Internet Mathematics Vol. 3, No. 3: 257-294

Approximating Personalized PageRank with Minimal Use of Web Graph Data

David Gleich and Marzia Polito

Abstract. In this paper, we consider the problem of calculating fast and accurate approximations to the personalized PageRank score of a webpage. We focus on techniques to improve speed by limiting the amount of web graph data we need to access.

Our algorithms provide both the approximation to the personalized PageRank score as well as guidance in using only the necessary information—and therefore sensibly reduce not only the computational cost of the algorithm but also the memory and memory bandwidth requirements. We report experiments with these algorithms on web graphs of up to 118 million pages and prove a theoretical approximation bound for all. Finally, we propose a local, personalized web-search system for a future client system using our algorithms.

1. Introduction and Motivation

To have web search results that are personalized, we claim that there is no need to access data from the whole web. In fact, it is likely that the majority of the webpages are totally unrelated to the interests of any one user.

In the original PageRank paper [Brin and Page 98], Brin and Page proposed a personalized version of the algorithm for the goal of user-specific page ranking. While the PageRank algorithm models a random surfer that teleports everywhere in the web graph, the random surfer in the personalized PageRank Markov chain only teleports to a few pages of personal interest. As a consequence, the personalization vector is usually sparse, and the value of a personalized score will be negligible or zero on most of the web.

In this paper, we present accurate and efficient algorithms for computing approximations of personalized PageRank without having to access the entire web graph matrix. Our algorithms iteratively divide the vertices of the web graph into an *active* set and an *inactive* set. At each iteration, the set of active vertices is expanded to include more pages that are likely to have a high personalized PageRank score. Only the set of active pages and their out-link information are actually involved in the computation, sensibly decreasing the computational time. A tolerance threshold is fixed in order to determine the termination of the approximation algorithm.

We provide theoretical bounds for such approximations, but we also show that in practice the approximation is even better than the bounds would suggest. We perform experiments on web graphs of up to 118 million nodes.

Our approach substantially differs from the one followed by Jeh and Widom [Jeh and Widom 03]. They developed a sophisticated computational scheme in order to efficiently compute a large number of such personalized PageRank vectors. This approach involves pre-computation and storage of a number of personalized vectors or their building blocks. We merely reduce the data used in the algorithm to save on computation time. In fact, we do need to have random access to a web graph, but in practice we only need to access a small part of it. A priori we do not know which part it is, but we iteratively discover this part while running the algorithms.

In the second set of algorithms, we use coarse level information from the *host graph* [Kamvar et al. 03a]. This graph has a vertex for each host, and its edges summarize all the links between individual pages. The storage space for the host graph as well as the computational time needed to deal with it are considerably less than the ones relative to the whole web graph.

In building our algorithms we were inspired and motivated by a possible new usage model, where the client machine hosts a fully-functional web-search system that employs, among many others, our algorithms. The analysis of such a system is beyond the scope of this paper, but we do provide a high-level vision of it.

We envision the incremental discovery of the web graph and the computation of personalized PageRank scores as performed by a client machine via a lightweight crawler integrated with the algorithm itself. Our algorithms make it feasible to have a personalized search mostly based on the client. Such a personalized search engine has important consequences for the privacy of users and for exploitation of the computational potential of client machines in web search. While we do not have a complete implementation of such a system yet, we believe it is a feasible goal. The next-generation machines will have significantly increased processor speed, memory, and disk-space. In light of these assumptions, we performed our experimentation on a high-end machine. We considered it as a proxy for a future client desktop or laptop of an average web user. Our algorithms are designed to provide a computational scheme with respect to this context and not to the standard server-based architecture of most popular search engines.

In Section 2 we give a detailed description of our four algorithms, while in Section 3 we state and prove theoretical approximation bounds for each algorithm. Then, in Section 4 we describe our experiments with the algorithms and show that the approximation achieved is acceptable. We also evaluate the different algorithms in terms of the trade-off between their accuracy and efficiency.

In Section 5 we give a high-level description of a personalized and privacy-protecting web-search system. This system uses our approximate algorithms for part of the ranking functionality and, in fact, motivated their development. Finally, we analyze the related work in Section 6 and summarize our conclusions in Section 7 as well as indicate future directions for the development of these ideas in Section 8.

2. Algorithms for Computing Personalized PageRank

Many authors have described the standard PageRank algorithm applied to an entire graph [Brin and Page 98, Kamvar et al. 03a, Arasu et al. 02]. We briefly review the algorithm here. Given an adjacency matrix W for a web graph $\mathcal{W}$ with vertex set $\mathcal{V}$, the PageRank algorithm computes the stationary distribution of a random-walk Markov chain where, with probability α, the walker follows the out-links of a given page and, with probability $1-\alpha$, the walker teleports to a page based on the probability distribution vector v. We summarize the relevant notation in Table 1.

Algorithm 1. (Standard PageRank algorithm.)

```
$x^{(0)} = v, k = 0$
**repeat**
  $y = \alpha W^T D_W^{-1} x^{(k)}$
  $\omega = 1 - ||y||_1$
  $x^{(k+1)} = y + \omega v$
  $\delta = ||x^{(k+1)} - x^{(k)}||_1, k = k + 1$
**until** $\delta$ is less than stopping tolerance.
```

Symbol	Meaning	Sec.
$1-\alpha$	random teleportation probability	2
A	PageRank Markov chain transition matrix	2
$\mathcal{B}$	set of host vertices in graph $\mathcal{C}$	2.3
C	combination page and host adjacency matrix	2.3
δ	total change in PageRank in an iteration	2
D_W	diagonal matrix of out-degrees for adjacency matrix W	2
D_W^{-1}	diagonal matrix of inverse out-degrees for adjacency matrix W	2
d_W	dangling node indicator vector for adjacency matrix W	2
e	vector of all ones	2
ε	the expansion tolerance	2.1
$\mathcal{F}$	frontier set of pages	2.2
H	host graph adjacency matrix	2.3
κ	smallest expansion criteria for the boundary set	2.2
L	adjacency matrix for a set of active pages	2.1
$\mathcal{L}$	set of active pages	2.1
l_v	number of in-links within a host for page v	2.3
$\mathcal{P}$	set of vertices corresponding to separated pages	2.3
p	a single personalization vertex	2.1
R	restriction matrix for the page to host restriction operator	2.3
$\mathcal{S}$	set of pages to separate	2.3
v	teleportation distribution	2
$\mathcal{V}$	set of vertices in a graph	2
W	adjacency matrix for web graph $\mathcal{W}$	2
$x(i)$	the entry in vector x for page i	2.1
∂	the boundary of a domain	2.2

Table 1. Summary of notation and the sections where it is introduced. In general, lower-case Greek letters are scalar values, lower-case letters are vectors or set elements, upper-case letters are matrices, and script letters are graphs or sets.

We can efficiently compute this stationary distribution by applying the power method to the stochastic transition matrix

$$A^T = \alpha(W^T D_W^{-1} + v d_W^T) + (1-\alpha) v e^T,$$

where for graph $\mathcal{W}$ with adjacency matrix W, D_W is the diagonal matrix of out-degrees for nodes in W, d_W is the dangling node indicator vector, and e is the vector of all ones. This idea gives rise to the standard PageRank algorithm in Algorithm 1. We use the notation D_W^{-1} to indicate the matrix with the inverse out-degree for each node. Also, δ, calculated in the algorithm, is the total change in PageRank in an iteration.

If v is the uniform distribution over all pages, then Algorithm 1 computes a *global PageRank vector*. Haveliwela computed and analyzed the *topic PageRank vector* by taking v as a uniform distribution over a subset of pages chosen to correspond to a particular topic [Haveliwala 02]. If v is a subset of pages chosen according to a user's interests, the algorithm computes a *personalized PageRank vector* (PPR) [Brin and Page 98].

In the remainder of this section, we will state four algorithms for approximate personalized PageRank computations. Then, we show that each algorithm terminates.

2.1. Restricted Personalized PageRank Algorithm

The restricted personalized PageRank algorithm (RPPR) attempts to determine the set of pages with high personalized PageRank and compute the PageRank vector corresponding to this limited set. We apply one iteration of the PageRank algorithm to the current set of active nodes and expand any nodes above the tolerance ε. The motivation for this algorithm is a formula from Jeh and Widom [Jeh and Widom 03, Section 4.1], which identifies the PageRank value for a single-page personalization vector with the inverse P-distance in the web graph. For simplicity of presentation, we assume that this algorithm is only applied to compute the personalized PageRank vector for a single page p.

In subsequent algorithms, we treat a vector x or y as a function of pages so that $x(i)$ is the entry in vector x associated with page i. The goal is to compute a set of active pages, $\mathcal{L}$, and a corresponding active link matrix, L. The matrix L is the adjacency matrix for all the out-links from pages in $\mathcal{L}$ and may reference pages not in $\mathcal{L}$ (but those pages will not have any out-links in L). We then compute one PageRank iteration on L from a vector x to a vector y and update both $\mathcal{L}$ and L with out-links from any page i that satisfies $y(i) > \varepsilon$.

Algorithm 2 is implemented in the `pagerank.m` function available from http://www.stanford.edu/~dgleich/programs.html.

Algorithm 2. (Restricted personalized PageRank algorithm.)

```
$x(p) = 1, \mathcal{L} = \{p\}, L =$ matrix of $p$ to its out-links
**repeat**
  $y = \alpha L^T D_L^{-1} x$ {The PageRank iteration is here.}
  $\omega = 1 - ||y||_1$
  $y(p) = y(p) + \omega$
  Add out-links for pages $v$ where $y(v) > \varepsilon$ to $\mathcal{L}$ and $L$, expanding $L$ as
  necessary.
  $\delta = ||x - y||_1$, $x = y$
**until** $\delta$ is less than stopping tolerance.
```

Algorithm 3. (Boundary-restricted personalized PageRank algorithm.)

```
$x(p) = 1$
**repeat**
  $y = \alpha L^T D_L^{-1} x$ {The PageRank iteration is here.}
  $\omega = 1 - ||y||_1$
  $y(p) = y(p) + \omega$
  Compute $y(\mathcal{F})$ {The total rank on the frontier.}
  **while** $y(\mathcal{F}) > \kappa$ **do**
    Find the page $v \in \mathcal{F}$ with maximum value in $y$.
    Add $v$ to $\mathcal{L}$ and $L$, remove the page from $\mathcal{F}$, and update $y(\mathcal{F})$.
  **end while**
  $\delta = ||y - x||_1$, $x = y$
**until** $\delta$ is less than stopping tolerance.
```

2.2. Boundary-Restricted Personalized PageRank Algorithm

Algorithm 3 is a slight modification to the previous algorithm and was suggested by our theoretical results about the approximation achieved by Algorithm 2.

Instead of expanding pages when the PageRank tolerance is above an expansion tolerance ε, we expand pages until the total rank on the frontier set of pages is less than κ. Let $\mathcal{F}$ denote the set of pages for the frontier. That is,

$$\mathcal{F} = \partial\mathcal{L}$$

is the set of pages that we know exist due to other out-links but have not yet visited. We borrow the notation ∂ to represent the boundary of a domain—in this case $\mathcal{L}$, the set of active pages. Algorithm 3 is also implemented in the `pagerank.m` function, which is available from http://www.stanford.edu/~dgleich/programs.html.

