Project Number 001907

DELIS
Dynamically Evolving, Large-scale Information Systems

Integrated Project
Member of the FET Proactive Initiative Complex Systems

Deliverable D1.2.3

Periodic crawls and sampling methods for the Web
Decomposition based techniques for network visualization
Start date of the project: January 2004
Duration: 48 months
Project Coordinator: Prof. Dr. math. Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute, University of Paderborn, Germany
Due date of deliverable: December 2006
Actual submission date: January 2007
Dissemination level: PU – public

Work Package 1.2: Monitoring and Measuring the Evolving Web
Participants: Università Degli Studi di Roma “La Sapienza” (UDRLS), Italy
Universität Paderborn (UPB), Germany
Universität Karlsruhe (UniKarls), Germany
Université de Paris-Sud XI (UPS), France
Universitat Pompeu Fabra (UPF), Spain
Università Degli Studi di Milano (UniMI), Italy

Authors of deliverable: Debora Donato(debora@yahoo-inc.com)
Robert Görke(rgoerke@ira.uka.de)
Stefano Leonardi(Stefano.Leonardi@dis.uniroma1.it)
Massimo Santini(santini@dsi.unimi.it)
Christian Sohler(csohler@upb.de)
1 Introduction

As illustrated in the implementation plan for months 25-42 the aim of the WP1.2 is to define a comprehensive set of concepts for monitoring, measuring and visualizing the evolving Web. The work of WP1.2 is done in cooperation between UPB, UDRLS, UniKarl, UPS and UniMi. During the third year of the project, the activity of the WP 1.2 partners was directed firstly to delving into the concepts and methodologies partially studied during the second year, secondly in the effort of constructing a reliable and consistent data set in order to monitor how the properties of the Web evolve over the time. For this second task, a cooperation between UDRLS and UniMi has led to the WebSnap project: This project aims at providing data and tools for the analysis of snapshots of the web. Since May of this year, monthly crawls of the .uk domain have been launched. This activity has required a significant amount of infrastructure and ad hoc algorithms and tools, which have been developed at the LAW (Laboratory for Web Algorithmics), UniMi.

Since global scale crawling is prohibitive, small but representative crawlings are required. We show in [DELIS-TR-0350] that in general the macroscopic properties of the Web are better represented by a shallow exploration of a large number of sites than by a deep exploration set of sites. We study the biases imposed by the crawling strategy in the observed measures and show that they can be significant even if the sample covers a large fraction of the collection. Moreover we propose a strategy [DELIS-TR-0348] that is efficient from the point of view of resource utilization, limiting the growth of the queue of pending pages during the crawling, while at the same time ensuring high coverage, politeness and good quality.

The analysis of the problems relating to efficient Web crawls led us to study Web spam, a growing phenomenon in the Web that reduces the precision of the search engines. The data collected at UniMi allow us to make available the first labeled collection [DELIS-TR-0405] for Web spam detection and demotion. In addition in [DELIS-TR-0341] we present two algorithms that result effective in fighting topological Web spam. This work has received the contribution of the UDRLS, UPF (Yahoo research center) and UniMi sites.

In the attempt of individuate significant properties for the study of Web dynamics, we have further investigated the properties of Wikipedia. We have already shown that Wikipedia and the Web are very similar since they share the same statical topological properties. The experiments performed on Wikipedia time evolution allows us to individuate a number of properties that can be used to determine if a network reached the maturity or is still evolving toward a more stable configuration.

We have already underlined as to perform an in-depth study of the baseline features of its dynamic, it is necessary to design a wide set of sampling and streaming algorithms for monitoring properties of evolving networks. Random sampling and streaming algorithms also allow to compute provably good approximations of the measures by observing only a subset of the pages. We extend the work done so far in cooperation between UDRLS and UPB on random sampling and streaming algorithms. In [DELIS-TR-0417] we provide methods to count the number of small dense subgraphs, to identify clusters in networks, to estimating the clustering coefficient, that is the average fraction of connected neighbor pairs of a vertex. Visualization techniques, based on a visualization-driven approach [DELIS-TR-0036] which relies on the concept of the k-kore-decomposition [BZ02, Sei83], were extensively applied during this year to investigate baseline features of largest P2P file sharing platforms in the Internet. The expertise acquired has been used in a twofold way. Within the WP1.3, a collaboration between TUM and UniKarl led to a study of the topology correlation of the overlay network of the Gnutella application and the underlying physical Internet (for more details see Deliverable D1.3 of WP1.3) Instead in [DELIS-TR-0402] we provided a visualization-driven analysis of the potential bias of the coreness, a graph-structural property that presents specific and interesting characteristics.
2 Main contributions

2.1 WebSnap: data and tools

The WebSnap project aims at providing data and tools for the analysis of snapshots of the web. This goal requires a significant amount of infrastructure and ad hoc algorithms and tools, which have been developed at the LAW (Laboratory for Web Algorithmics) of the Università degli Studi di Milano. The description of the data and tool used is part of [].

