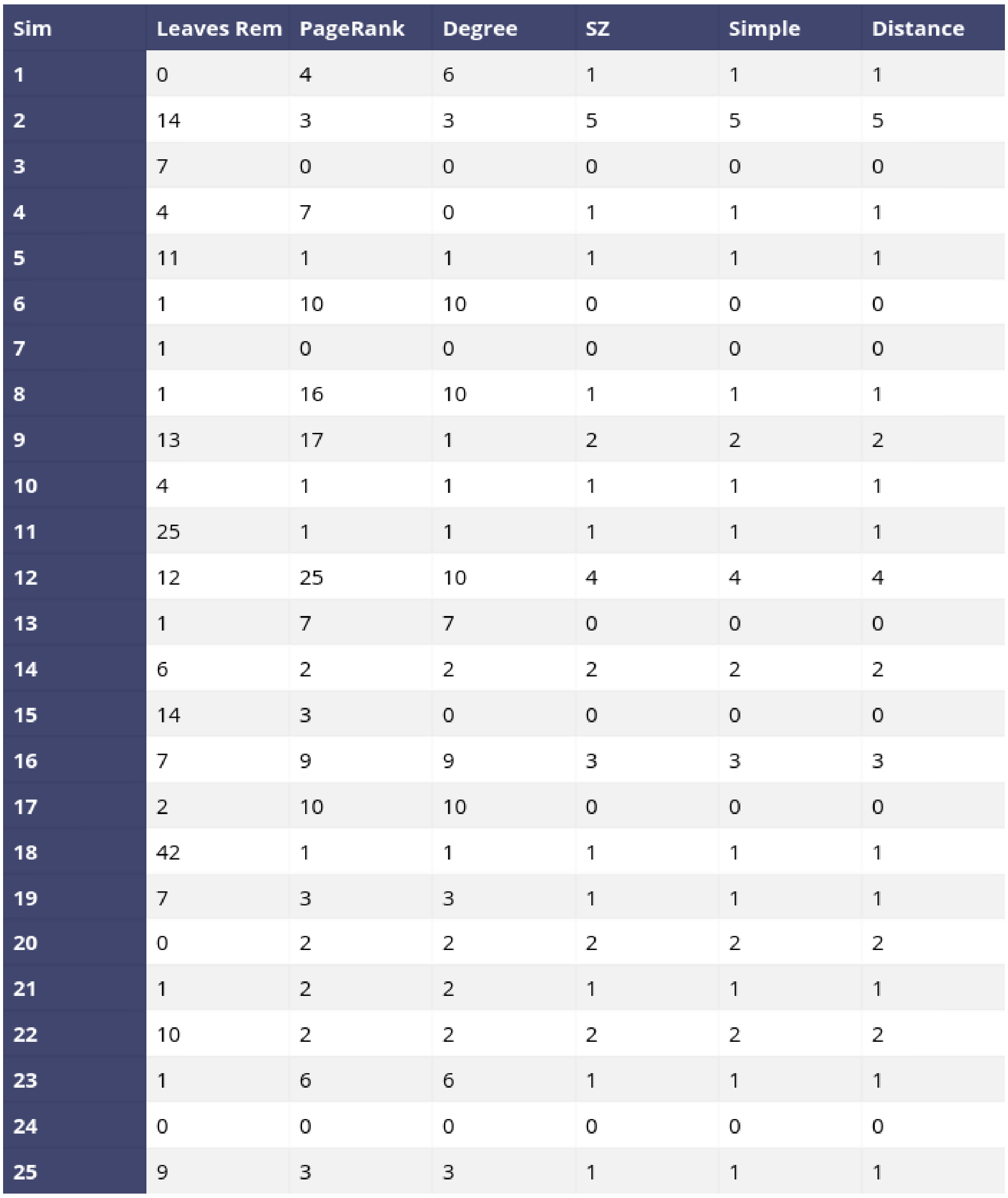


Figure 4.42: Tournament results for the 3 fastest estimators in P3R graphs



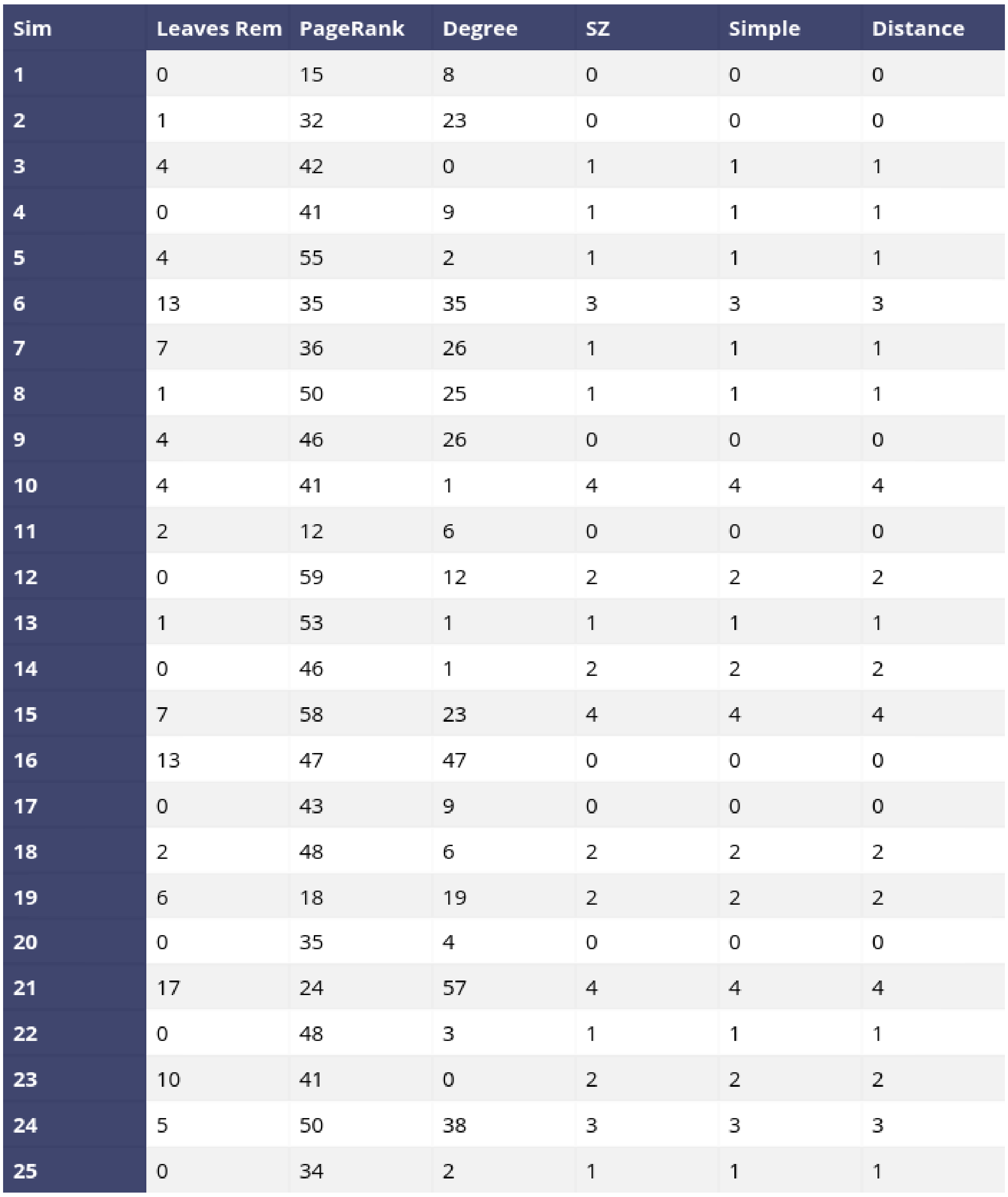
Sim	Leaves Rem	PageRank	Degree	SZ	Simple	Distance
1	19	2	2	0	0	0
2	1	0	0	0	0	0
3	5	2	2	0	0	0
4	1	0	0	0	0	0
5	52	1	1	1	1	1
6	0	4	4	4	4	4
7	1	2	2	0	0	0
8	2	0	0	0	0	0
9	0	2	2	1	1	1
10	3	0	0	3	3	3
11	6	6	6	0	0	0
12	2	1	1	1	1	1
13	14	10	2	3	3	3
14	6	0	0	0	0	0
15	17	0	0	0	0	0
16	1	1	1	1	1	1
17	1	3	1	1	1	1
18	1	1	1	1	1	1
19	1	1	1	1	1	1
20	12	0	0	0	0	0
21	20	4	4	4	4	4
22	1	1	1	1	1	1
23	1	8	8	2	2	2
24	8	1	1	0	0	0
25	18	1	1	1	1	1

Figure 4.43: Vertices returned by the estimators in the first 25 simulations for PA(75)



Sim	Leaves Rem	PageRank	Degree	SZ	Simple	Distance
1	0	4	6	1	1	1
2	14	3	3	5	5	5
3	7	0	0	0	0	0
4	4	7	0	1	1	1
5	11	1	1	1	1	1
6	1	10	10	0	0	0
7	1	0	0	0	0	0
8	1	16	10	1	1	1
9	13	17	1	2	2	2
10	4	1	1	1	1	1
11	25	1	1	1	1	1
12	12	25	10	4	4	4
13	1	7	7	0	0	0
14	6	2	2	2	2	2
15	14	3	0	0	0	0
16	7	9	9	3	3	3
17	2	10	10	0	0	0
18	42	1	1	1	1	1
19	7	3	3	1	1	1
20	0	2	2	2	2	2
21	1	2	2	1	1	1
22	10	2	2	2	2	2
