Estimating the Global PageRank of Web Communities

Jason V. Davis
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
jdavis@cs.utexas.edu

Inderjit S. Dhillon
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
inderjit@cs.utexas.edu

ABSTRACT

Localized search engines are small-scale systems that index a particular community on the web. They offer several benefits over their large-scale counterparts in that they are relatively inexpensive to build, and can provide more precise and complete search capability over their relevant domains. One disadvantage such systems have over large-scale search engines is the lack of global PageRank values. Such information is needed to assess the value of pages in the localized search domain within the context of the web as a whole. In this paper, we present well-motivated algorithms to estimate the global PageRank values of a local domain. The algorithms are all highly scalable in that, given a local domain of size n, they use $O(n)$ resources that include computation time, bandwidth, and storage. We test our methods across a variety of localized domains, including site-specific domains and topic-specific domains. We demonstrate that by crawling as few as n or $2n$ additional pages, our methods can give excellent global PageRank estimates.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra; G.3 [**Probability and Statistics**]: Markov Processes

General Terms

PageRank, Markov Chain, Stochastic Complementation

Keywords

Algorithms, Experimentation

1. INTRODUCTION

Localized search engines are small-scale search engines that index only a single community of the web. Such communities can be site-specific domains, such as pages within the `cs.utexas.edu` domain, or topic-related communities—for example, political websites. Compared to the web graph crawled and indexed by large-scale search engines, the size of such local communities is typically orders of magnitude smaller. Consequently, the computational resources needed to build such a search engine are also similarly lighter. By restricting themselves to smaller, more manageable sections of the web, localized search engines can also provide more precise and complete search capabilities over their respective domains.

One drawback of localized indexes is the lack of global information needed to compute link-based rankings. The PageRank algorithm [3], has proven to be an effective such measure. In general, the PageRank of a given page is dependent on pages throughout the entire web graph. In the context of a localized search engine, if the PageRanks are computed using only the local subgraph, then we would expect the resulting PageRanks to reflect the perceived popularity within the local community and not of the web as a whole. For example, consider a localized search engine that indexes political pages with conservative views. A person wishing to research the opinions on global warming within the conservative political community may encounter numerous such opinions across various websites. If only local PageRank values are available, then the search results will reflect only strongly held beliefs within the community. However, if global PageRanks are also available, then the results can additionally reflect outsiders' views of the conservative community (those documents that liberals most often access within the conservative community).

Thus, for many localized search engines, incorporating global PageRanks can improve the quality of search results. However, the number of pages a local search engine indexes is typically orders of magnitude smaller than the number of pages indexed by their large-scale counterparts. Localized search engines do not have the bandwidth, storage capacity, or computational power to crawl, download, and compute the global PageRanks of the entire web. In this work, we present a method of approximating the global PageRanks of a local domain while only using resources of the same order as those needed to compute the PageRanks of the local subgraph.

Our proposed method looks for a supergraph of our local subgraph such that the local PageRanks within this supergraph are close to the true global PageRanks. We construct this supergraph by iteratively crawling global pages on the current web frontier—i.e., global pages with inlinks from pages that have already been crawled. In order to provide

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

a good approximation to the global PageRanks, care must be taken when choosing which pages to crawl next; in this paper, we present a well-motivated page selection algorithm that also performs well empirically. This algorithm is derived from a well-defined problem objective and has a running time linear in the number of local nodes.

We experiment across several types of local subgraphs, including four topic related communities and several site-specific domains. To evaluate performance, we measure the difference between the current global PageRank estimate and the global PageRank, as a function of the number of pages crawled. We compare our algorithm against several heuristics and also against a baseline algorithm that chooses pages at random, and we show that our method outperforms these other methods. Finally, we empirically demonstrate that, given a local domain of size n, we can provide good approximations to the global PageRank values by crawling at most n or $2n$ additional pages.

The paper is organized as follows. Section 2 gives an overview of localized search engines and outlines their advantages over global search. Section 3 provides background on the PageRank algorithm. Section 4 formally defines our problem, and section 5 presents our page selection criteria and derives our algorithms. Section 6 provides experimental results, section 7 gives an overview of related work, and, finally, conclusions are given in section 8.

2. LOCALIZED SEARCH ENGINES

Localized search engines index a single community of the web, typically either a site-specific community, or a topic-specific community. Localized search engines enjoy three major advantages over their large-scale counterparts: they are relatively inexpensive to build, they can offer more precise search capability over their local domain, and they can provide a more complete index.

The resources needed to build a global search engine are enormous. A 2003 study by Lyman *et al.* [13] found that the 'surface web' (publicly available static sites) consists of 8.9 billion pages, and that the average size of these pages is approximately 18.7 kilobytes. To download a crawl of this size, approximately 167 terabytes of space is needed. For a researcher who wishes to build a search engine with access to a couple of workstations or a small server, storage of this magnitude is simply not available. However, building a localized search engine over a web community of a hundred thousand pages would only require a few gigabytes of storage. The computational burden required to support search queries over a database this size is more manageable as well. We note that, for topic-specific search engines, the relevant community can be efficiently identified and downloaded by using a focused crawler [21, 4].

For site-specific domains, the local domain is readily available on their own web server. This obviates the need for crawling or spidering, and a complete and up-to-date index of the domain can thus be guaranteed. This is in contrast to their large-scale counterparts, which suffer from several shortcomings. First, crawling dynamically generated pages—pages in the 'hidden web'—has been the subject of research [20] and is a non-trivial task for an external crawler. Second, site-specific domains can enable the robots exclusion policy. This prohibits external search engines' crawlers from downloading content from the domain, and an external search engine must instead rely on outside links and anchor text to index these restricted pages.

By restricting itself to only a specific domain of the internet, a localized search engine can provide more precise search results. Consider the canonical ambiguous search query, 'jaguar', which can refer to either the car manufacturer or the animal. A scientist trying to research the habitat and evolutionary history of a jaguar may have better success using a finely tuned zoology-specific search engine than querying Google with multiple keyword searches and wading through irrelevant results. A method to learn better ranking functions for retrieval was recently proposed by Radlinski and Joachims [19] and has been applied to various local domains, including Cornell University's website [8].

3. PAGERANK OVERVIEW

The PageRank algorithm defines the importance of web pages by analyzing the underlying hyperlink structure of a web graph. The algorithm works by building a Markov chain from the link structure of the web graph and computing its stationary distribution. One way to compute the stationary distribution of a Markov chain is to find the limiting distribution of a random walk over the chain. Thus, the PageRank algorithm uses what is sometimes referred to as the 'random surfer' model. In each step of the random walk, the 'surfer' either follows an outlink from the current page (i.e. the current node in the chain), or jumps to a random page on the web.

We now precisely define the PageRank problem. Let $\boldsymbol{U}$ be an $m \times m$ adjacency matrix for a given web graph such that $U_{ji} = 1$ if page i links to page j and $U_{ji} = 0$ otherwise. We define the PageRank matrix $\boldsymbol{P_U}$ to be:

$$\boldsymbol{P_U} = \alpha \boldsymbol{U} \boldsymbol{D_U}^{-1} + (1 - \alpha) \boldsymbol{v} \boldsymbol{e}^T, \tag{1}$$

where $\boldsymbol{D_U}$ is the (unique) diagonal matrix such that $\boldsymbol{U}\boldsymbol{D_U}^{-1}$ is column stochastic, α is a given scalar such that $0 \leq \alpha \leq 1$, $\boldsymbol{e}$ is the vector of all ones, and $\boldsymbol{v}$ is a non-negative, L_1-normalized vector, sometimes called the 'random surfer' vector. Note that the matrix $\boldsymbol{D_U}^{-1}$ is well-defined only if each column of $\boldsymbol{U}$ has at least one non-zero entry—i.e., each page in the webgraph has at least one outlink. In the presence of such 'dangling nodes' that have no outlinks, one commonly used solution, proposed by Brin *et al.* [3], is to replace each zero column of $\boldsymbol{U}$ by a non-negative, L_1-normalized vector.

