Agnostic Topology-Based Spam Avoidance in Large-Scale Web Crawls

Clint Sparkman[†]
Texas A&M University
College Station, TX 77843 USA
Email: sparkman@cse.tamu.edu

Hsin-Tsang Lee
Microsoft Corp.
Redmond, WA 98052 USA
Email: htlee@microsoft.com

Dmitri Loguinov[*]
Texas A&M University
College Station, TX 77843 USA
Email: dmitri@cse.tamu.edu

***Abstract*—With the proliferation of web spam and questionable content with virtually infinite auto-generated structure, large-scale web crawlers now require low-complexity ranking methods to effectively budget their limited resources and allocate the majority of bandwidth to reputable sites. To shed light on Internet-wide spam avoidance, we study the domain-level graph from a 6.3B-page web crawl and compare several agnostic topology-based ranking algorithms on this dataset. We first propose a new methodology for comparing the various rankings and then show that in-degree BFS-based techniques decisively outperform classic PageRank-style methods. However, since BFS requires several orders of magnitude higher overhead and is generally infeasible for real-time use, we propose a fast, accurate, and scalable estimation method that can achieve much better crawl prioritization in practice, especially in applications with limited hardware resources.**

I. Introduction

Competition for high placement in search engine results has recently created a financial incentive for many unethical Internet practices intended to deceive (i.e., spam) search engines and manipulate their ranking algorithms. In addition to adversely impacting the quality of commercial search results, web spam frequently impedes web exploration for various research purposes in areas of networking, data mining, information retrieval, and data-intensive computing. As a result of this adversarial nature of the web, many large-scale crawlers are now faced with challenges of detecting and avoiding "undesirable" content using real-time algorithms that must possess not only sufficient accuracy, but also reasonable overhead to be practical.

In a research setting, these tasks are especially difficult due to the lack of exorbitant financial resources needed to build enormous server clusters of commercial search engines and extreme secrecy surrounding the algorithms and data of these companies. Our thrust to overcome these challenges has led to a high-performance crawler called IRLbot [24] that can perform multi-billion-page web exploration using a single server, which is accomplished using several novel algorithms for verification of URL uniqueness, reputation ranking, and enforcement of budgets. Unlike previous crawlers [4], [5], [10], [12], [19], [20], [27], [28], [30], [31], [33] that manage spam only *after* the crawl has finished, IRLbot prioritizes queued URLs using budgets based on domain popularity estimated using real-time snapshots of the Pay-Level Domain (PLD) graph.

As the name suggests, PLDs are domains that must be purchased/acquired at a TLD or cc-TLD registrar (e.g., google.com, amazon.co.uk). IRLbot constructs PLD graphs by condensing all pages contained within a PLD into a single node and discarding duplicate edges in the resulting graph. It then uses in-degree (IN) at each node for determining the corresponding reputation. The number of incoming links can be viewed as a generic weight of endorsement from other domains. While this may be a poor technique at the page/site level, because it is susceptible to trivial inflation using dynamic scripts, rank manipulation at the PLD level becomes much more difficult as it requires hijacking links from a large number of legitimate PLDs and/or non-trivial expense associated with registering these domains using legal means. In addition, PLD graphs are dramatically smaller in size than alternative data structures used in prior work [3], [13], [14], [18], [38], which enables much more efficient ranking during large crawls.

A. Prioritization

Given the virtually unlimited web space, crawlers need methods to budget their finite resources so as to spend most of their time exploring valuable parts of the Internet. Considering that large sites (e.g., ebay.com with 260M pages or yahoo.com with 1.5B) can also contain legitimate information, the main problem examined in this work is *how to differentiate between domains that should be massively crawled and those that should not*.[1] There are two performance metrics in achieving this classification – overhead and accuracy – where the former term refers to the amount of processing needed to compute reputation and the latter term (discussed in more detail below) describes the method's ability to avoid over-allocation of resources to low-quality domains.

Consider a crawler that uses link structure of the seen web to dynamically compute domain *reputation scores* $s_1, \ldots, s_n$ using some prioritization function $\mathcal{P}$, where n is the total

[†]Major Clint Sparkman is an active duty member of the United States Air Force. The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

[*]Supported by NSF grants CNS-0720571 and CNS-1017766.

[1]Note that this problem is entirely different from search-related ranking of results, which involves fine-granular decisions about the quality of each page.

number of PLDs known to the crawler. Instead of directly using these scores, the crawler sorts set $\{s_i\}$ and obtains *reputation ranking* $r_1, \ldots, r_n$, where r_i is the position of domain i in the sorted list. This ranking is then used in the budgeting algorithm to decide the amount of crawling resources (e.g., bandwidth, pages downloaded) allocated to each domain. The budget function provides tiny resources to all nodes with $r_i \geq R$, where R is some threshold, and significantly larger budgets to the remaining domains.

As a result, the goal of the reputation algorithm is not to identify each spam page or detect malicious content [1], [2], [3], but rather to prevent domains with poor or unknown quality from admission into the top list. While [24] has shown that IRLbot's ranking function was correlated with the number of pages pulled from each domain, there is no evidence that domains with high ranking in the obtained dataset were in fact overwhelmingly reputable. Thus, our main goal in this paper is to experimentally study the IRLbot dataset, which is the largest non-commercial crawl to date, and understand how well the various ranking mechanism are able to detect reputable Internet resources, promote them to the top, and avoid spam, while remaining computationally feasible.

Since the baseline algorithm in IRLbot is completely *agnostic* (i.e., does not utilize human input or training on known spam) and relies on just the topological structure of the web, we compare its ranking technique IN to three other main methods in the same category – *Weighted In-Degree* (WIN) [9], PageRank [5], and level-2 *Supporters* (SUPP) [2]. While some of these techniques have been applied to page/host-level graphs, their usage on a PLD graph is novel.

B. Contributions

We first introduce a new comparison methodology that examines only the top-R nodes suggested by each algorithm, as opposed to random nodes drawn from the entire graph [3]. We start with manual spam analysis using $R = 1$K; however, since manual comparison is not only a labor-intensive, but also a rather subjective task, we later augment it with two automated approaches, neither of which has been attempted at this scale before. The first method relies on Google's opinion of each domain, which can be inferred from the corresponding Google Toolbar Rank (GTR) [15]. The second technique utilizes known spam lists to understand how many blacklisted PLDs are highly ranked by each studied technique.

