

# Approximating Personalized PageRank with Minimal Use of Webgraph Data

David Gleich\*

Institute for Computation  
and Mathematical Engineering  
Stanford University, Stanford, CA  
dgleich@stanford.edu

Marzia Polito

Applications Research Lab  
Microprocessor Technology Labs  
Intel Corporation, Santa Clara, CA  
marzia.polito@intel.com

## Abstract

In this paper, we consider the problem of calculating fast and accurate approximations to the personalized PageRank score ([8, 16]) of a webpage. We focus on techniques to improve speed by limiting the amount of webgraph data we need to access.

PageRank scores are mainly used for ranking purposes, and generally only the scores exceeding a given threshold are relevant. In practice, and relative to the size of the web, only a small number of pages have a non-negligible personalized PageRank score. We capitalize on this property of the PageRank score to reduce the amount of webgraph data needed for computation. 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.

Our algorithms assume that we have random access to a sparse representation of a webgraph (perhaps by crawling the web if necessary); some of them further require knowledge of a hostgraph ([19]). All of them provide a fast approximate calculation of the personalized PageRank score for pages where the score exceeds a given threshold.

We report experiments with these algorithms on webgraphs of up to 118 million pages and prove theoretical approximation bound for all. We conclude by proposing an application scenario of personalized Web Search that inspired and motivated our work.

## 1 Introduction and motivation

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

---

\*Work performed during a summer internship with Intel Corporation.

Jeh and Widom [16] developed personalized PageRank with the goal of personalizing web 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 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 webgraph matrix. Our algorithms iteratively divide the vertices of the webgraph into an *active* 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 outlinks 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 webgraphs of up to 118 million nodes.

Unlike [16], our approach does not involve pre-computation and storage of a number of personalized vectors, or building blocks for them. 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 webgraph, 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 our first approach, we do not use upfront any knowledge about the webgraph. In a second approach, we use coarse level information from the *hostgraph* (see [19]). This graph has a vertex for each host, and summarizes in its links all the links between single pages. The storage space for the hostgraph, as well as the computational time needed to deal with it, are considerably less than the ones relative to the whole webgraph.

The analysis of a fully-functional web search system using this technique is beyond the scope of this paper. But we do provide a high level vision of it. We envision the incremental discovery of the webgraph and the computation of personalized PageRank score as performed by a client machine via a lightweight crawling integrated with the algorithm itself. Our algorithms makes 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 exploiting of the computational potential of client machines in web search.

In section 2 we give a detailed description of our four algorithms, while in section 3 we state and prove their theoretical approximation bounds. 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 motivated their development. While we do not yet have a complete implementation

Symbol	Meaning	Sec.
$\alpha$	random teleportation probability	2
$A$	PageRank Markov chain transition matrix	2
$\mathcal{B}$	set of host vertices in graph $C$	2.3
$C$	combination page and host adjacency matrix	2.3
$\delta$	total change in PageRank in an iteration	2
$D_W$	diagonal matrix of outdegrees for graph $W$	2
$d_W$	dangling indicator vector for graph $W$	2
$e$	vector of all ones	2
$\varepsilon$	the expansion tolerance	2.1
$\mathcal{F}$	frontier set of pages	2.2
$\Gamma(v)$	neighbor set for vertex $v$	2.1
$H$	host graph adjacency matrix	2.3
$L$	adjacency matrix for a subset of active pages	2.1
$\mathcal{L}$	set of active pages	2.1
$l_v$	within host in-link for vertex $v$	2.3
$\mathcal{P}$	set of separated pages	2.3
$p$	a single personalization vertex	2.1
$R$	page to host restriction operator	2.3
$\mathcal{S}$	set of pages to separate	2.3
$W$	web graph adjacency matrix	2
$v$	Teleportation Distribution	2
$\mathcal{V}$	set of vertices in a graph	2
$x(v)$	the entry in vector $x$ for vertex $v$	2.1
$\partial\mathcal{L}$	the frontier for a set of pages $\mathcal{L}$	2.2

Table 1: Notation summary and the sections where it was introduced. In general, a lower-case Greek letters are scalar values, lower-case letters are vectors or set elements, upper-case letters are matrices, and script letters are sets.

of such a system, we believe it is a feasible goal. 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 this work.