The PageRank vector $\boldsymbol{r}$ is the dominant eigenvector of the PageRank matrix, $\boldsymbol{r} = \boldsymbol{P_U}\boldsymbol{r}$. We will assume, without loss of generality, that $\boldsymbol{r}$ has an L_1-norm of one. Computationally, $\boldsymbol{r}$ can be computed using the power method. This method first chooses a random starting vector $\boldsymbol{r}^{(0)}$, and iteratively multiplies the current vector by the PageRank matrix $\boldsymbol{P_U}$; see Algorithm 1. In general, each iteration of the power method can take $O(m^2)$ operations when $\boldsymbol{P_U}$ is a dense matrix. However, in practice, the number of links in a web graph will be of the order of the number of pages. By exploiting the sparsity of the PageRank matrix, the work per iteration can be reduced to $O(km)$, where k is the average number of links per web page. It has also been shown that the total number of iterations needed for convergence is proportional to α and does not depend on the size of the web graph [11, 7]. Finally, the total space needed is also $O(km)$, mainly to store the matrix $\boldsymbol{U}$.

ALGORITHM 1: A linear time (per iteration) algorithm for computing PageRank.

COMPUTEPR($\boldsymbol{U}$)
Input: $\boldsymbol{U}$: Adjacency matrix.
Output: $\boldsymbol{r}$: PageRank vector.
Choose (randomly) an initial non-negative vector $\boldsymbol{r}^{(0)}$ such that $\|\boldsymbol{r}^{(0)}\|_1 = 1$.
$i \leftarrow 0$
repeat
  $i \leftarrow i + 1$
  $\boldsymbol{\nu} \leftarrow \alpha \boldsymbol{U} \boldsymbol{D}_{\boldsymbol{U}}^{-1} \boldsymbol{r}^{(i-1)}$ {α *is the random surfing probability*}
  $\boldsymbol{r}^{(i)} \leftarrow \boldsymbol{\nu} + (1-\alpha)\boldsymbol{v}$ {$\boldsymbol{v}$ *is the random surfer vector.*}
until $\|\boldsymbol{r}^{(i)} - \boldsymbol{r}^{(i-1)}\| < \delta$ {δ *is the convergence threshold.*}
$\boldsymbol{r} \leftarrow \boldsymbol{r}^{(i)}$

4. PROBLEM DEFINITION

Given a local domain $\boldsymbol{L}$, let $\boldsymbol{G}$ be an $N \times N$ adjacency matrix for the entire connected component of the web that contains $\boldsymbol{L}$, such that $G_{ji} = 1$ if page i links to page j and $G_{ji} = 0$ otherwise. Without loss of generality, we will partition G as:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{L} & \boldsymbol{G}_{\text{out}} \\ \boldsymbol{L}_{\text{out}} & \boldsymbol{G}_{\text{within}} \end{bmatrix}, \tag{2}$$

where $\boldsymbol{L}$ is the $n \times n$ local subgraph corresponding to links inside the local domain, $\boldsymbol{L}_{\text{out}}$ is the subgraph that corresponds to links from the local domain pointing out to the global domain, $\boldsymbol{G}_{\text{out}}$ is the subgraph containing links from the global domain into the local domain, and $\boldsymbol{G}_{\text{within}}$ contains links within the global domain. We assume that when building a localized search engine, only pages inside the local domain are crawled, and the links between these pages are represented by the subgraph $\boldsymbol{L}$. The links in $\boldsymbol{L}_{\text{out}}$ are also known, as these point from crawled pages in the local domain to uncrawled pages in the global domain.

As defined in equation (1), $\boldsymbol{P}_{\boldsymbol{G}}$ is the PageRank matrix formed from the global graph $\boldsymbol{G}$, and we define the global PageRank vector of this graph to be $\boldsymbol{g}$. Let the n-length vector $\boldsymbol{p}^*$ be the L_1-normalized vector corresponding to the global PageRank of the pages in the local domain $\boldsymbol{L}$:

$$\boldsymbol{p}^* = \frac{\boldsymbol{E}_{\boldsymbol{L}}\boldsymbol{g}}{\|\boldsymbol{E}_{\boldsymbol{L}}\boldsymbol{g}\|_1},$$

where $\boldsymbol{E}_{\boldsymbol{L}} = [\ \boldsymbol{I} \mid \boldsymbol{0}\ ]$ is the restriction matrix that selects the components from $\boldsymbol{g}$ corresponding to nodes in $\boldsymbol{L}$. Let $\boldsymbol{p}$ denote the PageRank vector constructed from the local domain subgraph $\boldsymbol{L}$. In practice, the observed local PageRank $\boldsymbol{p}$ and the global PageRank $\boldsymbol{p}^*$ will be quite different. One would expect that as the size of local matrix $\boldsymbol{L}$ approaches the size of global matrix $\boldsymbol{G}$, the global PageRank and the observed local PageRank will become more similar. Thus, one approach to estimating the global PageRank is to crawl the entire global domain, compute its PageRank, and extract the PageRanks of the local domain.

Typically, however, $n \ll N$, i.e., the number of global pages is much larger than the number of local pages. Therefore, crawling all global pages will quickly exhaust all local resources (computational, storage, and bandwidth) available to create the local search engine. We instead seek a supergraph $\hat{\boldsymbol{F}}$ of our local subgraph $\boldsymbol{L}$ with size $O(n)$. Our goal

ALGORITHM 2: The FINDGLOBALPR algorithm.

FINDGLOBALPR($\boldsymbol{L}$, $\boldsymbol{L}_{\text{out}}$, T, k)
Input: $\boldsymbol{L}$: zero-one adjacency matrix for the local domain, $\boldsymbol{L}_{\text{out}}$: zero-one outlink matrix from $\boldsymbol{L}$ to global subgraph as in (2), T: number of iterations, k: number of pages to crawl per iteration.
Output: $\hat{\mathbf{p}}$: an improved estimate of the global PageRank of $\boldsymbol{L}$.
$\boldsymbol{F} \leftarrow \boldsymbol{L}$
$\boldsymbol{F}_{\text{out}} \leftarrow \boldsymbol{L}_{\text{out}}$
$\boldsymbol{f} \leftarrow$ COMPUTEPR($\boldsymbol{F}$)
for ($i = 1$ to T)
  {*Determine which pages to crawl next*}
  pages $\leftarrow$ SELECTNODES($\boldsymbol{F}$, $\boldsymbol{F}_{\text{out}}$, $\boldsymbol{f}$, k)
  Crawl pages, augment $\boldsymbol{F}$ and modify $\boldsymbol{F}_{\text{out}}$
  {*Update PageRanks for new local domain*}
  $\boldsymbol{f} \leftarrow$ COMPUTEPR($\boldsymbol{F}$)
end
{*Extract PageRanks of original local domain & normalize*}
$\hat{\boldsymbol{p}} \leftarrow \frac{\boldsymbol{E}_{\boldsymbol{L}}\boldsymbol{f}}{\|\boldsymbol{E}_{\boldsymbol{L}}\boldsymbol{f}\|_1}$

is to find such a supergraph $\hat{\boldsymbol{F}}$ with PageRank $\hat{\boldsymbol{f}}$, so that $\hat{\boldsymbol{f}}$ when restricted to $\boldsymbol{L}$ is close to $\boldsymbol{p}^*$. Formally, we seek to minimize

$$GlobalDiff(\hat{\boldsymbol{f}}) = \left\| \frac{\boldsymbol{E}_{\boldsymbol{L}}\hat{\boldsymbol{f}}}{\|\boldsymbol{E}_{\boldsymbol{L}}\hat{\boldsymbol{f}}\|_1} - \boldsymbol{p}^* \right\|_1. \tag{3}$$

We choose the L_1 norm for measuring the error as it does not place excessive weight on outliers (as the L_2 norm does, for example), and also because it is the most commonly used distance measure in the literature for comparing PageRank vectors, as well as for detecting convergence of the algorithm [3].