We find from our manual analysis of IRLbot's PLD graph with 89M nodes and 1.8B edges that WIN and PageRank produce significantly worse rankings than the other two methods, admitting 39+ spam PLDs into the top-1K list and exhibiting low-GTR domains throughout the entire ranking range $[1, R]$. In both cases, the highest ranked spam PLD appears in the top 10. We also discover that these two techniques have an almost identically poor ranking, despite the fact that PageRank requires substantially higher overhead. In contrast, IN allows only 9 spam PLDs in the top-1K list, with its first spam domain appearing in position 25. SUPP, being a BFS generalization of IN to two hops, performs even better, with only one spam PLD in position 718.

Leveraging the GTR data collected using an automated process and a popular email spam blocklist [32] with $60,701$ PLDs appearing in IRLbot data, we find that SUPP outperforms the other algorithms in every comparison and often admits 10 times fewer undesirable PLDs into its top-10K list. We also observe that IN again decisively outperforms PageRank and WIN, with its top-ranked sites not only carrying a higher GTR, but also containing fewer blacklisted and low-GTR PLDs. Interestingly enough, this occurs because PageRank and WIN both penalize nodes with high out-degree by splitting their support along the outgoing edges and thus under-represent the true value of extremely popular domains. This allows spam to rival them in the computed score and achieve unbelievingly high positions in the list.

Although SUPP is superior in every comparison, it requires an enormous amount of CPU processing due to elimination of duplicates in BFS and a huge number of random RAM accesses (i.e., 5 trillion in our case). This makes it difficult to incorporate into a high performance web crawler. To overcome this problem, we propose a novel technique to approximate SUPP, which we call *Top Supporters Estimation (TSE)*, and compare it against the exact SUPP algorithm and the Bit Vector (BV) estimation method from [2]. On the PLD graph, we find that TSE achieves less than 1% error while reducing the running time of SUPP by a factor of $36,000$. It is also 10 times faster than BV with 6 times lower error. We also discuss an external-memory application of TSE when the graph does not fit in RAM and show that its disk I/O overhead is $1,000$ times less than SUPP's, with $4,000$ times less RAM usage. TSE additionally reduces BV's RAM requirement by three orders of magnitude, all of which makes it highly appealing for real-time use in research crawlers with limited resources.

II. Related Work

Much academic and corporate effort has gone into designing web crawlers [4], [5], [10], [12], [21], [22], [27], [28], [29], [30], [31], [40]; however, reaching the full scale of the web using a non-commercial setup has turned out to be quite challenging. Typical university crawls report $100-150$M pages [10], [31], whereas common public datasets contain $10-100$M pages downloaded during the various years in the last decade [20], [23], [35], [39]. Industry labs report $1-5$B pages [16], [28] and commercial search engines were recently observed to index $30-40$B pages [24] with links to over 1T unique URLs [34]; however, not much is known about their implementations and domain coverage. With the exception of IBM's analysis in [13], which obtained 11 porn pages in the top-20 list sorted by PageRank, no prior literature has attempted to analyze the top-ranked nodes of any Internet-wide structure. Furthermore, PLD graphs have not been constructed or analyzed for any of the previous crawls.

We next briefly describe the ranking algorithms studied below. Assume an n-node directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. Denote by

$d(i,j)$ the shortest distance from i to j along the directed edges in G. Then, the total level-D support of j is defined as:

$$SUPP_D(j) = \sum_{i=1}^{n} \mathbf{1}_{d(i,j)=D}, \quad (1)$$

where $\mathbf{1}_A$ is an indicator variable of event A. Using $D = 1$, we obtain IN (already discussed above), which simply counts the in-degree at each node excluding self-loops [25], [26]. For $D \geq 2$, we call the algorithm *Supporters* (SUPP) [2] and note its equivalence to backwards BFS from each node to depth D. As we show below, this is a computationally expensive task, especially when G does not fit in RAM or is highly branching. To our knowledge, no prior paper has computed exact (i.e., non-estimated) SUPP$_D$ with $D \geq 2$ on real web graphs or attempted to use it for ranking as a standalone method.

Our third technique is the classic *PageRank* [5], which models a random walk on G, where the walker either traverses one of the outgoing edges with probability $\alpha = 0.85$ or teleports to a random node with probability $1 - \alpha$. The PageRank score π_j of page $j \in V$ is the stationary probability for the walker (i.e., the underlying discrete Markov chain) to be found at j and is given by the solution to the following recurrence:

$$\pi_j = \alpha \sum_{i:(i,j)\in E} \frac{\pi_i}{d_{out}(i)} + \frac{1-\alpha}{n} \quad (2)$$

where $d_{out}(i)$ is the out-degree of node i. While many modifications to PageRank exist, a vast majority of them require oracle input in the form of white/blacklists [6], [18], [37]. As a result, they are beyond the scope of this paper.

A single iteration of PageRank with $\alpha = 1$ and all π_i replaced with 1 leads to our last method, which is called *Weighted In-Degree* (WIN) in [9]. The rationale of this method is to assume that each node i in the graph has one unit of "authority" that it distributes equally between the nodes to which it links. WIN is often used as a low-overhead approximation to PageRank.

III. Methodology

We next outline our objectives in the evaluation and present our approach for checking the reputation of top domains.

A. Objectives

Controlling the reputation of downloaded content not only reduces bandwidth waste as discussed in the introduction, but also curtails the influence of unreputable pages on the collected dataset. Suppose G_∞ is the infinite web graph and $G_\mathcal{P} \subset G_\infty$ is its representation captured by a crawler with prioritization function $\mathcal{P}$. If $G_\mathcal{P}$ has a significant bias towards domains with undesirable content, it is possible that the search engine may have a hard time discerning good pages from bad in $G_\mathcal{P}$. For example, PageRank [5] teleports randomly to each page with probability $1/n$. Therefore, the more spam pages the crawler has visited, the larger their combined teleportation mass and the more the support to their target pages. In fact, a spammer may not need to attract many external links if the crawler covers its spam farm with sufficient diligence.

By altering the collected graph using a prioritization scheme $\mathcal{P}$, the crawler can reduce the weight of undesirable pages *preemptively* and provide a much cleaner input to the ranking algorithm used later by the search engine. However, very little evidence exists about what nodes dominate $G_\mathcal{P}$ among the existing crawlers [4], [5], [10], [12], [19], [20], [27], [28], [30], [31], [33], and no evaluation standard exists. Therefore, we first design a framework for understanding what domains are most prominent in $G_\mathcal{P}$ and then examine whether IRLbot's ranking function can be improved using other low-overhead agnostic techniques.