## 2 Algorithms for computing Personalized PageRank

Many authors have described the standard PageRank algorithm applied to an entire graph [8, 19, 4]. We briefly review the algorithm here. Given an adjacency matrix  $W$  for a webgraph with vertex set  $\mathcal{V}$ , the PageRank algorithm computes the stationary distribution of a random-walk Markov chain where, with probability  $\alpha$ , the walker follows the outlinks of a given page, and with probability  $1 - \alpha$ , the walker teleports to a prior distribution over pages,  $v$ . We summarize the remainder of the notation in table 1.

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  $D_W$  is the diagonal matrix of outdegrees 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. (We use the notation  $D_W^{-1}$  to indicate the matrix with the inverse outdegree for each node, which is the inverse if all elements are non-zero.)

$$x^{(0)} = v, k = 0$$

**repeat**

$$y = \alpha W^T D_W^{-1} x^{(k)}$$

$$\omega = 1 - \|y\|_1$$

$$x^{(k+1)} = y + \omega v$$

$$\delta = \|x^{(k)} - x^{(k-1)}\|_1, k = k + 1$$

**until**  $\delta$  is less than stopping tolerance.

If  $v$  is the uniform distribution over all pages, then the algorithm 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 [14]. If  $v$  is a subset of pages chosen according to a user's interests, the algorithm computes a *personalized PageRank vector* [16], or PPR.

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

## 2.1 Algorithm 1: Restricted Personal PageRank

The restricted personal PageRank algorithm (RPPR) attempts to determine the set of pages with high personal 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  $\varepsilon$ . The motivation for this algorithm is a formula from Jeh and Widom [16], section 4.1, which identifies the PageRank value for a single-page personalization vector with the inverse  $P$ -distance in the webgraph. For simplicity of presentation, we assume that this algorithm is only applied to compute the personalized PageRank vector for a single page  $p$ .

In the following algorithm definition, the symbol  $x(v)$  is the entry in vector  $x$  associated with page  $v$ . 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 outlinks from pages in  $\mathcal{L}$  and may reference pages not in  $\mathcal{L}$  (but these pages will not have any outlinks 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 any outlinks from page  $v$  with  $y(v) > \varepsilon$ .

$$x(p) = 1$$

**repeat**

$$y = \alpha L^T D_L^{-1} x \text{ \{The PageRank iteration is here.\}}$$

$$\omega = 1 - \|y\|_1$$

$y(p) = y(p) + \omega$   
 Add outlinks 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.

## 2.2 Algorithm 2: Boundary Restricted Personal PageRank

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

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

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

is the set of pages we know exist due to other outlinks but have not yet visited. (Here, we borrow the notation  $\partial$  to represent the boundary of a domain – in this case  $\mathcal{L}$ , the set of active pages.)

$x(p) = 1$   
**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.

In practice, 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 outlinks to  $\mathcal{L}$  until the rank on the remainder of the frontier is less than  $\kappa$ .

## 2.3 Algorithm 3: PageHostRank

PageHostRank (PHRank) is a hybrid algorithm that attempts to meld ideas from algorithms 1 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 [19]. That is, we view a host as a collection of webpages that all share the same host in the URL. For example, <http://www.intel.com/> and <http://www.intel.com/personal/index.htm> share the same host ([www.intel.com/](http://www.intel.com/)).

com), whereas `http://research.intel.com/ir/researchareas/index.asp` has a different host (`research.intel.com`). The host graph adjacency matrix  $H$  then has a weighted link between hosts  $I$  and  $J$  if host  $I$  contains any page that links to a page on host  $J$ . The weight on the link 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 page to host restriction operator.  $R_{iJ} = 1$  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 non-zero element in each row of  $R$ . The host graph adjacency matrix  $H$  is

$$H = R^T A R.$$

For the PageHostRank algorithm, we assume that the algorithm has two pieces of global information. The first is the host graph  $H$  itself, together with the hosts for each node in  $H$  and a count of dangling vertices for each host,  $d_H$ . The second is a vector  $l$  over all pages on the web specifying the number of inlinks 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  $i$ .

