Graph-based Seed Selection for Web-scale Crawlers

Shuyi Zheng[1] Pavel Dmitriev[2] C. Lee Giles[3]

[1,3]Department of Computer Science and Engineering
[3]College of Information Sciences and Technology
Pennsylvania State University, University Park, PA 16802

[2]Yahoo! Labs, Santa Clara, CA 95054

shzheng@cse.psu.edu dmitriev@yahoo-inc.com giles@ist.psu.edu

ABSTRACT

One of the most important steps in web crawling is determining the starting points, or *seed selection*. This paper identifies and explores the problem of seed selection in web-scale incremental crawlers. We argue that seed selection is not a trivial but very important problem. Selecting proper seeds can increase the number of pages a crawler will discover, and can result in a repository with more "good" and less "bad" pages. We propose a graph-based framework for crawler seed selection, and present several algorithms within this framework. Evaluation on real web data showed significant improvements over heuristic seed selection approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Crawler, Seed Selection, PageRank, Graph Analysis

1. INTRODUCTION

Crawling is not only one of the most important tasks of a search engine, but also an indispensable part of many other web applications. The breadth, depth, and freshness of the search results depend crucially on the quality of crawling. As the number of pages and sites on the web increases rapidly, deploying an effective and efficient crawling strategy becomes critical for a search engine.

A typical crawler [3] maintains a list of unvisited URLs called *frontier*, which is initialized with seed URLs. In each crawling loop, the crawler picks a URL from the frontier, fetches the corresponding page, parses the retrieved page to extract URLs, and adds unvisited URLs to the frontier. Typically, the crawler needs to keep revisiting some or all the URLs to check if they are updated since the last crawl. Due to the infinite nature of the web and the competition between getting new URLs and refreshing the old ones, even web-scale crawlers can never crawl "all" the URLs from the web.

Different crawling strategies resulting from different ways of ordering the URLs in the frontier can explore the web in different ways. However, all of them start from the seed pages and proceed by exploring the neighborhoods of the seed pages in one way or another. Thus, to a large extent, selecting good quality seed determines the quality of the crawl.

One may think that simply starting from root pages of several well known sites and crawling very deep will allow the crawler to reach all useful pages on the web. Unfortunately, this is not so. As Broder et. al [1] showed, even 9 years ago, close to half of all web pages could not be reached from the "central" strongly connected portion of the web. Moreover, the situation is likely to be even worse nowadays. Recently, many websites that contain millions of pages have emerged. Our study shows that many large websites are not strongly connected.

In this paper, we study the problem of crawler seed selection and propose a seed selection framework based on the analysis of the link structure of the web. We assume the incremental crawler model, where the crawler crawls continuously downloading new pages and refreshing the old ones, and seeds are updated periodically to keep up with the changes in the web.

2. THE PROBLEM OF SEED SELECTION

Since there exists a wide variety of crawling strategies, in this paper we make several assumptions about the crawling strategy. First, we assume that the number k of seeds is given. Second, we assume that a crawler crawls pages only within h hops from the seed and it always crawls all pages within h hops. We note that the values of h and k are related. Given a fixed repository size, the larger the number of seeds k is, the less hops h the crawler can crawl on average before it fills the repository. Third, we assume that the cost of downloading and processing a web page (measured in terms of time, CPU, and storage required) is constant.

As noted earlier, in an incremental crawler, seed selection

CIKM'09, November 2–6, 2009, Hong Kong, China.

happens periodically in order to keep up with the changes of the web. It is always based on the information from a portion of the web which is already crawled. Thus, an intuitive definition of the problem of seed selection is, given a currently known portion of the web and the desired number of seeds k, to select seeds so that as many as possible new good pages will get crawled, and as many as possible currently crawled good pages will be retained.

Several heuristic approaches can be used for seed selection. The simplest strategy, used as a baseline in our experiments, is to select the seeds at random. Another strategy is to select k pages with the highest PageRank. This makes sense since one can expect to find high quality pages around other high quality pages. Another intuitive strategy is to select k pages with most outlinks.

A heuristic which can improve the above strategies is to first split the known portion of the web into smaller units, such as web sites, and then select seeds independently for each web site. Such strategy allows distributing seeds "evenly" across the web, allocating to each web site a fraction of the total number of seeds, which is proportional to the site's importance. It also allows performing seed computation in parallel and on a smaller data set.