B. Manual Spam Evaluation

Evaluating any web-ranking algorithm is challenging for two reasons. The first problem is the absence of a common algorithm by which to measure ranking results in spam-avoidance applications. Related work [3], [6], [18], [37] usually selects a small random sample of the graph and manually classifies it to determine if each page is good or spam. Then, the entire ranking list is split into K buckets and the algorithms are compared against one another based on how many of the identified spam pages are contained within each bucket.

Due to our interest only in the most reputable PLDs, we offer a different approach. We first assign reputation scores to all nodes, sort the graph by the score, and then scrutinize the top-1K PLDs in each ranking for spam.[2] For each domain, we first attempt the PLD itself (e.g., http://amazon.com) and, if unsuccessful, then consult the list of hostnames seen by IRLbot from that PLD to determine the shortest alive website, which is then used as the representative for that domain. We follow all redirects and end up excluding 56 PLDs for which none of the websites are alive.

The second obstacle in comparing ranking is that there is no consensus on the definition of web spam. Some researchers in the field classify all pornographic pages as spam [6], while others define a site as spam if it employs techniques intended to trick search engines into ranking some pages higher than they deserve [17]. While we generally aim to identify the latter types of PLDs, it is impossible to detect through manual inspection cloaking, hidden links, and high-density farms/alliances pointing to some target page. We therefore rely on a *subjective* determination of what constitutes spam using two factors – attempts to perform malicious activities upon visit (e.g., install malware or viruses) and overwhelming presence of links whose main purpose is to create revenue from click-throughs (i.e., no immediately useful content can be discerned in the PLD).

In addition to classifying each PLD as spam or not, we also record the *Google Toolbar Rank* (GTR) for each opened page, which we use to estimate the relative value of each PLD and achieve an additional level of granularity in the

[2]While the top-1K domains in SUPP$_2$ ranking represent only 0.001% of the graph, they are responsible for 18% of Google's index (i.e., 5.5B pages).

TABLE I
TOP 15 RANKED PLDS ('S' IS SPAM, 'Q' IS QUESTIONABLE)

IN		PageRank		WIN		$SUPP_2$	
PLD	GTR	PLD	GTR	PLD	GTR	PLD	GTR
microsoft.com	9	microsoft.com	9	microsoft.com	9	google.com	10
google.com	10	adobe.com	10	information.com (S)	5	microsoft.com	9
yahoo.com	9	google.com	10	google.com	10	yahoo.com	9
adobe.com	10	information.com (S)	5	adobe.com	10	adobe.com	10
blogspot.com	9	macromedia.com	10	macromedia.com	10	macromedia.com	10
wikipedia.org	9	yahoo.com	9	yahoo.com	9	wikipedia.org	9
w3.org	10	sedoparking.com (S)	–	sedoparking.com (S)	–	blogspot.com	9
geocities.com	9	googlesyndication.com	–	miibeian.gov.cn	9	msn.com	8
msn.com	8	w3.org	10	googlesyndication.com	–	apple.com	9
amazon.com	9	miibeian.gov.cn	9	w3.org	10	geocities.com	9
aol.com	8	downloadrings.com (S)	1	ndparking.de (Q)	–	w3.org	10
myspace.com	9	chestertonholdings.com (Q)	–	statcounter.com	9	sourceforge.net	9
macromedia.com	10	juccoholdings.com (Q)	–	searchnut.com (S)	–	youtube.com	9
youtube.com	9	statcounter.com	9	revenuedirect.com (Q)	4	bbc.co.uk	9
tripod.com	7	linkz.com (Q)	3	myspace.com	9	netscape.com	8

comparison. Recall that Google offers a toolbar [15] that can be installed as a plugin to a user's web browser to provide, among other things, a rank between $0-10$ for each visited page that reflects Google's opinion of the browsed document. The toolbar returns no GTR for some pages, which occurs for resources that Google has not crawled, pages/domains that no longer exist or generate not-found/forbidden HTTP errors, and content purposely removed from the index. With the exception of [36] that studied a handful of GTRs for select Fortune-500 companies in 2003, no ranking in the literature has involved GTR-based comparison.

IV. MANUAL ANALYSIS

A. Dataset

Our data is obtained from an IRLbot web crawl that took place between June 9 and August 3, 2007. During these 41 days, IRLbot attempted to crawl 8.2B pages, issued 7.6B requests, and successfully downloaded 6.3B 200-OK HTML pages. The resulting webgraph has 41B nodes and 310B edges. The corresponding host graph contains 641M unique sites and 6.8B edges, while the PLD graph G, which we study in this paper, consists of 89M nodes and 1.8B edges. To prevent spammers from getting a boost from an unlimited number of duplicate inter-domain hyperlinks, graph G is *unweighted* (i.e., duplicate edges between the PLDs are removed). The average degree in G is 20.2, with the maximum in-degree $2,948,085$ (microsoft.com) and the maximum out-degree $1,496,327$ (snngr.com). While 89M nodes have non-zero in-degree, only approximately 30M PLDs (out of the 33M crawled) have non-zero out-degree.

Verification activities reported in this paper took place in August 2008 and have not been repeated to remain as close to the date of the original crawl as possible.

B. Top-Ranked PLDs

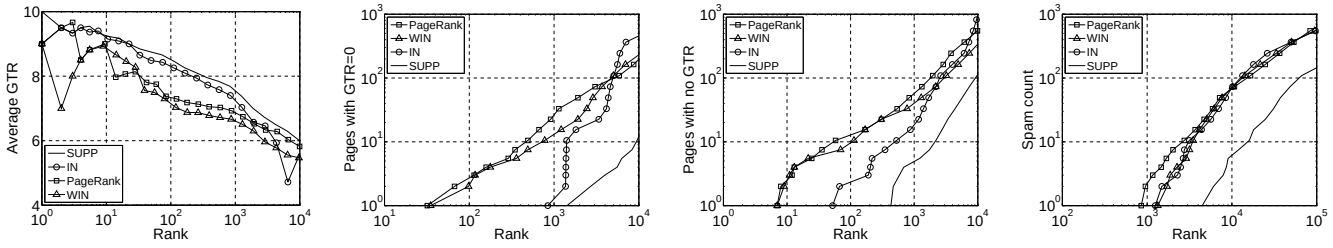
Table I shows the top 15 ranked PLDs for each candidate algorithm. The first part of the table shows that IN's top results are all well-known reputable domains and that 12 of them exhibit a GTR at least 9. The next two lists, belonging to PageRank and WIN, show a great deal of similarity – both rank sedoparking.com in position 7 and information.com in the top 4. After much scrutiny of these two domains, we label them both as spam (the former heavily promotes spam sites and is absent from Google's index; the latter is a revenue-marketing search engine hosting predominantly spam). Going down these ranking lists, one finds additional examples of spam – downloadrings.com and searchnut.com. Further examination of the two PageRank algorithms reveals at least five other suspicious PLDs with low or missing GTRs, raising serious doubts about the ability of these two methods to prevent undesirable PLDs from achieving high ranks in our (as well as any other) large domain-level dataset.