2.1.1 Data

The data collected is made of twelve snapshots of the .uk domain of about 100 Mpages each. The choice of the .uk domain was almost mandatory, as a full understanding of the page contents is necessary for any kind of assessment by human judgments. The chosen size is big enough to be significant (for a national domain), but manageable enough to be actually distributable to the project partners (consider that in the end we will collect more than one billion pages, which will have to be distributed in full length). Just for a comparison, at the time of the first crawls Google was claiming to have crawled 700 Mpages from .uk.

The snapshots are being taken at the start of each month, during a period of 7 – 10 days, using the bandwidth provided by the Università degli Studi di Milano and a cluster of PCs that has been funded by the DELIS project. The crawls was done using UbiCrawler’s [BCSV04] built-in per-host breadth-first visit, starting from a large seed of (150 000 elements) set of URLs obtained from the Open Directory Project. We limited each host to a maximum of 50 000 pages. This guarantees that we shall crawl at least 2 000 hosts, and limits the impact of web traps and database-driven sites.

Presently, about 14 TB of non-compressed data have been gathered. The amount of data will approximately double in the next months.

2.1.2 Software

The software used to gather the snapshot and compute derived data has been entirely developed at the LAW. In the last few years we have accumulated a number of well-tuned Java classes related to large data processing, and we used the established infrastructure, plus a significant amount of new code, to make WebSnap possible.

First, we review the existing software. fastutil provides high-performance containers for primitive types and very flexible, fast I/O wrappers (the standard Java buffered I/O classes are unusable for our purposes). MG4J [BV05] is a framework for full-text indexing and search engine construction, but it provides also very efficient data structures that are essential in handling web graphs — most notably, a fine-tuned implementation of minimal perfect order-preserving hash functions. The latter are fundamental in accessing the node of a graph corresponding to a given URL. WebGraph represent graphs in compressed form in main memory, still letting the user access successor lists at about 200 ns/link on a standard PC.

Last but not least, UbiCrawler [BCSV04] is our fault-tolerant, scalable and fully distributed crawler. It has been used to crawl the first six snapshot and we plan to complete WebSnap using it. However, whereas all previous software project are distributed under free licenses (GNU Public License or GNU Lesser Public License) UbiCrawler is not currently distributed, mainly for lack of documentation (however, we are presently working at a new crawler, BUbiNG, that will be freely available).

The output format of UbiCrawler is proprietary, and albeit a possible solution could have been that of distributing part of UbiCrawler’s code, we preferred to refer to the new upcoming proposed ISO standard for web archiving — WARC. It has been developed mainly by the people at the Internet Archive in the last few years.
WARC has many advantages: it is compressed page-by-page with GZip, resulting in a very good compression, but provides hooks to implement skipping mechanisms. In particular, the WARC files produced for WebSnap incorporate some metadata in the GZip comment field, making it possible to scan an archive, say for the list of URLs therein contained, very quickly (about 5 Kpages/s).

Nonetheless, no public code has been currently published that addresses the problem of reading WARC files. Thus, we developed a Java framework for WARC I/O that is highly efficient and easy to use. We extended some parts of fastutil and MG4J to make it possible, for instance, reading lines from a pure byte stream (in Java, line reading is relegated to character streams, but raw web data cannot be immediately translated into characters). At present time we have a converter that turns UbiCrawler data into WARC format, but the new crawler we are developing will use WARC as its native format.

One of the advantages of using a standard format is that we can provide utilities for examining, cutting, searching WARC files that will work independently of the crawler that gathered a snapshot.

2.1.3 Hardware
WebSnap is based partially on existing hardware at the LAW, and partially on new hardware directly acquired through DELIS funding. In particular, we bought eight Dell PowerEdge 1800 servers, with dual Xeon at 2.8 GHz, 4 GB of main memory and 1 TB raid disks. This cluster is used by eight UbiCrawler agents to perform the crawls, and to build the web graph of each snapshot. For the few tasks requiring more memory (about 20 GB for the construction of the global minimal perfect hash map) we use a Sun Fire V880 workstation with 16 GB of main memory.

2.2 Web crawling strategies analysis

2.2.1 Statistical analysis of subsamples of Web pages
The amount of resources needed for crawling and analyzing significant portion of the Web is considerable and can become even problematic when storing periodic snapshots is necessary to study Web dynamics. In this context, processing time, disk space and network usage can be reduced greatly by using adequate sampling methods, able to extract small but representative samples from the Web pages. In [DELIS-TR-0350] we compare different methods for Web sampling by sub-sampling a large Web collection and evaluate each method by studying several link-based metrics over the resulting sample.

**Sampling methods** Our sampling methods pick some nodes according to some schema, and then include an edge in the sampled graph if both its source and destination nodes were picked. We fixed the fraction of nodes to be picked by each sampling method to 0.1, 0.2, 0.5, 0.8 and 0.9.