We propose a greedy framework, given in Algorithm 2, for constructing $\hat{\boldsymbol{F}}$. Initially, $\boldsymbol{F}$ is set to the local subgraph $\boldsymbol{L}$, and the PageRank $\boldsymbol{f}$ of this graph is computed. The algorithm then proceeds as follows. First, the SELECTNODES algorithm (which we discuss in the next section) is called and it returns a set of k nodes to crawl next from the set of nodes in the current crawl frontier, $\boldsymbol{F}_{\text{out}}$. These selected nodes are then crawled to expand the local subgraph, $\boldsymbol{F}$, and the PageRanks of this expanded graph are then recomputed. These steps are repeated for each of T iterations. Finally, the PageRank vector $\hat{\boldsymbol{p}}$, which is restricted to pages within the original local domain, is returned. Given our computation, bandwidth, and memory restrictions, we will assume that the algorithm will crawl at most $O(n)$ pages. Since the PageRanks are computed in each iteration of the algorithm, which is an $O(n)$ operation, we will also assume that the number of iterations T is a constant. Of course, the main challenge here is in selecting which set of k nodes to crawl next. In the next section, we formally define the problem and give efficient algorithms.

5. NODE SELECTION

In this section, we present node selection algorithms that operate within the greedy framework presented in the previous section. We first give a well-defined criteria for the page selection problem and provide experimental evidence that this criteria can effectively identify pages that optimize our problem objective (3). We then present our main al-

gorithmic contribution of the paper, a method with linear running time that is derived from this page selection criteria. Finally, we give an intuitive analysis of our algorithm in terms of 'leaks' and 'flows'. We show that if only the 'flow' is considered, then the resulting method is very similar to a widely used page selection heuristic [6].

5.1 Formulation

For a given page j in the global domain, we define the expanded local graph $\boldsymbol{F}_j$:

$$\boldsymbol{F}_j = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{s} \\ \boldsymbol{u}_j^T & 0 \end{bmatrix}, \tag{4}$$

where $\boldsymbol{u}_j$ is the zero-one vector containing the outlinks from $\boldsymbol{F}$ into page j, and $\boldsymbol{s}$ contains the inlinks from page j into the local domain. Note that we do not allow self-links in this framework. In practice, self-links are often removed, as they only serve to inflate a given page's PageRank.

Observe that the inlinks into $\boldsymbol{F}$ from node j are not known until after node j is crawled. Therefore, we estimate this inlink vector as the expectation over inlink counts among the set of already crawled pages,

$$\boldsymbol{s} = \frac{\boldsymbol{F}^T\boldsymbol{e}}{\|\boldsymbol{F}^T\boldsymbol{e}\|_1}. \tag{5}$$

In practice, for any given page, this estimate may not reflect the true inlinks from that page. Furthermore, this expectation is sampled from the set of links within the crawled domain, whereas a better estimate would also use links from the global domain. However, the latter distribution is not known to a localized search engine, and we contend that the above estimate will, on average, be a better estimate than the uniform distribution, for example.

Let the PageRank of $\boldsymbol{F}$ be $\boldsymbol{f}$. We express the PageRank $\boldsymbol{f}_j^+$ of the expanded local graph $\boldsymbol{F}_j$ as

$$\boldsymbol{f}_j^+ = \begin{bmatrix} (1-x_j)\boldsymbol{f}_j \\ x_j \end{bmatrix}, \tag{6}$$

where x_j is the PageRank of the candidate global node j, and $\boldsymbol{f}_j$ is the L_1-normalized PageRank vector restricted to the pages in $\boldsymbol{F}$.

Since directly optimizing our problem goal requires knowing the global PageRank $\boldsymbol{p}^*$, we instead propose to crawl those nodes that will have the *greatest* influence on the PageRanks of pages in the original local domain $\boldsymbol{L}$:

$$\begin{aligned} \textit{influence}(j) &= \sum_{k \in \boldsymbol{L}} |f_j[k] - f[k]| \\ &= \|\boldsymbol{E}_{\boldsymbol{L}}(\boldsymbol{f}_j - \boldsymbol{f})\|_1. \end{aligned} \tag{7}$$

Experimentally, the *influence* score is a very good predictor of our problem objective (3). For each candidate global node j, figure 1(a) shows the objective function value *Global Diff*($\boldsymbol{f}_j$) as a function of the influence of page j. The local domain used here is a crawl of conservative political pages (we will provide more details about this dataset in section 6); we observed similar results in other domains. The correlation is quite strong, implying that the *influence* criteria can effectively identify pages that improve the global PageRank estimate. As a baseline, figure 1(b) compares our objective with an alternative criteria, outlink count. The outlink count is defined as the number of outlinks from the local domain to page j. The correlation here is much weaker.

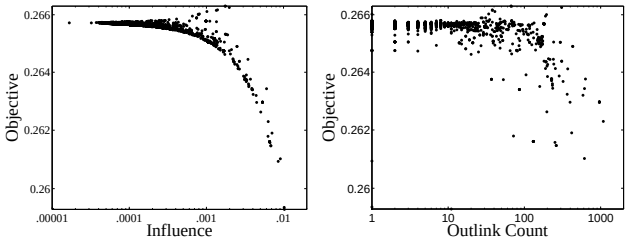


Figure 1: (a) The correlation between our *influence* page selection criteria (7) and the actual objective function (3) value is quite strong. (b) This is in contrast to other criteria, such as outlink count, which exhibit a much weaker correlation.

5.2 Computation

As described, for each candidate global page j, the *influence* score (7) must be computed. If $\boldsymbol{f}_j$ is computed exactly for each global page j, then the PageRank algorithm would need to be run for each of the $O(n)$ such global pages j we consider, resulting in an $O(n^2)$ computational cost for the node selection method. Thus, computing the exact value of $\boldsymbol{f}_j$ will lead to a quadratic algorithm, and we must instead turn to methods of approximating this vector. The algorithm we present works by performing one power method iteration used by the PageRank algorithm (Algorithm 1). The convergence rate for the PageRank algorithm has been shown to equal the random surfer probability α [7, 11]. Given a starting vector $\boldsymbol{x}^{(0)}$, if k PageRank iterations are performed, the current PageRank solution $\boldsymbol{x}^{(k)}$ satisfies:

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|_1 = O(\alpha^k \|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|_1), \tag{8}$$

where $\boldsymbol{x}^*$ is the desired PageRank vector. Therefore, if only one iteration is performed, choosing a good starting vector is necessary to achieve an accurate approximation.

We partition the PageRank matrix $\boldsymbol{P}_{\boldsymbol{F}_j}$, corresponding to the $\ell \times \ell$ subgraph $\boldsymbol{F}_j$ as:

$$\boldsymbol{P}_{\boldsymbol{F}_j} = \begin{bmatrix} \tilde{\boldsymbol{F}} & \tilde{\boldsymbol{s}} \\ \tilde{\boldsymbol{u}}_j^T & w \end{bmatrix}, \tag{9}$$

$$\begin{aligned} \text{where } \tilde{\boldsymbol{F}} &= \alpha \boldsymbol{F}(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1} + (1-\alpha)\frac{\boldsymbol{e}}{\ell+1}\boldsymbol{e}^T, \\ \tilde{\boldsymbol{s}} &= \alpha \boldsymbol{s} + (1-\alpha)\frac{\boldsymbol{e}}{\ell+1}, \\ \tilde{\boldsymbol{u}}_j &= \alpha(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1}\boldsymbol{u}_j + (1-\alpha)\frac{\boldsymbol{e}}{\ell+1}, \\ w &= \frac{1-\alpha}{\ell+1}, \end{aligned}$$

and $diag(\boldsymbol{u}_j)$ is the diagonal matrix with the $(i,i)^{th}$ entry equal to one if the i^{th} element of $\boldsymbol{u}_j$ equals one, and is zero otherwise. We have assumed here that the random surfer vector is the uniform vector, and that $\boldsymbol{L}$ has no 'dangling links'. These assumptions are not necessary and serve only to simplify discussion and analysis.

