Exploring and Analyzing the Tor Hidden Services Graph

Massimo Bernaschi
Institute for Applied Computing (IAC-CNR),
Via dei Taurini 19, Rome, Italy
massimo.bernaschi@cnr.it

Alessandro Celestini
Institute for Applied Computing (IAC-CNR),
Via dei Taurini 19, Rome, Italy
a.celestini@iac.cnr.it

Stefano Guarino
Institute for Applied Computing (IAC-CNR),
Via dei Taurini 19, Rome, Italy
s.guarino@iac.cnr.it

Flavio Lombardi
Institute for Applied Computing (IAC-CNR),
Via dei Taurini 19, Rome, Italy
flavio.lombardi@cnr.it

Abstract

The exploration and analysis of Web graphs has flourished in the recent past, producing a large number of relevant and interesting research results. However, the unique characteristics of the Tor network limit the applicability of standard techniques and demand for specific algorithms to explore and analyze it. The attention of the research community has focused on assessing the security of the Tor infrastructure (*i.e.*, its ability to actually provide the intended level of anonymity) and on discussing what Tor is currently being used for. Since there are no foolproof techniques for automatically discovering Tor hidden services, little or no information is available about the topology of the Tor Web graph. Even less is known on the relationship between content similarity and topological structure. The present paper aims at addressing such lack of information. Among its contributions: a study on automatic Tor Web exploration/data collection approaches; the adoption of novel representative metrics for evaluating Tor data; a novel in-depth analysis of the hidden services graph; a rich correlation analysis of hidden services' semantics and topology. Finally, a broad interesting set of novel insights/considerations over the Tor Web organization and content are provided.

***K*eywords** Web graphs, Network topology, Automatic Web exploration, Correlation analysis

1 Introduction

The Onion Router (Tor [1]) network is an anonymous communication network leveraging an implementation of the *onion routing* protocol [1]. Composed of a network of volunteer *routers*, Tor allows its users to access the Internet anonymously, evading traditional network surveillance and traffic analysis mechanisms. Tor can also provide anonymity to servers that offer various kinds of services (*e.g.*, websites) without revealing any network-level information about their locations. Such servers, configured to receive inbound connections only through Tor, are called hidden services and can only be accessed using a Tor-enabled browser. A hidden service is identified by its *onion url* and it is not associated with any (visible) IP address. Tor is able to interpret such urls and forward data packets to and from a hidden service, guaranteeing anonymity in both directions. Tor hidden services are part of the *dark Web*, *i.e.*, a subset of the *deep Web*.

[1] https://www.torproject.org/

In recent years, Tor attracted significant attention from both society and media, as well as from the research community. The main reason of such interest is that a completely anonymous network represents a perfect breading ground for illegal activities. Tor hidden services have been accused to provide protection to terrorists [2], and are known to host marketplaces for drugs, weapons and pedo-pornography [2]. Research work about Tor focused on two main goals: (i) assessing its actual level of anonymity, searching for Tor flaws and weaknesses, designing effective attacks and proposing possible countermeasures [3, 4, 5]; (ii) analysing and discussing the thematic organization of Tor hidden services and how their contents relate to their popularity, in order to understand to which extent Tor can be associated with illegality [6, 7].

Even though the analysis of the *surface* Web [3] graph has flourished in the past, producing a plethora of relevant research results, we could not find any similar result/work for the Tor Web. As a consequence, little or no information over the structure of the Tor hidden services graph is known. Furthermore, to the best of our knowledge, the literature lacks any results that relate the topological characteristics of the Tor Web to its security and to information extraction and content analysis of such a relevant and potentially dangerous network.

The present paper aims at addressing such deficiency of the Tor literature by collecting, analysing and discussing the hidden services graph using state-of-the-art techniques. Present work contributes as follows, by providing: (i) a survey over the possible Tor Web exploration approaches and their limitations; (ii) the adoption of a relevant subset of metrics for evaluating actual Tor hidden services data; (iii) a novel in-depth investigation over the Tor Web topology; (iv) an in-depth analysis of the relationship between the topics found in (English) Tor pages and the Tor Web topology.

The paper is organized as follows: Section 2 critically surveys related work, discussing both Tor and the surface Web; Section 3 details the objectives for this work, and motivates the choice of metrics; Section 4 presents the methodology we followed in our investigation; further, Section 5 discusses the collected experimental data; finally, Section 6 draws final conclusions and gives hints for future work.

2 Background and Related Work

The Tor network has been designed as a low-latency, anonymity-guaranteeing and censorship-resistant network. For the last ten years, Tor has attracted significant attention from the research community, interested both in assessing its security with respect to de-anonymization attacks, and in understanding which threats a publicly available anonymous communication system could expose society to. In the following, we provide an overview of what Tor is and how it works. Then, we summarize the main findings of previous research work concerning both Tor and the structure of the surface Web.

2.1 About Tor

The purpose of the Tor network is twofold: on the one hand, it makes it possible to access the Internet anonymously, guaranteeing resilience against standard network surveillance and traffic analysis techniques; on the other hand, it allows running anonymous and untraceable Web servers, called *hidden services*, that can be accessed only using a specific Tor-enabled browser. Tor anonymity is based on an implementation of the onion routing protocol [1], requiring each data packet to undergo multiple layers of encryption at the application layer of the communication protocol stack. Such layers are stripped off, one by one, along the route that connects the source to the destination, so that each router along the path is only aware of the previous and following step in the path. Tor routers are often referred to as *relays* and are managed by volunteers over the Internet. Each relay is identified by a long-term public signing key, and the list of all available relays is stored by special nodes of the network called *directory authorities*, endowed with their own directory signing key. The security of Tor therefore relies on cryptography at three different levels: (i) encryption for data privacy within Tor, (ii) authentication between clients and relays, and (iii) integrity/authenticity of the list of relays.

[2] http://www.bloomberg.com/news/articles/2014-10-15/how-anti-is-spies-fight-terrorism-with-digital-tools

[3] A term for the traditional WWW that is visible to everybody.

Tor Web usually indicates the network made of all Tor hidden services and their mutual hyperlinks, not to be confused with the network of Tor relays. The mechanism by which a client connects to a hidden service can be briefly described as follows. Each hidden service is identified by its *onion url*, a 80-bit excerpt of the SHA-1 hash digest of its public key. First, a hidden service advertises its existence by communicating its public key to a set of randomly picked relays, called *introduction points*. Next, the hidden service publishes a signed *descriptor*, containing its public key and its set of introduction points. This descriptor is stored in a distributed hash table, formed by special relays called *HSDirs*. To contact a hidden service, its onion url is used by the client to download the descriptor and thus learn the hidden service's introduction points and public key. Communication between client and hidden service takes place through secure circuits (relying on the aforementioned onion routing protocol) to a commonly known relay, known as *rendezvous point*. The rendezvous point is chosen by the client and communicated to the hidden service via one of the introduction points. As observed in [8], Tor effectiveness strongly depends on the choice of the relays and on their reliability and trustability, with special regard to the first and last relays in a Tor circuit (*entry guard* and *exit node*), which, being close to the source and to the destination, have access to the most sensitive information. However, directory authorities, in charge of distributing an always up-to-date list of valid and trusted relays, are actually responsible for the entire Tor network. Therefore, Tor uses a fixed set of 9 *trusted* directory authorities [4], whose list is hardcoded into each Tor client.

Searching through the Tor Web is not possible using standard approaches, since Tor hidden services are not indexed by traditional search engines like Google or Bing. Specific Tor search engines exist, accessible either through the surface Web (*e.g.*, Ahmia [5], Onion.link [6]), or through Tor only (*e.g.*, DuckDuckGo [7] TORCH [8]), but they are not likewise reliable, mostly because Tor is very volatile and not as connected as the surface Web. Even the size of the Tor Web can be hardly estimated, despite a few crawling attempts cited in the following.

2.2 On Tor Flaws

Most research work about Tor aims at either inding flaws and weaknesses of its anonymization scheme [8], or designing attacks [4],[5] and proposing effective counteractions [9],[3]. Perry *et al.* introduce TorFlow [8], a suite of tools for measuring the reliability, capacity and integrity of Tor routers. TorFlow aimed at making such measurements available to the Tor directory authorities, responsible for providing to Tor clients the list of available relays. In [10], the authors observe that a malicious router operator can counterfeit its information to attract tunnels for compromise. To address similar attacks, they propose an opportunistic bandwidth measurement algorithm to replace selfreported values, and a mechanism to let Tor users trade-off better performance against stronger anonymity. In [3], the authors present StegoTorus, a tool that comprehensively disguises Tor from protocol analysis. To foil analysis of packet contents, Tor traffic is steganographed to resemble an innocuous cover protocol, such as HTTP. To foil analysis at transport level, the Tor circuit is distributed over many shorter-lived connections with per-packet characteristics that mimic cover-protocol traffic. Byrukov [4] shows that it is possible to attack arbitrary hidden services by carefully analysing the HSDir selection process. By running a malicious HSDir, he effectively implemented an attack that allows to measure the popularity of a hidden service, take it down, and even deanonymize it. Arp and al [5], describe a deanonymization attack against Tor called TorBen. Such an attack is based on two notable weaknesses: Web pages can be easily manipulated to load content from untrusted origins, and, despite encryption, low-latency anonymization networks cannot effectively hide the size of request-response pairs. TorBen is a side channel attack, allowing short Web page markers to be transmitted to expose the Web page a user visits over Tor. In an empirical evaluation with 60,000 Web pages, TorBen was able to detect these markers with an accuracy of over 91% and no false positives.

[4] This number refers to the date this paper was submitted.