In our implementation, we sort the vector y first so that we can simply examine the pages in sorted order in y. There are no updates to the PageRank vector as we are adding pages to $\mathcal{L}$. We simply take the current set of values in y and add pages to $\mathcal{L}$ until the rank on the remainder of the frontier is less than κ. A complete sorting procedure for y is, in reality, unnecessary, since a limited number of highest-ranking pages will be removed from the frontier at each step.

2.3. PageHostRank Algorithm

PageHostRank (PHRank) is a hybrid algorithm that attempts to meld ideas from Algorithms 2 and 3 together with an extra set of global data. The global data comes in the form of a host graph.

We use the host graph in the sense of Kamvar et al. [Kamvar et al. 03a]. That is, we view a *host* as a collection of webpages that all share the same root in the URL. For example, the pages http://www.intel.com/ and http://www.intel.com/personal/index.htm share the same root (www.intel.com), whereas http://research.intel.com/ir/researchareas/index.asp has a different root (research.intel.com). The host graph adjacency matrix H then has a nonzero entry h_{IJ}, representing a weighted edge between the vertices for hosts I and J, if host I contains any page that links to a page on host J. The weight on the edge is the number of links from pages on host I to pages on host J.

We can formalize these ideas by using a restriction operator. Let R be the restriction matrix of the page to host the restriction operator such that $R_{iJ} = 1$ if webpage i belongs to host J. Then, R is an $n \times m$ matrix where n is the number of pages and m is the number of hosts. There is exactly one nonzero element in each row of R. The host graph adjacency matrix H is

$$H = R^T W R.$$

We assume that the PageHostRank algorithm has two pieces of global information. The first is the host graph $\mathcal{H}$ itself, together with the hosts for each node in $\mathcal{H}$ and a vector of counts of dangling vertices for each host, d_H. The second is a vector l over all pages on the web specifying the number of in-links from pages on the host to that page. Formally, if $R_{iJ} = 1$, then l_i is the number of links from pages in host J to page i.

In Section 3, we introduce a random surfer model that provides an intuitive justification for the PageHostRank algorithm. However, that model is not realistic. In our perspective, the PageHostRank algorithm is a way of representing the actions of a random surfer initially unaware of the content of a host but slowly discovering that content. This model may not reflect reality, but it does reflect the level of prior information that we assume when starting the algorithm.

At a high level, the PageHostRank algorithm runs iterations of personalized PageRank on the host graph. Whenever a host acquires sufficient rank, we separate all known pages from the host agglomeration and connect those pages based on their out-links. These separated pages become active. (In some sense, we reagglomerate the graph where all separated and active pages become individual hosts.)

To be precise, let us describe the process of separating one page from its host. Let $\mathcal{C}$ be the hybrid combination graph at the current iteration. Then, $\mathcal{C}$ is a weighted graph where the weight of an edge counts the number of aggregated links. Within $\mathcal{C}$ we have a set $\mathcal{P}$ of vertices corresponding to separated pages and a set $\mathcal{B}$ of hosts (or blocks of pages to understand the mnemonic). Let j be the page that we expand, and let J be the corresponding host ($R_{jJ} = 1$). Finally, let $\mathcal{F}$ be the frontier set of pages, the set of pages we know exist but have not yet separated. First, we remove j from $\mathcal{F}$, fetch the out-links for page j, and insert links with weight 1 for any links between page j and any pages in $\mathcal{P}$. Note that we can only ever separate pages in the frontier (pages in $\mathcal{F}$) because we do not know about other pages on the web. If j is a dangling page, subtract one from the dangling count for host J, $d_H(J)$. For any out-links to pages not in $\mathcal{P}$, we aggregate the out-links at the host level and insert weighted links between j and any of the hosts in $\mathcal{B}$. Further, we add all such pages to the frontier set $\mathcal{F}$.

Next, we insert any in-links from pages in the set $\mathcal{P}$ to j. Then, we add the page j to the set $\mathcal{P}$. Finally, we insert a weighted in-link from the host J to page j based on (a) the value l_j in the within-host in-link count and (b) the number of in-links to page j from pages in host J and in $\mathcal{P}$. For example, if we know that j has ten in-links from the host ($l_j = 10$), and we only see six in-links from pages in $\mathcal{P}$ and in J to j in $\mathcal{C}$, we'll connect four in-links from host J to page j.

There is one caveat at this last step; we always insist that there exists a weight-1 link from host J to page j. This condition ensures that any "long-distance" link from a page i not in $\mathcal{P}$ is represented in the path I, J, j, where host I contains page i.

Finally, we note that, when adding links to graph $\mathcal{C}$, we explicitly adjust the weight on the links between hosts as we connect j to the graph in the manner specified. For example, if page j links to page k in host K, then when we add the link from j to k, and we subtract one from the weight of the link from host J to host K. The only tricky adjustment is potentially between pages in $\mathcal{P}$ and on host J, which may already have links to page j represented.

For our experiments, we assigned the new rank of the page by setting $x(j) = x(J)/|J \cap \mathcal{F}|$ (we slightly abuse notation here and consider J as both a set and a node in this equation). That is, we gave page j the average rank for all the frontier nodes on host J.

We summarize this discussion with a concise procedure for separating a page. The procedure is intricate, so we provide a visual example for each of its steps along with a matching textual description. See Figure 1 for the visual example.

1. Subtract one from $d_H(J)$ if page j is dangling.

2. Adjust the weight of links from the current host J to other hosts (except host J) to account for all the out-links from page j. This step is a host-to-host ($\mathcal{B}$ to $\mathcal{B}$) adjustment. See Figure 1(c).

3. Subtract weight on the links from the host J to pages in the set $\mathcal{P}$ and in the out-links of j. This step is a host-to-page ($\mathcal{B}$ to $\mathcal{P}$) adjustment. Here,

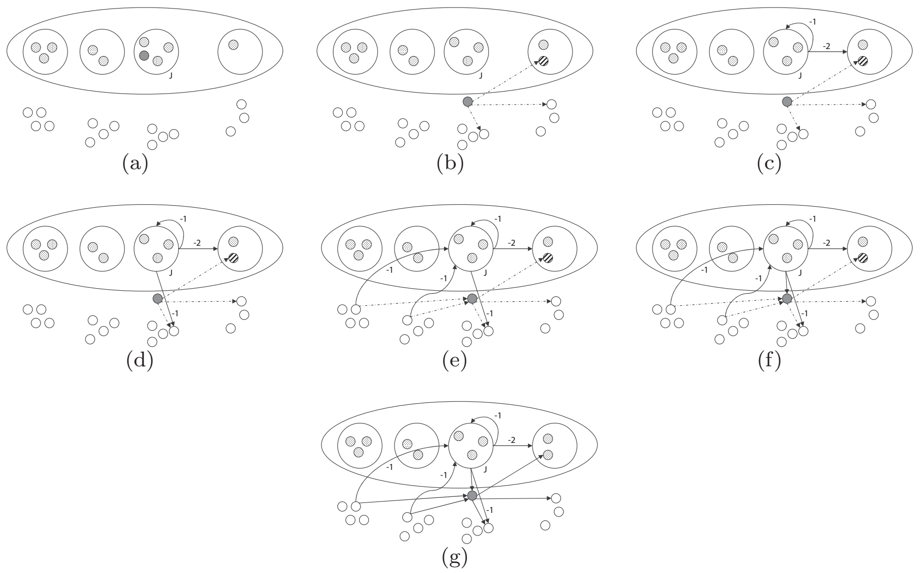


Figure 1. A visual example of the procedure for separating a page. (a) The combination web graph before running the separate page procedure. In this example, we separate the dark-shaded page. (b) This figure depicts the first step of separating the page by finding its out-links. As a visual cue, we represent these out-links with a dashed line to show that they exist but are not a part of the graph yet. (c) This picture describes the modification from Step 2 of the procedure. Because page j links to one page within host J and two pages in another host, we remove the weight on those links. (d) We know that page j linked to another page within host J, so we remove some of the weight on the link from host J to that page in accordance with Step 3 of the procedure. (e) The new dashed links represent the in-links to page j from other pages in the set $\mathcal{P}$. In this step, the algorithm adjusts the weight on the links from those pages to the host J. This figure corresponds to Step 4. (f) Finally, the algorithm adds the link from host J to page j following Step 5. (g) In Steps 6 and 7 the procedure adds all the in-links and out-links between pages in $\mathcal{P}$ and page j as represented by the dashed links turning to solid links. Finally, the algorithm adds the striped page to the frontier set.

we enforce the presence of a minimum of one link between host J and page j. See Figure 1(d).

4. For any in-links to j from pages in $\mathcal{P}$, adjust the weight on links from pages in $\mathcal{P}$ to host J. Here, we have a page-to-host ($\mathcal{P}$ to $\mathcal{B}$) adjustment. See Figure 1(e).

5. Do our best to estimate the number of in-links from host J to page j. Here, we know two things: (i) the number of total in-links from pages in host J to j (l_j), and (ii) the current number of in-links from pages in host J to page j; the difference indicates the number of in-links. However, we enforce that there is always one in-link to j from J. Add this link to graph $\mathcal{C}$. See Figure 1(f).

6. Add links from j to any pages in $\mathcal{P}$.

7. Add links from any pages in $\mathcal{P}$ to j.

8. For any out-links from j to pages not in $\mathcal{P}$, add those pages to the frontier set $\mathcal{F}$. Also, aggregate this set of links at the host level, and add links from j to the hosts in $\mathcal{B}$. See Figure 1(g).

9. Distribute some of the rank from the host vertex J to the newly created vertex j.

One property of this page-separating procedure is that, when we crawl all pages, we recover the original web graph adjacency matrix W in a permuted order. There is an extra set of host vertices included in the graph, but these vertices have no remaining in-links and therefore play no role in the computation of personalized PageRank. Put another way, if we separate all pages and run the algorithm (given fully in Algorithm 4), we recover the personalized PageRank scores exactly.

Once we have the algorithm to separate a page, the PageHostRank algorithm itself is fairly simple. We run iterations of PageRank on the adjacency matrix C of graph $\mathcal{C}$. At each iteration, we examine the rank on hosts in the set $\mathcal{B}$. For

Algorithm 4. (PageHostRank algorithm.)

```
  $C = H$, $\mathcal{S} = \{p\}$, $x(R(p)) = 1$, $\mathcal{F} = \{p\}$, $\mathcal{P} = \{\}$.
  repeat
    For all pages $s \in \mathcal{S}$, separate $s$ and update $C$, $x$, $d_H$, $\mathcal{F}$, and $\mathcal{P}$.
    $y = \alpha(C^T + e_p d_H^T) D^{-1}_{C+d_H e_p^T} x$
    $\omega = 1 - ||y||_1$
    $y(p) = y(p) + \omega$
    $\delta = ||y - x||_1$, $x = y$
    $\mathcal{S} = \{v \mid v \in \mathcal{F}, x(R(v)) > \varepsilon\}$
  until $\delta$ is less than stopping tolerance.
  $x(\mathcal{F}) = 0$.
```

any host I with rank that exceeds ε, we separate all pages in the set $\mathcal{F}$ and in host I and distribute the PageRank from this host to the separated pages.