A simple approach for estimating $\boldsymbol{f}_j$ is the following. First, estimate the PageRank $\boldsymbol{f}_j^+$ of $\boldsymbol{F}_j$ by computing one PageRank iteration over the matrix $\boldsymbol{P}_{\boldsymbol{F}_j}$, using the starting vector $\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}$. Then, estimate $\boldsymbol{f}_j$ by removing the last

component from our estimate of $\boldsymbol{f}_j^+$ (i.e., the component corresponding to the added node j), and renormalizing.

The problem with this approach is in the starting vector. Recall from (6) that x_j is the PageRank of the added node j. The difference between the actual PageRank $\boldsymbol{f}_j^+$ of $\boldsymbol{P}_{\boldsymbol{F}_j}$ and the starting vector $\boldsymbol{\nu}$ is

$$\begin{aligned}\|\boldsymbol{\nu} - \boldsymbol{f}_j^+\|_1 &= x_j + \|\boldsymbol{f} - (1-x_j)\boldsymbol{f}_j\|_1 \\ &\geq x_j + |\|\boldsymbol{f}\|_1 - (1-x_j)\|\boldsymbol{f}_j\|_1| \\ &= x_j + |x_j| \\ &= 2x_j.\end{aligned}$$

Thus, by (8), after one PageRank iteration, we expect our estimate of $\boldsymbol{f}_j^+$ to still have an error of about $2\alpha x_j$. In particular, for candidate nodes j with relatively high PageRank x_j, this method will yield more inaccurate results. We will next present a method that eliminates this bias and runs in $O(n)$ time.

5.2.1 Stochastic Complementation

Since $\boldsymbol{f}_j^+$, as given in (6) is the PageRank of the matrix $\boldsymbol{P}_{\boldsymbol{F}_j}$, we have:

$$\begin{aligned}\begin{bmatrix} \boldsymbol{f}_j(1-x_j) \\ x_j \end{bmatrix} &= \begin{bmatrix} \tilde{\boldsymbol{F}} & \tilde{\boldsymbol{s}} \\ \tilde{\boldsymbol{u}}_j^T & w \end{bmatrix} \begin{bmatrix} \boldsymbol{f}_j(1-x_j) \\ x_j \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\boldsymbol{F}}\boldsymbol{f}_j(1-x_j) + \tilde{\boldsymbol{s}}x_j \\ \tilde{\boldsymbol{u}}_j^T\boldsymbol{f}_j(1-x_j) + wx_j \end{bmatrix}.\end{aligned}$$

Solving the above system for $\boldsymbol{f}_j$ can be shown to yield

$$\boldsymbol{f}_j = (\tilde{\boldsymbol{F}} + (1-w)^{-1}\tilde{\boldsymbol{s}}\tilde{\boldsymbol{u}}_j^T)\boldsymbol{f}_j. \tag{10}$$

The matrix $\boldsymbol{S} = \tilde{\boldsymbol{F}} + (1-w)^{-1}\tilde{\boldsymbol{s}}\tilde{\boldsymbol{u}}_j^T$ is known as the stochastic complement of the column stochastic matrix $\boldsymbol{P}_{\boldsymbol{F}_j}$ with respect to the sub matrix $\tilde{\boldsymbol{F}}$. The theory of stochastic complementation is well studied, and it can be shown the stochastic complement of an irreducible matrix (such as the PageRank matrix) is unique. Furthermore, the stochastic complement is also irreducible and therefore has a unique stationary distribution as well. For an extensive study, see [15].

It can be easily shown that the sub-dominant eigenvalue of $\boldsymbol{S}$ is at most $\frac{\ell+1}{\ell}\alpha$, where ℓ is the size of $\boldsymbol{F}$. For sufficiently large ℓ, this value will be very close to α. This is important, as other properties of the PageRank algorithm, notably the algorithm's sensitivity, are dependent on this value [11].

In this method, we estimate the length ℓ vector $\boldsymbol{f}_j$ by computing one PageRank iteration over the $\ell \times \ell$ stochastic complement $\boldsymbol{S}$, starting at the vector $\boldsymbol{f}$:

$$\boldsymbol{f}_j \approx \boldsymbol{S}\boldsymbol{f}. \tag{11}$$

This is in contrast to the simple method outlined in the previous section, which first iterates over the $(\ell+1)\times(\ell+1)$ matrix $\boldsymbol{P}_{\boldsymbol{F}_j}$ to estimate $\boldsymbol{f}_j^+$, and then removes the last component from the estimate and renormalizes to approximate $\boldsymbol{f}_j$. The problem with the latter method is in the choice of the $(\ell+1)$ length starting vector, $\boldsymbol{\nu}$. Consequently, the PageRank estimate given by the simple method differs from the true PageRank by at least $2\alpha x_j$, where x_j is the PageRank of page j. By using the stochastic complement, we can establish a tight lower bound of zero for this difference. To see this, consider the case in which a node k is added to $\boldsymbol{F}$ to form the augmented local subgraph $\boldsymbol{F}_k$, and that the PageRank of this new graph is $\begin{bmatrix} (1-x_k)\boldsymbol{f} \\ x_k \end{bmatrix}$. Specifically, the addition of page k does not change the PageRanks of the pages in $\boldsymbol{F}$, and thus $\boldsymbol{f}_k = \boldsymbol{f}$. By construction of the stochastic complement, $\boldsymbol{f}_k = S\boldsymbol{f}_k$, so the approximation given in equation (11) will yield the *exact* solution.

Next, we present the computational details needed to efficiently compute the quantity $\|\boldsymbol{f}_j - \boldsymbol{f}\|_1$ over all known global pages j. We begin by expanding the difference $\boldsymbol{f}_j - \boldsymbol{f}$, where the vector $\boldsymbol{f}_j$ is estimated as in (11),

$$\begin{aligned}\boldsymbol{f}_j - \boldsymbol{f} &\approx \boldsymbol{S}\boldsymbol{f} - \boldsymbol{f} \\ &= \alpha\boldsymbol{F}(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1}\boldsymbol{f} + (1-\alpha)\frac{\boldsymbol{e}}{\ell+1}\boldsymbol{e}^T\boldsymbol{f} \\ &\quad + (1-w)^{-1}(\tilde{\boldsymbol{u}}_j^T\boldsymbol{f})\tilde{\boldsymbol{s}} - \boldsymbol{f}. \end{aligned}\tag{12}$$

Note that the matrix $(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1}$ is diagonal. Letting $o[k]$ be the outlink count for page k in $\boldsymbol{F}$, we can express the k^{th} diagonal element as:

$$(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1}[k,k] = \begin{cases} \frac{1}{o[k]+1} & \text{if } u_j[k] = 1 \\ \frac{1}{o[k]} & \text{if } u_j[k] = 0 \end{cases}$$

Noting that $(o[k]+1)^{-1} = o[k]^{-1} - (o[k](o[k]+1))^{-1}$ and rewriting this in matrix form yields

$$(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1} = \boldsymbol{D}_{\boldsymbol{F}}^{-1} - \boldsymbol{D}_{\boldsymbol{F}}^{-1}(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1}diag(\boldsymbol{u}_j). \tag{13}$$

We use the same identity to express

$$\frac{\boldsymbol{e}}{\ell+1} = \frac{\boldsymbol{e}}{\ell} - \frac{\boldsymbol{e}}{\ell(\ell+1)}. \tag{14}$$