The last part of the table shows the top domains ranked by $SUPP_2$. There is a substantial overlap with the IN list, but $SUPP_2$ is even more impressive with 13 PLDs with a GTR at least 9 and all 15 with a GTR at least 8. In general, one expects that a good ranking algorithm would place only *unquestionably reputable* domains near the top of the list. From the data in Table I, it is clear that IN and $SUPP_2$ both satisfy this requirement, while the other two methods by far do not.

This drastic difference stays virtually the same as we manually inspect the remaining 985 PLDs in each list. Considering the number of PLDs with GTRs no larger than 3, we find that WIN has the most with 54, followed by PageRank with 50, then IN with 5, and finally $SUPP_2$ with only 1 in position 624.

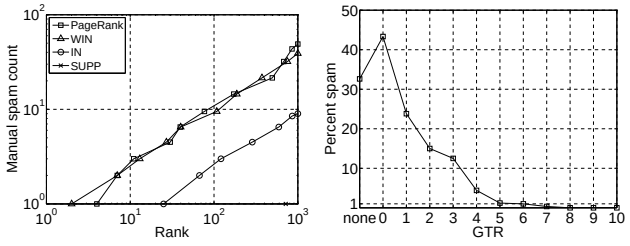
C. Spam Avoidance

The next part of our analysis compares the amount of spam found in the top-1K list of each algorithm during manual inspection. Define u_r to be the number of spam PLDs in position $[1,r]$ in a given ranked list. Figure 1(a) plots u_r vs r for each of the four techniques. As before, PageRank and WIN are significantly inferior to the other two methods, but it is also interesting that spam is discovered at approximately the same rate for these two algorithms as r increases. In total, PageRank and WIN admit respectively 49 and 39 spam PLDs in their top-1K lists, where the latter slightly outperforms the



(a) average GTR (b) GTR-0 (c) no GTR (d) blacklisted PLDs

Fig. 2. Comparison of ranking algorithms.



(a) spam count (b) spam probability

Fig. 1. Manual analysis of spam.

former despite its much lower complexity.

Interestingly, the figure shows that IN does considerably better than either of the PageRank-style techniques. It allows only 9 spam PLDs into the top-1K, the first being the same information.com in position 25. The real winner in this comparison, however, is SUPP_2 whose entire list contains only one spam PLD, which is linksynergy.com in position 718.

D. GTR and Spam

We finish this section by examining how well GTR values predict the occurrence of spam in the total $2,100$ domains verified manually. Figure 1(b) shows the percentage of PLDs with a given GTR that are spam according to our manual analysis. In the figure, the peak occurs at 43% for GTR-0 PLDs; however, it is also likely that this number should be higher given the rather conservative definition of spam used in our decisions. What can be clearly stated based on manual inspection of GTR-0 PLDs is that none of them are well-known, reputable domains. In fact, many of them are suspicious (e.g., related to gambling or pharmaceutical sales), while others are simply obscure. Similarly, PLDs not ranked by Google, which are responsible for the second largest point (i.e., 32%) in Figure 1(b), are again predominantly either spam or very obscure sites.

Following the rest of the curve in the figure, notice that it monotonically decays and becomes zero for GTRs larger than 7. In fact, almost no spam (i.e., less than 0.6%) is contained in PLDs with a GTR 5 or higher. Our general conclusion from this activity is that on average toolbar ranks accurately reflect overall domain value, despite occasional outliers. Since bulk GTR information is publicly available in certain portals on the web, we next leverage these findings to create an unbiased automated method for comparing ranking algorithms.

V. Automated Analysis

We first look at the running average of GTRs up to position $R = 10$K. We then switch to avoidance of GTR-0, no-GTR, and PLDs blacklisted by a well-known email spam project.

A. Average GTR

Our first goal is to determine the ability of each ranking algorithm to place the most valuable PLDs at the top of the list. Figure 2(a) plots a moving average of the numeric GTR for each algorithm. The graph shows that SUPP_2 maintains the highest average rank, while IN follows closely until approximately position 4K. PageRank and WIN track each other closely as well, although the latter takes hits from information.com and a few other questionable PLDs near the top of its ranking. All averages begin to converge around 10K, which can be considered inconsequential as we are interested in the separation only up to that point, where a web crawler is most impacted.

The sharp drop for IN's average GTR around position 4K comes from a large number of domains related to worldnews.com that have a zero GTR. These pages contain links to valid news articles and appear legitimate on the surface, except for their odd linking structure. Instead of using a single PLD with several subdirectories or multiple hosts to organize the site, this content is divided among hundreds of PLDs, all of which link to one another in a virtually complete graph. This tight link structure may appear as spam to Google and may be the reason they are all ranked with GTR 0.

The authors in [13] observe a similar case that arises from *link exchanges* (i.e., each user j buries links to $N-1$ other member sites within its directory structure, while each of the other users in return hosts a single link back to j). Link exchanges sometimes involve thousands of unique PLDs, which accumulate high PLD-level in-degree and successfully manipulate certain ranking schemes such as IN. While this does occur in practice, we found only one such example in the top-10K list. The SUPP_2 algorithm aims to overcome these exact problems and remains very resilient to manipulation since creating a large-enough link-exchange structure to be competitive with reputable sites at distance $D = 2$ hops requires much more resources (i.e., not thousands, but millions of unique PLDs).