In Algorithm 4, $R(i)$ maps from page i to host I, and $\mathcal{S}$ is the set of pages to separate.

2.4. Boundary PageHostRank Algorithm

Again, we modify the PageHostRank algorithm to restrict the total rank on the frontier set to κ. See Section 2.2 for more details.

Algorithm 5. (Boundary PageHostRank algorithm.)

```
$C = H$, $\mathcal{S} = \{p\}$, $x(R(p)) = 1$, $\mathcal{F} = \{p\}$, $\mathcal{P} = \{\}$.
**repeat**
    For all pages $s \in \mathcal{S}$, separate $s$ and update $C$, $x$, $\mathcal{F}$, $d_H$, and $\mathcal{P}$.
    $y = \alpha(C^T + e_p d_H^T) D^{-1}_{C + d_H e_p^T} x$
    $\omega = 1 - ||y||_1$
    $y(p) = y(p) + \omega$
    $\delta = ||y - x||_1$, $x = y$
    Sort the ranks on each host in decreasing order.
    Examine the hosts in the sorted order and add any host $J$ to an *expand
    host* set until the total rank on all remaining hosts is less than $\kappa$.
    Add any page in the set $\mathcal{F}$ and in a host in the *expand host* set to the
    separation set $\mathcal{S}$.
**until** $\delta$ is less than stopping tolerance.
$x(\mathcal{F}) = 0$.
```

Instead of directly adding pages from the frontier set $\mathcal{F}$ as in the boundary-restricted personalized PageRank algorithm (Section 2.2), we apply the same idea, but to hosts instead instead of pages. The selection yields a set of hosts that reduces the rank on the frontier to less than κ. We then expand any page in the frontier set on those hosts we selected.

2.5. Termination of Algorithms

In this section, we show that the algorithms terminate. In fact, this result is straightforward. In the matrix formulation of restricted personalized PageRank, we compute the exact personalized PageRank for the active page graph L. The largest L can be is the original adjacency matrix W, and we never remove any page once it is added to L; thus, Algorithm 2 terminates because the power method applied to the PageRank system always terminates. This argument also applies to the PageHostRank algorithm, which computes the exact PageRank on the combination graph $\mathcal{C}$. Although the graph may grow or change during the computation, at some point the graph must stop changing, and the so the PageRank algorithm will converge.

3. Approximation Bounds

In this section, we provide a set of theorems demonstrating that our approximate algorithms for PageRank computing are bounded approximations to the personalized PageRank vectors.

First, we prove an approximation bound for the simple restricted personalized PageRank algorithm (Algorithm 2). Then, we show the relationship between HostRank and PageRank. Finally, we expand the relationship between HostRank and PageRank to model the PageHostRank algorithm (Algorithm 4), and we prove an approximation bound for that algorithm.

3.1. Basic Approximation for Restricted PageRank

The lemma that we prove in this section is the basis of our approximation bounds; it states that we can express the difference between a modified and an exact solution in terms of the modified solution. Before we begin, let's establish some notation.

In contrast with Section 2, let

$$A^T = W^T D_W^{-1} + v d_W^T.$$

The matrix A^T is column-stochastic, but without the rank-1 modification in the PageRank Markov chain. In the theorems presented here, we will also be dealing with modified web graphs that will be denoted by $\widetilde{\mathcal{W}}$ and corresponding adjacency matrices denoted by $\widetilde{W}$. In the same manner,

$$\tilde{A}^T = \widetilde{W}^T D_{\widetilde{W}}^{-1} + v d_{\widetilde{W}}^T.$$

In general, the notation of the proofs is self-contained.

Lemma 3.1. *Let x^* be the exact PageRank vector for web graph $\mathcal{W}$ and personalization vector v:*

$$x^* = \alpha A^T x^* + (1-\alpha)v.$$

Also, let $\tilde{x}$ be the PageRank vector for a modification of web graph $\mathcal{W}$ to $\widetilde{\mathcal{W}}$ with the same nodes as W and the same personalization vector v:

$$\tilde{x} = \alpha \tilde{A}^T \tilde{x} + (1-\alpha)v.$$

Then,

$$r = \tilde{x} - x^* = \frac{\alpha}{1-\alpha} S^T \Delta^T \tilde{x},$$

where

$$\Delta^T = \tilde{A}^T - A^T$$

and S^T is composed of a set of column vectors that are rescaled solutions to a PageRank linear system,

$$\begin{aligned} S^T &= \begin{pmatrix} s_1 & s_2 & \dots & s_n \end{pmatrix}, \\ \left[I - \alpha A^T\right] s_i &= (1-\alpha) e_i, \\ S^T &= (1-\alpha)\left[I - \alpha A^T\right]^{-1}. \end{aligned}$$

Proof. First, note that the inverse in the definition of S^T is well defined because $I - \alpha A^T$ is strictly diagonally dominant for $\alpha < 1$. This proof proceeds by applying all the definitions in the statement of the lemma and some simple algebra. First, we rewrite the solution vectors x^* and $\tilde{x}$ as solutions to linear systems:

$$\left[I - \alpha A^T\right] x^* = (1-\alpha)v,$$

and

$$\left[I - \alpha \tilde{A}^T\right] \tilde{x} = (1-\alpha)v.$$

Recall that $r = \tilde{x} - x^*$ by definition, so $\tilde{x} = r + x^*$. Then,

$$\begin{aligned} \left[I - \alpha \tilde{A}^T\right] \tilde{x} &= (1-\alpha)v, \\ \left[I - \alpha \tilde{A}^T\right] (r + x^*) &= (1-\alpha)v, \\ \left[I - \alpha \tilde{A}^T - \alpha A^T + \alpha A^T\right] (r + x^*) &= (1-\alpha)v, \\ \left[I - \alpha A^T - \alpha\left(\tilde{A}^T - A^T\right)\right] (r + x^*) &= (1-\alpha)v, \\ \left[I - \alpha A^T - \alpha \Delta^T\right] (r + x^*) &= (1-\alpha)v, \\ \left[I - \alpha A^T\right] r + \left[I - \alpha A^T\right] x^* - \alpha \Delta^T (r + x^*) &= (1-\alpha)v, \\ \left[I - \alpha A^T\right] r + \left[I - \alpha A^T\right] x^* - \alpha \Delta^T \tilde{x} &= (1-\alpha)v. \end{aligned}$$

After we subtract the linear system formulation for the exact vector, we get

$$\left[I - \alpha A^T\right] r - \alpha \Delta^T \tilde{x} = 0.$$

Rearranging this last expression yields

$$\begin{aligned} \left[I - \alpha A^T\right] r &= \alpha \Delta^T \tilde{x}, \\ \left[I - \alpha A^T\right] r &= (1-\alpha) \frac{\alpha}{1-\alpha} \partial^T \tilde{x}. \end{aligned}$$

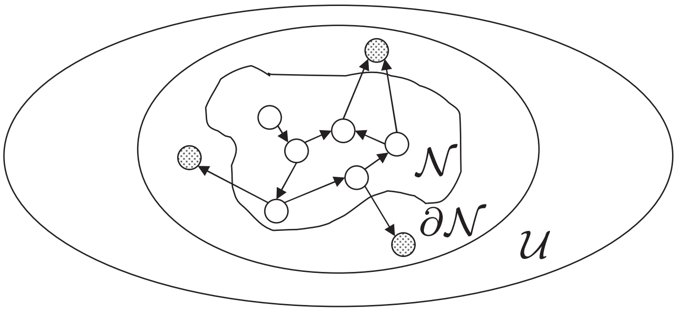


Figure 2. The hierarchy of sets we use to compute our bounds.

At this point, we have the change in PageRank vectors expressed as the solution of an unusual PageRank linear system. Multiplying by $\left[I - \alpha A^T\right]^{-1}$ and substituting the definition of S^T immediately yields the result. $\square$

This result was also shown in [Bianchini et al. 05].

In our approximation algorithms, we bound the maximum rank on all the unexpanded nodes. Let $\Gamma(\mathcal{N})$ denote the set of all adjacent vertices to vertices in $\mathcal{N}$. To be concise, call the set $\partial\mathcal{N} = \Gamma(\mathcal{N}) - \mathcal{N}$ and $\mathcal{U} = \mathcal{V} - \Gamma(\mathcal{N}) - \mathcal{N}$. (Mnemonically, $\mathcal{N}$ is the expanded neighborhood of the graph, $\partial\mathcal{N}$ is the boundary, and $\mathcal{U}$ is the unknown region.) We illustrate these sets in Figure 2.

In the next two theorems, we bound the results from the restricted personalized PageRank algorithms (Algorithms 2 and 3) in two steps. First, we show a theoretical bound for the PageRank system modeled in the restricted PageRank algorithms. Second, we bound the result achieved by each algorithm in light of the termination criteria for approximately solving the local PageRank problem.

Theorem 3.2. *Suppose that $\tilde{x}$ is the PageRank vector for the local graph computed in either the restricted or boundary-restricted personalized PageRank algorithm extended to a length-$|\mathcal{V}|$ vector with rank 0 on all nodes not in the local graph. Let $r = \tilde{x} - x^*$ and*

$$\kappa = \sum_{v \in \partial\mathcal{N}} \tilde{x}(v),$$

where $\mathcal{N}$ is the set from the restricted personalized PageRank algorithm. Then,

$$||r||_1 = \frac{2\alpha}{1-\alpha}\kappa.$$

Proof. We begin by using Lemma 3.1. In restricted personalized PageRank, the approximate vector $\tilde{x}$ comes from the solution of PageRank on a web graph $\widetilde{\mathcal{W}}$ with no links from the boundary set $\partial\mathcal{N}$. Hence, the lemma applies, and we can write

$$r = \frac{\alpha}{1-\alpha} S^T \Delta^T \tilde{x}.$$

Taking the norm, we have

$$\begin{aligned} ||r||_1 &= \frac{\alpha}{1-\alpha}||S^T \Delta^T \tilde{x}||_1 \\ &\leq \frac{\alpha}{1-\alpha}||S^T||_1 ||\Delta^T \tilde{x}||_1. \end{aligned}$$

To further specify and bound factors from this expression, first observe that $||S^T||_1 = 1$. This fact follows immediately from the definition of the columns of S^T (which are solutions of PageRank-like systems).