23	1	6	6	1	1	1
24	0	0	0	0	0	0
25	9	3	3	1	1	1

Figure 4.44: Vertices returned by the estimators in the first 25 simulations for UA(75)



Sim	Leaves Rem	PageRank	Degree	SZ	Simple	Distance
1	0	15	8	0	0	0
2	1	32	23	0	0	0
3	4	42	0	1	1	1
4	0	41	9	1	1	1
5	4	55	2	1	1	1
6	13	35	35	3	3	3
7	7	36	26	1	1	1
8	1	50	25	1	1	1
9	4	46	26	0	0	0
10	4	41	1	4	4	4
11	2	12	6	0	0	0
12	0	59	12	2	2	2
13	1	53	1	1	1	1
14	0	46	1	2	2	2
15	7	58	23	4	4	4
16	13	47	47	0	0	0
17	0	43	9	0	0	0
18	2	48	6	2	2	2
19	6	18	19	2	2	2
20	0	35	4	0	0	0
21	17	24	57	4	4	4
22	0	48	3	1	1	1
23	10	41	0	2	2	2
24	5	50	38	3	3	3
25	0	34	2	1	1	1

Figure 4.45: Vertices returned by the estimators in the first 25 simulations for P3R(75)

Conclusion

Thinking about finding the source of a rumor spread in a social network is a very illustrative way of presenting the studied family of root-finding problems. Identifying people who have the information as nodes in a graph and relations between them as its edge gives us a natural framework to work with in regards to the task of finding the first vertex of a random generated graph.