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 outlinks. These separated pages become active. (In some sense, we re-agglomerate 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  $C$  be the hybrid combination graph at the current iteration.  $C$  is a weighted graph where the weight on a link counts the number of aggregated links. Within  $C$  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 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 outlinks 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 outlinks to pages not in  $\mathcal{P}$ , we aggregate the outlinks 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 inlinks 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 inlink from the host  $J$  to page  $j$  based on a) the value  $l_j$  in the within-host inlink count and b) the number of inlinks to page  $j$  from pages in host  $J$  and in  $\mathcal{P}$ . For example, if we know that  $j$  has 10 inlinks from the host ( $l_j = 10$ ), and we only see 6 inlinks from pages in  $\mathcal{P}$  and in  $J$  to  $j$  in  $C$ , we'll connect 4 inlinks from  $J$  to page  $j$ .

There is one caveat at this last step; we always insist that there exist 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  $i$ .

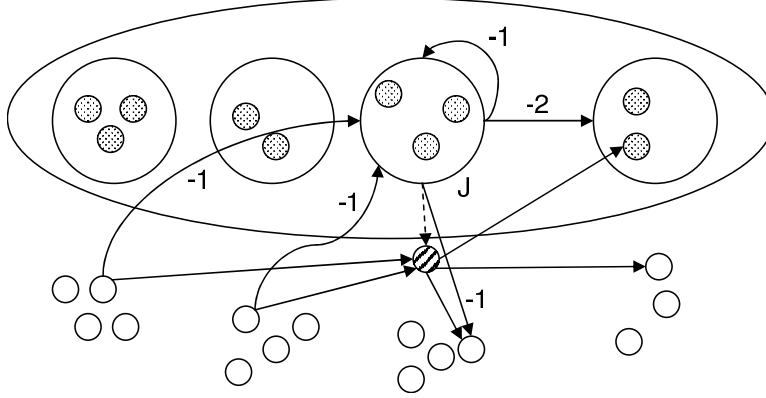


Figure 1: A visual depiction of separating a page in the PageHostRank algorithm. The dotted pages are the frontier pages which are represented as part of the hosts. The striped page is separated and we explicitly connect it with two other separated pages. Further, it has two links from separated pages. The links between the hosts with negative weight show that we remove some weight from each link because they are now represented in the actual page graph. (The host for each pages is the large node above it.) The dashed link is the inferred link based on the provided host-inlink count.

Finally, we note that when adding links to graph  $C$ , 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  $k$ , we remove 1 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. For our experiments, we assigned the new rank of the page by setting  $x(j) = x(J)/|J \cap \mathcal{F}|$ .

We summarize this discussion with the separate page algorithm.

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 outlinks from page  $j$ . This step is a host-to-host ( $\mathcal{B}$  to  $\mathcal{B}$ ) adjustment.
3. Subtract weight on the links from the host  $J$  to pages in the set  $\mathcal{P}$  and in the outlinks of  $j$ . This step is a host-to-page ( $\mathcal{B}$  to  $\mathcal{P}$ ) adjustment. Here, we enforce the presence of a minimum of one link between host  $J$  and page  $j$ .
4. For any inlinks 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.
5. Do our best to estimate the number of inlinks from host  $J$  to page  $j$ . Here, we know two things: i) the number of total inlinks from pages in host  $J$  to  $j$  ( $l_j$ ), and ii) the

current number of inlinks from pages in host  $J$  to page  $j$ ; the difference indicates the number of inlinks. However, we enforce that there is always one inlink to  $j$  from  $J$ . Add this link to graph  $C$ .

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 outlinks 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  $\mathcal{B}$ .
9. Distribute some of the rank from the host vertex  $J$  to the newly created vertex  $j$ .

One property of this separate page algorithm is that when we crawl all pages, we recover the original webgraph  $W$  in a permuted order. There is an extra set of host vertices included in the graph, but these vertices have no remaining inlinks, and therefore play no role in the computation of personal PageRank. Put another way, if we separate all pages and run the algorithm (given fully below), 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 graph  $C$ . At each iteration, we examine the rank on hosts in the set  $\mathcal{B}$ . For any host  $I$  with rank that exceeds  $\varepsilon$ , 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 the following algorithm,  $R(i)$  maps from page  $i$  to host  $I$ , and  $\mathcal{S}$  is the set of pages to separate.