[5] https://ahmia.fi

[6] https://onion.link

[7] http://3g2upl4pq6kufc4m.onion

[8] http://xmh57jrzrnw6insl.onion

2.3 On Tor Usage

The other main research direction over Tor infrastructure consists in studying if and how Tor is being (mis)used, and whether the accuses of providing protection and anonymity to criminals are well deserved or not. McCoy *et al.* [11] show an analysis obtained by taking part to the Tor network, with the primary goals of gaining a better understanding of how Tor was being used and by whom. They show that Web traffic makes up the majority of the connections and bandwidth, but non-interactive protocols consume a disproportionately large amount of bandwidth when compared to interactive protocols. Further, they develop a method for detecting exit router logging, and present evidence that Tor is used throughout the world, but router participation is limited to only a few countries. Spitters *et al.* [7] focus more on the contents of Tor hidden services. They apply classification and topic model-based text mining techniques to the contents of over a thousand Tor hidden services in order to model their thematic organization and linguistic diversity. Their results indicate that most hidden services exhibit illegal or controversial contents in their data set. Along the same line, Biryukov *et al.* [6] give an overview of Tor hidden services, studying 39824 hidden service descriptors collected in 2013, and analyzing and classifying their contents. At the time of the crawl, they were able to connect to 7114 destinations, but half of them were excluded because inappropriate for classification, ending up with 3050 destinations. Their results show that resources devoted to criminal activities (drugs, adult content, counterfeits, weapons, etc.) constitute 44%, whereas the remaining 56% are devoted to a number of different topics: "Politics" (9%) and "Anonymity" (8%) are among the most popular. Biryukov *et al.* also estimate the popularity of hidden services by looking at the request rates for hidden service descriptors by clients. They find that, while the content of Tor hidden services is rather varied, the most popular hidden services are related to botnets. While Biryukov *et al.* make use of topic modeling to analyse the content of the pages they examine, Owen and Savage [12] analysed hidden services' contents resorting to manual classification. In our present paper we adopt a different approach leveraging the reliable and well-known Cogito [9] semantic engine tool. This choice is motivated by the need to efficiently and effectively relate content and topological information. In addition, the reliability of the Cogito tool had been already proved during the IANCIS project [10]. A recent study carried out by Soska and Christin [13] on Tor marketplaces shows interesting evolutionary properties of the anonymous marketplace ecosystem. The authors perform a specific Tor Web crawling, collecting data from 16 different marketplaces over more than two years, without focusing on a specific category of products. The results of their study suggest that marketplaces are quite resilient to law enforcement take-downs and large-scale frauds. They also evidence that the majority of the products being sold belong to drugs category.

2.4 On Web Exploration and Topology

To the best of our knowledge, no research work has ever discussed the properties of the Tor Web graph, obtained associating a vertex to each hidden service or page, and an edge to each hyperlink between them. Some interesting initial work exists, such as Jansen's [14] aimed at building a simplified model of the Tor network. Jansen's toy Tor network model is used to perform simple small-scale simulations by leveraging existing Tor experimentation tools, namely Shadow [15], and ExperimenTor [16].

Over the last 20 years similar studies of the common *surface* Web graph allowed, among other things, to optimize search algorithms [17],[18] and Web data extraction [19], to find efficient compressed representations [20],[21],[22], to propose new suitable random graph models [23],[24], and to improve contents analysis [25]. In [26], the authors describe two algorithms for improved Web search and automatic community discovery, respectively. They report a number of measurements obtained while running such algorithms, that provide the first publicly known portrait of the Web graph. Along the same line, the authors of [27] report on experiments on local and global properties of the Web graph using two Altavista crawls each one with over 200 million pages and 1.5 billion links. The interest raised by such seminal papers suggested further rigorous studies on Web metrics, surveyed in [28]. New techniques introduced later allow to model graph structures, and to extract relevant measurements and information about the topology of the corresponding graph. Particular attention is dedicated to graph clustering, community detection, and community evaluation measures, well surveyed in [29]: communities often emerge spontaneously as sets of highly related

[9] `http://www.expertsystem.com/cogito`

[10] `http://www.iancis.eu`

pages with similar contents [25]. The other main topic is query processing: Google's search engine is based on a well-known Web graph metrics called PageRank [30], and query processing based on PageRank and similar metrics rely on ranking algorithms explicitly designed to be implemented over suitable representations of Web graphs, as discussed in [18]. Efficient representation of Web graphs is fundamental, due to the ever increasing size of the Web. In several papers [20],[21],[22], authors focus on this problem, agreeing that the ability to compress and easily navigate Web graphs depends on the topology of the graph, and on precise metrics able to capture its inner redundancy and to describe the existing correlation among adjacency lists of different vertices. Many researchers discuss the possibility of generating synthetic graphs, having the same properties of real-world Web graphs. Such models allow running simulations and test algorithms. The authors of [26], in fact, underline that traditional random graph models (*e.g.*, the well-known Erdős-Rényi model [31]) do not exhibit many properties emerged from their study of the Web graph. To provide better tools for further analysis of graph algorithms on the Web, they propose a new family of synthetic graph models based on an edge copying process. In [23], the authors speculate that the limits of traditional models are mainly caused by two design choices: the usage of a static set of vertices, and the independence of the edges. Therefore, they propose a new model where edges are statistically dependent and new vertices are added as time evolves, and they verify that graphs generated according to this model present many similarities with Web graphs. Many more models where introduced in the following years, and are mostly surveyed in [24].

In summary, it is clear that Tor represents a very interesting research topic, and that an accurate analysis of structural and topological properties of Web graphs is instrumental in reaching a clear understanding of their functioning, and of the behavior of exploration and data collection algorithms. Yet, no similar study has ever been performed on the Tor Web graph. The purpose of this paper is exactly to make a first step in order to fill this gap.

3 Analyzing the Tor graph

In this Section, we discuss how to properly characterize the Tor Web graph. In Section 3.1 we identify the main goals of our analysis. Such goals motivate the choice of suitable metrics, presented in Section 3.2. Finally, in Section 3.3 we briefly discuss how to formally describe the contents (*i.e.*, text semantic) of a Web page.

3.1 Goals

The purpose of our analysis is to gather information concerning the topology of the Tor Web. In addition, we aim at identifying potential relationships among topological properties and hidden services' contents. Since little or no information is available about the topology of the Tor Web graph, it is reasonable to start by measuring a set of general features of the graph. It is worth noticing that exploring the entire Tor Web is not feasible/practical for a number of reasons (it can be unstructured, not linked by any other page, or volatile/temporary; furthermore, relaying timeouts often occur) as shown by past attempts [11, 6]. Consequently, we will look at the portion of the Tor Web that is accessible from the surface Web, as in most other previous related studies [13]. We will particularly focus on measurements that allow us to verify the following hypotheses:

H1 We expect the Tor Web to be very topic-oriented, with most hidden services focusing on a specific topic, and only a few hubs, mostly marketplaces, forums, wikis, or link directories, that can be related to several different topics.

H2 Most hidden services are managed by single entities that offer services related to a specific topic. Hence we expect to find many pages with only a few hyperlinks, if any, to other pages controlled by the same entities.

H3 The hubs, that advertise and link to a large number of other pages, are much more likely to be known and accessible from the surface Web. This means that we expect to find most hubs among the root set used to seed our crawler (see Section 4.2).

H4 Considering all previous assumptions, we expect the topology of the Tor Web graph to be very similar to a "forest" of directed extended stars, where each weakly connected component is likely to coincide with the spanning tree obtained by a breadth first search started from, and rooted into, one of the seeds.

H5 The pages where most dangerous illegal activities are discussed or promoted are probably the hardest to access for the average user and for crawlers, due to the obvious attempt to conceal them.

3.2 Metrics

Let us briefly recall the definition of some useful metrics (see also [32]) that will be used later in the paper to characterize the topology of the Tor Web graph.

Degree distribution: the degree distribution is the statistical distribution of the number of edges per vertex (vertex degree). In undirected graphs, we define the fraction p_k of nodes of a graph of degree k, for all $k \geq 0$. When dealing with directed graphs, we need to define two different quantities p_k^{in} and p_k^{out} for the in-degree $\deg_{in}$ and out-degree $\deg_{out}$, respectively. While these two quantities can be separately studied, the study of their joint distribution usually provides more information about the structure of a network. In that case, we define $p_{k_{in},k_{out}}$ as the fraction of vertices of in-degree k_{in} and out-degree k_{out}, for all pairs (k_{in}, k_{out}). The degree distribution is one of the most studied characteristics of a graph [26, 27, 33, 34], as it provides a clear insight into its structure.

Centrality: the *centrality* of a vertex measures its importance in the graph, based on the rationale that the relevance of a node depends on how much the structure of the graph changes when we remove that vertex. Several centrality metrics have been proposed in the literature, such as the well-known *eigenvector centrality* [35], Katz centrality and Google's PageRank [30], to cite a few. We will focus on the most used *median* centrality metrics, the *betweenness centrality BC* [36], defined as the fraction of shortest paths between any two vertices of the graph that pass through the vertex considered. A vertex has high betweenness centrality if it takes on great importance in the transfer of information or objects in the network.

Connected components: the connected components of an undirected graph are defined as the equivalence classes induced by the adjacency relation. For directed graphs, there exist two different notions of connected components: *weakly* connected components (WCCs) are the equivalence classes induced by the relation "u is adjacent to v *or* vice versa", whereas *strongly* connected components (SCCs) are induced by the relation "u is adjacent to v *and* vice versa". When dealing with the structure of a graph, it is fundamental to find its number of components and their size, and to detect the presence of giant and/or small components. In our case, it is also important to compare the decomposition obtained by identifying connected components with that induced by content similarity.

3.3 Semantic Analysis

To implement a large scale semantic analysis, we had to rely on an automated semantic engine. A semantic engine can be fed with unformatted text, and can extract the meaning of that text, returning (partially) structured data that summarizes its content. Current semantic engines are not only capable of executing simple tasks such as detecting a document's language or recognizing mentions to known entities (*e.g.*, famous people, organizations, places), but they can also map documents to a set of predetermined categories, find related documents, or even extract complex knowledge assertions (*e.g.*, the fact that a company is buying another one for a specific amount).

In the present paper, we are mainly interested in two types of semantic analyses. On the one hand, we want to analyse the content of a few specific pages that play a relevant role in the Tor Web graph, such as those having highest in-degree, highest out-degree, or highest betweenness centrality score. Actually, that task can be easily performed manually, as it is limited to a small number of pages. On the other hand, we want to assess the relation between contents and connectivity, *e.g.*, verifying whether tightly connected graph regions correspond to set of pages concerning similar topics. In this case, a semantic analysis tool is required, because we need to analyse thousands of pages and extract structured and comparable/measurable information about their contents. To that purpose, we selected the Cogito semantic engine developed by Expert System, a company specialized in semantic technologies [11]. We chose Cogito, because it provides several out-of-the-box semantic intelligence features, such as categorization, text mining, fact mining, and entities extraction.