B. Unreputable GTR

Figure 2(b) plots the cumulative distribution of GTR-0 PLDs found in each list. Again, observe that SUPP_2 does the best at suppressing undesirable sites, where its first zero-ranked PLD (home.net, a link-spam site) does not appear until position $1,422$. Furthermore, this method admits only 11 GTR-0 PLDs in the top-10K. IN initially does an excellent job of suppressing unwanted domains, with only home-equity-loans-1.org in the top-1K at position 843; however, around 2K, worldnews.com-related PLDs quickly add up and after about 5K, IN performs the worst in this comparison, allowing 448 GTR-0 domains in the top-10K. WIN and PageRank are again very similar, with the former slightly outperforming the latter. Both have zero-ranked PLDs very high in their list – everytihng.com in position 32 for PageRank and home-equity-loans-1.org in position 35 for WIN.

Figure 2(c) plots the cumulative distribution of PLDs with no GTR in each list. SUPP_2 is again the clear winner, with the first no-GTR domain bfast.com in position 469, while allowing only 5 unranked PLDs in the top-1K and only 115 in the top-10K. The WIN and PageRank curves are very similar, both pushing 4 unranked PLDs in the top 15 (see Table I), and respectively admitting $39/54$ no-GTR PLDs in their top-1K and $329/556$ in their top-10K. IN initially splits the difference in performance between SUPP_2 and WIN with just 17 unranked PLDs in the top-1K; however, it eventually loses its advantages and finishes the top-10K slightly worse than PageRank.

C. Blacklisted Domains

In addition to PLDs with a zero or missing GTR, there are also publicly available blacklists that contain domains considered to be related to email spam (e.g., PLDs that originate email spam, host phishing content, distribute scams or viruses). The PLDs in this blacklist may or may not contain the kinds of link spam we commonly encounter for parked domains, but the fact that they rely on unsolicited bulk email suggests that they may also employ other unethical techniques such as web spamming or rank manipulation aimed to increase the visibility of their site.

We use a SpamAssassin blacklist [32] with $496,698$ domains that contains $60,701$ PLDs seen (not necessarily crawled) by IRLbot. We find the occurrence of these domains in each ranking and plot their count below each position r in Figure 2(d). SUPP_2 again does the best job of avoiding blacklisted domains, maintaining a clear separation from the other lists up through $R = 100$K. The first blacklisted PLD in the SUPP_2 list is topmeds10.com in position $4,459$ and only 7 appear in the top-10K. IN and WIN show the next-best performance with their first blacklisted PLDs yourmedpharm.info and naughty.com in positions $1,246$ and $1,334$, respectively. PageRank finishes last with webclients.net in position 852. However, by around $r = 5$K, it catches up to the other two methods and they stay together up to 100K.

D. High GTR

To understand whether any good domains have ended up at the bottom of our lists, we next examine the ranking produced by SUPP_2 to see if it might have missed anything important. We found 470 PLDs with a GTR 9 or 10 beyond the top-10K list and examined them manually. All of these cases can be classified into the following four categories: 1) redirects, which include a) misspelled names of famous companies (e.g., amozon.com, verisgn.com); b) unknown PLDs that redirect to Google, Adobe, Yahoo or some equally famous PLD (e.g., hospitalquality.org, defytherules.com, vknn.org, floralartbyamy.com); c) country versions of famous sites (e.g., reuters.cz); 2) mirrors that do not redirect through HTTP 301/302, but visually look identical to their main site (e.g., columbiauniversity.net, compap.com, cnn.co.il); 3) sites related to .gov and .edu, whose GTR is commonly inflated by Google based on their TLD; and 4) GTR anomalies that have since been corrected by Google and whose current GTR is much lower, usually $2-5$ or no GTR at all (e.g., laraweb.com.ar, absearecruitment.com, baileychevy.com).

Considering that IRLbot did not utilize static promotion of certain TLDs, redirect consolidation, or duplicate detection, we found no obviously mis-ranked GTR $9/10$ sites. Having these features in a crawler might be a good idea as they help reduce bandwidth wastage on mirrors and various duplicates.

E. Depth of Supporters

We next address the issue of whether $D = 2$ is the correct choice for Internet-wide PLD graphs. Exploring SUPP_3, we found it to be a poor indicator of site reputation due to the rapid explosion of supporter counts for popular PLDs and the lack of nodes for them to reach at depth 3. While the average in-degree $E[d_{in}]$ is only 20, the average level-2 supporter count $E[SUPP_2]$ is a massive 21K per node. This drastic difference comes from the fact that $E[SUPP_2] \sim E[d_{in}^2]$, which can be much larger than $E^2[d_{in}]$ if variance $Var[d_{in}]$ is huge (which it is).

As one example, google.com is a reputable site that is ranked highly by all of the algorithms we considered. It has 15.5M level-2 supporters, but only 6.2M level-3 supporters. Now consider hotsiteskey.info, a low value site that is ranked in position $192,056$ by SUPP_2. However, it manages to come up with 15.6M level-3 supporters, placing it above google.com and many other highly reputable PLDs. Additionally, the complexity of performing BFS walks from each node and counting unique supporters at distance $D \geq 3$ make these extensions extremely burdensome, even for offline simulations (see below).

This suggests that SUPP_D with $D \geq 3$ may be appropriate for slowly-branching graphs and those with much larger average distances, but clearly not the Internet PLD graph.

VI. Estimating Top Supporters

From the previous section, it is easy to see that SUPP_2 produces the best ranked PLD lists. However, calculating supporters directly does not scale well to large graphs because

of the enormous amount of CPU processing required to perform a limited-scope BFS flood from each node in the graph. However, the good news is that one often does not require SUPP_2 counts for nodes at the bottom of the list; instead, a high-performance crawler is interested in a *fast, accurate, and scalable technique for estimating supporters at the top of its ranking list*. We offer such an approach below.