The schemes used for picking nodes are the following:

**Uniform random sampling** Pages are chosen uniformly at random with a certain probability.

**Sampling by selecting entire sites** Sites are chosen uniformly at random with a certain probability, and all of the pages inside a site are included in the sample. We continued this process until we have a predefined fraction of the nodes in the graph. This is feasible in practice and the crawler must be instructed not to follow links outside the sampled sites.

**Sampling by breadth-first search** (BFS) All the initial pages of sites (the starting or home page, located in the root directory of the site and typically named “/index.*” or just “/”) were sampled. We consider those pages to be at depth equal to 1. All of the pages that are linked by those pages are considered to have depth equal to 2, and so on. This strategy simulates a BFS search that stops when a given threshold of nodes is reached.
Sampling by OPIC The OPIC algorithm (online page-importance computation) was introduced by Abiteboul et al. [APC03] as an algorithm for ranking pages while discovering them. It can be seen as a biased breadth-first search in which the pages that are highly linked are more likely to be chosen. To implement this algorithm in external memory, we approximated it by re-calculating page importance 20 times during the simulated crawl (instead of after inserting every node).

Evaluation For the empirical evaluation we use a collection of 50.6 million nodes and 1.9 billion edges obtained from a geographic crawl of Slovakia in June 2005. We concentrate over the overlap, a number of microscopic and macroscopic measures.

Overlap The overlap is measured as the fraction of nodes that are sampled by 2 sampling strategies. We observe that sampling by OPIC, by BFS and by sites behave in a similar way.

Microscopic measures When characterizing a large graph, many aspects of the topology of the graph can be studied. Reference [CRT05] presents a comprehensive survey on the type of statistics that can be extracted. All the sample strategies achieve good performances in approximating such statistics. The BFS is the method that seems more reliable for all the measures considered with the only exception of an overestimation of the average degree. This behavior is consistent with the observation that the pages closer to the home page are more connected than the ”deeper ones.

Macroscopic measures The macroscopic structure of the Web is captured by the bow-tie shape, firstly proposed in [BKM+00] and organized in five sets: CORE, IN, OUT, DISCONNECTED, TENDRILLS (for a complete description of the sets see [BKM+00]). The best approximation of the relative sizes of the components is given by the BFS method meanwhile sampling by sites performs very poorly. A similar results is obtained when we count the fraction of nodes that, respectively in IN and OUT are at distance k from the CORE, that is the nodes that are reachable starting from the CORE through a path of length k.

As overall result we can state that many characteristics of the connectivity of the Web arise from the interaction among many different sites. Hence a very deep crawl of few sites fails to capture some important features of the Web graphs.

2.2.2 A new strategy for controlling the queue size in Web crawling

In [DELIS-TR-0348] we study the size of the queue of pending pages during a crawl of a large subset of the Web. We show how an adequate handling of the larger hubs found on the Web can allow us to save 50% of the size of the queue while preserving both the coverage and quality of the collection. Reducing memory usage in Web crawling has a number of benefits. First, in practice most of the pages in the queue will never be visited; second, in the case of parallel crawlers [CGM02] that must exchange URLs, the amount of URLs exchanged can be reduced; third, in the case of focused crawlings [CvdBD99] by personal Web crawlers (that run on normal desktop PCs instead of large servers), keeping the usage of resources of the crawler small is important. The typical policy used in practice is the breadth-first search that generates large queue of pending pages. In Figure 1 a comparison between the queue size obtained simulating a breadth-first search and a depth-first search over the WebBase [HRGMP00] repository is shown. We present a strategy that has the good properties of breadth-first search in terms of quality and politeness with Web servers, but that uses a queue size comparable to that of depth-first search.

1Laboratory of Web Algorithmics, Dipartimento di Scienze dell’Informazione, Università degli studi di Milano, http://law.dsi.unimi.it/.
Secondary Queue for Large Hubs  Now we are going to show a strategy that can reduce the queue size while preserving coverage and quality of the collection and being polite with Web servers. The key is to remember the nodes with high outdegree and scanning their out-links again when we are out of links during the crawl.

If a node has less than $K$ out-links, then we add all of those links to the queue of pages to visit. If a node has more than $K$ out-links, we pick at random $K$ of them, but put the source node in a secondary queue for re-visiting it later. When the primary queue becomes empty, we take all the nodes in the secondary queue and re-visit them.

This strategy behaves basically as breadth-first and can provide full coverage using only half the size of the queue, as shown in Figure 2. In the figure, we are showing the combined size of both the primary and the secondary queue. The combined size is dominated by the size of the primary queue, as the out-degree exhibits a power-law so the fraction of nodes with many out-links is very small.