Moreover, if we permute Δ such that the ordering of vertices is $\mathcal{N}, \partial\mathcal{N}, \mathcal{U}$, then we have the following:

$$||\Delta^T \tilde{x}||_1 = \left\| \begin{pmatrix} 0 & \uparrow & \uparrow \\ 0 & \Delta^T_{\partial\mathcal{N}} & \Delta^T_{\mathcal{U}} \\ 0 & \downarrow & \downarrow \end{pmatrix} \begin{pmatrix} \tilde{x}_{\mathcal{N}} \\ \tilde{x}_{\partial\mathcal{N}} \\ 0 \end{pmatrix} \right\|_1 = ||\Delta^T_{\partial\mathcal{N}} \tilde{x}_{\partial\mathcal{N}}||_1 \leq \kappa ||\Delta^T_{\partial\mathcal{N}}||_1.$$

To understand this statement, let's study the structure of Δ^T. In the expanded neighborhood, we know all the out-links exactly. Because the personalization support is entirely within the set $\mathcal{N}$, the matrix Δ has only zeros in all rows corresponding to pages in $\mathcal{N}$ (respectively, in all columns of Δ^T). The notation $\Delta^T_{\partial\mathcal{N}}$ just refers to the subregion of the matrix associated with the out-links of pages in $\partial\mathcal{N}$ (similarly with $\Delta^T_{\mathcal{U}}$). In fact, we don't have to address $\Delta^T_{\mathcal{U}}$ at all because it drops out of the equation due to the zeroes in the $\tilde{x}$ vector.

We can further bound $||\Delta^T_{\partial\mathcal{N}}||_1$ by observing that the sum of each column of A^T and $\tilde{A}^T$ is 1, so the maximum of the difference of any column is 2. This difference is achieved by any page in $\partial\mathcal{N}$ with at least one out-link. Thus,

$$||\Delta^T_{\partial\mathcal{N}}||_1 \leq 2.$$

By combining these results, the theorem follows. □

In the next theorem, we adjust the previous result to take into consideration the fact that we do not solve for the exact PageRank vector of the local graph.

Theorem 3.3. *Let $\hat{x}$ be the vector computed by a restricted personalized PageRank algorithm with δ as the stopping tolerance. Let $r = \hat{x} - x^*$ and*

$$\kappa = \sum_{v \in \partial \mathcal{N}} \hat{x}(v),$$

where $\mathcal{N}$ is the set from the restricted personalized PageRank algorithm. Then,

$$||r||_1 \leq \frac{2\alpha}{1-\alpha}\kappa + \frac{1+\alpha}{(1-\alpha)^2}\delta.$$

Proof. First, the vector $\hat{x}$ is an approximate solution of the linear system

$$(I - \alpha \tilde{A}^T)\tilde{x} = (1-\alpha)v.$$

Note that $\tilde{x}$ and x^* are the same as in the previous proof. If we write

$$\hat{x} + \bar{x} = \tilde{x},$$

then $\bar{x}$ is the error in the approximate solution $\hat{x}$. Thus,

$$r = \hat{x} - x^* = \tilde{x} - x^* - \bar{x},$$

and by Theorem 3.2

$$||r||_1 \leq \frac{2\alpha}{1-\alpha}||\tilde{x}_{\partial\mathcal{N}}||_1 + ||\bar{x}||_1.$$

From

$$(I - \alpha\tilde{A}^T)(\hat{x} + \bar{x}) = (1-\alpha)v,$$

we find that

$$\begin{aligned}(I - \alpha\tilde{A}^T)\bar{x} &= (1-\alpha)v - (I - \alpha\tilde{A}^T)\hat{x} \\ &= \alpha\tilde{A}^T\hat{x} + (1-\alpha)v - \hat{x}.\end{aligned}$$

The right-hand side of this expression is the quantity $y - x$ from the algorithm. In fact, this expression is the residual $r_{\text{LS}} = b - Ax$ of the linear system,[1] so let $r_{\text{LS}} = y - x$ with y and x from a restricted personalized PageRank algorithm. Because we terminate when $||r_{\text{LS}}||_1 \leq \delta$, we immediately bound the right-hand side. Further, because

$$||(I - \alpha\tilde{A}^T)^{-1}||_1 = \frac{1}{1-\alpha}$$

(which follows from $||S^T||_1 = 1$ in the previous proofs), we find that

$$\bar{x} = (I - \alpha\tilde{A}^T)^{-1} r_{\text{LS}}$$

[1]Of course, the quantities here are expressed in terms of the standard $Ax = b$ linear system formulation. For our case, $b = (1-\alpha)v$ and $A = (I - \alpha\tilde{A}^T)$.

and

$$||\bar{x}||_1 \le ||(I - \alpha\tilde{A}^T)^{-1}||_1 ||r_{\text{LS}}||_1 \le \frac{1}{1-\alpha}\delta.$$

In the algorithm, we do not have $\kappa = \sum_{v\in\partial\mathcal{N}} \tilde{x}(v)$, but instead

$$\kappa = \sum_{v\in\partial\mathcal{N}} \hat{x}(v) = ||\hat{x}_{\partial\mathcal{N}}||_1.$$

(That is, κ comes from the approximate solution instead of the exact solution.) However, from the definition of the error $\bar{x}$,

$$\tilde{x}_{\partial\mathcal{N}} = \hat{x}_{\partial\mathcal{N}} - \bar{x}_{\partial\mathcal{N}}.$$

Applying norms yields

$$||\tilde{x}_{\partial\mathcal{N}}||_1 \le \kappa + ||\bar{x}_{\partial\mathcal{N}}||_1 \le \kappa + ||\bar{x}||_1 \le \kappa + \frac{1}{1-\alpha}\delta.$$

Substituting this expression into the previous bound on $||r||_1$ yields the theorem. □

Remark 3.4. Based on the previous proof, the restricted personalized PageRank algorithm (Algorithm 2) with expansion tolerance ε and stopping tolerance δ yields an approximate PageRank vector $\tilde{x}$, where

$$||x^* - \hat{x}||_1 \le \frac{2\alpha}{1-\alpha}\varepsilon|\Gamma(\mathcal{N}) - \mathcal{N}| + \frac{1+\alpha}{(1-\alpha)^2}\delta.$$

Remark 3.5. The boundary-restricted personalized PageRank algorithm (Algorithm 3) runs until

$$\sum_{v\in\partial\mathcal{N}} \hat{x}(v) \le \kappa$$

for a fixed κ. Thus, the boundary-restricted personalized PageRank algorithm yields an approximate personalized PageRank vector $\hat{x}$, where

$$||x^* - \hat{x}||_1 \le \frac{2\alpha}{1-\alpha}\kappa + \frac{1+\alpha}{(1-\alpha)^2}\delta.$$

3.2. HostRank and PageRank

In this section, we'll show the relationship between the PageRank vector and the HostRank vector. We first need to formally define HostRank. This section provides one step along the way to showing that the PageHostRank computation approximates the personalized PageRank vector.

Let R be the restriction matrix of the page to host the restriction operator as defined in Section 2.3. Then,

$$H = R^T W R$$

is the weighted adjacency matrix between hosts. Let $\widehat{\mathcal{W}}$ be the dangling-node adjusted web graph with adjacency matrix

$$\widehat{W} = W + d_W e_p^T,$$

where e_p is a vector with a one for each personalization page and zero elsewhere. Recall that d_W is the dangling-page indicator. In this modification, we explicitly add all the links back to the personalization pages from dangling nodes. Correspondingly, let

$$\hat{H} = R^T \widehat{W} R = H + (R^T d_W)(R^T e_p)^T.$$

The matrix $\hat{H}$ includes all the links added due to the dangling-node adjustment. With these matrices, we can define HostRank.

Definition 3.6. (HostRank.) Let B correspond to the random-walk scaling of $\hat{H}$,

$$B = D_{\hat{H}}^{-1} \hat{H}.$$

In this case, the scaling matrix $D_{\hat{H}}^{-1}$ has

$$(D_{\hat{H}}^{-1})_{ii} = \begin{cases} 0 & \sum_j \hat{H}_{ij} = 0 \\ \frac{1}{\sum_j \hat{H}_{ij}} & \text{otherwise.} \end{cases}$$

The HostRank vector h is the solution of the following equation:

$$h = \alpha B^T h + (1-\alpha) R^T v,$$

where v is a uniform distribution over all the personalization pages (and corresponds to the pages used in $\widehat{\mathcal{W}}$).

In the following discussion, we do our best to disambiguate between two uses of the symbol I. We use I as both a label of a host and as an index into a vector. In contrast, we use $\mathcal{I}$ to refer to the set of pages on host I. That is, h_I refers to the HostRank of host I, whereas $i \in \mathcal{I}$ refers to all pages agglomerated together in host I.

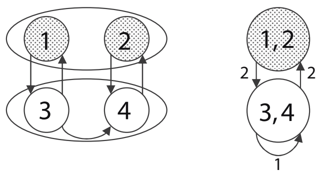


Figure 3. A counterexample for the simple relationship between HostRank and PageRank. If we personalize on the shaded pages, then we get a PageRank vector of $x^* = (0.1174 \quad 0.4002 \quad 0.0998 \quad 0.3826)^T$. The corresponding HostRank vector is $h = (0.4574 \quad 0.5426)^T$. Clearly, the statement is false.

Ideally, we would like to prove the following statement:

$$h_I = \sum_{i \in \mathcal{I}} x_i^*,$$

where x^* is the PageRank vector for web graph $\mathcal{W}$ with the same personalization. We call this the *summation property*.

Unfortunately, this statement is not generally true. See Figure 3 for a counterexample. Horton showed that a solution dependent restriction operator does give the summation property [Horton 97]. Nevertheless, this restriction operator depends upon the exact solution for PageRank—or a good approximation of it.

Instead, we show that if we compute PageRank on a deliberately modified web graph, then we keep the summation property.

We name our modification *silent side-step* PageRank. Silent side-step PageRank is a modeling artifact to explain *what* occurs when computing the PageRank of H instead of W. Eventually, this artifact will help relate the PageHostRank vector computed to the PageRank vector. In the silent side-step model, we change the behavior of the random surfer so

- with probability $(1 - \alpha)$, the surfer randomly jumps to any personalized page;
- with probability α, the surfer randomly jumps to any other page i' on the same host and randomly follows an out-link from that page.

In the second case, the surfer only makes one transition move in the Markov chain—that is, the surfer does not "stop" on page i'. The name comes from this silent within-host step.

Now, let U be the transition matrix for the silent side-step random surfer:

$$U_{i,j} = \frac{\sum_{i' \in \mathcal{I}} \hat{W}_{i',j}}{\sum_{i' \in \mathcal{I}, j'} \hat{W}_{i',j'}}$$

or

$$U = D^{-1}_{RR^T\hat{W}} RR^T\hat{W}.$$

In this equation, the matrix $D^{-1}_{RR^T\hat{W}}$ normalizes U to be a random-walk transition matrix. For each page $i \in \mathcal{I}$, the corresponding row of the transition matrix is identical. Likewise, for each page $i \in \mathcal{I}$, the normalizing factor in $D_{RR^T\hat{W}}$ is equal.

As we foreshadowed in Section 2, the random-surfer interpretation of silent side-step PageRank is not realistic. It is unlikely that a surfer would jump uniformly at random within the host and then take another step. A more realistic model would weight the intra-host connections, such as in [Kamvar et al. 03a]. However, the silent side-step model should be regarded as a modeling artifact along the way to showing that the PageHostRank results approximate the personalized PageRank results.