A. Analysis

To understand the infeasibility of SUPP_2, we start by analyzing its basic complexity under the assumption that the graph fits in RAM. We later extend this to external-memory scenarios. As shown in Fig. 3(a), SUPP_2 for each node x must count the number of unique nodes z whose shortest path to x is exactly two hops, which is usually accomplished by a BFS along the in-links. To calculate the overhead of SUPP_2, we need to define a new metric we call *Quick-Visit Supporters* (QVS), which counts the number of link traversals during BFS:

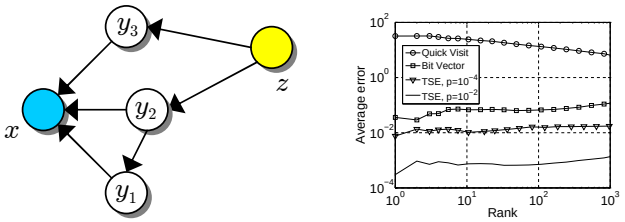
$$QVS(x) = \sum_{y:(y,x)\in E} d_{in}(y). \tag{3}$$

In Fig. 3(a), $SUPP_2(x) = 1$ but $QVS(x) = 3$ since it counts y_2 once and z twice. It then follows that a full-graph BFS requires $n(E[QVS(x)] + E[SUPP_2(x)])$ non-sequential RAM hits to mark all visited nodes and then clear the set bits. Since in our PLD graph $E[QVS(x)] = 34.4\text{K}$ and $E[SUPP_2(x)] = 21\text{K}$, we obtain that SUPP_2 issues almost 5 trillion random RAM lookups, which using 60-ns memory latency amounts to 82 hours (usually a little less due to caching). One may be tempted to approximate SUPP_2 using QVS, which runs a lot faster and obtains the estimates in just $nE[d_{in}] = 1.8\text{B}$ lookups. However, its performance is rather poor, keeping the average error above 200% throughout the top-1K list. The resulting ranking (using spam, no-GTR, GTR-0, and blacklisted domains) splits the difference between IN and SUPP_2, which we do not show for brevity.

Another method [2] that estimates SUPP_2 is *Bit Vector* (BV), in which nodes iteratively receive bit strings from their in-degree neighbors and apply a bitwise OR operation to them. Since OR is not additive, this process avoids the main pitfall of QVS (i.e., multiple counts of the same node z). BV requires $2rnE[d_{in}]$ lookups in RAM, the same number of OR operations, and generation of rbn random numbers, where $b = 64$ is the number of bits in each vector, $r = \log_2(S_{max})$ is the number of performed rounds [2], and S_{max} is the maximum SUPP_2 count. In our dataset, the original BV converges in 25 rounds and provides estimates for the entire n domains. However, since we are only interested in the top list, BV can be adapted to terminate earlier by performing only the last few rounds (i.e., 2 for the top-1K and 3 for the top-10K).

B. Top Supporters (TSE)

We next develop a new method called *Top Supporters Estimation (TSE)*, which allows very efficient computation of SUPP_2 counts for nodes with a large base of supporters. First intuition suggests to visit all in-neighbors y_i of x and then randomly subsample *their* in-neighbors z_j with probability p, which can be done efficiently by *skipping* through neighbor lists using a geometric random variable. This approach performs almost as fast as QVS; however, there unfortunately is no good way to reconstruct the number of *unique* supporters from the collected samples. It is well-know that given a multiset Γ (i.e., a set of items with repetition), a p-percent blind subsample of Γ cannot be used to accurately determine the number of unique items in Γ. This problem has been studied extensively in databases [7] and network flow sampling [11], without much promise for a solution. Of course if one is allowed to examine each item in Γ before deciding whether to sample it or not, accurate estimators exist; however, in such cases BFS has the same CPU overhead and there is no need for an estimator[3]. Instead, we use a novel two-pass technique that leverages both in/out-degree graphs.



(a) SUPP_2 (b) average error

Fig. 3. BFS-based SUPP_2 and estimation error.

Using the notation of Fig. 3(a), the first phase scans the out-graph and randomly retains in RAM a p-fraction of all nodes z with their entire out-neighbor adjacency lists $\{w_j\}$. This produces an unbiased random sample of all supporters z that x will later count. Then, the second phase reads sequentially the in-degree graph, examining each node x with its in-neighbors $\{y_i\}$. Assuming $x \neq z$ and $x \notin \{w_j\}$, any overlap between sets $\{w_j\}$ and $\{y_i\}$ indicates that z supports x at level-2. This detection is accomplished using hash-tables with efficient multi-CPU parallelization. Due to limited space, we do not dwell on the implementation. If z discovers that it supports x, it increments x's supporter counter by 1.

After all sampled PLDs z have been processed, the supporter counter of x is scaled by $1/p$, the result is written to disk, and the next node is read from the in-degree file (in practice, the file is read and written in chunks large enough to keep the I/O efficient). Neglecting certain small terms, TSE requires RAM lookup overhead $pn(E[QVS(x)] + E[SUPP_2(x)])$, which is a p-fraction of that of full BFS. It should be noted that since the crawler first builds the out-degree PLD graph as it parses pages, which is later inverted to obtain the in-degree PLD graph (used by all the other methods), there is no additional cost to obtain the two graphs needed for TSE's operation.

[3]Typical reasons for subsamping Γ are memory-related restrictions. In our problem, just building set Γ incurs prohibitive CPU overhead.

C. Comparison

We now compare the various RAM-only algorithms in terms of accuracy and run time. For these experiments, we use 64 bits for each node in BV and the adaptive approach recommended in [2]. We also retain level-1 BV vectors and remove any overlap with level-2 supporters so as to exclude immediate neighbors of each node x. For this comparison, the PLD graph is processed offline to re-write its 8-byte hashes using sequential 4-byte labels. After this procedure, it occupies approximately 8 GB in RAM. Non-sequential, larger IDs result in higher RAM consumption, less CPU cache locality, and more overhead for each algorithm. Such cases are not considered here, but they are straightforward extensions of our results below.

Figure 3(b) shows a sliding-window average of relative error for each estimation technique as a function of the node's rank (we use SUPP_2's order of ranking). Observe in the figure that QVS exhibits enormous error (i.e., over $1,000\%$) for the very top PLDs, which gradually reduces to about 200% near rank 1K. BV is second best with its error averaging 6.5% in this range, which is quite a bit better than the $15-17\%$ error found in [2]. However, TSE's error is even lower, ranging from about 1% for $p = 10^{-4}$ to 0.1% for $p = 10^{-2}$.

Table II shows the theoretical number of random RAM hits and various CPU operations (e.g., additions, ORs, random numbers generated) in each method. It also displays the actual running time of these algorithms on a quad-core AMD Opteron server, as well as the speedup factor compared to SUPP_2 (disk I/O is excluded). Each algorithm is optimized and multi-threaded to run on a quad-CPU server with enough memory to hold the entire PLD graph and each algorithm's other data structures. SUPP_2 runs for 70 hours, occupying all 4 CPUs at 100% utilization. This is over 35 times slower than 50-iteration PageRank. The default BV method (which requires 25 rounds) runs for 47 minutes; however, its scaled version that produces ranking only for the top-1K nodes (two rounds) terminates in 3.8 minutes, which is about 3 times faster than the most accurate version of TSE considered here (i.e., $p = 10^{-2}$). The other two TSE configurations in the table run much quicker, with $p = 10^{-4}$ being even faster than QVS.