[11] http://www.expertsystem.com

Cogito receives as input an unformatted text, and returns a structured document (XML or JSON) that contains the result of the semantic analysis of the input [12]. However, within the scope of the present paper we are only interested in Cogito's ability to suggest meaningful tags from a user defined taxonomy, or, more precisely, to assert the relevance of each category of the taxonomy with respect to the text under analysis. Therefore, we developed an application that "filters" Cogito's output, dropping any information not related to categorization. The result of our analysis is represented by a *semantic vector* S of size n, where n is the number of categories of the taxonomy, and $S[i]$ is the relative score of category i.

The taxonomy we use has been defined within the scope of the "IANCIS" ISEC Project [13], and it is based on previous works [7, 6] that discuss the thematic organization of Tor hidden services. The authors of [7] apply classification and topic modelling-based text mining techniques to the contents of over a thousand hidden services. To summarize their findings, they provide a taxonomy graph. The graph shows that most relevant topics are related to illicit or possibly illicit activities (mostly drugs, (child) pornography, cyber crimes, counterfeit, terrorism, weapons, racism), with only a few exceptions (business, technology, media, politics). Along the same line, the authors of [6] analyze and classify the contents of 3050 hidden services. Their results show that resources devoted to criminal activities (again, mostly drugs, adult content, counterfeits, weapons) constitute 44%, whereas the remaining 56% are devoted to a number of different topics, with politics (9%) and anonymity (8%) being the most popular. For our classification, we selected 17 categories, as a trade-off among the above findings, the objectives of the "IANCIS" Project, and the domain ontologies developed by our partner Expert System that develops and sells the semantic engine *Cogito*. The rationale was to cover a wide range of topics, from media and IT services to common felonies and terrorism, with a focus on criminal activities that have been frequently associated to dark networks. The selected topics are the following:

- Cyber attack
- Cyber deception
- Cyber security
- Fraud
- Gambling
- Illicit drugs
- Illicit trafficking in weapons, munitions, and explosives
- Information systems
- IT services companies
- Media companies
- Murder, grievous bodily injury
- Racism and xenophobia
- Rape
- Sexual exploitation of children and child pornography
- Social networks
- Terrorism
- Weaponry

In the literature, alternative approaches, mostly based on topic modeling, have been successfully implemented for the analysis and classification of Tor hidden services [7]. However, we believe it is an advantage to use a well-established product whose performance can be considerably refined thanks to the careful work of a group of linguists, that allows to specify in advance a list of categories of interest for us (partly based on previous findings [7, 6]). It is important to take into account that the ontologies developed by Expert System for the "IANCIS" Project, and herein used, only consider English. As a consequence, we filtered out all non English text found before feeding it to Cogito. Considering other languages is a viable direction for future activities.

We highlight that the semantic vectors can be easily used to assess the similarity of two pages. Intuitively, two pages p_1 and p_2 are *semantically equivalent* if their vectors S_{p_1} and S_{p_2} coincide. More generally, a natural way to compare the content of two pages is to consider the *cosine similarity* [14] of their vectors, that is, the cosine of the angle between the two vectors. For instance, assume that our taxonomy consists of only two categories, "food" and "drugs". If the text contained in page p_1 can be associated with food but not with drugs (*i.e.*, page p_1 somehow talks about food but not about drugs), the semantic vector of page p_1 will be $S_{p_1} = (1, 0)$. Now, if the same holds for page p_2, we have $cosine(S_{p_1}, S_{p_2}) = cosine((1, 0), (1, 0)) = \cos(0) = 1$, which means that the two pages are semantically equivalent. If p_2 talks about food as much as it talks about drugs, we

[12] See http://www.intelligenceapi.com/demo/ for a demo of Cogito's functioning.

[13] DG Home Affairs ISEC Programme 2013 - Project IANCIS "Indexing of Anonimous Networks for Crime Information Search", 2014-2016, GA n. HOME/2013/ISEC/AG/INT/4000005222 - www.iancis.eu

[14] https://en.wikipedia.org/wiki/Cosine_similarity

Table 1: Number of nodes and arcs of the page (PG), host (HG), and service (SG) graphs.

Graph	# Nodes	# Arcs
Directed Page Graph (PG)	918885	20446513
Undirected Page Graph (PG*)	918885	17963147
Directed Host Graph (HG)	5420	65716
Undirected Host Graph (HG*)	5420	64379
Directed Service Graph (SG)	5144	60688
Undirected Service Graph (SG*)	5144	59492

have $S_{p_2} = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$, so that $cosine(S_{p_1}, S_{p_2}) = cosine((1,0), (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})) = \cos(\frac{\pi}{4}) = \frac{\sqrt{2}}{2}$. Finally, if p_2 talks about drugs and not about food, we have $S_{p_2} = (0,1)$, so that $cosine(S_{p_1}, S_{p_2}) = cosine((1,0),(0,1)) = \cos(\frac{\pi}{2}) = 0$, which means that there is no similarity at all between the content of the two pages. Since we are considering only vectors with non negative components, the angle between two vectors cannot be larger than $\frac{\pi}{2}$, hence the cosine similarity is a perfect normalized metrics of similarity. In Section 5.2 we will discuss more in details how to analyse the semantic of several pages so as to infer general information about the graph and the relation between topological and semantic aspects.

4 Dataset and Tor Graphs

To collect all data needed for our analysis we made use of the BUbiNG crawler [37] and leveraged the Polipo [15] HTTP proxy. We chose BUbiNG as it is a high-performance, scalable, distributed and open-source crawler [16].

We extracted the Tor graph from the data we gathered in the crawling phase. We analysed only onion links, we did not add to the graph nodes in the surface Web or arcs from/to the surface Web, possibly linked by onion pages. We built three different graphs: a Page Graph (PG), a Host Graph (HG), and a Service Graph (SG). In the Page Graph each node represents a page and each arc represents the existence of one or more hypertextual links between pages. In the Host Graph each node represents the set of pages belonging to a single host, *i.e.*, any Tor domain or subdomain, and each arc represents the existence of one or more hypertextual links between hosts, *i.e.*, the existence of one or more hypertextual links among pages belonging to different hosts. Finally, in the Service Graph, each node represents the set of pages belonging to a hidden service, *i.e.*, a Tor domain identified by a sequence of 16 characters (base32 encoded). Each arc represents the existence of one or more hypertextual links among hidden services, *i.e.*, the existence of one or more hypertextual links among pages belonging to different hidden services. As better described later in Section 5, for all three graphs we built both a *directed* version, denoted PG, HG, and SG, respectively, and a *undirected* version, denoted PG*, HG*, and SG*, respectively. Table 1 shows the size of each graph in terms of number of nodes and arcs.

4.1 The Crawling Process

In BUbiNG, a pool of software agents are responsible for both exploring the Web and collecting (and partially elaborating) the data. Such agents work in parallel, each handling in turn several threads. According to its default configuration, BUbiNG implements a *breadth-first-search*, and we did not modify this aspect. However, to gain an insight about the *depth* of our crawling, we measured the diameter of the undirected version of each graph and the eccentricity of vertices from the root set, see Table 2 and Table 3. In our experiments, BUbiNG was mainly used as follows:

- A predetermined set of hidden services, the *root set*, is inserted in an *url list*.
- The first *fetching thread* available extracts the first onion url from the list and exports the content of the corresponding hidden service in main memory.
- The first *parsing thread* available analyses that content aiming at extracting new onion urls to visit.

[15] `http://www.pps.univ-paris-diderot.fr/~jch/software/polipo/`

[16] Developed by the Laboratory for Web Algorithmics (LAW) at the Computer Science Department of the Università degli studi di Milano

Table 2: Diameter of the undirected page (PG*), host (HG*), and service (SG*) graphs

Graph	Diameter
Undirected Page Graph (PG*)	95
Undirected Host Graph (HG*)	4
Undirected Service Graph (SG*)	4

Table 3: Eccentricity of selected vertices from the root set, for the undirected page (PG*), host (HG*), and service (SG*) graphs

Node	Eccen. (PG*)	Eccen. (HG*)	Eccen. (SG*)
1	66	3	3
2	66	3	3
3	66	3	3
4	66	3	3
5	66	3	3
6	66	3	3
7	66	3	3
8	65	3	3
9	66	3	3
10	66	3	3
11	65	3	3
12	66	3	3

- The new onion urls found are passed to a *sieve* able to verify whether such urls were already visited before.
- If so, the urls are discarded, otherwise they are added in the tail of the url list.
- *Fetching threads* attempt to contact each url for a maximum of three times, after that the url is considered not available.

The url list is continuously updated with the new urls identified, and it is always kept ordered in the same order in which the urls are found. In other words, the list is handled with a simple first-in-first-out approach. We used also several filters to crawl only useful resources, *e.g.*, only onion addresses are followed by the crawler and media files (images, videos, *etc.*) are not downloaded. Including the outer border of the Tor Web, *i.e.*, pages of the surface Web that link to or are linked by Tor pages could be interesting, but falls beyond the scope of the present work and will be considered as a possible subject of future work.

If the Web graph is strongly connected, the composition of the root set does not affect the final outcome of the crawl, though it may affect its speed. However, without any legitimate assumption on the topology of the Tor Web graph, our analysis is potentially influenced by the choice of the root set, as we discuss in the next Section.

We ran the crawler for about six weeks, and given the high latency of Tor network, we set the connection and socket timeout to 360 seconds. At the end of the process our dataset contained 1119048 records, thus the crawler tried to connect to 1119048 urls. Only 824324 urls replied with a success HTTP status code (200), whereas 94561 urls replied with a 3xx status code [17], 96816 urls with a 4xx status code and 103347 urls with a 5xx status code.

4.2 On the Root Set Choice

One of the main goals of the present work is to provide the first characterization of the Tor Web graph, so we are faced with the problem of how to guarantee a fully representative exploration of the Tor Web, *i.e.*, how to visit all, or at least a large enough set of, active hidden services. The crawler finds a hidden service if and only if that hidden service can be reached, only following hyperlinks, from one of the elements of the root set. Since many hidden services could deliberately try to conceal themselves by limiting the number of ingoing hyperlink, we argue that a complete

[17] A status code 3xx is not actually an error code, since these codes are related to Web redirection.

crawl of the Tor Web is only possible supporting the crawler with some sort of brute force searcher, that continuously feeds the url list of the crawler with new hidden services found by trying to access random onion urls.