Theorem 3.7. *HostRank and silent side-step PageRank have the summation property.*

Proof. The silent side-step PageRank vector u satisfies the following equation:

$$u = \alpha U^T u + (1-\alpha)v.$$

If we apply the restriction operator, we have

$$\begin{aligned} R^T u &= \alpha R^T U^T u + (1-\alpha)R^T v \\ &= \alpha R^T \hat{W}^T RR^T D^{-1}_{RR^T\hat{W}} u + (1-\alpha)R^T v. \end{aligned}$$

First, recall that $R^T\hat{W}^T R = H^T$. Now, $R^T D^{-1}_{RR^T\hat{W}} u = \hat{D}^{-1}_{R^T\hat{W}R} R^T u$. To justify this fact, observe that, for each page of one host, the normalization constant is the same. Further, that normalization constant is the same as the corresponding entry in $\hat{D}^{-1}_{R^T\hat{W}R}$, i.e., the normalization based on the total (weighted) count of all out-links from a host. Therefore, because all the constants are the same, we can agglomerate first and divide second. Formally,

$$(R^T D^{-1}_{RR^T\hat{W}} u)_i = \sum_{i\in\mathcal{I}} \frac{u_i}{\sum_{i'\in\mathcal{I}} \hat{W}_{i',j}} = \frac{1}{\sum_{i'\in\mathcal{I}} \hat{W}_{i',j}} \sum_{i\in\mathcal{I}} u_i = (\hat{D}^{-1}_{R^T\hat{W}R} R^T u)_i.$$

At this point, we are effectively done. To complete the proof, we have

$$\begin{aligned} R^T u &= \alpha H^T \hat{D}^{-1}_{R^T\hat{W}R} R^T u + (1-\alpha)R^T v \\ &= \alpha B^T R^T u + (1-\alpha)R^T v. \end{aligned}$$

Because there is a unique stationary distribution for the HostRank Markov chain, we know that $R^T u = h$. $\square$

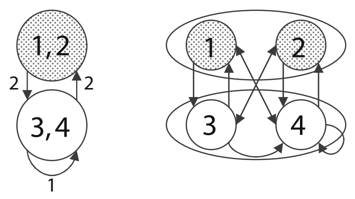


Figure 4. In this figure, we show the agglomerated HostRank graph and the silent side-step PageRank graph. In the silent side-step PageRank graph, each node collects all the out-links from each vertex agglomerated together. Recall that $h = (0.4574 \quad 0.5426)^T$. Now we have that $u = (0.2287 \quad 0.2287 \quad 0.1944 \quad 0.3481)^T$ and $R^T u = (0.4574 \quad 0.5426)^T$.

See Figure 4 for an example that shows that the summation property does hold.

3.3. PageHostRank

In this section, we'll provide bounds on the PageRank approximation achieved by the PageHostRank algorithms. First, we'll show that a theorem similar to the result relating silent side-step PageRank and HostRank holds for PageHostRank. That is, there is a more complicated modification of the Markov chain that keeps the summation property. Next, we'll use Lemma 3.1 to relate the modified graph back to the original web graph, which will provide the approximation bound for the PageHostRank algorithm (Algorithm 4).

First, we describe the adjacency matrix C for the combination graph $\mathcal{C}$ in Algorithm 4. To do this, we partition the nodes for each host into two sets: $\mathcal{I}_a$ is the set of agglomerated nodes and $\mathcal{I}_e$ is the set of expanded nodes. In this section, we use the index I_a to denote the index for the agglomerated node corresponding to the set $\mathcal{I}_a$. Thus, for each host I, we have

$$\mathcal{I} = \mathcal{I}_a \cup \mathcal{I}_e, \mathcal{I}_a \cap \mathcal{I}_e = \emptyset,$$

where $\mathcal{I}$ is the set of all pages on host I.

For a pair of hosts I and J, we describe the connectivity in C. In the following, $i \in \mathcal{I}_e$ and $j \in \mathcal{J}_e$ unless otherwise indicated:

$$C_{I_a,J_a} = \begin{cases} \sum_{i\in\mathcal{I},j\in\mathcal{J}} W_{i,j} - \sum_{i\in\mathcal{I}_e,j\in\mathcal{J}_a\cup\mathcal{F}} W_{i,j} & I \neq J, \\ \max\left(\sum_{i\in\mathcal{I}_a,j\in\mathcal{I}_a} W_{i,j} - \left|\{k| \sum_{i'\in\mathcal{I}_a} W_{i',k} = 0\}\right|, 0\right) & I = J; \end{cases}$$

$$\begin{aligned} C_{I_a,j} &= \max\bigl(\sum_{i\in\mathcal{I}_a} W_{i,j}, 1\bigr) \text{ for } j \in \mathcal{I} \text{ or } 0 \text{ if } j \notin \mathcal{I}; \\ C_{i,J_a} &= \sum_{j\in\mathcal{J}_a\cup\mathcal{F}} W_{i,j}; \\ C_{i,j} &= W_{i,j} \text{ if } j \in \mathcal{J}_e. \end{aligned}$$

Starting from the top, the weight on the connection between two hosts I and J is the total weight from all links between I and J without all links from the nodes expanded from host I. The weight between a host node I_a and itself is the weight between all agglomerated nodes in $\mathcal{I}_a$ without the nodes with artificial in-links. The max function enforces that the weight does not go negative. The weight between a host node and an expanded node (on that host) is at least one or the weight from all agglomerated nodes. There are no links between a host node and expanded nodes on other hosts. The weight from a page to other hosts nodes is simply the aggregation of all the links to those pages. Finally, the weight between expanded pages is 1, if the pages link together.

We next describe an expanded web graph that has a summation property with the combination graph $\mathcal{C}$. Let $\tilde{\mathcal{C}}$ be a web graph with $|\mathcal{V}| + |\mathcal{B}|$ nodes (recall that each host is an entry in $\mathcal{B}$). Each extra node corresponds to a special in-link aggregation node for links from unexpanded pages to expanded pages on separate hosts. The web graph $\tilde{\mathcal{C}}$ is the original web graph $\mathcal{W}$ with two modifications. First, let $\tilde{I}$ be the node added for host I. The node $\tilde{I}$ steals in-links from any node in $\mathcal{J}_a$ to a node in $\mathcal{I}_e$. That is, for any node $j \in \mathcal{J}_a$, instead of linking to $i \in \mathcal{I}_e$ as in the original web graph, the node $\tilde{I}$ steals this in-link so that j links to $\tilde{I}$ in $\tilde{C}$ (and not to i). Next, the node $\tilde{I}$ links to any page with a "virtual" in-link constructed when we invoked the rule that each expanded node on a host must have one in-link from the agglomerated host node. That is,

$$\tilde{C}_{\tilde{I},i} = 1 \text{ if } \sum_{i'\in\mathcal{I}_a} W_{i',i} = 0.$$

The second modification of $\tilde{\mathcal{C}}$ with respect to W is that all nodes in a set $\mathcal{I}_a \cup \tilde{I}$ copy out-links. That is, the row of the adjacency matrix $\tilde{C}$ for any node in $\mathcal{I}_a \cup \tilde{I}$ is identical and is equal to the sum of all rows in $\mathcal{I}_a \cup \tilde{I}$ in the original adjacency matrix W modified with the extra host node as above,[2] i.e.,

$$\tilde{C}_{i\in\mathcal{I}_a\cup\tilde{I},:} = \sum_{i\in\mathcal{I}_a\cup\tilde{I}} W_{i,:}.$$

[2]Technically, the correct way of describing this modification is by first introducing the extra host nodes and then copying out-links in a modification of the modified graph, but we do both steps at once to slightly ease the notation.

This leads to the next theorem.

Theorem 3.8. *If x is a PageHostRank vector, then there exists a web graph $\widetilde{\mathcal{W}}$ with $|\mathcal{V}| + |\mathcal{B}|$ vertices, such that the personalized PageRank vector on $\widetilde{W}$, $\tilde{x}$, satisfies*

$$\begin{aligned} x_I &= \tilde{x}_{\tilde{I}} + \sum_{i \in \mathcal{I} \cap (\mathcal{V} - \mathcal{P})} \tilde{x}_i && \text{for all } I \in \mathcal{B}, \\ x_i &= \tilde{x}_i && \text{for all } i \in \mathcal{P}, \end{aligned}$$

where $\mathcal{P}$ is from the PageHostRank algorithm.

Proof. The modified web graph $\widetilde{\mathcal{W}}$ equals $\tilde{\mathcal{C}}$ as defined above. The summation property follows because we defined $\tilde{\mathcal{C}}$ such that all nodes within a host are lumpable [Kemeny and Snell 83]. Thus, when we aggregate them together, the corresponding stationary probability vectors have the summation property. □

We could have used the lumpable property in the proof for the HostRank algorithm. However, we believe that presenting that result with an explanation such as the silent side-side step move gives more intuition about what occurs.

Using Theorem 3.8, we can conclude that PageHostRank gives us an approximation to personalized PageRank.

Theorem 3.9. *Suppose that $\tilde{x}$ is the solution from the full PageHostRank model on $|\mathcal{V}| + |\mathcal{B}|$ vertices, x is the PageRank vector for the PageHostRank model on the small graph $\mathcal{C}$, and x^* is the exact PageRank vector extended with $|\mathcal{B}|$ extra nodes with rank 0. Let $r = \tilde{x} - x^*$ and*

$$\kappa = \sum_{v \notin \mathcal{P}} x(v),$$

where $\mathcal{P}$ is the expanded page set from the PageHostRank algorithm. Then,

$$||r||_1 \le \frac{2\alpha}{1-\alpha}\kappa.$$

Proof. By the summation property from Theorem 3.8, we know that

$$\kappa = \sum_{v \notin \mathcal{P}} \tilde{x}(v)$$

as well. The remainder of the proof is virtually identical to the analogous bound for restricted PageRank. We begin with

$$r = \frac{\alpha}{1-\alpha} S^T \Delta^T \tilde{x},$$

where $\Delta^T = \tilde{A}^T - A^T$ for $\tilde{A}$ corresponding to $\tilde{C}$. This lemma still applies because we can model x^* as the exact solution of personalized PageRank on a web graph with $|\mathcal{B}|$ fictitious, unlinked nodes.

Getting the final bound for this problem is slightly easier than for restricted PageRank. We find that again, for all pages in $\mathcal{P}$, all the out-links are represented exactly. So,

$$||\Delta^T \tilde{x}||_1 = \left\| \begin{pmatrix} 0 & \Delta^T_{\mathcal{V}\cup\mathcal{B}-\mathcal{P}} \end{pmatrix} \begin{pmatrix} \tilde{x}_{\mathcal{P}} \\ \tilde{x}_{\mathcal{V}\cup\mathcal{B}-\mathcal{P}} \end{pmatrix} \right\|_1 \leq 2||\tilde{x}_{\mathcal{V}\cup\mathcal{B}-\mathcal{P}}||_1 = 2\kappa.$$

Hence,

$$||r||_1 \leq \frac{\alpha}{1-\alpha}||S^T||_1||\Delta^T\tilde{x}||_1 \leq \frac{2\alpha}{1-\alpha}\kappa. \qquad \square$$

In our algorithm implementation, we set $x(\neg\mathcal{P}) = 0$ (i.e., everything not in $\mathcal{P}$) as the final step because we want the rank on the separated set of pages to come from a PageRank vector. (The fictitious "hosts" modeled in the algorithm are not actual pages.) Technically, this means our bound only applies the total change in the PageHostRank vector on the pages $\mathcal{P}$ in relationship with the exact PageRank on $\mathcal{P}$, i.e.,

$$||x_{\mathcal{P}} - \tilde{x}_{\mathcal{P}}||_1 \leq ||x - \tilde{x}||_1.$$

In the next theorem, we relate the vector computed from the algorithm, including the error due to the stopping criteria.