We tested this technique in other Web collections made available by the Laboratory of Web Algorithmics (Dipartimento di Scienze dell’Informazione, USM, available online at http://law.dsi.unimi.it/). We used two snapshots of 40 million pages from the .it domain and 18 million pages from the .uk domain. In both cases, we observe the same behavior in terms of queue size and coverage, showing the robustness of our approach.

Only a few nodes have to be visited more than once. In Table 1 we show the maximum queue size, and the fraction of nodes that have to be re-visited in the WebBase collection, for different values of $K$. There is an interesting trade-off between the number of maximum out-links taken per page and the maximum queue size. For instance, taking $K = 16$, the maximum queue size grows from 50% to 64% but only 1% of the pages have to be visited more than once.

Table 1: Maximum queue size and fraction of re-visits for the strategy with a secondary queue. The coverage is always 100%.

<table>
<thead>
<tr>
<th>Max. outlinks ($K$)</th>
<th>All</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. queue size (w.r.t. BFS)</td>
<td>0.14</td>
<td>0.12</td>
<td>0.11</td>
<td>0.09</td>
<td>0.07</td>
</tr>
<tr>
<td>Re-visited nodes</td>
<td>100%</td>
<td>86%</td>
<td>79%</td>
<td>64%</td>
<td>50%</td>
</tr>
</tbody>
</table>

2.3 Analysis of Wikipedia time evolution

In many aspects the Web graph and the Wikigraphs are very similar, and to a certain extent this study provides several hints about how the Web at large is evolving and may continue to evolve.
in the future, in particular with respect to its connectivity. In [DELIS-TR-0347] we try to gain insights about the evolution of an hyperlink structure from the study of a coherent, information-rich corpus. We performed a series of measurements and observed that the Wikigraph resembles many characteristics of the Web graph. The core of this study was the temporal analysis of Wikigraphs, where we made a large number of experiments on the evolution over time of the topological and statistical properties of Wikigraphs and made several observations on the frequency of update of the articles of Wikipedia.

**Wikipedia:** On the Wikipedia, our previous paper [CSC\textsuperscript{+}06] studies the bow-tie structure in the non-English Wikipedias, and Reference [VWD04] deals with the graphical representation of the history of an article.

The Wikipedia is also an excellent source of data for other Information Retrieval tasks. For instance, currently the INEX initiative [http://inex.is.informatik.uni-duisburg.de/2006/](http://inex.is.informatik.uni-duisburg.de/2006/) provides a dataset of Wikipedia articles annotated with topic and relevance assessments for research purposes.


**Remarkable findings:** The observation of Wikipedia provides mixed signals of growth and maturity of this collection.

**Signs of transient regime (growth):**

- The number of articles, updates, visitors and editors is still growing exponentially.
- The size of articles is still growing linearly.
- The number of links per article is also growing linearly, slowly than the amount of text.
- The number of reverts is growing slowly, which may signal more vandalism, but the number of double reverts (revert wars) has stabilized.

**Signs of permanent regime (maturity):**

- There is a clear power-law distribution of the indegree and outdegree.
- The average edits per user has been mostly constant in the last two years.
- There is a high correlation between PageRank and indegree, indicating that the microscopic connectivity of the encyclopedia resembles its mesoscopic properties.
- The clustering coefficient and edge reciprocity of links have remained basically constant during the last two years.
- Over 2/3 of the articles belong now to the larger strongly connected component.

These are the first observations with this degree of detail of the evolution of a large hyperlinked corpus. In the future, we expect to relate this study with the observed evolution of large samples of pages from the Web.
2.4 Spamindexing: a growing phenomenon in the Web

Since its inception, the Web has been considered as a privileged means for free exchange of information and resources among users. Unfortunately this feature has been revealed also its major drawback, since it has catalyzed the (more or less legitimate) interests of ventures seeking for easy profits against low investments and risks.

The economic connotation of the Web is nowadays so strong to have led to new economic paradigms as e-business, e-commerce and so on. Also the definition of “spam” that has been commonly used to refer to unsolicited (and possibly commercial) bulk messages received by e-mail, has assumed a different meaning. In [GGM05], Web spamming is defined as “any deliberate action that is meant to trigger an unjustifiably favorable relevance or importance for some Web page, considering the page’s true value”. A spam page is a page that is used for spamming or receives a substantial amount of its score from other spam pages. This is due to the fact that the most of the informational and transactional needs of people all over the world are satisfied by querying search engines. Since the most of the users click on the first few results in a search engine, there is an economic incentive for manipulating search engine’s listings by creating pages that score high independently of their real merit. We refer to search engine spamming also known as spamindexing, as the excessive manipulation to influence search engine ranking often for pages which contain little or no relevant content. Search Engine spam has not to be confused with legitimate search engine optimization (SEO) that involves getting a site the exposure it deserves.