To verify the feasibility of a similar approach, we developed a test application that continuously checks random hidden services' urls. As a matter of fact, random, but plausible urls can be easily generated, as onion urls are *de facto* 80-bit hash digests (in their readable form, 16 5-bit characters). Our application hence uses m independent threads working in parallel as follows:

- Each thread computes a hash-chain, starting from a given seed.
- For each hash digest h generated, the thread translates the 80 bits of h into a 16 5-bit characters word $word(h)$ and tests the reachability of the hidden service $word(h).onion$.
- The thread periodically writes information about the status of the search on a file named $serviceURL\_info.csv$. The information written are: the number of hash computed since application launch, the number of hash computed since last valid hidden service found, the number of valid hidden services found, and the date.

The result of our test was quite unsatisfactory: our application ran for 1 month, trying to reach approximately 1 million (2^{20}) different onion urls, and finding 0 active hidden services. Actually, with a similar number of trials, it is perfectly possible to find no active hidden service. The Tor Project estimates that about 30000 hidden services announce themselves to the Tor network every day [18], so the chance to find an active hidden service by randomly selecting an onion url is approximately 2^{-65}. What was surprising is that the application was able to check only 2^{20} urls in a whole month. The reason is that the latency of the Tor network is indeed much higher than expected, and the connection timeout results the bottleneck of any similar brute-force attempt. Our test hence led us to the conclusion that a brute-force search of active hidden services is unfeasible.

Hence, in our analysis we fed BUbiNG only with a (large) set of hidden services found "by hand" in the surface Web, mostly wikis and link directories, like "The Hidden Wiki" page. With "by hand" we mean that we merged together several lists of hidden services obtained using both standard (*e.g.*, Google) and Tor-specific (*e.g.*, Ahmia) search engines. In particular, our root set contains the complete list of hidden services indexed by Ahmia at the time our crawling activity started. Despite its simplicity, we have at least two good reasons to assert that our approach is effective. On the one hand, the quality of our root set is supported by the numbers of our crawl: the amount of hidden services we were able to analyse is comparable to those used in previous similar attempts in the literature [7, 6]. On the other hand, the way our root set was composed guarantees several advantages in terms of *what* portion of the Tor Web we explored. First, we focus on the portion of the Tor Web that is accessible to "common" users [19]. While such a portion can be expected to exhibit interesting topological characteristics, the least connected, most isolated and (probably) most volatile hidden services are not very interesting from this point of view, and their presence in the sample set risks to water down our analysis. Further, only using a root set of onion urls available on the surface Web allows us to measure the size of the "public" part of the Tor Web and compare it with the estimated total size. Even if limited to a sector of the Tor Web, our results pave the way for a characterization of the Tor Web graph. The possibility to analyse and compare other approaches to extend the root set is the first direction we identify for future work in the area.

5 Experiments, Analysis, and Discussion

Hereafter, we present and discuss the findings emerged from the analysis we carried out over the set of hidden services collected by our crawl. More precisely, we consider three graphs that describe the Tor Web at three different layers:

Page Graph (PG) This is the graph built on the relations among the visited Web pages. This graph describes Tor at the lowest level, associating a vertex to each Tor page and an edge to each hyperlink.

[18] https://blog.torproject.org/blog/some-statistics-about-onions

[19] Our notion of a common user includes even advanced users that know how to run a crawler, but not users who are given the url of a hidden service by the owner of that service.

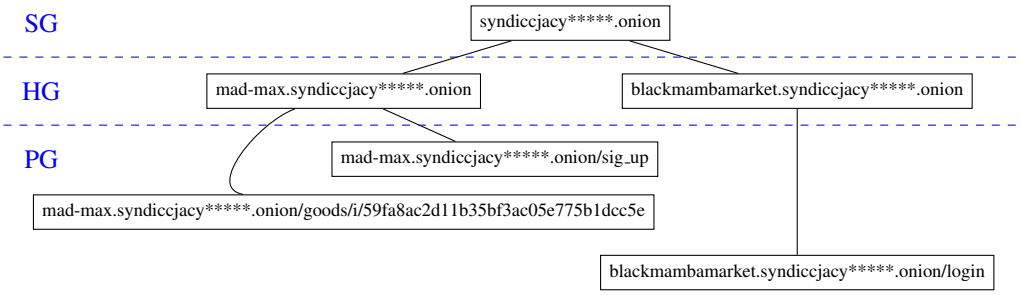


Figure 1: An example of graph construction

Host Graph (HG) This is the graph composed of the relationships among Tor hosts. In this graph, vertices correspond to domains and subdomains of the Tor Web, with an edge between host A and host B if there is at least one hyperlink between a page of A and a page of B.

Service Graph (SG) This is the graph summarizing/clustering the above information on a per-service basis. This graph is the highest level representation of the Tor Web, in which vertices correspond to Tor hidden services, and an edge connects service A to service B if there is at least one hyperlink between a page of A and a page of B.

To further clarify the differences among these three graphs, let us consider the following example, depicted in Fig. 1. We found that the hidden service

S `syndiccjacy*****.onion`

accommodates two different subdomains:

H_1 `mad-max.syndiccjacy*****.onion`;
H_2 `blackmambamarket.syndiccjacy*****.onion`.

In turn, `mad-max.syndiccjacy*****.onion` hosts two pages:

$P_{1,1}$ `mad-max.syndiccjacy*****.onion/sig_up`;
$P_{1,2}$ `mad-max.syndiccjacy*****.onion/goods/i/59fa8ac2d11b35bf3ac05e775b1dcc5e`;

whereas `blackmambamarket.syndiccjacy*****.onion` hosts a single page:

P_2 `blackmambamarket.syndiccjacy*****.onion/login`.

To represent these resources, we therefore use a single node in the SG (corresponding to service S), two nodes in the HG (corresponding to hosts H_1 and H_2), and three nodes in the PG (corresponding to pages $P_{1,1}$, $P_{1,2}$, and P_2).

For all three graphs, we need to tell apart their *directed* and *undirected* version. The directed graphs will be denoted PG, SG, and HG, respectively, and the direction of each edge is induced by the corresponding hyperlink. The respective undirected graphs will be denoted PG*, SG*, and HG*. We first focus on the topology of the aforementioned graphs. With the help of plots and tables, we analyse the connectivity of those graphs, the degree distributions of both the directed and undirected version, and the centrality of their vertices. Then, we focus on the semantic analysis of the content extracted from all visited pages. Even though previous work already applied classification and topic-modelling text mining techniques to discuss the content of Tor pages [7, 6], our analysis provides novel insights thanks to the use of a well-established semantic engine that relies on a taxonomy specifically defined for Tor. Finally, our main contribution consists in the first ever attempt to relate topological and semantic properties of the Tor Web. Among other findings, this comparison allows us to assess the nature of the most relevant (in terms of degree and centrality) pages of the network, and to evaluate whether central nodes of the graph connect semantically uniform components, *i.e.*, whether set of pages with similar content can be identified by cutting off suitably selected vertices and edges. Overall, the analysis exhibited in this Section provides an interesting snapshot of the Tor Web graph, and broadly confirms our intuitions presented in Section 3.

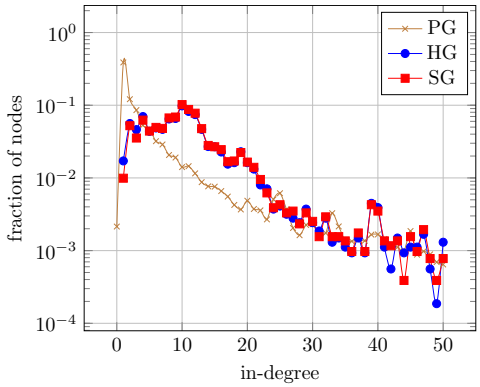
5.1 Topology of the Tor Web Graph

In the following, we focus on the three metrics, defined in Section 3.2, representing the most important characteristics of the structure of the Tor Web graph.

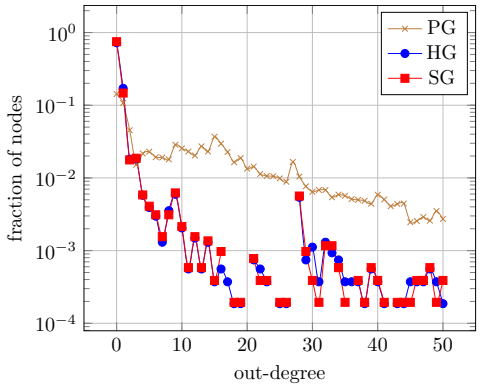
5.1.1 Degree Distribution

We start presenting two plots that show a comparison of the in-degree distribution (Fig. 2a) and of the out-degree distribution (Fig. 2b) for the PG, HG, and SG.

From Fig. 2a, we observe that almost 40% of the PG have in-degree 1. If we add those having in-degree 2 or 3 we arrive to $\sim 60\%$, and if we sum up all nodes having in-degree at most 10 we reach $\sim 80\%$ of the graph. This confirms the intuition, underlying our hypotheses *H2*, *H4*, and *H5*, that most Tor pages are "hard to find", meaning that they are linked by only a few other pages. For what concerns the SG and the HG, first of all we observe that the two probability distributions are almost identical. In comparison with the PG, the in-degree distribution of hosts/services is less concentrated, mostly because having in-degree 0 or 1 is far less likely for a host/service. Despite significantly more hosts/services than pages have in-degree in $[10, 20]$, the average in-degree of a page results more than twice as large (~ 25 *vs.* ~ 12). This is quite reasonable, since a single hyperlink to any of the pages of a host/service is enough to induce an edge in the HG/SG, but multiple hyperlinks among pages of the same pair of hosts/services merge into a single edge in the HG/SG. We underline that, for the sake of clarity, Fig. 2a only shows the most relevant part of the plots. There are a few special cases of pages, services and hosts with much larger in-degree. More specifically, the largest in-degree page has in-degree 18358, whereas the largest in-degree host and service have in-degree 303 and 181, respectively.



(a) Comparison of the in-degree distribution for the Page Graph (PG), Host Graph (HG), and Service Graph (SG).



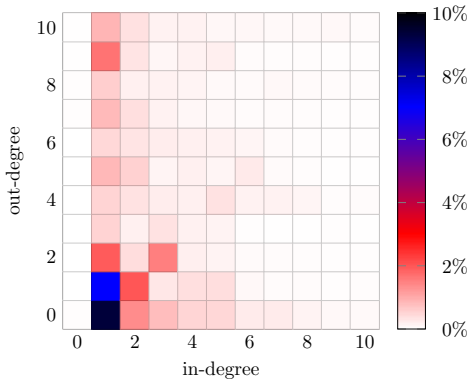
(b) Comparison of the out-degree distribution for the Page Graph (PG), Host Graph (HG), and Service Graph (SG).