3. GRAPH-BASED SEED SELECTION

While the above heuristics make sense intuitively, they do not directly optimize the criteria we are interested in. To formally define the problem of seed selection, we assume that every crawled web page has an associated value. Higher value indicates higher quality or higher potential to discover new pages calculated, for example, as the number of uncrawled URLs the page links to. In addition, the value can be negative if the page is undesirable, such as a spam page. We can now define the seed selection problem as maximizing the value of the portion of the web graph "covered" by the seeds.

DEFINITION 1 (SEED SELECTION PROBLEM). *Given a directed graph $G = (V, E)$, a function $w : V \to \mathbb{R}$, assigning a weight $w(v)$ to every $v \in V$, and edges unweighted, given the number of seeds k, and the number of hops a seed covers h, select the seeds so that $w(\cup_{i=1}^{k} A_i)$ is maximized, where $A_i = \{v | v \in V, v$ within h hops of seed $i\}$, $w(A) = \sum_{v \in A} w(v)$.*

In this formalization, the seed selection problem is an instance of the *Maximum K-Coverage Problem*, which is known to be NP-hard [2]. However, a greedy iterative approximation exists for this problem, which achieves $1 - \frac{1}{e}$ approximation [2]. The algorithm is shown below.

Unfortunately, performing even a single iteration of the algorithm is computationally expensive for large h, due to the exponential in h complexity of step 2. For example, if the number of outlinks of a page on the website is l, finding the optimal seed v^* will take $O(n \cdot l^h)$. For a rather typical scenario of $l = 20$, $h = 5$, $n = 1,000,000$, we need $O(10^{12})$ operations to find v^*.

Because of the high complexity of step 2, we resort to an approximation again. We propose and evaluate four approximation algorithms for finding the vertex in the graph that covers maximal value within h hops. These algorithms are described below.

Algorithm 1 Seed Selection Algorithm

Input: Weighted Graph G of (a portion of) the web, Maximal Seed Number k, Number of hops allowed h
Output: Selected Seeds $\mathcal{S}$

Algorithm:

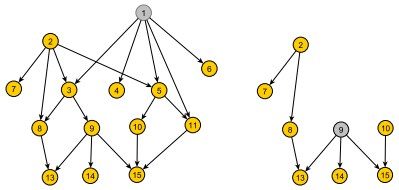
1: **FOR** $i = 1$ to k
2: Find vertex v^* which covers maximal value within h hops
3: Make h hops from v^* on graph G
4: Set the values of all covered vertices to zero
5: Add v^* to $\mathcal{S}$
6: **IF** values of all vertices are zero **THEN**
7: BREAK
8: **ENDIF**
9: **ENDFOR**

3.1 Maximal Out-degree First

The first algorithm, called *MaxOut*, always selects as a seed the page that has the largest out-degree. While this algorithm does not look at the values of the pages, it is based on the intuition that covering more pages in the first hop will lead to covering more value in h hops.

This algorithm is different from the heuristic *OutDegree* algorithm, because after each step it updates the graph by removing the covered vertices. As a result, the out-degree of remaining vertices will decrease if they have out-links pointing to the removed vertices. Therefore, the algorithm will select seeds which cover substantially disjoint portions of the web graph.

Here we use the following example to illustrate the process of this algorithm and show the difference with the heuristic *OutDigree* algorithm. For this example, we assume the number of seeds, k, is 2 and the number of hops, h, is 1. In Figure 1(a), the top two nodes (pages) with highest out-degree is node 1 (out-degree=5) and node 2 (out-degree=4). These two nodes will be selected as seeds if we use the heuristic *OutDegree* algorithm. On the contrary, if we use *MaxOut* algorithm, node 1, 3, 4, 5, 6, and 11 will all be removed after the first step because they are all covered within 1 hop from the first selected node 1. In step 2 (Figure 1(b)), node 2 is no longer the one with the highest out-degree because some nodes it links to have already been covered by previous selected seeds. Instead, node 9 is the best one for this step.



(a) Step 1 (b) Step 2

Figure 1: An Example of MaxOut Algorithm

3.2 Maximal Depth-d Weight First

This algorithm, called *MaxWeight*, selects as a seed the page that has the maximal depth-d weight, where depth-d weight is defined as follows.

DEFINITION 2 (DEPTH-d WEIGHT). *Given vertex v,*

$$\textit{depth-}d\textit{ weight of } v = w(v) + \sum_{u \in C} w(u)$$

where C is the set of vertices within d hops from v.

The *MaxWeight* algorithm tries to approximate the optimal seed, which has a maximal depth-h weight, with a seed that has a maximal depth-d weight, $d < h$. Since we always set the values of all covered vertices to zero after we select each seed, we have to recalculate the depth-d weight for all vertices of the input graph.