The three studied models for growing trees assumed as ground truth showed us how different could be output results when talking about source estimators. In the case where the underlying structure of the graph came from regular trees, we saw that the behavior of the SZ-estimator had two different faces. We proved that, intuitively, if the involved people only send the rumor to one of its neighbors in the network, then the probability of correct source detection by the SZ-estimator goes to 0 as the number of people knowing the rumor grows. On the other hand, if people propagate the information to at most two neighbors in the network, then the probability that this estimator detects the root is greater or equal than $1/4$ in the limit. To prove such results it was vital the used of the nice properties of the ML estimator. Relating them together with a novel measure of SZ-centrality, we were capable of giving explicit formulas for computing the probabilites of seeing particular order of rumor infections. Although they were useful from a theoretical point of view, they were computationally expensive. For this reason, we gave an algorithm to compute such probabilities based on the idea of passing messages. Utilizing this technique, it was possible to implement the SZ-estimators using Python tools, and to test it by simulations.

For the case where the estimators were allowed to return a set of candidates for the root, we saw that some of them were able to give such sets with size independent of the rate of growth of the underlying network. We call these kind of algorithms the root-finding estimators. We also present some properties of a specific function, the simple estimator, which computes the maximum subtree size in the network, for a given vertex v, and we proved that it is a root-finding estimator in the case where the graph follows Uniform and Preferential mechanisms. We used, once again, the properties of the ML estimator to give lower and upper bounds for output size set given by any root-finding algorithm.

In order to explore better the studied theory, we presented other algorithms: Distance, PageRank, Degree and Leaves Removal. The first three are part of the famous estimators while the latter is perhaps a not-so-well-studied one. We tested some theoretical results proposing three kinds of simulation tasks. Results showed that there are 3 estimators of these 6 mentioned before that outstand in terms of time performance: PageRank, Degree and Leaves Removal. It was possible to do around 500 simulations on graph which sizes went from 100 up to 5000. For the remaining ones, results in terms of time were not so satisfactory: it was only possible to do 50 simulations for each one in graphs which maximum size were equal to 100. Letting aside time issues, we saw that the slowest estimators were more accurate than the fastest ones: They found the source many more times than the others.

Appendix A

Pólya urns model and Dirichlet distribution

First, let's intuitively define a random process by induction in the following way: Begin with the tuple $(1,0) \in \{0,1\}^2$. Next, construct a new tuple $(1,0,a_1,a_2,...,a_{n-2}) \in \{0,1\}^n$ from $(1,0,a_1,a_2,...,a_{n-3}) \in \{0,1\}^{n-1}$ adding $a_{n-2} \in \{0,1\}$ according to the next probabilities:

$$\mathbb{P}(a_{n-2} = 1|(1,0,a_1,a_2,...,a_{n-3})) = \frac{|\{a_i : a_i = 1\}| + 1}{n-1},$$
$$\mathbb{P}(a_{n-2} = 0|(1,0,a_1,a_2,...,a_{n-3})) = \frac{|\{a_i : a_i = 0\}| + 1}{n-1}$$

A random process built following the previous steps is known as a *standard Pólya urn model with 2 colors.* Intuitively, the elements of the tuple $(1,0,a_1,a_2,...,a_{n-3})$ are colored balls in a box. If $a_j = 1$, then the $(j+2)^{th}$ added ball is black and $a_j = 0$, then the $(j+2)^{th}$ added ball is white, and what the process does is telling us that the chance of adding a new black or white ball is going to be proportional to the amount of black or white balls that are already in the box.

A standard Pólya urn model with k colors follows the same rules but the elements cosidered to built the sequence are taken from the set $[0, k-1]$.

Definition A.0.1. *Let R_k be a matrix that satisfies*

$$R_k = (a_{ij})_{1 \le i,j \le k}, \quad \textit{with } a_{ij} \in \mathbb{N}, \forall 1 \le i,j \le k.$$

Let P_0 be a vector that satisfies

$$P_= (b_1, b_2, ..., b_k)^T, \quad \textit{with } b_i \in \mathbb{N}, \forall 1 \le i \le k.$$

The Pólya urn process

$$(P_n)_n = \begin{pmatrix} P_n^{(1)} \\ P_n^{(2)} \\ \vdots \\ P_n^{(k)} \end{pmatrix}$$

with replacement matrix R_k and initial composition vector P_0 is a Markov process defined according to the following distribution

$$\mathbb{P}\left(P_{n+1} = P_n + (a_{ij})_{1 \le i \le k} |P_n\right) = \frac{P_n^{(j)}}{\sum_{j=1}^k P_n^{(j)}}, \quad \forall 1 \le j \le k.$$

Sometimes, when the replacement and initial matrices are not relevant, this process receives the name of Pólya urn scheme of k colors.

The entry (R_{ij}) of the replacement matrix says that R_{ij} balls of the i^{th} color should be place in the box after one step. The vector P_0 says how many balls of each color the box initially contains.

For the intuitive explanation given at the begining of this section, we have that values of R_k and P_0 are

$$R_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and $P_0 = (1,1)^T$.

For $b \in \mathbb{N}$, considerer the *simplex of dimension* b defined as

$$\Delta_b = \left\{ (q_1, ..., q_b) \in \mathbb{R}^b : \quad \sum_{i=1}^{k-1} q_i = 1, \; q_i \geq 0 \text{ for } 1 \leq i \leq b \right\}.$$

Theorem A.0.1. *Let $(P_n)_n$ be a Pólya urn process of b colors and with initial composition vector P_0. Suppose that $P_0^{(i)} = 1$, for all $1 \leq i \leq b$. Then,*

$$\left(\frac{P_n^{(1)}}{n}, \frac{P_n^{(2)}}{n}, ..., \frac{P_n^{(b)}}{n} \right) \underset{a.s.}{\rightarrow} Y, \quad \text{as } n \to +\infty,$$

where Y satisfies the following two properties

1. *$Y \in \Delta_b$.*
2. *Y is uniform, i.e. if*
$$Y = (Y_1, .., Y_b),$$
then $(Y_1, .., Y_{b-1})$ is uniform over
$$\overline{\Delta}_{b-1} = \left\{ (q_1, ..., q_{b-1}) \in \mathbb{R}^{b-1} : \quad \sum_{i=1}^{b-1} q_i \leq 1, \; q_i \geq 0 \text{ for } 1 \leq i \leq b-1 \right\}.$$

Remarks:

(i) It can be shown that the random vector Y satisfies

$$Y =_d (U_{(1)}, U_{(2)} - U_{(1)}, ..., 1 - U_{(b-1)}),$$

where $U_{(1)}, U_{(2)}, ..., U_{(b-1)}$ is the order statistics of $(b-1)$ i.i.d. random variables that are uniform in $[0,1]$.