Figure 2: In-degree and out-degree distributions for the directed graphs.

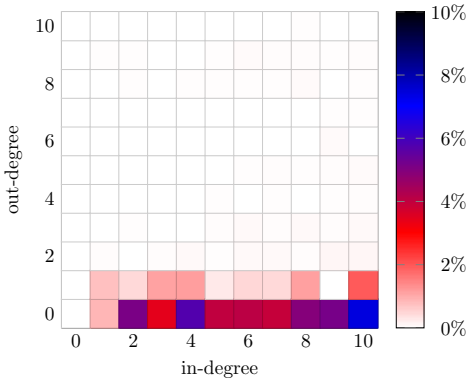
For what concerns the out-degree, from Fig. 2b we see that the HG and the SG are once again almost identical. What leaps out is that the great majority of the nodes ($\sim 72\%$ for the HG, $\sim 75\%$ for the SG) have out-degree 0, *i.e.*, no outgoing hyperlinks. In both cases, adding up nodes with out-degree 1 we reach $\sim 90\%$ of the whole graph. Significantly, hosts/services with out-degree 2 or 3 are one order of magnitude rarer than those with out-degree 1. This confirms our hypothesis *H2*, namely, that most hidden services of the Tor Web graph offer specific services and have very low, if any, interest in providing links to any other service, including related ones. Interestingly, about thirty hosts present hyperlinks to a few hundred other hosts, whereas 10 hosts have more than 3000 outgoing edges, thus covering more than half of the whole graph, with one reaching out-degree as large as 4751. Focusing on the PG, we observe an analogous qualitative behaviour, with nodes of out-degree 0 and 1 weighing significantly more than all others. However, in the case of Tor pages the difference is less pronounced, and a non-negligible portion of the PG links to a few tens of other pages. There are more than 500 pages with hundreds of hyperlinks, about a hundred pages with out-degree > 1000, but only 4 pages with out-degree > 5000, having out-degree 9156, 9669, and 10253 (two different pages), respectively. Overall, the out-degree distribution of these three graphs

shows the importance that specific link directories have in the topology of the Tor Web graph (as stressed later in Section 5.1.2) other than the importance of the root set on the result of the crawl, supporting our hypotheses *H1*, *H2*, and *H4*.

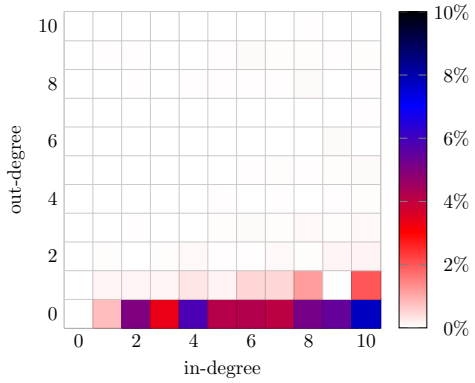
Notwithstanding the precious insight provided by Figs. 2a and 2b, the joint in-out-degree distribution provides even more information about the structure of the graph. Once again, the joint distribution of the SG and the HG, depicted in Fig. 3b and in Fig. 3c, respectively, are very similar. We observe that almost half of the hosts have out-degree 0 and in-degree ≤ 10: such combinations together account for $\sim 45\%$ of the whole graph. The joint in-out-degree distribution of the PG, depicted in Fig. 3a, is instead concentrated along the axis with in-degree 1, with combinations of in-degree 1 and out-degree ≤ 10 accounting for $\sim 25\%$ of the whole graph, and the single combinations $(1, 0)$ and $(1, 1)$ weighing more than 9% and 7%, respectively. Once again, the experiments seem to support our hypothesis *H4*, namely, that the topology of the Tor Web resembles a "forest" of directed extended stars, where a few hosts (the centres of the stars) link to a very large number of other hosts with very small in-degree and no outgoing edges.



(a) Joint in-out-degree distribution for the Page Graph (PG).



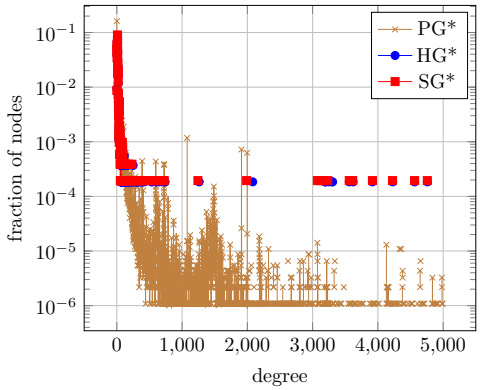
(b) Joint in-out-degree distribution for the Host Graph (HG).



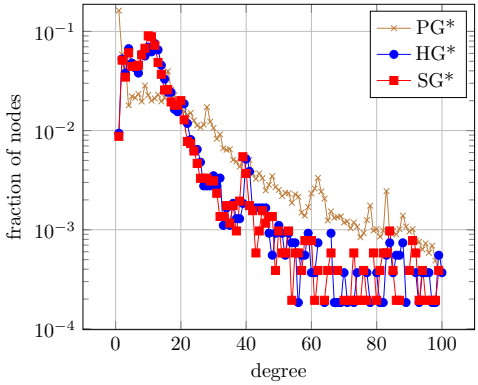
(c) Joint in-out-degree distribution for the Service Graph (SG).

Figure 3: Joint in-out-degree distribution for the directed graphs. Values on the color-bars refer to fractions of nodes of the graph.

The analysis of the in-degree and out-degree distribution of the PG, HG, and SG provides many interesting sparks for gaining a better understanding of their structure. One of the main insights gained so far is that the Tor network can be rightfully expected to be very disconnected if represented as a *directed* graph, as we will show later in Section 5.1.3. This is not surprising if we consider the volatility of the Tor network, and that even the surface Web has been shown to present similar properties [26]. In Figures 4a and 4b we therefore depict the degree distribution of the *undirected* graphs PG*, HG*, and SG*, up to degree 5000 (Fig. 4a) and up to degree 100 (Fig. 4b), respectively. From the figures, we observe that the undirected graphs are significantly more connected than their directed counterpart.



(a) Comparison of the degree distribution for the *undirected* Page Graph (PG*), Host Graph (HG*), and Service Graph (SG*) - Degrees $\leq$ 5000.

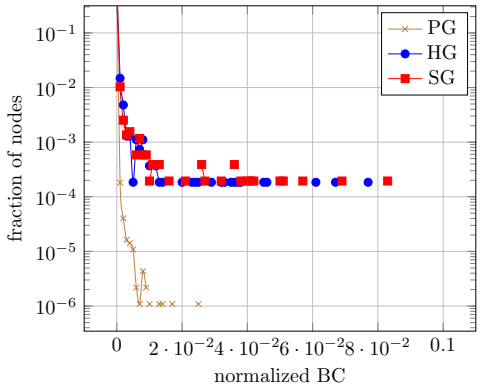


(b) Comparison of the degree distribution for the *undirected* Page Graph (PG*), Host Graph (HG*), and Service Graph (SG*) - Degrees $\leq$ 100.

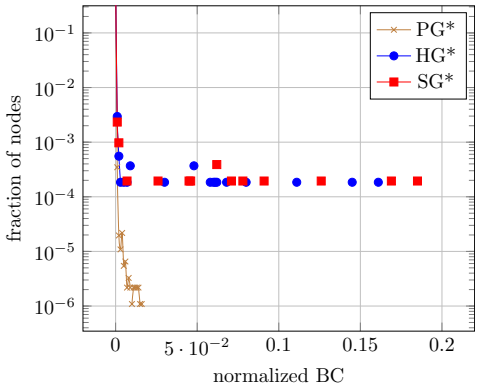
Figure 4: Degree distribution for the undirected graphs.

5.1.2 Betweenness Centrality

Studying the degree distribution of a graph does not only give a first glimpse about the connectivity of the network, but it is also a way to assess the *importance* of single nodes. However, as discussed in 3.2, there are more specific metrics able to capture the *centrality* of a node. Among them, the Betweenness Centrality (BC) is the most widely known and used. Roughly speaking, the BC score of a node quantifies the importance of that node in the overall connectivity of the graph and in the way information flow through it. For the computation of the BC we resorted to a multi-GPU implementation [38, 39]. In the following, we present some plots that allow us to compare and discuss the BC of the nodes of the three graphs. We will especially point our attention to the undirected versions PG*, SG*, and HG* of the three graphs since the BC depends on the number of paths in the graph: the dominance of nodes with very low in-degree and out-degree severely limits the number of paths, making the analysis of the BC and other similar topological metrics of modest interest for the PG, HG, and SG.



(a) Comparison of the normalized betweenness centrality distribution for the *directed* Page Graph (PG), Host Graph (HG), and Service Graph (SG).



(b) Comparison of the normalized betweenness centrality distribution for the *undirected* Page Graph (PG*), Host Graph (HG*), and Service Graph (SG*)

Figure 5: Normalized betweenness centrality distribution for all graphs.

In Figures 5a and 5b we present a comparison of the normalized BC for the three graphs, for the directed and undirected cases, respectively. The overall trend depicted in the two figures is the same: there are a few very central nodes surrounded by a vast majority of peripheral nodes.

Since the degree distribution and the betweenness centrality distribution can be directly compared, it is natural to check whether nodes having the highest BC score have also the largest degree. To this end, in Table 4 we report the top 10 nodes in the ranking of the page graph for what concerns both degree and centrality, and for both the directed and undirected versions PG and PG*. What emerges is that nodes with the highest BC score do not have the largest degree. Since the BC of a node is proportional to the amount of shortest paths passing through that node, the intuition here is that most central nodes are those nodes that connect otherwise disconnected regions, and those nodes do not necessarily have a very large degree. Somewhat more surprisingly, the rankings vary significantly from the directed to the undirected version of the graph, underlying once more the significant difference between the topology of these two graphs. It is worth noticing that most top-ranked nodes in terms of out-degree in Table 4 belonged to our root set, thus supporting our hypothesis *H3*.