2.4.1 A reference collection for Web Spam

We made available a public collection for research on Web spam. This collection is the result of efforts by a team of volunteers. The corpus is a large set of Web pages in 11,000 .uk hosts downloaded in May 2006 by the Laboratory of Web Algorithmics, USM. The labeling process was coordinated by UDRLS. Volunteers were provided with a set of guidelines and were asked to mark a set of hosts as either normal, spam, or borderline. The collection includes about 6,700 judgments done by the volunteers and can be used for testing link-based and content-based Web spam detection and demotion techniques. Volunteers were provided with a set of guidelines and were asked to mark a set of hosts as either normal, spam, or borderline.

The resulting dataset is composed of three parts:

1. Human-assigned labels for a set of hosts.
2. URLs and hyperlinks, available from the download page of UK-2006 at the Laboratory of Web Algorithmics, USM. The graph can be read using the tools of the WebGraph framework.
3. HTML page contents.

A description of this dataset can be found in [DELIS-TR-0405].

2.4.2 Using rank propagation and probabilistic counting for link-based detection

It is well-known that search engines exploit the link structure of the Web to rank results of users’ queries. Therefore a quite straightforward technique to increase the importance of a page is by creating densely connected set of pages able to push score from a group of pages toward the spam page. An example of these structures, known as link farm, is depicted in Figure 3. A page that participates in a link farm may have a high in-degree, but little relationship with the rest of the graph. Heuristically, we call spamming achieved by using link farms topological spamming. In particular, a topological spammer gets its goal since a link farm has topological and spectral properties that statistically differ from those exhibited by non spam pages.
Figure 3: Schematic depiction of the neighborhood of a page participating in a link farm (left) and a normal page (right).

Table 2: Summary of the performance of the different metrics, the ranges in the error rate correspond to a simple classifier with a few rules.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Detection rate</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree (D)</td>
<td>73-74%</td>
<td>2-3%</td>
</tr>
<tr>
<td>D + PageRank (P)</td>
<td>74-77%</td>
<td>2-3%</td>
</tr>
<tr>
<td>D + P + TrustRank</td>
<td>77%</td>
<td>2-3%</td>
</tr>
</tbody>
</table>

This is the kind of page we are interested in. The targets of our spam-detection algorithms are the pages that receive most of their ranking by participating in link farms. Link farms can receive links from non-spam sites by buying advertising, or by buying expired domains used previously for legitimate purposes.

Topological spamming is only one of the possible techniques. Other common approaches are content-based [GGM05]; in this second case, a spam page may use a large number of keyword, repetitive text, meta-tags or content hiding.

Link-based and content-based analysis offer two orthogonal approaches. We think that these approaches are not alternative and should probably be used together since neither of them are capable to capture all possible cases of spamming.

2.4.3 Link analysis for Web Spam Detection

The insertion of structures artificially generated around a page introduces local variation in some of the topological properties of the graph. In [BCD+06a] we showed that a simple classifier built using common metrics as degree, edge reciprocity, PageRank and TrustRank can be used to detect a rate of spam pages between 73% to 77% with a false positive rate of 2% to 3%. The performance of the different classifiers we built is summarized in Table 2.

But a significant improvement in the classification accuracy has been achieved using two link-based algorithms that we presented in [BCD+06b] and we briefly explain in Paragraph 2.4.3 and 2.4.3. Our main contributions are:

- We introduce a damping function for rank propagation [BYBC06] that provides a metric that helps in separating spam from non-spam pages.
- We propose a new technique for link spam detection that exploits the distribution of the number of Web page supporters with respect to distance. To this purpose, we present an improved approximate neighborhood counting algorithm [PGF02].
- We suggest an automatic classifier that only uses link attributes, without looking at Web page content, still achieving a precision that is equivalent to that of the best spam classifiers that use content analysis. This is an important point, since in many cases spam pages exhibit pretty “normal” contents.

Our algorithms are tested on a large sample of the .uk domain where thousands of domains have been inspected and manually classified as spam or non-spam domains. This sample was downloaded in 2002 by the Dipartimento di Scienze dell’Informazione, Università degli studi di Milano.

**Truncated PageRank** In [BCD+06b] we described Truncated PageRank, a link-based ranking function that decreases the importance of neighbors that are topologically “close” to the target node. In [ZGG+04] it is shown that spam pages should be very sensitive to changes in the damping factor of the PageRank calculation; in our case with Truncated PageRank we modify not only the damping factor but the whole damping function.

Intuitively, a way of demoting spam pages is to consider a damping function that removes the direct contribution of the first levels of links, such as:

\[
damping(t) = \begin{cases} 
0 & t \leq T \\
C\alpha^t & t > T 
\end{cases}
\]

Where \( C \) is a normalization constant and \( \alpha \) is the damping factor used for PageRank. This function penalizes pages that obtain a large share of their PageRank from the first few levels of links; we call the corresponding functional ranking the **Truncated PageRank** of a page. The calculation of Truncated PageRank is described in detail in [BCD+06b]. There is a very fast method for calculating Truncated PageRank. Given a PageRank computation, we can store “snapshots” of the PageRank values at different iterations and then take the difference and normalize those values at the end of the PageRank computation. Essentially, this means that the Truncated PageRank can be calculated for free during the PageRank iterations.