Interestingly, TSE with $p = 10^{-4}$ is more accurate than BV not just for the top-1K, but also for the top-10M PLDs. Thus, if estimates for such a large top-list are needed for some reason, TSE can be a more impressive alternative to BV since the latter now requires 13 iterations to obtain convergence for the top-10M, placing its running time at roughly 25 minutes. Despite these findings, BV has merit in its ability to tackle $D \geq 3$ with little additional overhead, while TSE may require significantly lower p as D increases to control the explosion of BFS. As we encounter future graphs that require SUPP_D estimation at $D \geq 3$, we will revisit this issue and study TSE's accuracy in those settings against BV's.

D. External Memory

We finish by briefly discussing handling large graphs that do not fit in RAM, where the main performance metric is the amount of disk I/O performed (i.e., CPU overhead is neglected). The most straightforward SUPP_2 method, which we call SUPP_2-A, loads a sequential chunk of vectors $(x, y_1, y_2, \ldots)$ from the in-graph into some buffer Q and then re-scans the entire in-graph to check all in-neighbors of level-1 supporters $\{y_i\} \in Q$. After one full pass, the method accumulates all unique supporters z of each node $x \in Q$.

TABLE II
MEMORY HITS, CPU OPERATIONS, AND RUNNING TIME ON THE PLD GRAPH (2.8 GHZ QUAD-CORE OPTERON)

Algorithm	Hits	Ops	Time	Speedup
SUPP_2	4.9T	1.9T	70 hrs	–
TSE ($p = 10^{-2}$)	49B	19B	11 min	381
Bit Vector ($r = 2$)	7.1B	11B	3.8 min	1,113
TSE ($p = 10^{-3}$)	4.9B	1.9B	70 sec	3,600
Quick Visit	1.8B	1.8B	55 sec	4,581
TSE ($p = 10^{-4}$)	490M	190M	7.5 sec	33,600

Assuming F is the size of the in-graph in bytes and RAM size is Ω, the total I/O overhead in SUPP_2-A consists of read-only operations and equals (certain straightforward manipulations omitted):

$$D = \frac{F^2(E[d_{in}] + E[SUPP_2])}{E[d_{in}]\Omega}. \quad (4)$$

Another approach, which we call SUPP_2-B, performs much better when Ω is small. It scans simultaneously both in/out graphs (assumed to be in sorted order) and writes out all pairs (x, z), where z is x's level-2 supporter. Assuming h is the node hash size, this method initially writes $D = 2nhE[QVS(x)]$ bytes to disk, which represent $m = D/\Omega$ sorted blocks that need to be k-way merged to remove duplicates. Using $M =$ 256 KB per buffer for each open file handle, SUPP_2-B uses $k = \Omega/M$ concurrent handles and $\lceil \log_k m \rceil$ merge phases.

QVS reads the file twice (i.e., $D = 2F$) and stores the last two vectors of in-degree counts and hashes in RAM (i.e., $\Omega = 2(h+m)n$, where $m = 4$ the number of bytes needed for each supporter counter). BV has been proposed [2] under the assumption that entire bit vectors from the last two iterations and all supporter counts fit in RAM. Its RAM consumption is therefore $\Omega = (2b + m)n$ and its disk overhead is $D = 2rF$. Finally, TSE reads $D = 2F$ bytes and maintains $\Omega = pF$, without ever requiring that all supporter counts fit in RAM. Note that since both BV and QVS are iterative methods similar to PageRank, their Ω can be reduced at the expense of certain offline pre-processing on the graph [8]. We do not consider this in our comparison.

Table III shows the resulting I/O complexity for all studied methods using the PLD graph with 8-byte hashes (i.e., $F = 15.8$ GB). We ignore the small memory needed for one or two file handles (SUPP_2-A, QVS, BV, TSE) and the buffer through which each graph is sequentially scanned (all methods). SUPP_2-A in the table with 8 GB of RAM (i.e., half the graph fits in memory) requires a surprising 32 TB of disk I/O (i.e., it reads the graph $2,000$ times). As even less RAM becomes available, its performance dramatically reduces and reaches 2.6 PB for $\Omega = 100$ MB. While 70 hours of CPU time

TABLE III
DISK I/O AND RAM FOR EXTERNAL-MEMORY PROCESSING

Algorithm	Disk read	Disk write	RAM	Phases
$SUPP_2$-A	32 TB	–	8 GB	–
	130 TB	–	2 GB	–
	2.6 PB	–	100 MB	–
$SUPP_2$-B	49 TB	49 TB	8 GB	1
	98 TB	98 TB	2 GB	2
	147 TB	147 TB	100 MB	3
Bit Vector ($r = 2$)	63 GB	–	1.9 GB	–
Quick Visit	31.4 GB	–	2.1 GB	–
TSE ($p = 10^{-2}$)	31.4 GB	–	157 MB	–
TSE ($p = 10^{-3}$)	31.4 GB	–	16 MB	–
TSE ($p = 10^{-4}$)	31.4 GB	–	1.6 MB	–

in the previous section may seem excessive, reading petabytes from disk is even worse.

For small Ω, $SUPP_2$-B maintains much better scalability (see the 100-MB case in the table), but requires enough disk space to write 49 TB of pairs (x, z) and then perform a 3-phase merge with 294 TB of combined read/writes. Both BV and QVS perform well, scanning the graph just $2 - 4$ times and maintaining 2 GB in RAM. TSE lowers their Ω by an additional $1 - 3$ orders of magnitude (depending on p) and clearly represents the most scalable solution.

VII. CONCLUSIONS

This paper compared various agnostic algorithms for ranking the web at the PLD level using manual analysis and Google Toolbar Ranks (GTRs). As $SUPP_2$ decisively outperformed the other methods, but was infeasible in practice, we proposed a fast, scalable, and accurate estimator for its top-ranked PLDs. It is shown to achieve a 1% error in the top-1K list with $4 - 5$ orders of magnitude less overhead than $SUPP_2$.

Future work involves studying the host graph and oracle-based extension of $SUPP_2$ to fight spam.

REFERENCES

[1] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates, "Link-based Characterization and Detection of Web Spam," in *Proc. AIRWeb*, Aug. 2006.

[2] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates, "Using Rank Propagation and Probabilistic Counting for Link-Based Spam Detection," in *Proc. WebKDD*, Aug. 2006.