Table 4: Degree and Betweenness Centrality Top 10

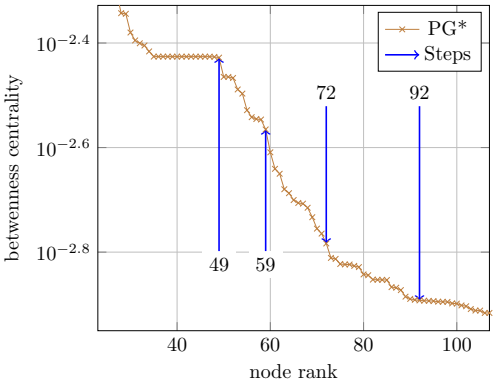
Rank	Node Index				
	In-degree (PG)	Out-degree (PG)	Degree (PG*)	BC (PG)	BC (PG*)
1	255	6855	255	142	154
2	3399	496	496	452	461
3	13499	8107	3399	653	4217
4	341315	42224	13499	422	15324
5	344053	218797	341315	12630	28608
6	341666	103530	343628	3467	37137
7	790729	131172	344053	1558	39453
8	343628	130099	341666	348	40354
9	342870	128671	790729	1913	42400
10	411	128096	6855	1198	46429

5.1.3 Connected Components

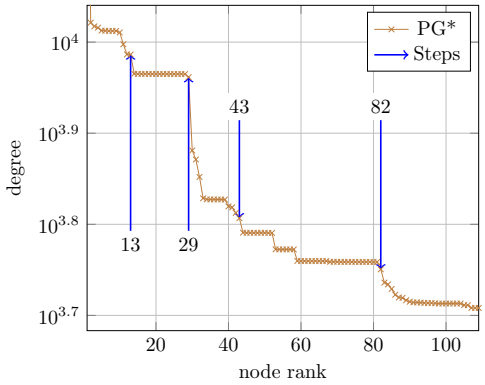
To complete the topological analysis of the Tor network, we focused on the number and size of connected components of the three graphs under study. As we explained in Section 3.2, if we consider the directed version of such graphs, we need to distinguish between strong and weak connection, whereas in undirected graphs the notion of connectivity is unambiguous. However, as we already mentioned in Section 5.1.2, the structure of the PG, SG, and HG is scarcely relevant. This is especially true when it comes to consider connected components: for instance, in the SG we found 4422 Strongly Connected Components (SCCs) for a graph of 5144 total nodes, with almost all SCCs consisting of a single service. Besides, even seminal papers [26, 27] that analysed the connected components of the surface Web focused on the undirected graph because of the very low connectivity of the directed one. However, if we focus on the undirected graphs PG*, HG*, and SG*, we observe a completely opposite phenomenon: all three graphs exhibit a single giant connected component that comprehends all nodes of the network [20] , *i.e.*, of size 918885, 5420, and 5144, respectively.

To gain additional information about the structure of the graph, we decided to verify how the connectivity of the network depends on its most important nodes, identified in Section 5.1.2. To this end, we incrementally removed from the PG* the nodes with larger BC and larger degree, together with all their edges, and studied the number and size of connected components of the remaining graph. The first cuts took into account the BC and degree rankings, cutting off at points where those rankings exhibit a significant drop, as shown in Fig. 6a and in Fig. 6b. Table 5 summarizes our findings, showing that the number of connected components increases quickly, but the size of the largest component decreases pretty slowly. Let us underline that the column SRCC reports the number of *Semantically Relevant Connected Components*, that is, the number of components that

[20] More precisely, we found a few micro-components of 1 or 2 nodes. We realized that such components were artefacts of the procedure used to reconstruct the graph. Indeed, the only explanation of the effective existence of such components would have been the presence of their nodes in the root set. We checked such possibility and we didn't find any evidence. Therefore, we considered such isolated pages as some sort of noise introduced by our implementation. As such, we focused on the portion of the graph generated by our crawler that seemed ascribable to a descriptive pattern of the network.



(a) Normalized betweenness centrality for the *undirected* Page Graph (PG*) - zoom on topological cuts.



(b) Degree for the *undirected* Page Graph (PG*) - zoom on topological cuts.

Figure 6: Topological cuts for the undirected Page Graph (PG*).

include at least 2 pages (*i.e.*, at least 1 possible comparison) to which it is possible to associate a semantic vector [21].

Table 5: Connected Components after Cuts (PG*)

Cut (BC)	# CC	# SRCC	Max Size	Cut (Degree)	# CC	# SRCC	Max Size
49	462	32	876348	13	13433	5	903680
59	4792	34	872008	29	14464	6	902323
72	4874	41	860346	43	16055	9	895265
92	9672	73	851268	82	16902	10	894294
500	41966	444	685351	500	43023	53	847006
1000	58646	861	550912	1000	48287	66	835470
5000	137732	1395	322329	5000	86071	343	727876
10000	171310	1553	261276	10000	115122	536	652547
50000	307000	1785	70993	50000	265580	3696	261239
100000	390179	2137	12127	100000	396926	5866	103085

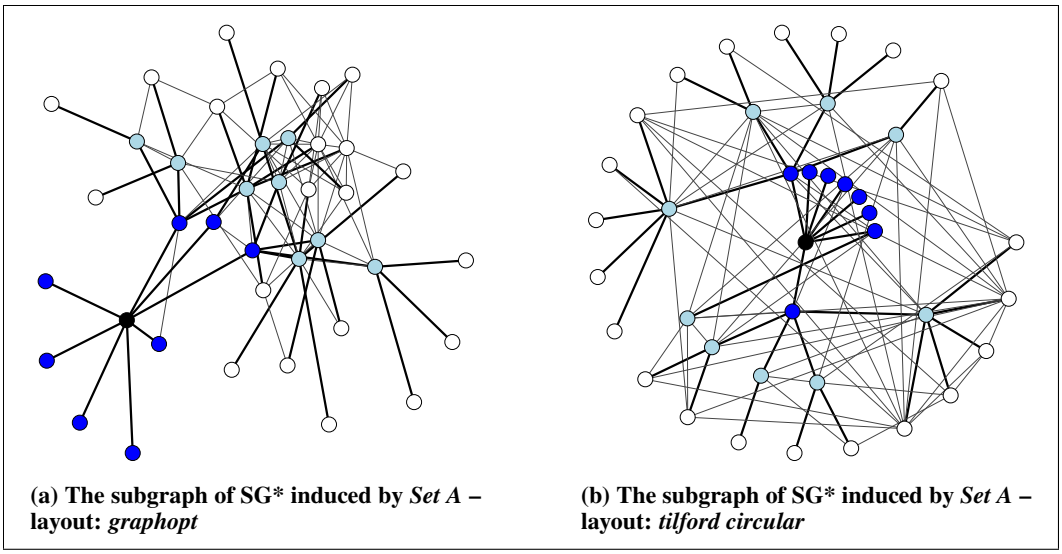
Figure 7, showing two excerpts of the SG*, represents a visual support to our findings. We selected two subgraphs, respectively induced by the following two sets of vertices:

Set A The vertex v with maximal BC, 8 random neighbors $v_1, \ldots, v_8$ of v, $\min(4, \deg(v_i))$ random neighbors $v_{i,j}$ of each v_i, and $\min(5, \deg(v_{i,j}))$ random neighbors of each $v_{i,j}$ – this graph is depicted in Fig. 7a and Fig. 7b using two different graph layouts;

Set B The two vertices v, v' having largest BC, 8 random neighbors $v_1, \ldots, v_8$ of v and 8 random neighbors $v'_1, \ldots, v'_8$ of v', $\min(4, \deg(v_i))$ random neighbors of each v_i and $\min(4, \deg(v'_i))$ random neighbors of each v'_i – this graph is depicted in Fig. 7c and Fig. 7d using two different graph layouts.

In the first subgraph we highlighted the minimum spanning tree (of depth 3) rooted in v, whereas in the second one we highlighted two minimum spanning trees (of depth 2) rooted in v and v', respectively. Although Fig. 7 is obtained from the undirected service graph SG*, whereas our hypothesis *H4* referred to the directed version of the Tor Web graph, the comparison between the whole subgraphs and their minimum spanning trees suggests that the structure of the Tor Web graph is not far from our prediction.

[21] Many Tor pages do not include any text or are not in English, preventing Cogito from producing a semantic vector.



(a) The subgraph of SG* induced by *Set A* – layout: *graphopt*

(b) The subgraph of SG* induced by *Set A* – layout: *tilford circular*

(c) The subgraph of SG* induced by *Set B* – layout: *graphopt*

(d) The subgraph of SG* induced by *Set B* – layout: *tilford circular*

Figure 7: Excerpts of the SG*, using the *graphopt* and the *tilford circular* visualization layouts (graphs in the same box are different visualizations of the very same graph). Nodes' colors get lighter as distance from the root node(s) with largest BC increases.

5.2 Semantic Analysis of the Tor Web Graph

In this Section, we discuss the semantic organization of the Tor Web graph, and correlate it with our findings about its topology. To that purpose, we first need to identify a suitable way to: (i) analyse the content of a single page, (ii) compare the contents of two pages, and (iii) assess the semantic uniformity of a set of pages. More in detail:

(i) As outlined in Section 3.3, with the aid of the Cogito semantic engine, for each page p, we obtain a vector S_p of size n, whose i^{th} component $S_p[i]$ quantifies to which extent the content of the page can be associated to the i^{th} category of our taxonomy. The taxonomy, defined within the scope of the "IANCIS" ISEC Project [22], is reported and discussed in Section 3.3. Let us recall that we consider only English text, as already motivated in Sec-

[22] DG Home Affairs ISEC Programme 2013 - Project IANCIS "Indexing of Anonimous Networks for Crime Information Search", 2014-2016, GA n. HOME/2013/ISEC/AG/INT/4000005222 - www.iancis.eu

tion 3.3. Additionally, we ignore all pages with a null semantic vector, *i.e.*, whose content is irrelevant with respect to any category of our taxonomy.

(ii) To compare the contents of two pages, we compute the cosine similarity of the two corresponding vectors, $cosine(S_{p_1}, S_{p_2})$, defined as the cosine of the angle between the two vectors. If $cosine(S_{p_1}, S_{p_2}) = 0$ the two vectors are orthogonal and the two pages have nothing in common. Conversely, if $cosine(S_{p_1}, S_{p_2}) = 1$ the two vectors coincide, *i.e.*, the two pages can be considered fully equivalent, at least according to our taxonomy.

(iii) We compute the pairwise cosine similarity of the semantic vectors associated to a set of pages with the goal of measuring their semantic uniformity. This allows drawing statistics and using the average similarity as a synthetic score for the whole set.