Note that as the number of indirect neighbors also depends on the number of direct neighbors, reducing the contribution of the first level of links by this method does not mean that we are calculating something completely different from PageRank. In fact, for most pages, both measures are strongly correlated, as shown in [BCD+06b].

![Figure 4: Histogram of the ratio between TruncatedPageRank at distance 4 and PageRank in the home page.](image1)

![Figure 5: Maximum ratio change of the TruncatedPageRank from distance \( i \) to distance \( i - 1 \).](image2)

In practice, we observe that for the spam hosts in our collection, the Truncated PageRank is smaller than the PageRank, as shown in Figure 4. There is a sharp peak for the spam pages in low values,
meaning that many spam pages lose a large part of their PageRank when Truncated PageRank is used. We also found that studying the ratio of Truncated PageRank at distance $i$ versus Truncated PageRank at distance $i - 1$ also helps in identifying Web spam, as shown in Figure 5. A classifier using Truncated PageRank, as well as PageRank and degree-based attributes (60 features in total) can identify 76.9% to 78.0% of the spam hosts with 1.6% to 2.5% of false positives.

Estimation of supporters  Following [BCSU05], we call $x$ a supporter of page $y$ at distance $d$, if the shortest path from $x$ to $y$ formed by links in $E$ has length $d$. The set of supporters of a page are all the other pages that contribute to its link-based ranking.

A natural way of fighting link spam is to count the supporters. The naive approach is to repeat a reverse breadth-first search from each node of the graph, up to a certain depth, and mark nodes as they are visited [LN89]. Unfortunately, this is infeasible unless a subset of “suspicious” node is known a priori. A method for estimating the number of supporters of each node in the graph is described in [BCD+06b] which improves [PGF02].

The general algorithm (described in detail in [BCD+06b]) involves the propagation of a bit mask. We start by assigning a random vector of bits to each page. We then perform an iterative computation: on each iteration of the algorithm, if page $y$ has a link to page $x$, then the bit vector of page $x$ is updated as $x ← x OR y$. After $d$ iterations, the bit vector associated to any page $x$ provides information about the number of supporters of $x$ at distance $\leq d$. Intuitively, if a page has a larger number of supporters than another, more 1s will appear in the final configuration of its bit vector.

The algorithm is described in detail in [BCD+06b]. In order to have a good estimation, $d$ passes have to be repeated $O(\log N)$ times with different initial values, because the range of the possible values for the number of supporters is very large. We have observed that counting supporters from distances $d$ from 1 to 4 give good results in practice. We measured how the number of supporters change at different distances, by measuring, for instance, the ratio between the number of supporters at distance 4 and the number of supporters at distance 3. The histogram for the minimum and maximum change is shown in Figure 6.

![Figure 6: Histogram of the ratio minimum change in the size of the neighborhood in the first few level vs distance 3.](image6)

![Figure 7: Histogram of the number of different neighbor hosts at distance 4 vs distance 3.](image7)

This algorithm can be extended very easily to consider the number of different hosts contributing to the ranking of a given host. To do so, in the initialization the bit masks of all the pages in the same host have to be made equal. In Figure 7, we plot the number of supporters at distance 4 considering different hosts contributing towards the ranking of the home pages of the marked hosts. We observed anomalies in this distribution for the case of the spam pages, and these anomalies are more evident by counting different hosts than by counting different pages. Considering distance 4, the estimation of supporters based on pages (62 attributes) yields a classifier with 78.9% to 77.9% of detection rate
and 1.4% to 2.5% of false positives. If we base the estimation on hosts (67 attributes, slightly more because in-degree is not the number of neighbors at distance one in this case) allows us to build a classifier for detecting 76.5% to 77.4% of the spam with an error rate from 1.3% to 2.4%.

The detection rate is two to three percentage points lower if distance 2 is considered, with roughly the same false positives ratio.

2.5 Enumerating dense clusters in data streams

The analysis of the structure of large networks often requires the computation of network indices based on counting the number of certain small subgraphs. In the analysis of complex networks, the clustering coefficient is an important measure of the density of clusters in graphs and the degree at which clusters decompose into communities. The clustering coefficient of a graph is the average fraction of connected neighbor pairs of a vertex. Related to the clustering coefficient is the transitivity coefficient, which is the ratio between three times the number of triangles and the number of paths of length two in the graph.

Frequent subgraphs in networks are also called motifs. Motifs are considered as the building blocks of universal classes of complex networks. Detecting motifs can, for example, shed light on the process of network formation. Specific motifs can be found with similar frequency in complex networks originated from the same application domain, as for instance in biochemistry, neurobiology, ecology, and engineering.