Recall that, by definition, we have $\boldsymbol{P}_{\boldsymbol{F}} = \alpha\boldsymbol{F}\boldsymbol{D}_{\boldsymbol{F}}^{-1} + (1-\alpha)\frac{\boldsymbol{e}}{\ell}$. Substituting (13) and (14) in (12) yields

$$\begin{aligned}\boldsymbol{f}_j - \boldsymbol{f} &\approx (\boldsymbol{P}_{\boldsymbol{F}}\boldsymbol{f} - \boldsymbol{f}) \\ &\quad -\alpha\boldsymbol{F}\boldsymbol{D}_{\boldsymbol{F}}^{-1}(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1}diag(\boldsymbol{u}_j)\boldsymbol{f} \\ &\quad -(1-\alpha)\frac{\boldsymbol{e}}{\ell(\ell+1)} + (1-w)^{-1}(\tilde{\boldsymbol{u}}_j^T\boldsymbol{f})\tilde{\boldsymbol{s}} \\ &= \boldsymbol{x} + \boldsymbol{y} + (\tilde{\boldsymbol{u}}_j^T\boldsymbol{f})\boldsymbol{z}, \end{aligned}\tag{15}$$

noting that by definition, $\boldsymbol{f} = \boldsymbol{P}_{\boldsymbol{F}}\boldsymbol{f}$, and defining the vectors $\boldsymbol{x}, \boldsymbol{y}$, and $\boldsymbol{z}$ to be

$$\boldsymbol{x} = -\alpha\boldsymbol{F}\boldsymbol{D}_{\boldsymbol{F}}^{-1}(\boldsymbol{D}_{\boldsymbol{F}} + diag(\boldsymbol{u}_j))^{-1}diag(\boldsymbol{u}_j)\boldsymbol{f} \tag{16}$$

$$\boldsymbol{y} = -(1-\alpha)\frac{\boldsymbol{e}}{\ell(\ell+1)} \tag{17}$$

$$\boldsymbol{z} = (1-w)^{-1}\tilde{\boldsymbol{s}}. \tag{18}$$

The first term $\boldsymbol{x}$ is a sparse vector, and takes non-zero values only for local pages k' that are siblings of the global page j. We define $(i,j) \in \boldsymbol{F}$ if and only if $\boldsymbol{F}[j,i] = 1$ (equivalently, page i links to page j) and express the value of the component $x[k']$ as:

$$x[k'] = -\alpha \sum_{k:(k,k')\in\boldsymbol{F}, u_j[k]=1} \frac{f[k]}{o[k](o[k]+1)}, \tag{19}$$

where $o[k]$, as before, is the number of outlinks from page k in the local domain. Note that the last two terms, $\boldsymbol{y}$ and $\boldsymbol{z}$ are not dependent on the current global node j. Given the function $h_j(\boldsymbol{f}) = \|\boldsymbol{y} + (\tilde{\boldsymbol{u}}_j^T\boldsymbol{f})\boldsymbol{z}\|_1$, the quantity $\|\boldsymbol{f}_j - \boldsymbol{f}\|_1$

can be expressed as

$$\begin{aligned}
\|\boldsymbol{f}_j - \boldsymbol{f}\|_1 &= \sum_k \left|x[k] + y[k] + (\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}) z[k]\right| \\
&= \sum_{k:x[k]=0} \left|y[k] + (\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}) z[k]\right| \\
&\quad + \sum_{k:x[k]\neq 0} \left|x[k] + y[k] + (\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}) z[k]\right| \\
&= h_j(\boldsymbol{f}) - \sum_{k:x[k]\neq 0} \left|y[k] + (\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}) z[k]\right| \\
&\quad + \sum_{k:x[k]\neq 0} \left|x[k] + y[k] + (\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}) z[k]\right| . \quad (20)
\end{aligned}$$

If we can compute the function h_j in linear time, then we can compute each value of $\|\boldsymbol{f}_j - \boldsymbol{f}\|_1$ using an additional amount of time that is proportional to the number of non-zero components in $\boldsymbol{x}$. These optimizations are carried out in Algorithm 3. Note that (20) computes the difference between all components of $\boldsymbol{f}$ and $\boldsymbol{f}_j$, whereas our node selection criteria, given in (7), is restricted to the components corresponding to nodes in the original local domain $\boldsymbol{L}$.

Let us examine Algorithm 3 in more detail. First, the algorithm computes the outlink counts for each page in the local domain. The algorithm then computes the quantity $\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}$ for each known global page j. This inner product can be written as

$$(1-\alpha)\frac{1}{\ell+1} + \alpha \sum_{k:(k,j)\in \boldsymbol{F}_{\text{out}}} \frac{f[k]}{o[k]+1},$$

where the second term sums over the set of local pages that link to page j. Since the total number of edges in $\boldsymbol{F}_{\text{out}}$ was assumed to have size $O(\ell)$ (recall that ℓ is the number of pages in $\boldsymbol{F}$), the running time of this step is also $O(\ell)$.

The algorithm then computes the vectors $\boldsymbol{y}$ and $\boldsymbol{z}$, as given in (17) and (18), respectively. The L_1NormDiff method is called on the components of these vectors which correspond to the pages in $\boldsymbol{L}$, and it estimates the value of $\|\boldsymbol{E}_{\boldsymbol{L}}(\boldsymbol{y} + (\tilde{\boldsymbol{u}}_j^T \boldsymbol{f})\boldsymbol{z})\|_1$ for each page j. The estimation works as follows. First, the values of $\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}$ are discretized uniformly into c values $\{a_1, ..., a_c\}$. The quantity $\|\boldsymbol{E}_{\boldsymbol{L}}(\boldsymbol{y} + a_i\boldsymbol{z})\|_1$ is then computed for each discretized value of a_i and stored in a table. To evaluate $\|\boldsymbol{E}_{\boldsymbol{L}}(\boldsymbol{y} + a\boldsymbol{z})\|_1$ for some $a \in [a_1, a_c]$, the closest discretized value a_i is determined, and the corresponding entry in the table is used. The total running time for this method is linear in ℓ and the discretization parameter c (which we take to be a constant). We note that if exact values are desired, we have also developed an algorithm that runs in $O(\ell \log \ell)$ time that is not described here.

In the main loop, we compute the vector $\boldsymbol{x}$, as defined in equation (16). The nested loops iterate over the set of pages in $\boldsymbol{F}$ that are siblings of page j. Typically, the size of this set is bounded by a constant. Finally, for each page j, the `scores` vector is updated over the set of non-zero components k of the vector $\boldsymbol{x}$ with $k \in \boldsymbol{L}$. This set has size equal to the number of *local* siblings of page j, and is a subset of the total number of siblings of page j. Thus, each iteration of the main loop takes constant time, and the total running time of the main loop is $O(\ell)$. Since we have assumed that the size of $\boldsymbol{F}$ will not grow larger than $O(n)$, the total running time for the algorithm is $O(n)$.

Algorithm 3: Node Selection via Stochastic Complementation.

SC-Select($\boldsymbol{F}$, $\boldsymbol{F}_{\text{out}}$, $\boldsymbol{f}$, k)
Input: $\boldsymbol{F}$: zero-one adjacency matrix of size ℓ corresponding to the current local subgraph, $\boldsymbol{F}_{\text{out}}$: zero-one outlink matrix from $\boldsymbol{F}$ to global subgraph, $\boldsymbol{f}$: PageRank of $\boldsymbol{F}$, k: number of pages to return
Output: pages: set of k pages to crawl next
{*Compute outlink sums for local subgraph*}
foreach (page $j \in \boldsymbol{F}$)
  o[j] $\leftarrow \sum_{k:(j,k)\in \boldsymbol{F}} F[j,k]$
end
{*Compute scalar $\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}$ for each global node j*}
foreach (page $j \in \boldsymbol{F}_{\text{out}}$)
  g[j] $\leftarrow (1-\alpha)\frac{1}{\ell+1}$
  foreach (page $k : (k,j) \in \boldsymbol{F}_{\text{out}}$)
    g[j] $\leftarrow$ g[j] $+ \alpha \frac{f[k]}{o[k]+1}$
  end
end
{*Compute vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ as in (17) and (18)*}
$\boldsymbol{y} \leftarrow -(1-\alpha)\frac{\boldsymbol{e}}{\ell(\ell+1)}$
$\boldsymbol{z} \leftarrow (1-w)^{-1}\tilde{\boldsymbol{s}}$
{*Approximate $\|\boldsymbol{y} + g[j] * \boldsymbol{z}\|_1$ for all values $g[j]$*}
norm_diffs $\leftarrow L_1$NormDiffs(g, $\boldsymbol{E}_{\boldsymbol{L}}\boldsymbol{y}$, $\boldsymbol{E}_{\boldsymbol{L}}\boldsymbol{z}$)
foreach (page $j \in \boldsymbol{F}_{\text{out}}$)
  {*Compute sparse vector $\boldsymbol{x}$ as in (19)*}
  $\boldsymbol{x} \leftarrow \boldsymbol{0}$
  foreach (page $k : (k,j) \in \boldsymbol{F}_{\text{out}}$)
    foreach (page $k' : (k,k') \in \boldsymbol{F}$))
      x[k'] $\leftarrow$ x[k'] $- \frac{f[k]}{o[k](o[k]+1)}$
    end
  end
  $\boldsymbol{x} \leftarrow \alpha\boldsymbol{x}$
  scores[j] $\leftarrow$ norm_diffs[j]
  foreach ($k : x[k] > 0$ and page $k \in \boldsymbol{L}$)
    scores[j] $\leftarrow$ scores[j] $- |y[k] + g[j] * z[k]|$
      $+|x[k] + y[k] + g[j] * z[k])|$
  end
end
Return k pages with highest scores