We consider only the undirected graph PG*, since we can rightfully associate a semantic vector per node only for pages, and the topology of the directed graph PG is of limited interest. Even though our framework allows to establish what topics are more relevant in a service or host, it is not likewise accurate in associating a single semantic vector to each of them.

We start looking at the entire PG*, that, as we saw, is composed of a single connected component. First of all, in Fig. 8 we show the statistical distribution of the cosine similarity between any two vectors of the graph. Fig. 8 highlights that in most cases $cosine(S_{p_1}, S_{p_2}) \approx 1$ or $cosine(S_{p_1}, S_{p_2}) = 0$. The insight here is that most semantic vectors are significantly unbalanced (*i.e.*, the corresponding page can be associated with only a small subset of the categories considered), so that any two vectors are either very similar or completely unrelated. The semantic analysis thus seems to confirm our hypothesis *H1*.

It is therefore interesting to understand how close to each border are the semantic vectors of our dataset, *i.e.*, how relevant is each of the topic considered in the dataset. To this end, in Fig. 9 we present graphs showing the statistical distribution of the cosine similarity among pages of PG* and semantic vectors representing the single categories of our taxonomy. More precisely, since each category corresponds to a dimension of our vector space, the standard basis vectors $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ model mono-thematic pages in which only topic i is considered. The statistical distribution of the cosine similarity among pages of PG* and e_i tells us how relevant is topic i in our dataset, and if the relevance of the topic is significantly dependent on the particular page considered. Let us underline that the cosine similarity between a vector $x = (x_1, \ldots, x_n)$ and e_i is simply $\frac{x_i}{||x||}$: this means that it does not depend on which other topics that page includes, *i.e.*, on x_j, $j \neq i$, but only on the relative "weight" of topic i.

What emerges is that only "information systems" is a very diffused topic in our dataset, whereas almost all other topics are completely unrelated to the vast majority of the pages. At the same time, most topics have a non-negligible spike at 1, meaning that there are many pages that can be associated to a single category of our taxonomy. Again, these results support our hypothesis *H1*.

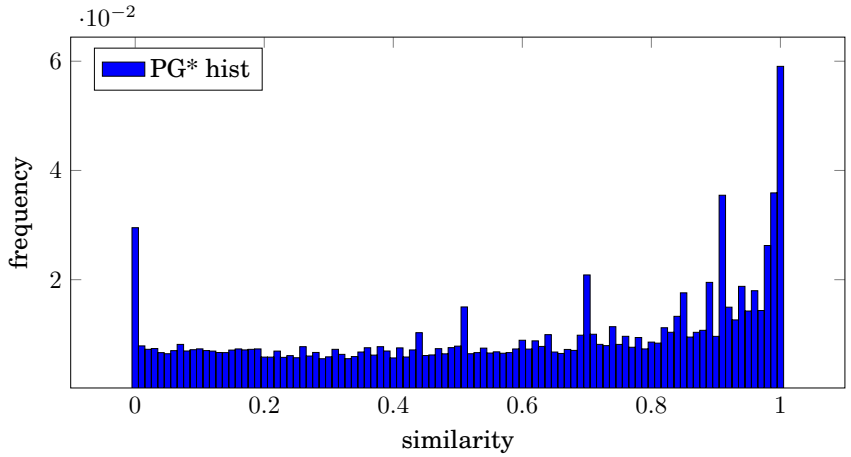


Figure 8: The statistical distribution of the cosine similarity among pages of the PG*.

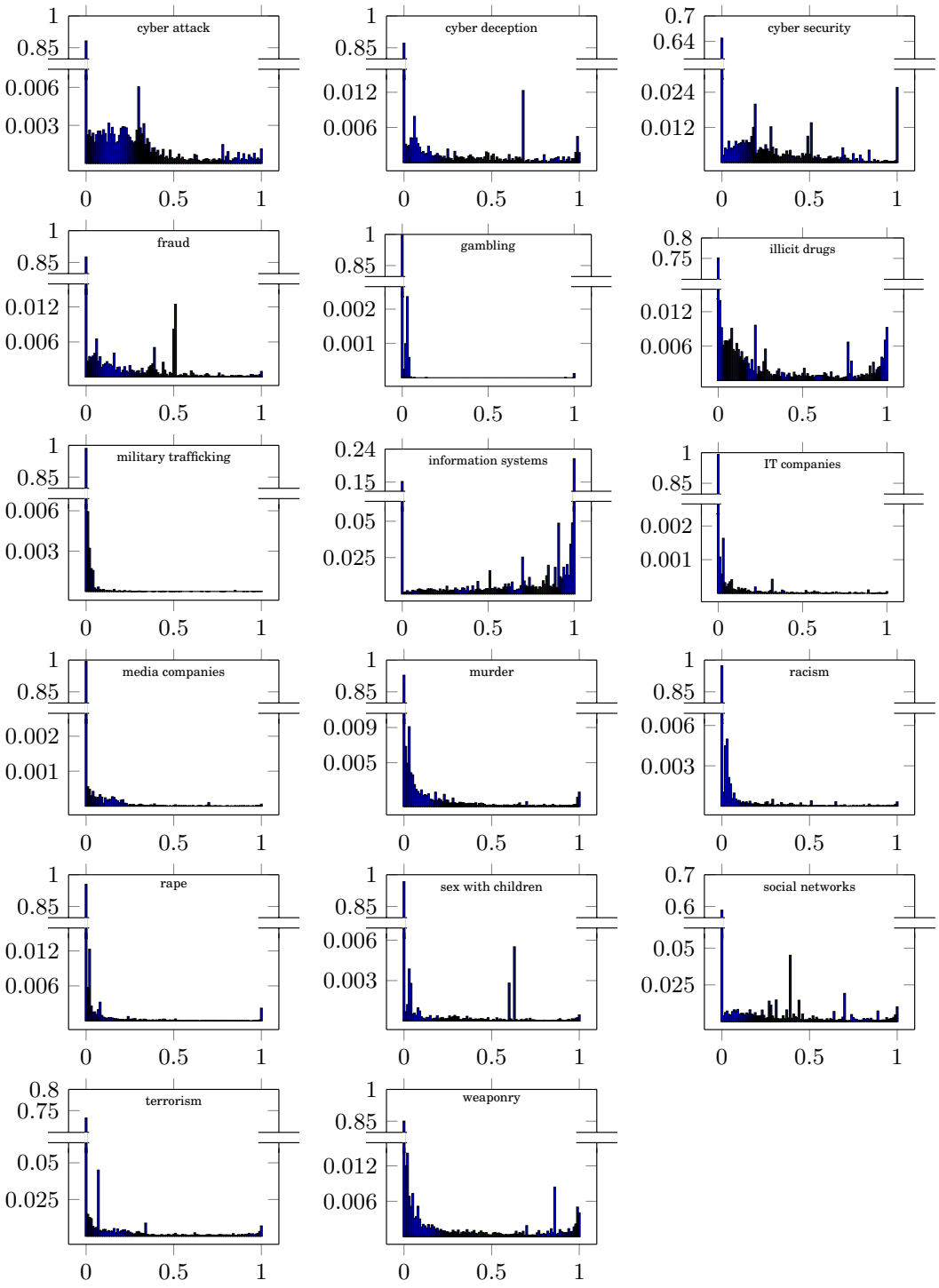


Figure 9: The statistical distribution of the cosine similarity between pages of the PG* and semantic vectors that represent the single categories of our taxonomy.

Now, we want to assess whether the Tor PG* contains smaller tightly connected components that exhibit a larger semantic uniformity with respect to the whole graph. To that purpose, we repeatedly cut off the pages with largest BC score and largest degree so as to split the original single connected component into a number of smaller components, as described in Section 5.1.3. For each of those components, we find the average cosine similarity of that component. Since the number of components grows quickly, we need a synthetic representation of what happens with different cuts. To this end, in both Fig. 10 and Fig. 11, for each cut we show a bubble per component: the size of the bubble is the number of semantic vectors of that component [23], and the center of the bubble is located according to the average semantic similarity of the component. We compare the bubbles, *i.e.*, single components, with their average (the average of the average) cosine similarity. More precisely, both figures show both the simple (shortened "Avg") and the *weighted* (shortened "W. Avg") arithmetic mean of the average similarities, where in the latter each component's weight is the number of semantic vectors it contains. The two figures clearly show that we can identify components with increasing semantic similarity by cutting off the graph nodes with increasing BC score or increasing degree. Additionally, by comparing the two figures, it emerges that the BC is a better performing metrics than the degree to the end of identifying semantic uniform sub-components. Although these results partially support the intuition formalized by hypothesis *H1*, they also suggest that many link directories and wikis (that, having the largest degree, but not the largest BC, are not cut off by BC cuts) are monothematic. While we expected link directories and wikis to be mostly responsible for connecting regions of the graph that cover different topics, the ability of pages having large BC score in connecting otherwise disconnected components eventually results more effective.

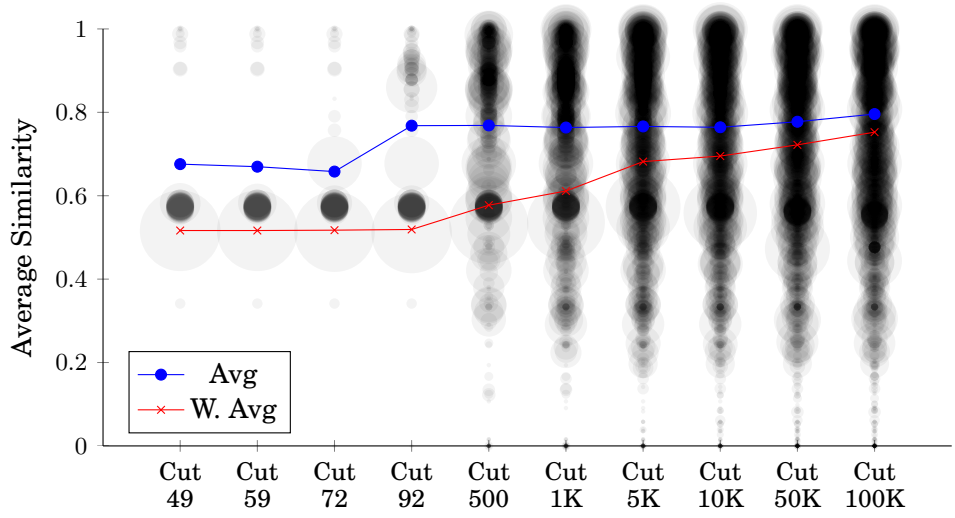


Figure 10: Bubble chart for the cosine similarity of different BC cuts. Each bubble corresponds to a component and the size of a bubble is the log of the number of semantic vectors of its component. "Avg" denotes the average for that cut, "W. Avg" the weighted average, where each component weighs as its number of semantic vectors.