$$C = H, \mathcal{S} = \{p\}, x(R(p)) = 1, \mathcal{F} = \{p\}, \mathcal{P} = \{ \}.$$

**repeat**

For all pages  $s \in \mathcal{S}$ , separate  $s$  and update  $C$ ,  $x$ ,  $\mathcal{F}$ , and  $\mathcal{P}$ .

$$y = \alpha(C^T + e_p d_H^T) D_C^{-1} x$$

$$\omega = 1 - \|y\|_1$$

$$y(p) = y(p) + \omega$$

$$\delta = \|y - x\|_1, x = y$$

$$\mathcal{S} = \{v \mid v \in \mathcal{F}, x(R(v)) > \varepsilon\}$$

**until**  $\delta$  is less than stopping tolerance.

$$x(\mathcal{F}) = 0.$$

## 2.4 Algorithm 4: Boundary PageHostRank

Again, we modify the PageHostRank algorithm to restrict the total rank on the frontier set to  $\kappa$ . See section 2.2 for more details.

$$C = H, \mathcal{S} = \{p\}, x(R(p)) = 1, \mathcal{F} = \{p\}, \mathcal{P} = \{ \}.$$

**repeat**

For all pages  $s \in \mathcal{S}$ , separate  $s$  and update  $C$ ,  $x$ ,  $\mathcal{F}$ , and  $\mathcal{P}$ .

$$y = \alpha(C^T + e_p d_H^T) D_C^{-1} x$$

$$\omega = 1 - \|y\|_1$$

$$y(p) = y(p) + \omega$$

$$\delta = \|y - x\|_1, x = y$$

Sort the ranks on each host in decreasing order.

Examine hosts in 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 personal PageRank algorithm (section 2.2), we determine which hosts have pages that should be interesting, and expand any pages in the frontier on those hosts.

## 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 personal PageRank, we compute the exact personal PageRank for the active page graph  $L$ . The largest  $L$  can be is the original web graph  $W$ , and we never remove any page once it is added to  $L$ ; thus the algorithm terminates. This argument also applies to the PageHostRank algorithm on the graph  $C$ .

## 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 personal PageRank algorithm. Then, we show the relationship between HostRank and PageRank. Finally, we expand the relationship between HostRank and PageRank to model the PageHostRank algorithm, and we prove an approximation bound for that algorithm.

### 3.1 Basic Approximation for Restricted PageRank

We prove here the lemma that is the basis for our approximation bounds. This lemma states that we can express the difference between a modified and exact solution in terms of the modified solution. Before we begin, let's establish some notation.

In contrast with section 2, let

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

$A^T$  is row-stochastic, but without the rank-1 modification in the PageRank Markov chain. In these theorems, we will also be dealing with modified webgraphs which will be denoted by  $\tilde{W}$ . In the same manner,

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

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

**Lemma 3.1.** *Let  $x^*$  be the exact PageRank vector for webgraph  $W$  and personalization vector  $v$ ,*

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

*Also, let  $\tilde{x}$  be the PageRank vector for a modification of webgraph  $W$  to  $\tilde{W}$  with the same nodes as  $W$  and the same personalization vector  $v$ ,*

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

*Then,*

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

*where*

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

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

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

*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,

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

and

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

Recall that  $\delta = \tilde{x} - x^*$  by definition, so  $\tilde{x} = \delta + x^*$ , then

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

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

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

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

$$[I - \alpha A^T - \alpha \Delta^T] (\delta + x^*) = (1 - \alpha)v.$$

$$[I - \alpha A^T] \delta + [I - \alpha A^T] x^* - \alpha \Delta^T (\delta + x^*) = (1 - \alpha)v.$$

$$[I - \alpha A^T] \delta + [I - \alpha A^T] x^* - \alpha \Delta^T \tilde{x} = (1 - \alpha)v.$$

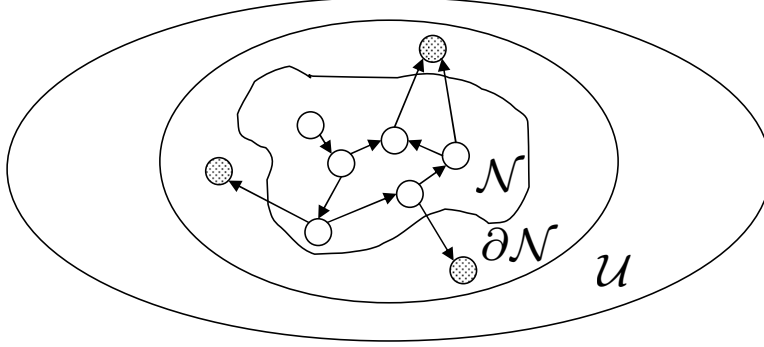


Figure 2: The hierarchy of sets we use to compute our bounds.