Theorem 3.10. *Let $\hat{x}$ be the vector computed in PageHostRank algorithm with stopping tolerance δ, then*

$$||\hat{x}_{\mathcal{P}} - x^*_{\mathcal{P}}||_1 \leq \frac{2\alpha}{1-\alpha}\kappa + \frac{1+\alpha}{(1-\alpha)^2}\delta.$$

Proof. The proof is identical to the analogous result for restricted PageRank, Theorem 3.3, after using the result Theorem 3.9. $\square$

4. Experimentation on Web Graphs

We used our approximate PageRank algorithms on a few publicly available data sets. The corresponding web graphs had variable size from 10,000 to 118 million vertices. We report on the experiments with the largest data set, the WebBase graph of approximately 118 million vertices. Sources of our data sets were [Kamvar 03] and [Boldi and Vigna 04, Vigna 06].

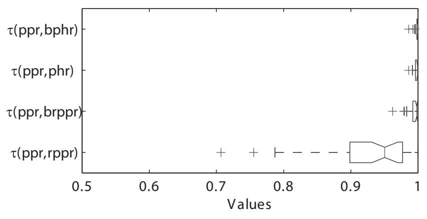
We performed the experiments on an Intel$^{\text{TM}}$ Itanium$^{\text{TM}}$ 2 system, with 32 GB of shared RAM. Because of the large size of the memory, it was possible to keep the adjacency matrix of the web graph, in sparse format, in memory and to work efficiently with it. We used the values $\kappa = 0.001$, $\varepsilon = 0.0001$, and $\alpha = 0.85$ for the approximation algorithms.

Because of the additivity property of personalized PageRank with respect to the personalization vector, we investigated the approximation accuracy of PPR vectors where the personalization vector is supported on a single page. We experimented with 40 different vectors.

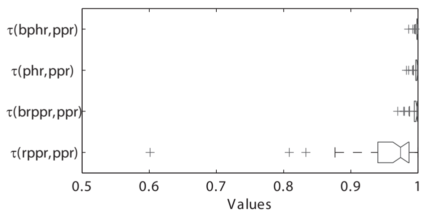
For each one of these experiments corresponding to a single personalization page, we computed the exact PPR value on the whole web graph, then computed restricted personalized PageRank (RPPR, Section 2.1), boundary-restricted personalized PageRank (BRPPR, Section 2.2), PageHostRank (PHR, Section 2.3), and boundary PageHostRank (BPHR, Section 2.4).

We chose to evaluate the difference between exact and approximate scoring with several metrics. In Figures 5 and 6, we use box-plots to represent compactly the results of all of our 40 experiments. The box-plot represents the values in a sample by using a horizontal box, two dashed lines, and a set of outliers marked as crosses. The lines in the box denote the lower quartile, median, and the upper quartile. The dashed lines extend 1.5 quartiles beyond the lower and upper quartiles to show reasonable values for the rest of the data. Outliers are values outside of the dashed lines.

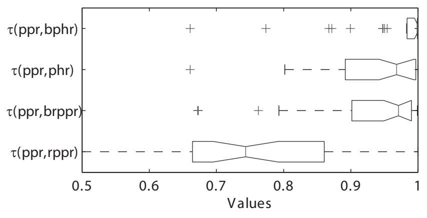
In Figure 5 we show a set of results indicating the correlation in ranking between the exact and approximate personalized PageRank vectors. Each figure displays a box-plot of the results over all 40 experiments, where the box encloses the region between the upper and lower quartiles. The values computed for each experiment are Kendall's τ correlation coefficients. A value of 1 indicates a perfect rank correlation, thus distributions of values near 1 are better results. In the figures, $\mathcal{T}_n$ is the set of n pages with the highest personalized PageRank and $\hat{\mathcal{T}}_n$ is the set of n pages with highest approximate personalized PageRank scores. Likewise, the vector x^* is the exact personalized PageRank vector and $\hat{x}$ is an approximate personalized PageRank vector. We show the distribution of the Kendall τ coefficient for the first 100 and 1000 pages according to each method. The τ coefficient is based on the number of transpositions between different rankings, and we choose to evaluate this measure at two different sets of pages to provide a comparison of the algorithms at different levels of precision.



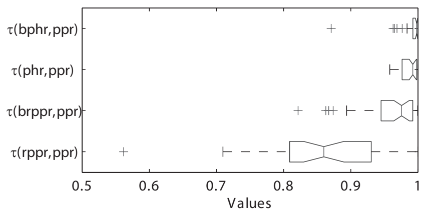
(a)



(b)



(c)



(d)

Figure 5. The ranking changes between the exact personalized PageRank vector and the approximations computed by the four different algorithms. The quantities computed for each plot are (a) $\tau(x^*(\mathcal{T}_{100}), \hat{x}(\mathcal{T}_{100}))$, (b) $\tau(\hat{x}(\hat{\mathcal{T}}_{100}), x^*(\hat{\mathcal{T}}_{100}))$, (c) $\tau(x^*(\mathcal{T}_{1000}), \hat{x}(\mathcal{T}_{1000}))$, and (d) $\tau(\hat{x}(\hat{\mathcal{T}}_{1000}), x^*(\hat{\mathcal{T}}_{1000}))$. In all cases, the approximate algorithm is identified by the label on the left of the plot.

We have omitted the results from the first 10 pages because they were almost always ranked identically for all algorithms.

Figure 6 presents on the 1- and ∞-norm distances between PageRank vectors. We calculate these distances on the whole vector, as well as on the subset of entries corresponding to active and frontier pages. Each box-plot shows the distribution of results over all 40 experiments. A value of 0 indicates that we computed the result exactly. The first two figures, (a) and (b), show the difference in the 1-norm and ∞-norm on the set of pages in the active set, that is, the subset of pages where we know the out-links. The second set of figures shows the difference in each norm on the frontier set, the pages that we know exist but we do not know their out-links. The final set, (e) and (f), show the difference in norm over all pages.

The results in Figure 5 show that the ranking of the top 100 pages for the BRPPR, PHR, and BPHR algorithms are similar. The results from the top 1000 results show that the algorithms are missing some pages with respect to the exact solution. This follows from the more negative results of Figure 5(c) compared with Figure 5(d). The most likely explanation of these results is that there are a few pages with high personalized PageRank known to the exact

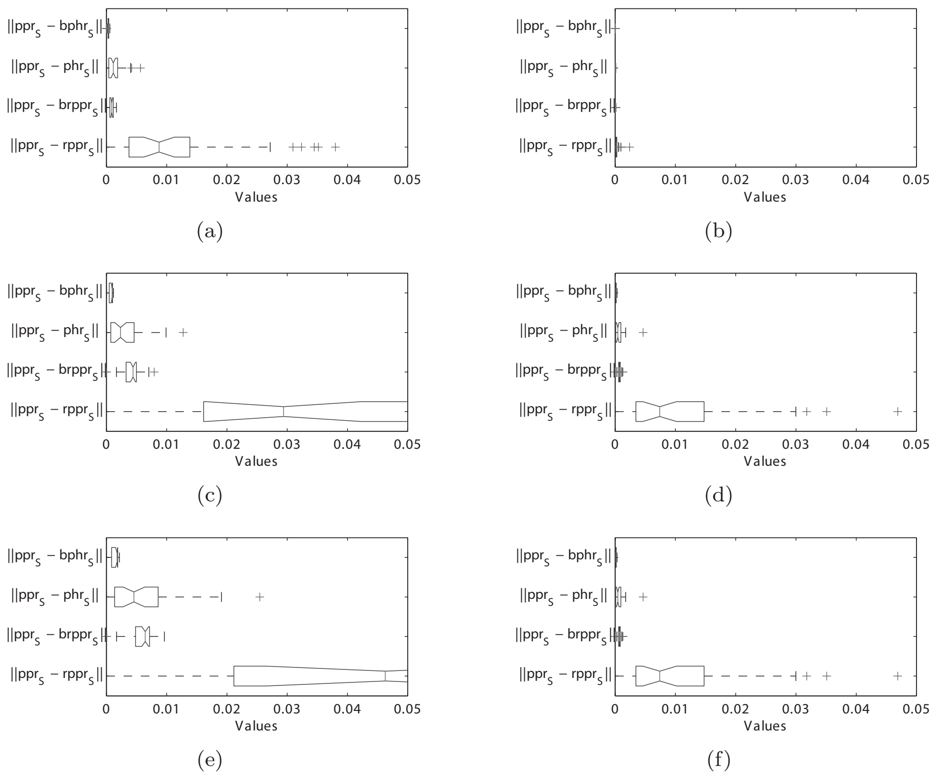


Figure 6. The difference in norm between the exact personalized PageRank (PPR) vector and the approximate vectors. The norms and sets for each figure are (a) 1-norm, S = active set; (b) ∞-norm, S = active set; (c) 1-norm, S = frontier set; (d) ∞-norm, S = frontier set; (e) 1-norm, S = all pages; and (f) ∞-norm, S = all pages.

algorithm ranked between 100–1000 that are unknown to the approximations. The difference in ranking of these few pages causes the τ results to shift to the left away from 1. However, from Figure 5(d), among the top 1000 pages found by each algorithm, the ranking is approximately preserved.

The results from Figure 6 reveal a similar pattern. Again, the BRPPR, PHR, and BPHR algorithms perform the best. On the active set of pages, the difference between the exact and approximate vectors is extremely small (Figures 6(a) and (b)). Further, even among all pages, the difference between the personalized PageRanks of any two pages is small (Figure 6(f)). Interestingly, according to the τ measure, the PHR algorithm is better than the BRPPR algorithm, but according to the norm measures, the BRPPR algorithm is better.

Method	Median	Mean	Std. Dev.
RPPR	0.11	0.16	0.15
BRPPR	1.02	8.21	16.05
PHRank	965.58	969.02	362.05
BPHRank	1343.16	9813.91	23,730.21

Table 2. Each row summarizes the execution times of all experiments for each approximation algorithm. All values are in seconds.

In aggregate, RPPR is the worst algorithm. In contrast, BPHR is the best algorithm. That said, even the approximations from RPPR are good, whereas those from the other three algorithms are excellent. These results verify our intuition that using some coarse-level information about the set of pages helps these algorithms achieve better approximations.

However, using that coarse information is expensive. In Table 2 we summarize the time necessary for the experiments.