6 Conclusions

We have identified and addressed a relevant lack of knowledge regarding the Tor seen as a graph. Starting from state-of-the-art studies and tools, we have devised and implemented a novel approach for the exploration of the Tor Web, that led to us collecting a rich set of interesting data over the Tor Web topology and contents. A further in-depth analysis of the obtained experimental data helped discovering and describing novel relationships between topology and semantics, and allowed a careful reasoning of these results. Indeed, this work shows interesting characteristics of the Tor Web graph that relate topic semantics and Web graph topology. Such findings contribute to a better understanding of the Tor usage and contained information. Future work will aim at improving the coverage of Tor nodes, in particular the relation between the composition of the root set used to start

[23] As discussed previously in this Section, not all pages are associated with a semantic vector.

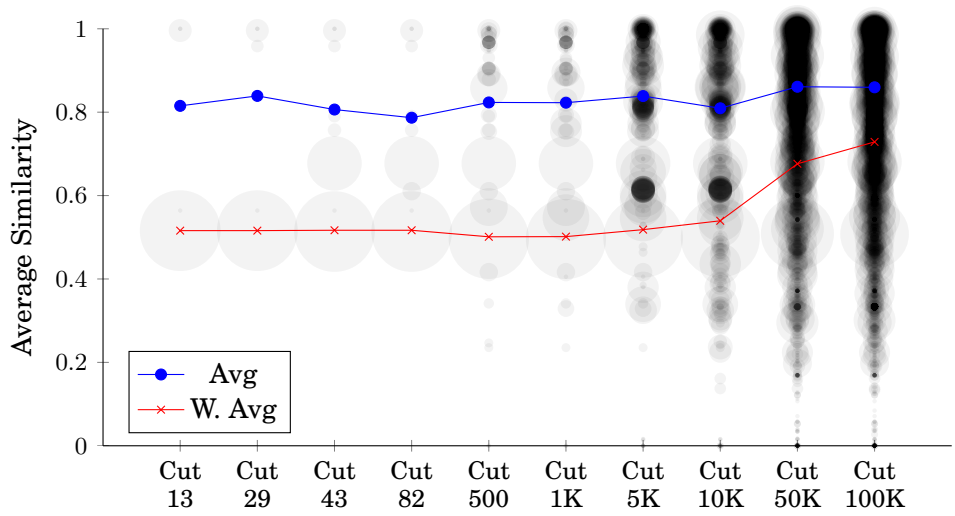


Figure 11: Bubble chart for the cosine similarity of different degree cuts. Each bubble corresponds to a component and the size of a bubble is the log of the number of semantic vectors of its component. "Avg" denotes the average for that cut, "W. Avg" the weighted average, where each component weighs as its number of semantic vectors.

the crawling and the outcome of the Tor Web exploration. We also aim at analysing the evolution of the Tor Web graph over time, both from a topological and a semantic point of view.

Acknowledgments

This research work is partially funded by IANCIS [24], a EU ISEC project that involves IAC-CNR, Expert System and the Italian "Arma dei Carabinieri", and whose purpose is to build a tool that, based on Cogito, can crawl and semantically index and cluster Tor pages.
We thank Flavio Vella for his help in measuring the betweenness centrality scores of the graphs under study.

References

[1] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Usenix Security Symposium*, 2004.

[2] Monica J. Barrat. Silk road: Ebay for drugs. *Addiction*, 107(3):683–683, 2012.

[3] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. Stegotorus: A camouflage proxy for the tor anonymity system. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 109–120, New York, NY, USA, 2012. ACM.

[4] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for tor hidden services: Detection, measurement, deanonymization. In *Proc. Symposium on Security and Privacy*, SP '13, pages 80–94, Washington, DC, USA, 2013. IEEE Computer Society.

[5] Daniel Arp, Fabian Yamaguchi, and Konrad Rieck. Torben: A practical side-channel attack for deanonymizing tor communication. In *Proc. 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, pages 597–602, New York, NY, USA, 2015. ACM.

[6] Alex Biryukov, Ivan Pustogarov, Fabrice Thill, and Ralf-Philipp Weinmann. Content and popularity analysis of tor hidden services. In *Distributed Computing Systems Workshops (ICDCSW), 2014 IEEE 34th International Conference on*, pages 188–193, June 2014.

[24] http://www.iancis.eu/

[7] Martijn Spitters, Stefan Verbruggen, and Mark van Staalduinen. Towards a comprehensive insight into the thematic organization of the tor hidden services. In *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, pages 220–223, Sept 2014.

[8] Mike Perry. Torflow: Tor network analysis. `http://fscked.org/talks/TorFlow-HotPETS-final.pdf`, 2009.

[9] Robin Snader and Nikita Borisov. Improving security and performance in the tor network through tunable path selection. *IEEE Transactions on Dependable and Secure Computing*, 8(5):728–741, 2011.

[10] Robin Snader and et al. A tune-up for tor: Improving security and performance in the tor network. `https://www.internetsociety.org/doc/tune-tor-improving-security-and-performance-tor-network-paper`, 2008.

[11] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the tor network. In *Privacy Enhancing Technologies*, volume 5134 of *LNCS*, pages 63–76. Springer Berlin Heidelberg, 2008.

[12] Gareth Owen and Nick Savage. Empirical analysis of tor hidden services. *IET Information Security*, 10(3):113–118, 2016.

[13] Kyle Soska and Nicolas Christin. Measuring the longitudinal evolution of the online anonymous marketplace ecosystem. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 33–48, 2015.

[14] Rob Jansen, Kevin Bauer, Nicholas Hopper, and Roger Dingledine. Methodically modeling the tor network. In *Proc. 5th USENIX Conference on Cyber Security Experimentation and Test*, CSET'12, pages 8–8, Berkeley, CA, USA, 2012. USENIX Association.

[15] Rob Jansen and Nicholas Hopper. Shadow: Running tor in a box for accurate and efficient experimentation. In *Proc. 19th Symp. on Network and Distributed System Security (NDSS)*. Internet Society, February 2012.

[16] Kevin Bauer, Micah Sherr, Damon McCoy, and Dirk Grunwald. Experimentor: A testbed for safe and realistic tor experimentation. In *CSET*, 2011.

[17] Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. *SIGKDD Explor. Newsl.*, 2(1):1–15, June 2000.

[18] Soumen Chakrabarti, Amit Pathak, and Manish Gupta. Index design and query processing for graph conductance search. *The VLDB Journal*, 20(3):445–470, June 2011.

[19] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, 70:301 – 323, 2014.

[20] Paolo Boldi and Sebastiano Vigna. The webgraph framework i: Compression techniques. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 595–602, New York, NY, USA, 2004. ACM.

[21] Francisco Claude and Gonzalo Navarro. Fast and compact web graph representations. *ACM Trans. Web*, 4(4):16:1–16:31, September 2010.

[22] Francisco Claude and Susana Ladra. Practical representations for web and social graphs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1185–1190, New York, NY, USA, 2011. ACM.

[23] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D Sivakumar, Andrew Tomkins, and Eli Upfal. Stochastic models for the web graph. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 57–65, 2000.

[24] Anthony Bonato. A survey of models of the web graph. In Alejandro López-Ortiz and AngèleM. Hamel, editors, *Combinatorial and Algorithmic Aspects of Networking*, volume 3405 of *Lecture Notes in Computer Science*, pages 159–172. Springer Berlin Heidelberg, 2005.

[25] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans M. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35:66–71, 2002.

[26] JonM. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and AndrewS. Tomkins. The web as a graph: Measurements, models, and methods. In *Computing and Combinatorics*, volume 1627 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 1999.

[27] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer Networks*, 33(1–6):309 – 320, 2000.

[28] Devanshu Dhyani, Wee Keong Ng, and Sourav S. Bhowmick. A survey of web metrics. *ACM Comput. Surv.*, 34(4):469–503, December 2002.

[29] Christos Giatsidis, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. Advanced graph mining for community evaluation in social networks and the web. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 771–772, New York, NY, USA, 2013. ACM.

[30] Massimo Franceschet. Pagerank: Standing on the shoulders of giants. *Commun. ACM*, 54(6):92–101, June 2011.

[31] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[32] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[33] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In Philip S. Yu, Jiawei Han, and Christos Faloutsos, editors, *Link Mining: Models, Algorithms, and Applications*, pages 337–357. Springer New York, 2010.

[34] Evgeniy A. Grechnikov. Degree distribution and number of edges between nodes of given degrees in the buckley–osthus model of a random web graph. *Internet Mathematics*, 8(3):257–287, 2012.

[35] Phillip Bonacich. Some unique properties of eigenvector centrality. *Social Networks*, 29(4):555 – 564, 2007.

[36] Dimitrios Prountzos and Keshav Pingali. Betweenness centrality: Algorithms and implementations. *SIGPLAN Not.*, 48(8):35–46, feb 2013.

[37] Paolo Boldi, Andrea Marino, Massimo Santini, and Sebastiano Vigna. Bubing: Massive crawling for the masses. In *Proc. 23rd International Conference on World Wide Web Companion*, pages 227–228, 2014.

[38] Flavio Vella, Giancarlo Carbone, and Massimo Bernaschi. Algorithms and heuristics for scalable betweenness centrality computation on multi-gpu systems. *CoRR*, abs/1602.00963, 2016.

[39] Massimo Bernaschi, Giancarlo Carbone, and Flavio Vella. Scalable betweenness centrality on multi-gpu systems. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '16, pages 29–36, New York, NY, USA, 2016. ACM.

Appendix: Software Tools

To explore Tor, extract text and visualize the obtained graphs we made use of a set of open source tools, namely: Polipo, Bubing, Apache Tika, igraph. To extract metrics from the graphs, we developed custom software in several languages (mainly C and Java), including the aforementioned Cuda code for calculating the BC. Finally, we made use of the Cogito proprietary software to construct the semantic vectors. Readers interested in additional details and in getting our tools are welcome to contact the authors.