In the domain of Web applications, certain dense subgraphs of small size in the Webgraph, the graph formed by Web pages and hyperlinked connections, have been considered in the attempt of tracing the emergence of hidden cyber-communities. These subgraphs are typically dense bipartite cliques of small size, that are interpreted as cores of the communities. The vertices on the left side of the clique are considered member pages all pointing to a set of centers/authorities for the community. A large number of these subgraphs has been observed in large web crawls. A well-known model of the process of growth of the hyperlinked structure of the Web, denoted by copying model, uses these dense subgraphs as building blocks of the process of formation of the Webgraphs.

Counting the number of certain subgraphs in a large graph is a challenging computational task. The current state of the art provides methods that are either computational unfeasible on large data sets or do not provide any guarantee on the accuracy of the estimation. The best known methods for the solution of the simplest non trivial version of this problem, i.e. counting the number of triangles in a subgraph, reduces to matrix multiplication. This is not computational feasible even on graphs of medium size, because of time complexity and the space required to store the whole graph and the related data structures in main memory.

A natural way to address the problem of computing with massive data sets is to resort to the data stream model. In this model data arrives in a stream, one item at a time, and the algorithms are required to use very little space and per-item processing time. Secondary and slower memory storage devices naturally produce data streams for which multiple passes of computation are usually prohibitive due to the volumes of stored data. In several network contexts, the application receive data without pace from remote sources. Data stream computation allows also to compute on-line relevant quantities without incurring a large cost for organizing and storing data. We refer for instance to distributed crawlers collecting Web pages and their links, and performing structural analysis of the Webgraph prior to transfer data to a storage device.

In [DELIS-TR-0417] we consider the following model of computation for directed and undirected graphs in data streams. Let \( G = (V, E) \) be a directed graph without self-loops. Our input data is a stream of directed edges. For our algorithms we either assume that these edges are either grouped by their source or by destination vertex. The order among source or destination vertices can be arbitrary.

In this model we present random sampling data stream algorithms for computing the clustering
coefficient, transitivity coefficient, and the number of bipartite cliques in a graph in the incidence stream model. Our algorithms find applications to the problems of detecting the existence of dense clusters in a graph.

To compute the transitivity coefficient it essentially suffices to compute the number of triangles in a graph. We present a data structure for this task that uses $O\left(\frac{1}{\varepsilon^2} \log(|V|) (1 + \frac{|T_3|}{|T_2|})\right)$ memory cells. Observe that $\frac{|T_3|}{|T_2|}$ is exactly equal to $\frac{1}{3}$ of the inverse of the transitivity coefficient of the graph, an universal measure whose value for networks of practical interest is hardly bigger than $10^5$. We present a 1-pass streaming algorithm which with probability $1 - \delta$ returns a $(1 \pm \varepsilon)$-approximation on the clustering coefficient $C_G$ of a graph $G$ when the graph is given as a incidence stream. It needs $O\left(\log \log |V| \cdot \frac{1}{\varepsilon^2} \cdot C_G\right)$ memory cells.

Denote by $K_{i,j}$ the set of complete bipartite cliques in the graph where each of $i$ vertices link to all of $j$ vertices. As a further contribution we provide a data stream algorithm that provides an approximation of the number of $K_{3,3}$ of the graph in the incidence stream model ordered by destination nodes with outdegree bounded by $\Delta$ which needs $O\left(\log(|V|) \cdot \frac{|K_{3,3}| \cdot \Delta^2 \ln(\frac{1}{\delta})}{|K_{3,3}| \cdot \varepsilon^2}\right)$ memory cells. We also provide an optimized implementation of the two pass version of the presented data stream algorithms and a test on networks including large web graphs, graphs of the largest online encyclopedia Wikipedia, graphs of collaborations between actors and authors.

Our algorithm for approximating the clustering and transitivity coefficient provide excellent approximations with a sample of size $10^5$. For the algorithm that estimates the number of bipartite cliques, we find out that a number of $10^5$ samples already suffices to detect a large number of bipartite cliques.

### 2.6 Decomposition-based analytic Visualization of Data Bias for Topological Analysis
#### 2.6.1 Comparison of Data Sources for the AS Network Based on Analytic Visualizations

The contemporary Internet is a collection of segregated routing domains called Autonomous Systems (AS), each having their own administration and independent routing policies. Routing information between ASes is exchanged via an exterior gateway protocol such as BGP [BGP]. The graph of the ASes, where nodes represent different ASes, and edges correspond to traffic trade agreements between the ASes, provides us with an abstraction of the Internet underlay. The source of data we used to model the Internet at the level of the Autonomous Systems, was the data from the Oregon Routeviews Project [Routeviews]. The Oregon Routeviews Project is one of the major repositories for snapshots of the AS network using looking glasses. Another reliable source for data on the AS network is the DIMES data set from the EVERGROW framework. In contrast to Routeviews, DIMES extracts AS relations by traceroute experiments. Note that a similar model for the exploration methods of DIMES is analyzed in WP1.1 by the group at ETHZ. Comparing the Routeviews AS topological map in Figure 2.6.1 with that of DIMES in Figure 2.6.1, we observe that the visualizations do not indicate a significant difference. In Figure 9 we visualize both the intersection of the two data sets. Here, different grey correspond to the different edge sets, i.e., light grey to DIMES, dark grey to Routeviews and black to the intersection. Again both the union and the intersection strongly suggest, that the core structure discovered by the the two data sources are highly similar.