While the median time for RPPR and BRPPR was below two seconds, the median time for the PHRank algorithms was around 20 minutes (with the longest run at 34 hours). This still compares favorably to the exact algorithm, which always takes over one hour to compute and requires all the data in the graph.

5. An Application to Privately Personalized Web Search

In this section, we envision a personalized web search scenario involving the use of our approximation algorithms for PageRank. We indicate how, in the near future, our algorithms can help provide personalized search results to a web user without violating the user's privacy. Concurrently, our model exploits the under-utilized potential of the client machine.

It is beyond the scope of this paper to describe and evaluate an actual implementation of the envisioned personalized web search engine, as we will treat this topic in a separate work. However, we want to provide further motivation for our current work and explain how our algorithm will be essential in such a scenario.

On personal machines, users store an abundance of information. Much of this information is about the taste and preferences of the users. Many users bookmark pages that contain significant information and value. We propose to utilize these pages for a personalized PageRank algorithm. We use the bookmarks as the reset distribution for personalized PageRank.

An exact computation of personalized PageRank requires the link structure for the entire web. However, we presented algorithms that compute a good approximation of the personalized PageRank examining only a portion of the web that is *close* to the favorites themselves, where the distance is measured in terms of *clicks away*. We propose a definition of Local Web based on this criterion.

Definition 5.1. The *Local Web* of a user is formed by the webpages that have a personalized PageRank score larger than a fixed threshold ϵ, where the personalization vector is uniform on the user's favorites.

With our definition of the Local Web, we need to state how it can be built and used. Initially, in our system, the client machine performs a targeted crawling from the user's favorites. The crawling algorithm is guided by the approximate algorithms from Section 2. For each page, the client stores the URL and the outgoing links. Later, the system computes an approximation of the personalized PageRank score for the page. Because the personalized PageRank scores are additive with respect to the personalization vector, it suffices to work with only one favorite at a time [Jeh and Widom 03].

As an initial estimate to investigate the feasibility of the system, we wanted to estimate the size of the Local Web. Toward that end, we verified in smaller-scale experiments the claim that there is only a small percentage of webpages with nonnegligible personalized PageRank scores. Thus, we anticipate that the Local Web will not exceed 0.1–1% of the total pages.

The current estimate of the size of the crawled web is around 20 billion pages [Mayer 05]; this yields a Local Web of 20–200 million pages. We estimate that the storage required for just the link structure of such a web is less than 20 GB.

The scenario we envision, therefore, requires higher computational power and memory than the current average desktop or laptop. However, this amount is feasible for current high-end machines, and it is reasonable to assume that soon we will have local machines able to deal with such data structures in memory.

When the user decides to add a favorite, two things need to be updated:

- the personalized score of the pages currently in the Local Web and
- the set of pages forming the Local Web and their personalized scores.

Our approximate algorithms (Section 2) perform the first task in real time. The second task, guided by the same approximate algorithms, could be performed offline. Hopefully, this task will last at most overnight.

Introducing the Local Web motivates a new usage model for web search. In this new model, the user inputs a query, and the client machine receives a set of URLs relevant to the query from the server—as it is done today. In the new scenario, however, the URLs would be accompanied by some score of their relevance and are not simply preranked.

The next step is to select results from the Local Web relevant to the query, together with their personal ranking. After the client machine has selected some local results, the client combines the result lists from the search server and the local personalization information to generate the final set of results. In the following paragraphs, we discuss a few possibilities for how these operations might be implemented.

We envision two possibilities for selecting a set of local pages relevant to the query. In the first, the client machine has a Local Web search engine, which locally retrieves the pages related to the query. This option guarantees that we fully exploit the potential of the Local Web but requires larger storage space for an inverted index. A second option is based on the assumption that the client is able to receive a large number or URLs from the server quickly. In this scenario, the client only uses the subset of returned results contained within the Local Web. This requires the client machine to only store the web graph and the URLs of the local pages. In this scenario, however, it is possible that we lose some pages that are contained in the Local Web but were not included in the large list communicated by the server.

Once we have selected a potential set of pages from the Local Web, the client still needs to combine the global search data with the local personalization data. For each page of the Local Web, we have a score—the approximate personalized PageRank score. In a complete web-search scenario, we will have more scoring components associated with each page—as is the case for scoring systems of popular web search engines. In this paper, we focus on link analysis scores; hence, we'll indicate one combination procedure for PageRank scores. Similar procedures can be performed with combinations of other scores.

At this point, we have a set of pages relevant to the query, coming from the server, from the Local Web, or from both. Each one has two PageRank scores: one from the server relative to a global teleportation vector and one privately stored on the client relative to a personalized teleportation vector. This second score is indeed an accurate approximation of the exact score, computed locally and without disclosing personalization parameters. We propose to use a weighted linear combination of the two. By the linearity property of the solution of the PageRank linear system, this corresponds to computing a PageRank vector with a teleportation distribution that is a weighted combination of a uniform vector and the personalized one.

6. Related Work

While we believe our work in this area to be novel, there are some strong relationships with previous work in five areas: personalized PageRank computing, focused crawling, community finding, private personalized search, and decentralized search. Further, there are many excellent survey results about PageRank and personalized PageRank computation. Two we wish to highlight are Berkhin [Berkhin 05] and Langville [Langville and Meyer 04b].

6.1. Personalized PageRank

Jeh and Widom created a hub and skeleton system to efficiently compute a large number of personalized PageRank vectors [Jeh and Widom 03]. In this system, each user has a unique personalized PageRank vector that is biased toward his or her interests. There are two problems with this approach. First, users must disclose a set of pages representing their interests to the search engine. The search engine must keep these records on file in order to produce the personalization at query time. Second, it creates a large computation task for the web search server, which we believe would be more appropriate to perform on the client. Other work on PageRank acceleration focuses on using the full web graph data in the most effective manner for global PageRank [Kamvar et al. 03b, Langville and Meyer 04a, McSherry 05]. Our ideas focus on using less web graph data to quickly compute an approximation to personalized PageRank.

Kamvar et al. exploited the block structure of the web to speed up the computation of PageRank; they also propose a fast computation of personalized PageRank, with the restriction that the user is allowed to choose only hosts as personal teleportation targets [Kamvar et al. 03a]. All these methods are very appealing if the goal is to compute, store, and retrieve tens of thousands of different personalized vectors on a server; however, we believe that our algorithms bring considerable advantages when computing a few hundreds of personalized PageRank vectors on a client-side machine.

A different approach that does not require full storage of the web graph was proposed by Abiteboul et al. [Abiteboul et al. 03]. Their algorithm continually updates a page ranking vector as a crawler processes through the web. While this method does not compute a personalized PageRank score, it computes something with similar properties.

An alternative approach that does compute an approximation to personalized PageRank was presented by Fogaras et al. [Fogaras et al. 04]. To compute personalized PageRank scores, they use a Monte Carlo approximation to precompute a manageable-size index. This index provides personalization scores for online queries. Their idea differs from previous approaches and yields personalized PageRank scores for any personalization vector, instead of personalization vectors that combine a few topic vectors or have a limited number of personalization pages. Our work differs because our focus is to reduce the use of web graph data and our framework does not involve any precomputation or a central server or any randomness. Thus, our work is decentralized and allows total privacy of the personalization parameters. Another related approximation method was recently proposed [Sarlós et al. 06].

We also found that Chien et al. used an algorithm similar to our restricted personalized PageRank algorithm to compute the change in PageRank following small changes in web graph structure [Chien et al. 04].

Other flavors of personalized or topic-biased PageRank exist: Haveliwala's topic-sensitive PageRank enforces random restarts on a selected number of pages relative to a specific topic in order to obtain a PageRank biased toward that topic [Haveliwala 02]. Richardson and Domingos proposed a modified PageRank algorithm where a probabilistic model of the relevance of the page to the query "guides" the "random surfer" [Richardson and Domingos 01]. However, an efficient execution requires precomputing of a considerable number of such PageRank vectors, for several possible query words.

Both Berkhin [Berkhin 06] and Andersen et al. [Andersen et al. 06] proposed approximate personalized PageRank algorithms based on similar PageRank properties to the ones that we exploit. The key difference is that these algorithms focus on "single-page" modifications to the PageRank vector. That is, rather than taking a power method step on the local graph as in our algorithms, Berkhin and Andersen analyze "micro-steps" or "push" operations that correspond to the portion of the step that a power method takes. Both of these papers provide detailed runtime bounds for their algorithms. As the vision of a Local Web search system running on a client motivated the development of our algorithm, we decided instead to focus on empirical evaluation of computational times in common cases rather than on a theoretical analysis of it. Nevertheless, we gave theoretical bounds on their accuracy.

6.2. Focused Crawling

One possible interpretation of the goal of our approximate algorithms is to build a focused crawler for the user's personalization interests. Cho et al. used an algorithm similar to our restricted personalized PageRank algorithm to guide a crawler to find all the *hot* pages on the web (pages with high PageRank) [Cho et al. 98]. Our work differs in that we are concerned only with personalized PageRank, which represents a smaller and more tractable goal from a crawling perspective. Chakrabarti et al. used a classification process to guide the crawler to pages that are likely to be related [Chakrabarti et al. 99].

6.3. Community Finding

Another set of related work stems from community finding on the web. We can equivalently cast our algorithms as attempts to find the community of pages related to the user's interests. This follows from the assumption that the community of pages forming the user's Local Web (and therefore, influencing the personalized PageRank score) is clustered. Because the PageRank vector is related to a random walk on the underlying web graph, we would suspect that the PageRank values would experience a step-like threshold if there are good cuts in the graph near the personalization pages. Flake et al. used graph cuts to infer community structure in the web graph [Flake et al. 00], and hence, we can view our approximate algorithms as personalized community search. Andersen and Lang expanded a seed set into a community with small conductance and a strong relationship to the seed, while examining only a small neighborhood of the entire graph [Andersen and Lang 06]. Later, the same authors proposed a local graph partitioning algorithm based on personalized PageRank [Andersen et al. 06].

6.4. Personalization Privacy

The concern of protecting user's privacy in personalizing web search results was expressed by Teevan et al. [Teevan et al. 05]. They proposed a solution that would involve the computational potential of the client. They focused on extracting personalization information from web data already present on the desktop and on reranking the web search results on the client side. Pitkow et al. proposed a similar method as well as a query-modification method based on personalization data [Pitkov et al. 02]. Our approach for using our algorithms in the context of personalized search differs in the sense that the client machine is involved in *discovering* new personally relevant data, namely the Local Web. This goes beyond the previously proposed scenario where the client machine utilizes data already present on the desktop to retrieve a personal profile of the user.

6.5. Decentralized Search

Recently, several models and prototypes have been proposed for peer-to-peer web search systems (see, e.g., [Bender et al. 05, Suel et al. 03, Reynolds and

Vadhat 03]). Those system do not rely on a central server to provide the user with web search results, but rather they explore distributed indexing and peer-to-peer communication techniques. In each of these models, peers contain collections of webpages. The research into these models is analyzing how to search different peer collections. However, there is no indication of what a single collection should contain. The experiments presented in these papers use thematic collections of webpages on each peer. Our work is complementary to these ideas. We provide algorithms for guiding the collection of a local set of pages that have a personal value and are a good approximation to the set of pages with high personalized PageRank.