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

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

Re-arranging this last expression yields

$$[I - \alpha A^T] \delta = \alpha \Delta^T \tilde{x}.$$

$$[I - \alpha A^T] \delta = (1 - \alpha) \frac{\alpha}{1 - \alpha} \Delta^T \tilde{x}.$$

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

In our approximation algorithms, we bound the maximum rank on all the unexpanded nodes. Let  $\Gamma(\mathcal{N})$  to 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.

**Theorem 3.1.** *Suppose that  $\tilde{x}$  is the solution from the restricted personalized PageRank algorithm. Let  $\delta = \tilde{x} - x^*$  and*

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

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

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

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

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

Taking the norm, we have

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

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

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

To understand this statements, let's study the structure of  $\Delta^T$ . In the expanded neighborhood, we know all the outlinks exactly. Because the personalization support is entirely within the set  $\mathcal{N}$ , the matrix  $\Delta$  is 0 in all rows corresponding to pages in  $\mathcal{N}$  (respectively, 0 in all columns of  $\Delta^T$ ). The notation  $\Delta_{\partial\mathcal{N}}^T$  just refers to the subregion of the matrix associated with the outlinks of pages in  $\partial\mathcal{N}$  (the same with  $\Delta_{\mathcal{U}}^T$ ). In fact, we don't have to address  $\Delta_{\mathcal{U}}^T$  at all because it drops out of the equation due to the 0's in the  $\tilde{x}$  vector.

We can further bound  $\|\Delta_{\partial\mathcal{N}}^T\|_1$  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 outlink. Thus,

$$\|\Delta_{\partial\mathcal{N}}^T\|_1 \leq 2.$$

By combining these results, the proof follows. □

This result was also shown in [6].

**Remark 3.1.** *Based on the previous proof, the restricted personal PageRank algorithm with expansion tolerance  $\varepsilon$  yields an approximate PageRank vector  $\tilde{x}$ , where*

$$\|x - \tilde{x}\|_1 \leq \frac{2\alpha}{1-\alpha} \varepsilon |\Gamma(\mathcal{N}) - \mathcal{N}|.$$

**Remark 3.2.** *In the boundary restricted personalized PageRank algorithm, we bound  $\kappa$  explicitly. Thus,*

$$\|x - \tilde{x}\|_1 \leq \frac{2\alpha}{1-\alpha} \kappa.$$

We can slightly tighten the previous theorem with a more careful accounting. In the following lemma, we reuse the notation from the previous theorem for brevity.

**Lemma 3.2.** *If we permute  $\tilde{x}$  to write*

$$\tilde{x}^T = \left( \tilde{x}_{\mathcal{N}}^T \quad \overbrace{\tilde{x}_{\text{dangling}}^T \quad \tilde{x}_{\text{link to } p}^T \quad \tilde{x}_{\text{no link to } p}^T}^{\tilde{x}_{\partial\mathcal{N}}^T} \quad 0^T \right)^T$$

(where  $p$  is a single personalization page) and outlinks are weighted uniformly, then

$$\|\Delta^T \tilde{x}\|_1 \leq \left( 1 + \frac{|\mathcal{V}| - 2}{|\mathcal{V}|} \right) \|\tilde{x}_{\text{link to } p}\|_1 + 2\|\tilde{x}_{\text{no link to } p}\|_1 \leq 2\|\tilde{x}_{\partial\mathcal{N}}\|_1.$$