The data sets correspond to the period of March to June 2005. We obtain 48,073 edges (corresponding to 20,406 ASes) from Routeviews and 38,928 edges (corresponding to 14,154 ASes) from DIMES. Of these, 21,725 edges exclusively belong to Routeviews, and 12,580 edges exclusively to DIMES. The rest of the edges are common to both data sets. The union of the two data sets thus results in 60,653 unique edges (corresponding to 20,612 ASes). Two interesting, but peripheral, observations we addressed are that as much as 58% of the edges appear in only one data set, and that many edges only discovered by DIMES are incident to the core. Since visualizations cannot
fully replace a mathematical analysis, we corroborate our observed indications by an examination of the core distributions in both data sets. Basically we compare essential features, such as the degree and the core distribution, yet cores are more important to our analysis than degree, thus we show the core distributions in Figures 11 and 10. The Routeviews data sample is plotted as a solid line, while the DIMES sample is dotted. Figure 2.6.1 plots the maximum coreness of the end-nodes (as $y$-axis), while Figure 2.6.1 shows the minimum coreness. We repeat these distribution plots for the sets of edges that are exclusive to Routeviews and Dimes in Figures 2.6.1 and 2.6.1. Summing things up, the distributions in Routeviews and DIMES are very similar, except for the broad tail of the Routeviews distribution observed in Figure 2.6.1, which is an interesting observation requiring further investigation. However, the overall similarity of the plots and the resembling visualizations reveal that Routeviews and DIMES data is indeed similar, hence our topological analysis is unaffected by the source of data.

2.6.2 Stability of the AS Core Structure Over Time

To ensure that an analysis of the AS graph structure is not biased by the time of measurement, we analyze the macroscopic temporal evolution of the AS graph obtained from Routeviews over a longer period of time. First, we observe that during the period of April 2001 to April 2005, the number of nodes in the AS graph increases by about 2000 nodes per year, the number of edges increases by 4800 edges per year and the maximum core number has increased from 18 to 26. Although the network grows in absolute terms and especially, the individual core levels grow, their relative sizes remain stable, as we now show. Note that we cannot analyse DIMES in the same way due to the lack of available data.

Similar to the rings of a tree trunk, Figure 12 illustrates the temporal evolution of the relative proportions of the k-shells, i.e., collection of nodes with coreness k. In this figure, the thickness of
Figure 10: Comparison of coreness distributions of the edges in Routeviews (solid) and DIMES (dotted). The x-axis denotes the number of edges, and y-axis the minimum or maximum endnode coreness.

Figure 11: Comparison of coreness distributions of the exclusive sets of edges in Routeviews (solid) and DIMES (dotted). The x-axis denotes the number of edges, and y-axis the minimum or maximum endnode coreness.

one strip corresponds to the fraction of nodes that have a given coreness. The lowest strip represents the maximum core while the highest strip reflects the 1-shell. Although the evolution of the core structure obviously is subject to major fluctuations, the visualization clearly reveals a stability of the relative core sizes over time.

Figure 13 is a top-down view of the visualization technique [DELIS-TR-0036], relating the coreness of an AS to its position in the layout very well: nodes with large coreness (dark) are placed in the center while nodes with small coreness (light) are placed in the periphery.

With this background, we present snapshots of the AS network over four years in Figure 14, where only nodes with coreness two (white) and three (grey) are drawn. The black color in the center is due to edges. It is apparent that the annuli remain constant over time. Most nodes of the two- and three-shell are placed in the periphery of the layout which reflects their position in the AS hierarchy.
Figure 12: Relative size of cores. X-axis denotes time and y-axis (logarithmically scaled) denotes the fraction of nodes in k-shell.

Figure 13: Visualization (top-down) of the AS network (Jan 1, 2005). Small white nodes have small coreness while big black nodes have large coreness.

References


Figure 14: Snapshots of the AS network (2002-2005) where only the second (white) and third (grey) shells are drawn.


[DELIS-TR-0417] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, and Christian Sohler. Enumerating dense clusters in data streams. A short abstract of this paper has been presented to the European Conference on Complex systems (ECCS 06), 2006.

[Gnutellav0.6] Gnutella v0.6 RFC. http://www.the-gdf.org/.