5.2.2 PageRank Flows

We now present an intuitive analysis of the stochastic complementation method by decomposing the change in PageRank in terms of 'leaks' and 'flows'. This analysis is motivated by the decomposition given in (15). PageRank 'flow' is the increase in the local PageRanks originating from global page j. The flows are represented by the non-negative vector $(\tilde{\boldsymbol{u}}_j^T \boldsymbol{f})\boldsymbol{z}$ (equations (15) and (18)). The scalar $\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}$ can be thought of as the total amount of PageRank flow that page j has available to distribute. The vector $\boldsymbol{z}$ dictates how the flow is allocated to the local domain; the flow that local page k receives is proportional to (within a constant factor due to the random surfer vector) the expected number of its inlinks.

The PageRank 'leaks' represent the decrease in PageRank resulting from the addition of page j. The leakage can be quantified in terms of the non-positive vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ (equations (16) and (17)). For vector $\boldsymbol{x}$, we can see from equation (19) that the amount of PageRank leaked by a local page is proportional to the weighted sum of the Page-

Ranks of its siblings. Thus, pages that have siblings with higher PageRanks (and low outlink counts) will experience more leakage. The leakage caused by $\boldsymbol{y}$ is an artifact of the random surfer vector.

We will next show that if only the 'flow' term, $(\tilde{\boldsymbol{u}}_j^T \boldsymbol{f})\boldsymbol{z}$, is considered, then the resulting method is very similar to a heuristic proposed by Cho *et al.* [6] that has been widely used for the "Crawling Through URL Ordering" problem. This heuristic is computationally cheaper, but as we will see later, not as effective as the Stochastic Complementation method.

Our node selection strategy chooses global nodes that have the largest *influence* (equation (7)). If this influence is approximated using only 'flows', the optimal node j^* is:

$$\begin{aligned} j^* &= \text{argmax}_j \|\boldsymbol{E_L}\tilde{\boldsymbol{u}}_j^T \boldsymbol{f}\boldsymbol{z}\|_1 \\ &= \text{argmax}_j \tilde{\boldsymbol{u}}_j^T \boldsymbol{f} \|\boldsymbol{E_L}\boldsymbol{z}\|_1 \\ &= \text{argmax}_j \tilde{\boldsymbol{u}}_j^T \boldsymbol{f} \\ &= \text{argmax}_j \langle \alpha(\boldsymbol{D_F} + diag(\boldsymbol{u}_j))^{-1}\boldsymbol{u}_j + (1-\alpha)\frac{\boldsymbol{e}}{\ell+1}, \boldsymbol{f}\rangle \\ &= \text{argmax}_j \boldsymbol{f}^T(\boldsymbol{D_F} + diag(\boldsymbol{u}_j))^{-1}\boldsymbol{u}_j. \end{aligned}$$

The resulting page selection score can be expressed as a sum of the PageRanks of each local page k that links to j, where each PageRank value is normalized by $o[k]+1$. Interestingly, the normalization that arises in our method differs from the heuristic given in [6], which normalizes by $o[k]$. The algorithm PF-SELECT, which is omitted due to lack of space, first computes the quantity $\boldsymbol{f}^T(\boldsymbol{D_F}+diag(\boldsymbol{u}_j))^{-1}\boldsymbol{u}_j$ for each global page j, and then returns the pages with the k largest scores. To see that the running time for this algorithm is $O(n)$, note that the computation involved in this method is a subset of that needed for the SC-SELECT method (Algorithm 3), which was shown to have a running time of $O(n)$.

6. EXPERIMENTS

In this section, we provide experimental evidence to verify the effectiveness of our algorithms. We first outline our experimental methodology and then provide results across a variety of local domains.

6.1 Methodology

Given the limited resources available at an academic institution, crawling a section of the web that is of the same magnitude as that indexed by Google or Yahoo! is clearly infeasible. Thus, for a given local domain, we approximate the global graph by crawling a local neighborhood around the domain that is several orders of magnitude larger than the local subgraph. Even though such a graph is still orders of magnitude smaller than the 'true' global graph, we contend that, even if there exist some highly influential pages that are very far away from our local domain, it is unrealistic for *any* local node selection algorithm to find them. Such pages also tend to be highly unrelated to pages within the local domain.

When explaining our node selection strategies in section 5, we made the simplifying assumption that our local graph contained no dangling nodes. This assumption was only made to ease our analysis. Our implementation efficiently handles dangling links by replacing each zero column of our adjacency matrix with the uniform vector. We evaluate the algorithm using the two node selection strategies given in Section 5.2, and also against the following baseline methods:

- RANDOM: Nodes are chosen uniformly at random among the known global nodes.
- OUTLINKCOUNT: Global nodes with the highest number of outlinks from the local domain are chosen.

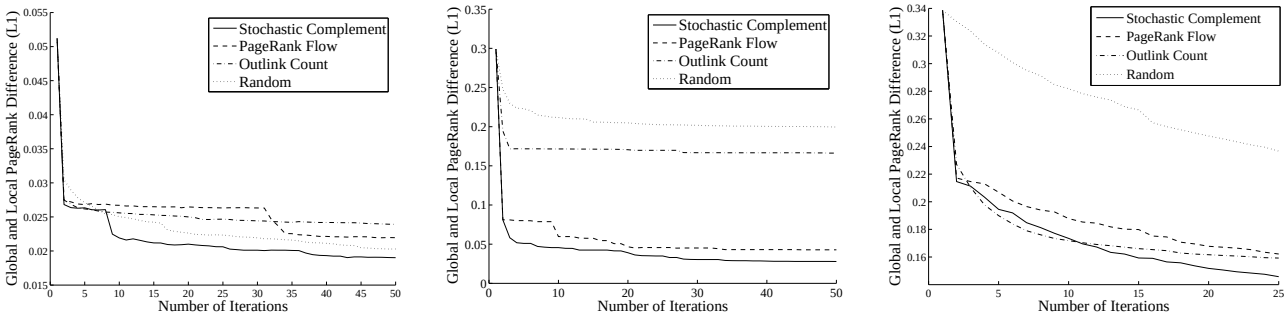
At each iteration of the FINDGLOBALPR algorithm, we evaluate performance by computing the difference between the current PageRank estimate of the local domain, $\frac{\boldsymbol{E_L f}}{\|\boldsymbol{E_L f}\|_1}$, and the global PageRank of the local domain $\frac{\boldsymbol{E_L g}}{\|\boldsymbol{E_L g}\|_1}$. All PageRank calculations were performed using the uniform random surfer vector. Across all experiments, we set the random surfer parameter α, to be .85, and used a convergence threshold of 10^{-6}. We evaluate the difference between the local and global PageRank vectors using three different metrics: the L_1 and L_∞ norms, and Kendall's tau. The L_1 norm measures the sum of the absolute value of the differences between the two vectors, and the L_∞ norm measures the absolute value of the largest difference. Kendall's tau metric is a popular rank correlation measure used to compare PageRanks [2, 11]. This metric can be computed by counting the number of pairs of pairs that agree in ranking, and subtracting from that the number of pairs of pairs that disagree in ranking. The final value is then normalized by the total number of $\binom{n}{2}$ such pairs, resulting in a $[-1, 1]$ range, where a negative score signifies anti-correlation among rankings, and values near one correspond to strong rank correlation.