[3] A. A. Benczur, K. Csalogany, T. Sarlos, and M. Uher, "Spamrank – Fully Automatic Link Spam Detection," in *Proc. AIRWeb*, May 2005.

[4] P. Boldi, B. Codenotti, M. Santini, and S. Vigna, "UbiCrawler: A Scalable Fully Distributed Web Crawler," *Software: Practice & Experience*, vol. 34, no. 8, pp. 711–726, Jul. 2004.

[5] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," in *Proc. WWW*, Apr. 1998, pp. 107–117.

[6] J. Caverlee and L. Liu, "Countering Web Spam with Credibility-Based Link Analysis," in *Proc. ACM PODC*, Aug. 2007, pp. 157–166.

[7] A. Chao and S.-M. Lee, "Estimating the Number of Classes via Sample Coverage," *J. American Statistical Association*, vol. 87, Mar. 1992.

[8] Y.-Y. Chen, Q. Gan, and T. Suel, "I/O-Efficient Techniques for Computing PageRank," in *Proc. CIKM*, Nov. 2002.

[9] Y.-Y. Chen, Q. Gan, and T. Suel, "Local Methods for Estimating PageRank Values," in *Proc. CIKM*, Nov. 2004.

[10] J. Cho, H. Garcia-Molina, T. Haveliwala, W. Lam, A. Paepcke, and S. R. G. Wesley, "Stanford WebBase Components and Applications," *ACM Trans. Internet Technology*, vol. 6, no. 2, pp. 153–186, May 2006.

[11] N. Duffield, C. Lund, and M. Thorup, "Estimating Flow Distributions from Sampled Flow Statistics," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 325–336.

[12] D. Eichmann, "The RBSE Spider – Balancing Effective Search Against Web Load," in *Proc. WWW*, May 1994.

[13] N. Eiron, K. S. McCurley, and J. A. Tomlin, "Ranking the Web Frontier," in *Proc. WWW*, May 2004, pp. 309–318.

[14] G. Feng, T.-Y. Liu, Y. Wang, Y. Bao, Z. Ma, X.-D. Zhang, and W.-Y. Ma, "AggregateRank: Bringing Order to Web Sites," in *Proc. ACM SIGIR*, Aug. 2006, pp. 75–82.

[15] Google Toolbar. [Online]. Available: http://toolbar.google.com.

[16] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien, "How to Build a WebFountain: An Architecture for Very Large-Scale Text Analytics," *IBM Systems Journal*, vol. 43, no. 1, pp. 64–77, 2004.

[17] Z. Gyöngyi and H. Garcia-Molina, "Web Spam Taxonomy," in *Proc. AIRWeb*, May 2005.

[18] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating Web Spam with TrustRank," in *Proc. VLDB*, Aug. 2004, pp. 576–587.

[19] Y. Hafri and C. Djeraba, "High Performance Crawling System," in *Proc. ACM MIR*, Oct. 2004, pp. 299–306.

[20] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke, "WebBase: A Repository of Web Pages," in *Proc. WWW*, May 2000, pp. 277–293.

[21] Internet Archive. [Online]. Available: http://www.archive.org/.

[22] K. Koht-arsa and S. Sanguanpong, "High Performance Large Scale Web Spider Architecture," in *Proc. IEEE ISCIT*, Oct. 2002.

[23] Laboratory for Web Algorithmics, "Web Datasets," Sep. 2008. [Online]. Available: http://law.dsi.unimi.it/.

[24] H.-T. Lee, D. Leonard, X. Wang, and D. Loguinov, "IRLbot: Scaling to 6 Billion Pages and Beyond," in *Proc. WWW*, Apr. 2008, pp. 427–436.

[25] R. Lempel and S. Moran, "SALSA: The Stochastic Approach for Link-Structure Analysis," *ACM Trans. Information Systems*, vol. 19, no. 2, pp. 131–160, Apr. 2001.

[26] Y. Li, "Toward a Qualitative Search Engine," *IEEE Internet Computing*, vol. 2, no. 4, pp. 24–29, Jul. 1998.

[27] O. A. McBryan, "GENVL and WWWW: Tools for Taming the Web," in *Proc. WWW*, May 1994.

[28] M. Najork and A. Heydon, "High-Performance Web Crawling," Compaq Systems Research Center, Tech. Rep. 173, Sep. 2001. [Online]. Available: http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-173.pdf.

[29] M. Najork and J. L. Wiener, "Breadth-First Search Crawling Yields High-Quality Pages," in *Proc. WWW*, May 2001, pp. 114–118.

[30] B. Pinkerton, "Finding What People Want: Experiences with the Web Crawler," in *Proc. WWW*, Oct. 1994.

[31] V. Shkapenyuk and T. Suel, "Design and Implementation of a High-Performance Distributed Web Crawler," in *Proc. IEEE ICDE*, Mar. 2002, pp. 357–368.

[32] W. Stearns, "SpamAssassin Domain Blacklist," Aug. 2008. [Online]. Available: http://www.stearns.org/sa-blacklist/.

[33] T. Suel, C. Mathur, J. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram, "ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval," in *Proc. WebDB*, Jun. 2003, pp. 67–72.

[34] The Official Google Blog, "We knew the web was big..." Jul. 2008. [Online]. Available: http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html.

[35] The Stanford WebBase Project, "WebBase Archive," Sep. 2008. [Online]. Available: http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/.

[36] T. Upstill, N. Craswell, and D. Hawking, "Predicting Fame and Fortune: PageRank or Indegree," in *Proc. 8th Australasian Document Computing Symposium*, Dec. 2003.

[37] B. Wu, V. Goel, and B. Davison, "Topical TrustRank: Using Topicality to Combat Web Spam," in *Proc. WWW*, May 2006.

[38] J. Wu and K. Aberer, "Using SiteRank for Decentralized Computation of Web Document Ranking," in *Proc. Adaptive Hypermedia*, Aug. 2004, pp. 265–274.

[39] Yahoo Research Barcelona, "Datasets for Research on Web Spam Detection," Sep. 2008. [Online]. Available: http://barcelona.research.yahoo.net/webspam/datasets/.

[40] H. Yan, J. Wang, X. Li, and L. Guo, "Architectural Design and Evaluation of an Efficient Web-crawling System," *Journal of Systems and Software*, vol. 60, no. 3, pp. 185–193, Feb. 2002.