(ii) The distribution of the random vector Y is known as the *Dirichlet distribution with parameter* $\alpha = ((1)_{i=1}^b)$. We denote this kind of distribution as $\text{Dir}((1)_{i=1}^b)$.

Now, it is all set up to prove Theorem A.0.1:

Proof. (i) First, we are going to show that, given $\in \mathbb{N}$,

$$\mathbb{P}(P_n^{(1)} = m_1, ..., P_n^{(b)} = m_b)$$

is equal for all choices of $(m_1, .., m_b) \in \mathbb{N}_+^b$ with $m_1 + m_2 + ... + m_b = b + n$.

For $n = 0$, we know that if $m_i \geq 1$, for all $1 \leq i \leq b$, and

$$m_1 + m_2 + \ldots + m_b = b,$$

then we must have $m_i = 1$, for all $1 \leq i \leq b$. Therefore,

$$\mathbb{P}(P_0^{(1)} = 1, ..., P_0^{(b)} = 1) = 1,$$

which proves the induction basis. Now, define

$$\Lambda_n = \left\{ (m_1, ..., m_b) \in \mathbb{N}_+^b : \sum_{i=1}^{b} m_i = b + n \right\},$$

and assume that

$$\mathbb{P}(P_n = (m_i)_{i=1}^b) = C_n, \quad \forall (m_i)_{i=1}^n \in \Lambda_n,$$

for some positive constant $C_n > 0$.
Let $(m'_1, ..., m'_b) \in \Lambda_{n+1}$. We know that

$$\mathbb{P}(P_{n+1} = (m'_i)_{i=1}^b) = \sum_{(m_i)_{i=1}^b \in \Lambda_n} \mathbb{P}\left(P_n = (m_i)_{i=1}^b\right) \mathbb{P}\left(P_{n+1} = (m'_i)_{i=1}^b | P_n = (m_i)_{i=1}^b\right).$$

For any $(m_i)_{i=1}^b \in \Lambda_n$, we must have that

$$(m'_i)_{i=1}^b = (m_i)_{i=1}^b + e_j,$$

for some $1 \leq j \leq b$, since $\{P_n\}_n$ is a standard Pólya urn with replacement matrix $I_b = (e_1, e_2, ..., e_b)$. Therefore, we only have b possible choices for $(m_i)_{i=1}^b \in \Lambda_n$, once that $(m'_i)_{i=1}^n$ is fixed. Then

$$\mathbb{P}(P_{n+1} = (m'_i)_{i=1}^b) = \sum_{1 \leq j \leq b} C_n \mathbb{P}\left(P_{n+1} = (m'_i)_{i=1}^b | P_n = (m'_i)_{i=1}^b - e_j\right).$$

By the definition of Pólya urn,

$$\mathbb{P}\left(P_{n+1} = (m'_i)_{i=1}^b | P_n = (m'_i)_{i=1}^b - e_j\right) = \frac{m'_j - 1}{n + b},$$

and thus,

$$\begin{aligned} \mathbb{P}(P_{n+1} = (m'_i)_{i=1}^b) &= \sum_{1 \leq j \leq b} C_n \mathbb{P}\left(P_{n+1} = (m'_i)_{i=1}^b | P_n = (m'_i)_{i=1}^b - e_j\right) \\ &= \frac{C_n(n+1)}{n+b}, \end{aligned}$$

which is independent of $(m'_i)_{i=1}^b$.

(ii) Now, let's prove that for $i \in \{1, ..., b\}$, $\left\{ \frac{P_n^{(i)}}{n+b} \right\}_{n \geq 0}$ is a martingale with respect to the filtration $\mathcal{F}_n^i = \sigma(P_1^{(i)}, ..., P_n^{(i)})$.:

Note that

$$\begin{aligned}
\mathbb{E}\left(\frac{P^{(i)}_{n+1}}{n+1+b} \mid \mathcal{F}^i_n\right) &= \frac{1}{n+1+b}\mathbb{E}((P^{(i)}_n+1)\mathbb{1}_{\{P^{(i)}_{n+1}=P^{(i)}_n+1\}} \mid \mathcal{F}^i_n) \\
&\quad + \frac{1}{n+1+b}\mathbb{E}(P^{(i)}_n\mathbb{1}_{\{P^{(i)}_{n+1}=P^{(i)}_n\}} \mid \mathcal{F}^i_n) \\
&= \frac{P^{(i)}_n+1}{n+1+b}\mathbb{P}(P^{(i)}_{n+1}=P^{(i)}_n+1 \mid \mathcal{F}^i_n) \\
&\quad + \frac{P^{(i)}_n}{n+1+b}\mathbb{P}(P^{(i)}_{n+1}=P^{(i)}_n \mid \mathcal{F}^i_n) \\
&= \frac{P^{(i)}_n+1}{n+1+b}\frac{P^{(i)}_n}{n+b} + \frac{P^{(i)}_n}{n+1+b}\frac{n+b-P^{(i)}_n}{n+b} \\
&= \frac{P^{(i)}_n}{n+b}.
\end{aligned}$$