In our experiments we used values 2 and 3 for d. We found that using values larger than 3 results in prohibitively long running times.

3.3 Maximal Weighted SCC First

In this algorithm, called *MaxSCC*, we first calculate the *Strongly Connected Components (SCC)* of the link graph. A strongly connected component is a subgraph consisting of all vertices such that there is a path from each vertex in the subgraph to every other vertex. The algorithm then selects as the seed a page that has the maximal depth-d weight in the maximal weight SCC. Similarly, we also need to recalculate the weight of all vertices after we select each seed.

This algorithm is based on the intuition that selecting the seed within the highest-weight SCC will lead to crawling all the pages within that SCC, if the diameter of the SCC is less or equal than h. If the diameter is greater than h, the algorithm tries to select a seed that covers the most high-weight portion of the SCC, based on the *MaxWeight* algorithm.

3.4 Root SCC First

Given a link graph G, we can form a higher level graph G' as follows. We collapse each strongly connected component in G into a vertex. Then, we add a directed edge $e(c_1, c_2)$, connecting the vertex corresponding to SCC c_1 to the vertex corresponding to SCC c_2 if and only if there exists at least one hyperlink from a page in c_1 to a page in c_2. It is not difficult to see that G' is a directed acyclic graph. Given two components c_1, c_2 in G', we say that c_1 is a parent of c_2, and c_2 is a child of c_1, if there is an edge $e(c_1, c_2) \in G'$. Components without parents are called root components.

Suppose there is no limit on the number of hops. Then any seed selected in a parent component will cover all the pages in its child components. This motivates the *RootSCC* algorithm (Algorithm 2), which selects as a seed a page that has the maximal depth-d weight in the maximal weight root SCC of the maximal weight SCC in G.

Algorithm 2 Root SCC First

Input: Weighted Graph G of a website, Maximal Seed Number k, Number of hops allowed h, Number of hops to examine d
Output: Selected Seeds $\mathcal{S}$ of this website

Algorithm:

```
1: Calculate strongly connected components of G
2: Build a directed acyclic graph G' by collapsing each SCC into a vertex
3: FOR each SCC c
4:     Calculate total weight of all vertices in c
5: ENDFOR
6: FOR i = 1 to k
7:     Pick SCC c1 which has the largest total weight
8:     Get SCC c2 which is the maximal weight root SCC for c1 in G'
9:     Calculate depth-d weight for all vertices of c2
10:    Get seed s from c2 which has the largest depth-d weight
11:    Make h hops on graph G, mark all covered vertices
12:    Set the values of all marked vertices to zero
13:    Recalculate total weight for each SCC
14:    Add s to S
15:    IF the weight of all vertices in G is zero THEN
16:        BREAK
17:    ENDIF
18: ENDFOR
```

4. EXPERIMENTAL RESULTS

4.1 Experiment Setup

To evaluate the performance of the algorithms, we selected a random sample of 2000 web sites from Malaysian web, each containing at least 100 pages and having at least 1 external link into the site. We used an experimental web-scale crawl from the summer 2008 to obtain all pages crawled from these web sites. We also obtained other page attributes, such as whether the page was determined to be spam, whether the page was clicked on in search engine results, and how many uncrawled pages the page links to. We assigned a value to each page according to the following rules:

For every page p set $w(p) = 1$. If p is spam, set $w(p) = -10$. If p was clicked, set $w(p) = 10$. Let n be the number of uncrawled pages the page links to. Set $w(p) = w(p) + n * P_{uniq}$, where P_{uniq} is the probability of uniqueness of a newly crawled page on the site. This probability is calculated based on past crawl statistics.

We implemented the three heuristic approaches and the five graph-based algorithms described above. We use *Random* algorithm as a baseline, and report the average performance of the algorithms over the 2000 sites in terms of the percentage of improvement over *Random*. We vary the number of seeds generated for each site from 1 to 10, keeping the number of hops fixed at 5, and vary the number of hops from 1 to 5 keeping the number of seeds fixed at 5.

4.2 Experiment Results

Figure 2 shows the results for the page coverage and Figure 3 shows the results for the value coverage for the fixed number of hops and varying number of seeds (error bars indicate standard errors). The following conclusions can be drawn from these graphs. First, *Random* performs rather poorly. All other strategies outperform it significantly. As one may expect, the improvement over *Random* is larger when the number of seeds is small, and it decreases as the number of seeds grows. One can also see that *PageRank*, while outperforming *Random*, performs much worse than the other heuristic approach, *OutDegree*, in terms of both the page coverage and the value coverage.