6.2 Results

Our experiments are based on two large web crawls and were downloaded using the web crawler that is part of the Nutch open source search engine project [18]. All crawls were restricted to only 'http' pages, and to limit the number of dynamically generated pages that we crawl, we ignored all pages with urls containing any of the characters '?', '*', '@', or '='. The first crawl, which we will refer to as the 'edu' dataset, was seeded by homepages of the top 100 graduate computer science departments in the USA, as rated by the US News and World Report [16], and also by the home pages of their respective institutions. A crawl of depth 5 was performed, restricted to pages within the '.edu' domain, resulting in a graph with approximately 4.7 million pages and 22.9 million links. The second crawl was seeded by the set of pages under the 'politics' hierarchy in the dmoz open directory project[17]. We crawled all pages up to four links away, which yielded a graph with 4.4 million pages and 17.3 million links.

Within the 'edu' crawl, we identified the five site-specific domains corresponding to the websites of the top five graduate computer science departments, as ranked by the US News and World Report. This yielded local domains of various sizes, from 10,626 (UIUC) to 59,895 (Berkeley). For each of these site-specific domains with size n, we performed 50 iterations of the FINDGLOBALPR algorithm to crawl a total of $2n$ additional nodes. Figure 2(a) gives the (L_1) difference from the PageRank estimate at each iteration to the global PageRank, for the Berkeley local domain.

The performance of this dataset was representative of the typical performance across the five computer science site-specific local domains. Initially, the L_1 difference between the global and local PageRanks ranged from .0469 (Stanford) to .149 (MIT). For the first several iterations, the



(a) www.cs.berkeley.edu (b) www.enterstageright.com (c) Politics

Figure 2: L_1 difference between the estimated and true global PageRanks for (a) Berkeley's computer science website, (b) the site-specific domain, www.enterstageright.com, and (c) the 'politics' topic-specific domain. The stochastic complement method outperforms all other methods across various domains.

three link-based methods all outperform the random selection heuristic. After these initial iterations, the random heuristic tended to be more competitive with (or even outperform, as in the Berkeley local domain) the outlink count and PageRank flow heuristics. In all tests, the stochastic complementation method either outperformed, or was competitive with, the other methods. Table 1 gives the average difference between the final estimated global PageRanks and the true global PageRanks for various distance measures.

Algorithm	L_1	L_∞	Kendall
Stoch. Comp.	**.0384**	**.00154**	**.9257**
PR Flow	.0470	.00272	.8946
Outlink	.0419	.00196	.9053
Random	.0407	.00204	.9086

Table 1: Average final performance of various node selection strategies for the five site-specific computer science local domains. Note that Kendall's Tau measures similarity, while the other metrics are dissimilarity measures. Stochastic Complementation clearly outperforms the other methods in all metrics.

Within the 'politics' dataset, we also performed two site-specific tests for the largest websites in the crawl: www.adamsmith.org, the website for the London based Adam Smith Institute, and www.enterstageright.com, an online conservative journal. As with the 'edu' local domains, we ran our algorithm for 50 iterations, crawling a total of $2n$ nodes. Figure 2 (b) plots the results for the www.enterstageright.com domain. In contrast to the 'edu' local domains, the RANDOM and OUTLINKCOUNT methods were not competitive with either the SC-SELECT or the PF-SELECT methods. Among all datasets and all node selection methods, the stochastic complementation method was most impressive in this dataset, realizing a final estimate that differed only .0279 from the global PageRank, a ten-fold improvement over the initial local PageRank difference of .299. For the Adam Smith local domain, the initial difference between the local and global PageRanks was .148, and the final estimates given by the SC-SELECT, PF-SELECT, OUTLINKCOUNT, and RANDOM methods were .0208, .0193, .0222, and .0356, respectively.

Within the 'politics' dataset, we constructed four topic-specific local domains. The first domain consisted of all pages in the dmoz politics category, and also all pages within each of these sites up to two links away. This yielded a local domain of 90,811 pages, and the results are given in figure 2 (c). Because of the larger size of the topic-specific domains, we ran our algorithm for only 25 iterations to crawl a total of n nodes.

We also created topic-specific domains from three political sub-topics: liberalism, conservatism, and socialism. The pages in these domains were identified by their corresponding dmoz categories. For each sub-topic, we set the local domain to be all pages within three links from the corresponding dmoz category pages. Table 2 summarizes the performance of these three topic-specific domains, and also the larger political domain.

To quantify a global page j's effect on the global PageRank values of pages in the local domain, we define page j's *impact* to be its PageRank value, $g[j]$, normalized by the fraction of its outlinks pointing to the local domain:

$$impact(j) = \frac{o_L[j]}{o[j]} \cdot g[j],$$

where, $o_L[j]$ is the number of outlinks from page j to pages in the local domain $\boldsymbol{L}$, and $o[j]$ is the total number of j's outlinks. In terms of the random surfer model, the *impact* of page j is the probability that the random surfer (1) is currently at global page j in her random walk and (2) takes an outlink to a local page, given that she has already decided not to jump to a random page.

For the politics local domain, we found that many of the pages with high *impact* were in fact political pages that should have been included in the dmoz politics topic, but were not. For example, the two most influential global pages were the political search engine www.askhenry.com, and the home page of the online political magazine, www.policyreview.com. Among non-political pages, the home page of the journal "Education Next" was most influential. The journal is freely available online and contains articles regarding various aspect of K-12 education in America. To provide some anecdotal evidence for the effectiveness of our page selection methods, we note that the SC-SELECT method chose 11 pages within the www.educationnext.org domain, the PF-SELECT method discovered 7 such pages, while the OUTLINKCOUNT and RANDOM methods found only 6 pages each.

For the conservative political local domain, the socialist website www.ornery.org had a very high *impact* score. This

All Politics:

Algorithm	L_1	L_2	Kendall
Stoch. Comp.	**.1253**	**.000700**	**.8671**
PR Flow	.1446	.000710	.8518
Outlink	.1470	.00225	.8642
Random	.2055	.00203	.8271

Conservativism:

Algorithm	L_1	L_2	Kendall
Stoch. Comp.	**.0496**	**.000990**	**.9158**
PR Flow	.0554	.000939	.9028
Outlink	.0602	.00527	.9144
Random	.1197	.00102	.8843

Liberalism:

Algorithm	L_1	L_2	Kendall
Stoch. Comp.	**.0622**	**.001360**	**.8848**
PR Flow	.0799	.001378	.8669
Outlink	.0763	.001379	.8844
Random	.1127	.001899	.8372

Socialism:

Algorithm	L_1	L_∞	Kendall
Stoch. Comp.	.04318	.00439	**.9604**
PR Flow	.0450	.004251	.9559
Outlink	**.04282**	**.00344**	.9591
Random	.0631	.005123	.9350

Table 2: Final performance among node selection strategies for the four political topic-specific crawls. Note that Kendall's Tau measures similarity, while the other metrics are dissimilarity measures.

was largely due to a link from the front page of this site to an article regarding global warming published by the National Center for Public Policy Research, a conservative research group in Washington, DC. Not surprisingly, the global PageRank of this article (which happens to be on the home page of the NCCPR, `www.nationalresearch.com`), was approximately .002, whereas the local PageRank of this page was only .00158. The SC-SELECT method yielded a global PageRank estimate of approximately .00182, the PF-SELECT method estimated a value of .00167, and the RANDOM and OUTLINKCOUNT methods yielded values of .01522 and .00171, respectively.