7. Conclusion

In this paper we proposed efficient algorithms that allow us to compute accurate approximations of personalized PageRank scores by utilizing only the strictly necessary information out of the web graph data. We evaluated these algorithms on various web graphs, with size as large as 118 million nodes. One of our algorithms (Algorithm 3, boundary-restricted personalized PageRank) showed very good performance in terms of approximation and can almost run in real time to update scores following an update of the personalization parameters. Another (Algorithm 5, boundary PageHostRank) demonstrated excellent performance. It requires a larger, yet manageable, amount of data from the web graph to be used. While this algorithm takes considerably longer to run, we believe a parallel implementation can be developed to reduce the runtime.

Our algorithms also support the possibility of building a web search system that offers personalized search results to a web user without violating the user's privacy. Our model harnesses extra computation power on the client computer in a hybrid client-server usage model.

8. Future Work

We are currently working on the parallelization potential of our approximation algorithms, in order to fully take advantage of current and future client machine features and to improve the speed of PageHostRank.

More experiments are needed to establish accuracy and efficiency of the approximation algorithms when varying the tolerance parameter. In particular, we need to perform a comparison between the approximation algorithms under different fixed conditions than the tolerance parameter, such as the time that the algorithm is allowed to run and the number of active pages allowed.

Finally, there are many details inside the algorithms that we have heuristically chosen for this paper. First, in the boundary-restricted personalized PageRank algorithm, the theoretical result only depends on pushing the rank on the frontier below κ and does not depend at all on the details of how this occurs. This suggests that some heuristics may yield algorithms with better performance in terms of the number of pages explored. Likewise, the PageHostRank algorithm may perform better with more sophisticated models of the host graph. The theoretical bounds apply as long as there is some way to relate the agglomerated graph to a lumpable graph over all vertices along with an additional set of host vertices.

A specific crawling system needs to be built in order to construct data sets that represent the Local Web exactly following our definition. Moreover, the client machine contains more information such as the web cache, or click history, that could indicate more personalization direction.

We designed the algorithms in this paper to be accurate and efficient inside of a client-side Local Web search system. However, these algorithms may also be useful in a server-based system. The key issue is how best to use them inside of a server-based system in conjunction with other methods such as those suggested by Jeh and Widom [Jeh and Widom 03].

Furthermore, the Local Web itself needs further analysis. We can calculate other link-analysis scoring algorithms—even significantly more expensive algorithms than are not possible on the full web. For example, Kleinberg's HITS algorithm [Kleinberg 99] could be computed locally in response to a query, and the hub and authority scores could be substituted for personalized PageRank scores to compute a final ranking. Also, extremely intricate content analysis is possible if we compute and store an inverted index along with the link structure. We need to perform user-evaluation experiments with suitable evaluation criteria.

The coarse-level vision that we provided in Section 5 needs to be refined, and more components than PageRank need to be integrated. Moreover, our envisioned system stresses privacy and does not address the need for collaboration between users that has recently emerged in popular applications and in research work on P2P web search (e.g., [Bender et al. 05, Suel et al. 03, Reynolds and Vadhat 03]). The concepts and algorithms presented in this paper need to be integrated in a larger system, where peers voluntarily share (partial) information of each one's Local Web and personalized ranking.

Acknowledgements. We are grateful to Ara Nefian and Carole Dulong for the numerous discussions and inputs in the definition and development stages of this project. We would also like to thank the reviewers for their insightful suggestions and comments on the first version of this paper. Work by both authors was performed at the Application Research Lab, Microprocessor Technology Labs, Intel Corporation, Santa Clara, CA 95052.

References

[Abiteboul et al. 03] S. Abiteboul, M. Preda, and G. Cobena. “Adaptive On-Line Page Importance Computation.” In *Proceedings of the 12th International Conference on World Wide Web*, pp. 280–290. New York: ACM Press, 2003.

[Andersen and Lang 06] R. Andersen and K. J. Lang. “Communities from Seed Sets.” In *Proceedings of the 15th International Conference on World Wide Web*, pp. 223–232. New York: ACM Press, 2006.

[Andersen et al. 06] R. Andersen, F. Chung, and K. J. Lang. “Local Graph Partitioning Using PageRank Vectors.” In *47th Annual IEEE Symposium on Foundations of Computer Science*, pp. 475–486. Los Alamitos, CA: IEEE Press, 2006.

[Arasu et al. 02] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. “PageRank Computation and the Structure of the Web: Experiments and Algorithms.” In *The 11th International Conference on World Wide Web Posters Proceedings.* Available online (http://www2002.org/CDROM/poster/173.pdf), 2002.

[Bender et al. 05] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. “MINERVA: Collaborative P2P Search.” In *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 1263–1266. New York: VLDB Endowment, 2005.

[Berkhin 05] P. Berkhin. “A Survey on PageRank Computing.” *Internet Mathematics* 2:1 (2005), 73–120.

[Berkhin 06] P. Berkhin. “Bookmark Coloring Approach to Personalized PageRank.” *Internet Mathematics* 3:1 (2006), 41–62.

[Bianchini et al. 05] M. Bianchini, M. Gori, and F. Scarselli. “Inside PageRank.” *ACM Trans. Inter. Tech.* 5:1 (2005), 92–128.

[Boldi and Vigna 04] P. Boldi and S. Vigna. “The WebGraph Framework I: Compression Techniques.” In *Proceedings of the 13th International Conference on World Wide Web*, pp. 595–602. New York: ACM Press, 2004.

[Brin and Page 98] S. Brin and L. Page. “The Anatomy of a Large-Scale Hypertextual (Web) Search Engine.” *Computer Networks* 30 (1998), 107–117.

[Chakrabarti et al. 99] S. Chakrabarti, M. van den Berg, and B. Dom. “Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery.” *Proc. WWW8, Computer Networks* 31:11–16 (1999), 1623–1640.

[Chien et al. 04] S. Chien, C. Dwork, R. Kumar, D.R. Simon, and D. Sivakumar. “Link Evolution: Analysis and Algorithms.” *Internet Mathematics* 1:3 (2004), 277–304.

[Cho et al. 98] J. Cho, H. García-Molina, and L. Page. “Efficient Crawling Through URL Ordering.” *Proc. WWW7, Computer Networks* 30:1–7 (1998), 161–172.

[Flake et al. 00] G. Flake, S. Lawrence, and C. Lee Giles. "Efficient Identification of Web Communities." In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 150–160. New York: ACM Press, 2000.

[Fogaras et al. 04] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. "Towards Scaling Fully Personalized PageRank." In *Algorithms and Models for the Web-Graph: Third International Workshop, WAW 2004, Rome, Italy, October 16, 2004, Proceedings*, Lecture Notes in Computer Science 3243, pp. 105–117. Berlin: Springer Verlag, 2004.

[Haveliwala 02] T. Haveliwala. "Topic-Sensitive PageRank." In *Proceedings of the 11th International Conference on World Wide Web*, pp. 517–526. New York: ACM Press, 2002.

[Horton 97] G. Horton. "On the Multilevel Solution Algorithm for Markov Chains." Technical Report NASA CR-201671 ICASE Report No. 97-17, NASA Langley Research Center, 1997.

[Jeh and Widom 03] G. Jeh and J. Widom. "Scaling Personalized Web Search." In *Proceedings of the 12th international conference on World Wide Web*, pp. 271–279. New York: ACM Press, 2003.

[Kamvar 03] Sepandar D. Kamvar. *Sepandar D. Kamvar's homepage.* Data section includes Stanford and Stanford-Berkeley data sets. Available online (http://www.stanford.edu/~sdkamvar/research.html), 2003.

[Kamvar et al. 03a] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. "Exploiting the Block Structure of the Web for Computing PageRank." Technical report, Stanford University, 2003.

[Kamvar et al. 03b] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. "Extrapolation Methods for Accelerating PageRank Computations." In *Proceedings of the 12th International Conference on World Wide Web*, pp. 261–270. New York: ACM Press, 2003.

[Kemeny and Snell 83] J. Kemeny and J. L. Snell. *Finite Markov Chains*, Undergraduate Texts in Mathematics. New York: Springer Verlag, 1983.

[Kleinberg 99] J. M. Kleinberg. "Authoritative Sources in a Hyperlinked Environment." *Journal of the ACM* 46:5 (1999), 604–632.

[Langville and Meyer 04a] A. Langville and C. Meyer. "Updating PageRank with Iterative Aggregation." In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, pp. 392–393. New York: ACM Press, 2004.

[Langville and Meyer 04b] Amy N. Langville and Carl D. Meyer. "Deeper Inside PageRank." *Internet Mathematics* 1:3 (2004), 335–380.

[Mayer 05] Tim Mayer. "Our Blog is Growing Up—And So Has Our Index," *Yahoo! Search Blog.* Available online (http://www.ysearchblog.com/archives/000172.html), 2005.

[McSherry 05] F. McSherry. "A Uniform Approach to Accelerated PageRank Computation." In *Proceedings of the 14th International Conference on World Wide Web*, pp. 575–582. New York: ACM Press, 2005.

[Pitkov et al. 02] J. Pitkov, H. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. "Personalized Search." *Communications of the ACM* 45:9 (2002), 50–55.

[Reynolds and Vadhat 03] P. Reynolds and A. Vadhat. "Efficient Peer-to-Peer Keyword Searching." In *Middleware 2003: ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, Brazil, June 16–20, 2003, Proceedings*, Lecture Notes in Computer Science 2672, pp. 21–40. New York: Springer, 2003.

[Richardson and Domingos 01] M. Richardson and P. Domingos. "The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank." *Advances in Neural Information Processing Systems* 14. Available online (http://books.nips.cc/papers/files/nips14/AA57.pdf), 2001.

[Sarlós et al. 06] T. Sarlós, A. A. Benczúr, K. Csalogány, D. Fogaras, and B. Rácz. "To Randomize or Not To Randomize: Space Optimal Summaries for Hyperlink Analysis." In *Proceedings of the 15th International Conference on World Wide Web*, pp. 297–306. New York: ACM Press, 2006.

[Suel et al. 03] T. Suel, C. Mathur, J.-W. Wu, J. Zhang, A. Delis, M. Kharrazi, X. Long, and K. Shanmugasundaram. "ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrival." *International Workshop on the Web and Databases (WebDB).*

[Teevan et al. 05] J. Teevan, S. T. Dumais, and E. Horvitz. "Personalizing Search via Automated Analysis of Interests and Activities." In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 449–456. New York: ACM Press, 2005.

[Vigna 06] Sebastiano Vigna. "WebGraph." Available online (http://webgraph.dsi.unimi.it/), 2006.

David Gleich, Institute for Computation and Mathematical Engineering, Stanford University, Stanford, CA 94305 (dgleich@stanford.edu)

Marzia Polito, ABInventio, Altadena, CA 91101 (marzia@abinventio.com)

Received April 14, 2006; accepted October 4, 2006.