Thus, $\left\{\frac{P^{(i)}_n}{n+b}\right\}_{n\geq 0}$ is a bounded martingale and therefore, there exist random variables $Y^{(i)}$, for $i \in \{1,2,...,b\}$, such that

$$\frac{P^{(i)}_n}{n+b} \underset{\text{a.s.}}{\rightarrow} Y^{(i)}, \quad \text{for all } i \in \{1,2,...,b\}.$$

Define $Y = (Y^{(1)}, Y^{(2)}, ..., Y^{(b)})$. Then, clearly

$$\frac{P_n}{n+b} \rightarrow Y \quad \text{a.s.}.$$

(iii) Note that $Y \in \Delta_b$ a.s. This reason is that since

$$\sum_{i=1}^{b} P^{(i)}_n = b+n, \qquad \text{for all } n \geq 0,$$

it is obvious that

$$\sum_{i=1}^{b} \frac{P^{(i)}_n}{b+n} = 1, \qquad \text{for all } n \geq 0,$$

and therefore,

$$\lim_{n\to+\infty} \sum_{i=1}^{b} \frac{P^{(i)}_n}{b+n} = 1,$$

which shows that

$$\sum_{i=1}^{b} Y^{(i)} = 1, \quad \text{a.s.}$$

(iv) To conclude it is necessary to prove that

$$(Y^{(1)}, Y^{(2)}, ..., Y^{(b-1)}) \sim \text{Unif}(\overline{\Delta}_{b-1}).$$

Let $f : \overline{\Delta}_{b-1} \to \mathbb{R}$ be a continuous function. We want to prove that

$$\mathbb{E}\left(f(Y^{(1)}, Y^{(2)}, ..., Y^{(b-1)})\right) = \int \cdots \int_{\overline{\Delta}_{b-1}} f(x_1, ..., x_b) dx_1 ... dx_{b-1}. \tag{A.1}$$

Note that

$$\lim_{n\to+\infty} f\left(\frac{P_n^{(1)}}{n+b},...,\frac{P_n^{(b)}}{n+b}\right) = f(Y^{(1)}, Y^{(2)}, ..., Y^{(b-1)}),$$

by the continuity of f.

Since $\overline{\Delta}_{b-1}$ is compact, we have that f is bounded and therefore,

$$\lim_{n\to+\infty} \mathbb{E}\left(f\left(\frac{P_n^{(1)}}{n+b},...,\frac{P_n^{(b)}}{n+b}\right)\right) = \mathbb{E}\left(f(Y^{(1)}, Y^{(2)}, ..., Y^{(b-1)})\right),$$

by the Dominated Convergence Theorem.

Now, define

$$\overline{\Lambda}_b = \left\{(m_1, ..., m_b) \in \mathbb{N}_+^b : \sum_{i=1}^{b} m_i \le b+n\right\}.$$

By (i),

$$\mathbb{E}\left(f\left(\frac{P_n^{(1)}}{n+b},...,\frac{P_n^{(b)}}{n+b}\right)\right) = \frac{1}{|\Lambda_{b-1}|}\sum_{(m_1,...,m_{b-1})\in\Lambda_{b-1}} f\left(\frac{m_1}{n+b},...,\frac{m_{b-1}}{n+b}\right)$$

and since

$$\frac{1}{|\Lambda_{b-1}|}\sum_{(m_1,...,m_{b-1})\in\Lambda_{b-1}} f\left(\frac{m_1}{n+b},...,\frac{m_{b-1}}{n+b}\right)$$

is a Riemann sum for

$$\int\cdots\int_{\overline{\Delta}_{b-1}} f(x_1, ..., x_b)dx_1...dx_{b-1},$$

we see that A.1 is true and therefore,

$$(Y^{(1)}, Y^{(2)}, ..., Y^{(b-1)}) \sim \text{Unif}(\overline{\Delta}_{b-1}).$$

$\square$

It is possible to define Dirichlet distributions of parameter α for $\alpha \in \mathbb{N}^* \setminus \{(1, 1, ..., 1)\}$. The way of doing this is simply adding and deleting some of the coordinates of the parameter α. For instance, a random vector Z has distribution $\text{Dir}(\beta)$, for $\beta = (2, 1, ..., 1) \in \mathbb{R}^{b-1}$ with if

$$Z =_d (Y_1 + Y_2, Y_3, Y_4, ..., Y_b)$$

for $Y \sim \text{Dir}(\alpha)$ where $\alpha = (1)_{i=1}^{b-1}$.

Intuitively, we merge the first two coordinates of the parament α to obtain a Dirichlet distribution with parameter β which first coordinate is equal to 2 and the remaining $b-2$ are equal to 1.

This last property can be stated as follows

Proposition A.0.1. *Let $Z = (Z_1, ..., Z_m)$, $\alpha = (\alpha_1, \alpha_2, ..., \alpha_m)$ and suppose that $Z \sim$ Dir(α), then*

(i) For $i, j \in \{1, 2, ..., m\}$ with $i \le j$,

$$Z' = (Z'_1, Z'_2, ..., Z'_{m-1}) \sim Dir(\alpha'_1, \alpha'_2, ..., \alpha'_{m-1}),$$

where

- $Z'_k = Z_k$ *and* $\alpha'_k = \alpha_k$ *for* $k \in \{1, ..., i-1, i+1, ..., j-1\}$,
- $Z'_i = Z_i + Z_j$, $\alpha'_i = \alpha_i + \alpha_j$, *and*
- $Z'_k = Z_{k+1}$, *and* $\alpha'_k = \alpha_{k+1}$, *for* $k \in \{j, j+1, ..., m-1\}$.