Second, all four of the graph based algorithms outperform the heuristic seed selection approaches. The *MaxOut* algo-

rithm, which only uses outdegree of a page to select the best seed, performs best in terms of the coverage of pages. However, as expected it performs much worse than most of the graph-based strategies in terms of the coverage of value.

Third, the algorithms using simple depth-d approximation perform better than the algorithms that try to utilize the knowledge of strongly connected components and their connectivity. In fact, the *RootSCC* algorithm performs very poorly in terms of the value coverage, worse than the heuristic *OutDegree* algorithm. The reason for this might be that, as we showed earlier, most of web sites tend to consist of many small-size SCCs, so selecting the seeds based on the direct estimation of the depth-d value is better than selecting seeds based on the the value of the SCC the seed belongs to.

Finally, the two *MaxWeight* algorithms using $d = 2$ and $d = 3$ perform similar, suggesting that using 2 hop approximation is enough to identify a good seed.

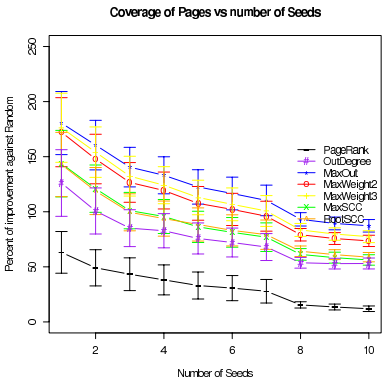


Figure 2: Coverage of Pages

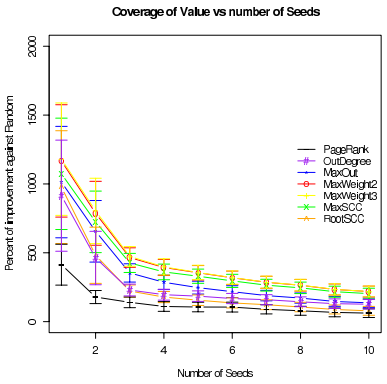


Figure 3: Coverage of Value

Figures 4 and 5 show the results for the page coverage and the value coverage for the fixed number of seeds and varying number of hops. As once can see, improvement of the algorithms over *Random* increases slightly with the number of hops increasing. The increase is higher for the page coverage, and smaller for the value coverage. This is expected because due to the nature of our algorithms most high-value pages should be covered within a few hops. The relative order of the algorithms is the similar to the one on figures 2 and 3.

Overall, the results show that the graph-based strategies significantly outperform the heuristic approaches, demonstrating the advantage of the graph-based seed selection framework. While we only evaluated the algorithms for one specific setting of page values applicable in a web-scale crawler setting, we believe that the results will generalize to other settings, such as the ones arising in a news or blog search engine.

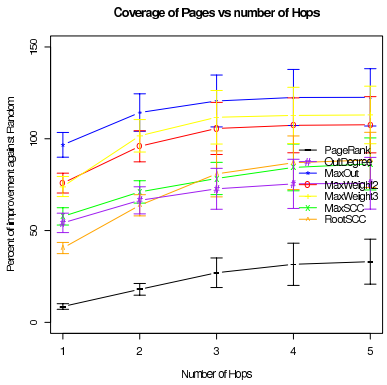


Figure 4: Coverage of Pages

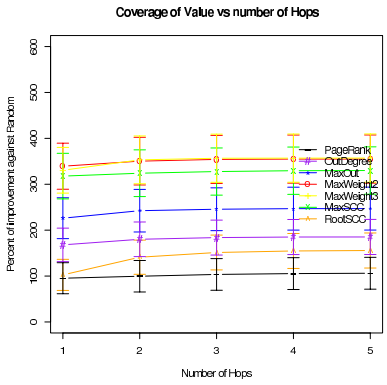


Figure 5: Coverage of Value

5. CONCLUSIONS AND FUTURE WORK

In this paper we discuss the problem of seed selection for a web-scale crawler. We formalize this problem as a graph theoretic problem, analyze its complexity and propose several approximate algorithms. Experimental results on a dataset of 2000 real web sites demonstrates that our algorithms significantly outperform the naive seed selection strategies.

6. REFERENCES

[1] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the Web. *Computer Networks*, 33(1-6):309–320, 2000.

[2] D. Hochbaum and A. Pathria. Analysis of the Greedy Approach in Problems of Maximum k-Coverage. *Naval Research Logistics*, 45(6):615–627, 1998.

[3] G. Pant, P. Srinivasan, and F. Menczer. Crawling the Web. *Web Dynamics*, pages 153–178, 2004.