7. RELATED WORK

The node selection framework we have proposed is similar to the url ordering for crawling problem proposed by Cho *et al.* in [6]. Whereas our framework seeks to minimize the difference between the global and local PageRank, the objective used in [6] is to crawl the most highly (globally) ranked pages first. They propose several node selection algorithms, including the outlink count heuristic, as well as a variant of our PF-SELECT algorithm which they refer to as the 'PageRank ordering metric'. They found this method to be most effective in optimizing their objective, as did a recent survey of these methods by Baeza-Yates *et al.* [1]. Boldi *et al.* also experiment within a similar crawling framework in [2], but quantify their results by comparing Kendall's rank correlation between the PageRanks of the current set of crawled pages and those of the entire global graph. They found that node selection strategies that crawled pages with the highest *global* PageRank first actually performed worse (with respect to Kendall's Tau correlation between the local and global PageRanks) than basic depth first or breadth first strategies. However, their experiments differ from our work in that our node selection algorithms do not use (or have access to) global PageRank values.

Many algorithmic improvements for computing *exact* PageRank values have been proposed [9, 10, 14]. If such algorithms are used to compute the global PageRanks of our local domain, they would all require $O(N)$ computation, storage, and bandwidth, where N is the size of the global domain. This is in contrast to our method, which *approximates* the global PageRank and scales linearly with the size of the local domain.

Wang and Dewitt [22] propose a system where the set of web servers that comprise the global domain communicate with each other to compute their respective global PageRanks. For a given web server hosting n pages, the computational, bandwidth, and storage requirements are also linear in n. One drawback of this system is that the number of distinct web servers that comprise the global domain can be very large. For example, our 'edu' dataset contains websites from over 3,200 different universities; coordinating such a system among a large number of sites can be very difficult.

Gan, Chen, and Suel propose a method for estimating the PageRank of a single page [5] which uses only constant bandwidth, computation, and space. Their approach relies on the availability of a remote connectivity server that can supply the set of inlinks to a given page, an assumption not used in our framework. They experimentally show that a reasonable estimate of the node's PageRank can be obtained by visiting at most a few hundred nodes. Using their algorithm for our problem would require that either the entire global domain first be downloaded or a connectivity server be used, both of which would lead to very large web graphs.

8. CONCLUSIONS AND FUTURE WORK

The internet is growing exponentially, and in order to navigate such a large repository as the web, global search engines have established themselves as a necessity. Along with the ubiquity of these large-scale search engines comes an increase in search users' expectations. By providing complete and isolated coverage of a particular web domain, localized search engines are an effective outlet to quickly locate content that could otherwise be difficult to find. In this work, we contend that the use of global PageRank in a localized search engine can improve performance.

To estimate the global PageRank, we have proposed an iterative node selection framework where we select which pages from the global frontier to crawl next. Our primary contribution is our stochastic complementation page selection algorithm. This method crawls nodes that will most significantly impact the local domain and has running time linear in the number of nodes in the local domain. Experimentally, we validate these methods across a diverse set of local domains, including seven site-specific domains and four topic-specific domains. We conclude that by crawling an additional n or $2n$ pages, our methods find an estimate of the global PageRanks that is up to ten times better than just using the local PageRanks. Furthermore, we demonstrate that our algorithm consistently outperforms other existing heuristics.

Often times, topic-specific domains are discovered using a focused web crawler which considers a page's content in conjunction with link anchor text to decide which pages to crawl next [4]. Although such crawlers have proven to be quite effective in discovering topic-related content, many irrelevant pages are also crawled in the process. Typically, these pages are deleted and not indexed by the localized search engine. These pages can of course provide valuable information regarding the global PageRank of the local domain. One way to integrate these pages into our framework is to start the FindGlobalPR algorithm with the current subgraph $\boldsymbol{F}$ equal to the set of pages that were crawled by the focused crawler.

The global PageRank estimation framework, along with the node selection algorithms presented, all require $O(n)$ computation per iteration and bandwidth proportional to the number of pages crawled, Tk. If the number of iterations T is relatively small compared to the number of pages crawled per iteration, k, then the bottleneck of the algorithm will be the crawling phase. However, as the number of iterations increases (relative to k), the bottleneck will reside in the node selection computation. In this case, our algorithms would benefit from constant factor optimizations. Recall that the FindGlobalPR algorithm (Algorithm 2) requires that the PageRanks of the current expanded local domain be recomputed in each iteration. Recent work by Langville and Meyer [12] gives an algorithm to quickly recompute PageRanks of a given webgraph if a small number of nodes are added. This algorithm was shown to give speedup of five to ten times on some datasets. We plan to investigate this and other such optimizations as future work.

In this paper, we have objectively evaluated our methods by measuring how close our global PageRank estimates are to the actual global PageRanks. To determine the benefit of using global PageRanks in a localized search engine, we suggest a user study in which users are asked to rate the quality of search results for various search queries. For some queries, only the local PageRanks are used in ranking, and for the remaining queries, local PageRanks and the approximate global PageRanks, as computed by our algorithms, are used. The results of such a study can then be analyzed to determine the added benefit of using the global PageRanks computed by our methods, over just using the local PageRanks.

Acknowledgements. This research was supported by NSF grant CCF-0431257, NSF Career Award ACI-0093404, and a grant from Sabre, Inc.

9. REFERENCES

[1] R. Baeza-Yates, M. Marin, C. Castillo, and A. Rodriguez. Crawling a country: better strategies than breadth-first for web page ordering. *World-Wide Web Conference*, 2005.

[2] P. Boldi, M. Santini, and S. Vigna. Do your worst to make the best: paradoxical effects in pagerank incremental computations. *Workshop on Web Graphs*, 3243:168–180, 2004.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 33(1–7):107–117, 1998.

[4] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *World-Wide Web Conference*, 1999.

[5] Y. Chen, Q. Gan, and T. Suel. Local methods for estimating pagerank values. *Conference on Information and Knowledge Management*, 2004.

[6] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. *World-Wide Web Conference*, 1998.

[7] T. H. Haveliwala and S. D. Kamvar. The second eigenvalue of the Google matrix. Technical report, Stanford University, 2003.

[8] T. Joachims, F. Radlinski, L. Granka, A. Cheng, C. Tillekeratne, and A. Patel. Learning retrieval functions from implicit feedback. http://www.cs.cornell.edu/People/tj/career.

[9] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Exploiting the block structure of the web for computing pagerank. *World-Wide Web Conference*, 2003.

[10] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Extrapolation methods for accelerating pagerank computation. *World-Wide Web Conference*, 2003.

[11] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 2004.

[12] A. N. Langville and C. D. Meyer. Updating the stationary vector of an irreducible markov chain with an eye on Google's pagerank. *SIAM Journal on Matrix Analysis*, 2005.

[13] P. Lyman, H. R. Varian, K. Swearingen, P. Charles, N. Good, L. L. Jordan, and J. Pal. How much information 2003? *School of Information Management and System, University of California at Berkely*, 2003.

[14] F. McSherry. A uniform approach to accelerated pagerank computation. *World-Wide Web Conference*, 2005.

[15] C. D. Meyer. Stochastic complementation, uncoupling markov chains, and the theory of nearly reducible systems. *SIAM Review*, 31:240–272, 1989.

[16] US News and World Report. http://www.usnews.com.

[17] Dmoz open directory project. http://www.dmoz.org.

[18] Nutch open source search engine. http://www.nutch.org.

[19] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

[20] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the Twenty-seventh International Conference on Very Large Databases*, 2001.

[21] T. Tin Tang, D. Hawking, N. Craswell, and K. Griffiths. Focused crawling for both topical relevance and quality of medical information. *Conference on Information and Knowledge Management*, 2005.

[22] Y. Wang and D. J. DeWitt. Computing pagerank in a distributed internet search system. *Proceedings of the 30th VLDB Conference*, 2004.