It can be proven that if $Y \sim Dir((\alpha_i)_{i=1}^b)$, then its probability density function is

$$f_Y(q;\alpha) = \frac{\Gamma(\sum_{i=1}^k \alpha_i)}{\prod_{i=1}^k \Gamma(\alpha_i)} \prod_{i=1}^k q_i^{\alpha_i - 1},$$

for all $q \in \Delta_b$ and $f_Y(q;\alpha) = 0$ otherwise, where Γ is the *gamma function* defined as

$$\Gamma(\beta) = \int_0^\infty x^{\beta-1} e^{-x} dx, \quad \forall \beta > 0.$$

When $k = 2$ and $\alpha = (a, b)$, we have

$$f((x,y);\alpha) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}\, y^{b-1}, \qquad \text{for } (x,y) \in \Delta_2,$$

but since $(x, y) \in \Delta_2$, implies that $x + y = 1$, then we can rewrite the previous density function as

$$f(x;\alpha) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}\, (1-x)^{b-1}$$

which is the density function of the Beta distribution of parameters a and b. Sometimes, we write $B(a, b)$ for

$$\frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}.$$

A random variable X is said to have a *gamma distribution with parameters a and b*, if its density function is given by

$$f(x;a,b) = \frac{b^a x^{a-1} e^{-bx}}{\Gamma(a)}, \qquad \text{for } x > 0.$$

Under the same hypothesis of the Proposition A, we have that

Corollary A.0.1. *For* $i \in \{1, ..., m-1\}$,

$$Z'' = (Z_1 + ... + Z_i, Z_{i+1} + ... + Z_m) \sim Dir(\alpha_1 + ... + \alpha_i, \alpha_{i+1} + ... + \alpha_m),$$

which corresponds to a Beta distribution with parameters $\alpha_1 + ... + \alpha_i$ *and* $\alpha_{i+1} + ... + \alpha_m$.

Proposition (and thus, Corollary A.0.1) is a consequence of the following theorem ([10] Theorem 4.1. p. 594):

Theorem A.0.2. *Let* $Y_1, ..., Y_{k+1}$ *be independent gamma random variables with parameters* $a_i > 0$. *Define* $Y = \sum Y_i$ *and* $X_i = Y_i/Y$ *for* $1 \leq i \leq k$. *Then* $(X_1, ..., X_k) \sim Dir(a_1, ..., a_{k+1})$ *and* $(X_1, ..., X_k)$ *is independent of* Y.

Conversely, if Y *is gamma* $(\sum a_i)$, *and* Y *is independet of* $(X_1, ..., X_k) \sim Dir(a_1, ..., a_{k+1})$, *then the random variables* $YX_1, ..., YX_k, Y(1 - \sum_{i=1}^k X_i)$ *are independet gamma random variables with parameters* $a_1, ..., a_{k+1}$.

Finally, by A.0.1 it is clear that

Corollary A.0.2. *If* (P_n) *is a Pólya urn of 2 colors with position vector* $P_0 = (a, b)$, *then*

$$\frac{P_n^{(1)}}{n} \Rightarrow Be(a, b).$$

Appendix B

Upper bound for the product of exponential random variables

A random variable E is said to be *exponentially distributed with parameter* $\alpha > 0$ ($E \sim Exp(\alpha)$) if its density function is equal to

$$f(x;\alpha) = \alpha e^{-\alpha x}\mathbb{1}_{\{x\geq 0\}}, \quad \text{for } x \in \mathbb{R}.$$

This section contains the proof of a useful result. Before proving it, we state the following standard result ([10]).

Lemma B.0.1. *Let $E_1, E_2, ..., E_n$ be an i.i.d. sequence of random variables such that $E_1 \sim Exp(1)$ and let $U_1, U_2, ..., U_n$ is an i.i.d. sequence of uniform random variables in $[0,1]$. Then, for $k < n$,*

$$\left(\frac{\sum_{k=1}^{i} E_k}{\sum_{k=1}^{n} E_k}\right)_{i=1,...,n} =_d (U_{(i)})_{i=1,...,n},$$

where $(U_{(i)})_{i=1,...,n}$ is the order statistics of $(U_1, U_2, ..., U_n)$.

In other words, the previous lemma says that, for any $k < n$, the random variable

$$\frac{\sum_{k=1}^{i} E_k}{\sum_{k=1}^{n} E_k}$$

has the same distribution that the i^{th} smallest number out of n numbers chosen independently and uniformly between 0 and 1.

Now, we can present the main result of this section:

Lemma B.0.2. *Let $E_1, E_2, ...$ be an i.i.d. sequence of random variables such that $E_1 \sim Exp(1)$. Let*

$$X = \prod_{i=1}^{+\infty} \min\left(\sum_{k=1}^{i} E_k, 1\right),$$

then for any $t > 0$,

$$F_X(t) = \mathbb{P}(X \leq t) \leq 6t^{1/4}.$$

Proof. Note that almost surely, there exists l such that

$$\sum_{k=1}^{l+1} E_k > 1,$$

and therefore,

$$\prod_{i=1}^{+\infty} \min\left(\sum_{k=1}^{i} E_k, 1\right) = \prod_{i=1}^{l} \min\left(\sum_{k=1}^{i} E_k, 1\right).$$

Thus, for $t \in (0,1)$,

$$\begin{aligned}\mathbb{P}(X \le t) &= \mathbb{P}\left(\exists l : \sum_{k=1}^{l+1} E_k > 1 \wedge \sum_{k=1}^{l} E_k \le 1 \wedge \prod_{i=1}^{l}\left(\sum_{k=1}^{i} E_k\right) \le t\right) \\ &\le \sum_{l=1}^{+\infty} \min\left\{\mathbb{P}\left(\sum_{k=1}^{l} E_k \le 1\right), \mathbb{P}\left(\sum_{k=1}^{l+1} E_k > 1 \wedge \prod_{i=1}^{l}\left(\sum_{k=1}^{i} E_k\right) \le t\right\}\right). \qquad (B.1)\end{aligned}$$

Note also that

$$\mathbb{P}\left(\sum_{k=1}^{l} E_k \le 1\right) = \int_0^1 \frac{x^{l-1}\exp(-x)}{(l-1)!}dx \le \int_0^1 \frac{x^{l-1}}{(l-1)!}dx = \frac{1}{l!}. \qquad (B.2)$$

On the other side, note that

$$\prod_{i=1}^{l}\left(\sum_{k=1}^{i} E_k\right) = \left(\sum_{k=1}^{l+1} E_k\right)^{l} \prod_{i=1}^{l}\left(\frac{\sum_{k=1}^{i} E_k}{\sum_{k=1}^{l+1} E_k}\right).$$

Therefore, by Lemma B.0.1, we have

$$\mathbb{P}\left(\sum_{k=1}^{l+1} E_k > 1 \wedge \prod_{i=1}^{l}\left(\sum_{k=1}^{i} E_k\right) \le t\right) = \mathbb{P}\left(\sum_{k=1}^{l+1} E_k > 1 \wedge \left(\sum_{k=1}^{l+1} E_k\right) \prod_{i=1}^{l} U_i \le t\right)$$

and consequently, using also the Markov inequality

$$\begin{aligned}\mathbb{P}\left(\sum_{k=1}^{l+1} E_k > 1 \wedge \prod_{i=1}^{l}\left(\sum_{k=1}^{i} E_k\right) \le t\right) &\le \mathbb{P}\left(\prod_{i=1}^{l} U_i \le t\right) \\ &\le \mathbb{E}\left(\frac{\sqrt{t}}{\sqrt{\prod_{i=1}^{l} U_i}}\right)\end{aligned}$$

Since, for any $1 \le i \le l$,

$$\mathbb{E}\left(\frac{1}{\sqrt{U_i}}\right) = \int_0^1 \frac{1}{\sqrt{x}}dx = 2$$

and the random variables U_i are i.i.d., we can conclude that

$$\mathbb{E}\left(\frac{\sqrt{t}}{\sqrt{\prod_{i=1}^{l} U_i}}\right) = 2^l\sqrt{t}.$$

This way

$$\mathbb{P}\left(\sum_{k=1}^{l+1} E_k > 1 \wedge \prod_{i=1}^{l}\left(\sum_{k=1}^{i} E_k\right) \le t\right) \le 2^l\sqrt{t}. \qquad (B.3)$$

Finally, combining B.1, B.2 and B.3 and the fact that $\min(a, b) \leq \sqrt{ab}$ we can see that

$$\mathbb{P}(X \leq t) \leq \sum_{l=1}^{+\infty} \sqrt{\frac{2^l \sqrt{t}}{l!}} \leq 6t^{1/4}.$$

□

Appendix C

An upper bound for a Beta distribution

This section provides a proof of the following useful bound:

Theorem C.0.1. *Given $\eta \in (0,1)$, there exists $K(\eta)$ such that*

$$\mathbb{P}\left(Beta\left(K' - \frac{deg_{PA(K')}(0)}{2}, \frac{deg_{PA(K')}(0)}{2} \right) \leq 1-\eta \right) \leq \eta \qquad \forall K' \geq K(\eta).$$

For now on, we write $deg(0)$ for $deg_{PA(K')}(0)$.

Before proving Theorem C.0.1, we have to present another important theorem [1]:

Theorem C.0.2. *Given $\eta \in (0,1)$, there exists K', such that*

$$\mathbb{P}\left(\max_{0\leq i \leq K'} deg(i) \leq \sqrt{K' \log(K')} \right) \geq 1-\eta.$$

Proof of Theorem C.0.1. Let $K' \in \mathbb{N}$ and define $\kappa = \sqrt{K' \log(K')}$.

Note that

$$\begin{aligned}
&\mathbb{P}\left(\text{Beta}\left(K' - \frac{deg(0)}{2}, \frac{deg(0)}{2} \right) \leq 1-\eta \right)\\
=&\sum_{j=1}^{K'} \mathbb{P}\left(deg(0)=j\right) \mathbb{P}\left(\text{Beta}\left(K' - \frac{j}{2}, \frac{j}{2} \right) \leq 1-\eta \right)\\
\leq &\sum_{j=1}^{\kappa} \mathbb{P}\left(deg(0)=j\right) \mathbb{P}\left(\text{Beta}\left(K' - \frac{j}{2}, \frac{j}{2} \right) \leq 1-\eta \right) + \sum_{j=\kappa+1}^{K'} \mathbb{P}\left(deg(0)=j\right).
\end{aligned}$$

By Theorem C.0.2, we see that for K' large enough,

$$\sum_{j=\kappa+1}^{K'} \mathbb{P}\left(deg(0)=j\right) = \mathbb{P}\left(\kappa+1 \leq deg(0) \leq K'\right) \leq \frac{\eta}{2}. \tag{C.1}$$

On the other hand,

$$\sum_{j=1}^{\kappa} \mathbb{P}\left(deg(0)=j\right) \mathbb{P}\left(\text{Beta}\left(K' - \frac{j}{2}, \frac{j}{2} \right) \leq 1-\eta \right) \leq \max_{1\leq j\leq\kappa} \mathbb{P}\left(\text{Beta}\left(K' - \frac{j}{2}, \frac{j}{2} \right) \leq 1-\eta \right).$$

Define

$$g(\theta) = \left(K' - \frac{j}{2} - 1\right)\log(\theta) + \left(\frac{j}{2} - 1\right)\log(1-\theta), \quad \forall \theta \in (0,1).$$

Since

$$\mathbb{P}\left(\text{Beta}\left(K' - \frac{j}{2}, \frac{j}{2}\right) \leq 1-\eta\right) = \int_0^{1-\eta} \frac{\theta^{K'-j/2-1}(1-\theta)^{j/2-1}}{B(K'-j/2, j/2)}\, d\theta,$$

then we see

$$\mathbb{P}\left(\text{Beta}\left(K' - \frac{j}{2}, \frac{j}{2}\right) \leq 1-\eta\right) = \int_0^{1-\eta} \frac{\exp(g(\theta))}{B(K'-j/2, j/2)}\, d\theta.$$

Finally, using the facts that $\theta_* := \arg\ \max_{0<\theta<1}\ g(\theta) = (1-K'+d/2)/(2-K')$, $1-\theta_* = O(\kappa/K')$, and $g''(\theta) \leq -cK'$, for some positive constant c indepedent of $j \in \{0,1,...,\kappa\}$ (i.e. $c = c(\eta)$), we have that

$$\exp(g(\theta)) \leq \exp(g(\theta_*) - cK'(\theta-\theta_*)^2),$$

and therefore,

$$\int_0^{1-\eta} \exp(g(\theta))d\theta \leq \int_0^{1-\eta} \exp(g(\theta_*))e^{-cK'(\theta-\theta_*)^2}\, d\theta \leq \exp(g(\theta_*))e^{-cK'(\eta - O(\kappa/K')^2)} \leq e^{-cK'}\exp(g(\theta_*)).$$

Besides,

$$\begin{aligned} B\left(K' - \frac{j}{2}, \frac{j}{2}\right) &= \int_0^1 \exp(g(\theta))d\theta \\ &\geq \int_{\theta_*-1/K'}^{\theta_*} \exp(g(\theta))\, d\theta \\ &\geq \frac{1}{K'}\exp(g(\theta_*)). \end{aligned}$$

Thus,

$$\mathbb{P}\left(\text{Beta}\left(K' - \frac{deg(0)}{2}, \frac{deg(0)}{2}\right) \leq 1-\eta\right) \leq K'e^{-cK'},$$

and therefore,

$$\mathbb{P}\left(\text{Beta}\left(K' - \frac{deg(0)}{2}, \frac{deg(0)}{2}\right) \leq 1-\eta\right) \leq \frac{\eta}{2},$$

for K' large enough. This last bound, together with C.1 gives the result. □

Bibliography

[1] Backhausz, A., & Mori, T. F. (2014). Weights and degrees in a random graph model based on 3-interactions. *Acta Mathematica Hungarica,* 143(1), 23-43.

[2] Bai, W. J., Zhou, T., and Wang, B. H. (2007). Immunization of susceptible–infected model on scale-free networks. *Physica A: Statistical Mechanics and its Applications*, 384(2), 656-662.

[3] Bailey, N. T. (1975). *The mathematical theory of infectious diseases and its applications.* Charles Griffin and Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE..

[4] Barabási, A. L., and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439), 509-512.

[5] Bartholomew, D. J., and Bartholomew, D. J. (1967). *Stochastic models for social processes.* London: Wiley.

[6] Bubeck, S., Devroye, L., & Lugosi, G. (2017). Finding Adam in random growing trees. *Random Structures & Algorithms,* 50(2), 158-172.

[7] Bundy, A., and Wallen, L. (1984). Breadth-First Search. In *Catalogue of Artificial Intelligence Tools* (pp. 13-13). Springer Berlin Heidelberg.

[8] Capasso, V., & Capasso, V. (1993). *Mathematical structures of epidemic systems* (Vol. 88). Berlin: Springer.

[9] Chierichetti, F., Lattanzi, S., and Panconesi, A. (2009). Rumor spreading in social networks. In *International Colloquium on Automata, Languages, and Programming* (pp. 375-386). Springer Berlin Heidelberg.

[10] Devroye, L. (1986) *Non-Uniform Random Variate Generation.* Springer-Verlag New York.

[11] Kay, S. M. (1993). *Fundamentals of statistical signal processing.* Prentice Hall PTR.

[12] Kermack, W. O., & McKendrick, A. G. (1932, October). Contributions to the mathematical theory of epidemics. II.—The problem of endemicity. *In Proc. R. Soc. Lond. A* (Vol. 138, No. 834, pp. 55-83). The Royal Society.

[13] Levin, D. A., and Peres, Y. (2017). Markov chains and mixing times (Vol. 107). American Mathematical Soc..

[14] van Lint, J. H., and Wilson, R. M. (2001). *A course in combinatorics.* Cambridge university press.

[15] Millar, R. B. (2011). *Maximum likelihood estimation and inference: with examples in R, SAS and ADMB* (Vol. 111). John Wiley & Sons.

[16] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.

[17] Serazzi, G., and Zanero, S. (2004). Computer virus propagation models. In *Performance Tools and Applications to Networked Systems* (pp. 26-50). Springer Berlin Heidelberg.

[18] Shah, D., and Zaman, T. (2011). Rumors in a network: who's the culprit?. *IEEE Transactions on Information Theory,* 57(8), 5163-5181.

[19] Shah, D., and Zaman, T. (2012). Finding rumor sources on random graphs. *arXiv preprint arXiv:1110.6230.*

[20] Streftaris, G., and Gibson, G. J. (2002). Statistical inference for stochastic epidemic models. In *Proc. of the 17th Int'l Workshop on Statistical Modelling. Chania* (Vol. 609, p. 616).

[21] Teixeira, A. (2016) *Notas de aula Probabilidade I.*

[22] Watts, D. J., and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *nature*, 393(6684), 440-442.