*Proof.* This follows directly from permuting the boundary elements on the matrix to achieve the ordering of  $\tilde{x}$  given in the statement of the lemma, and from writing down explicitly the  $\Delta$  matrix. For pages that are dangling,  $\tilde{x}_{\text{dangling}}$ , the columns of  $\Delta^T$  are 0. For pages that link to  $p$ , they must have less than  $|\mathcal{V}|$  neighbors, thus the minimum weight outlink is  $1/|\mathcal{V}|$ . Because these page links to  $p$ , the total column sum of  $\Delta^T$  is less than

$$1 - (1/|\mathcal{V}|) + (|\mathcal{V}| - 1)/|\mathcal{V}| = 1 + (|\mathcal{V}| - 2)/|\mathcal{V}|.$$

Finally, for pages that do not link to  $p$ , the sum of absolute values of the elements in the corresponding column of  $\Delta^T$  is exactly 2.  $\square$

**Remark 3.3.** *If any page in  $\partial\mathcal{N}$  links to  $p$  or if we have a dangling page,*

$$\|\delta\|_1 < \frac{2\alpha}{1 - \alpha} \kappa.$$

## 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.

Let  $R$  be the page to host restriction operator as defined in section 2.3. Then

$$H = R^T W R$$

is the weighted adjacency matrix between hosts. Let  $\hat{W}$  be the dangling-node adjusted webgraph

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

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

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

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

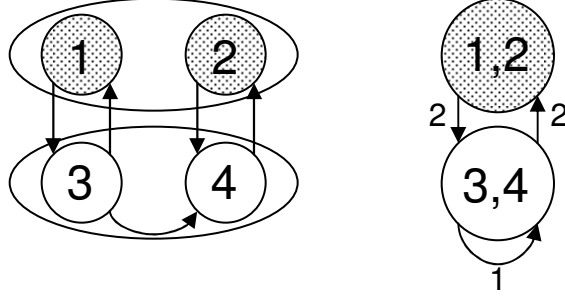


Figure 3: A counter example for the simple relationship between HostRank and PageRank. If we personalize on the shaded pages, then we get a PageRank vector of  $x^* = (0.1174 \ 0.4002 \ 0.0998 \ 0.3826)^T$ . The corresponding HostRank vector is  $h = (0.4574 \ 0.5426)^T$ . Clearly, the statement is false.

**Definition 3.1** (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  $\hat{W}$ ).*

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  $I$ .

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 webgraph  $W$  with the same personalization. We call this the summation property.

Unfortunately, this statement is not generally true. See figure 3 for a counter-example. However, Horton showed that a solution dependent restriction operator does give the summation property in [15]. Unfortunately, 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 webgraph, then we keep the summation property.

We name our modification *silent side-step* PageRank. In this model, we change the behavior of the random surfer.

- With probability  $(1 - \alpha)$ , the surfer randomly jumps to any personalized page.
- With probability  $\alpha$ , the surfer randomly jumps to any other page  $i'$  on the same host, and randomly follows an outlink 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 second part of the silent 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,

$$U = D_{RR^T \hat{W}}^{-1} R R^T \hat{W}.$$

In this equation, the matrix  $D_{RR^T \hat{W}}^{-1}$  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.

**Theorem 3.2.** *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

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

$$R^T u = \alpha R^T \hat{W}^T R R^T D_{RR^T \hat{W}}^{-1} u + (1 - \alpha)R^T v.$$

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

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

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

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

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

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

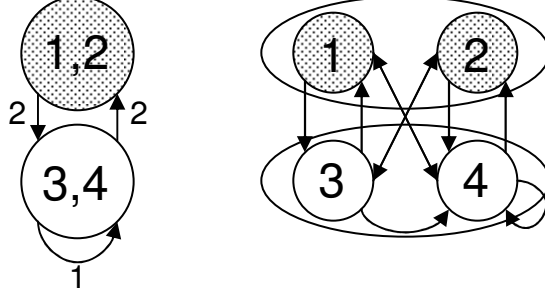


Figure 4: In this figure, we show the agglomerated HostRank graph and the silent side-step PageRank graph. In the silent-step PageRank graph, each node collects all the outlinks from each vertex agglomerated together. Recall that  $h = (0.4574 \ 0.5426)^T$ . Now we have that  $u = (0.2287 \ 0.2287 \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

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 our modified webgraph lemma to relate the modified graph back to the original webgraph which will provide the approximation bound for the PageHostRank algorithm.

First, we describe the adjacency matrix  $C$  for the combination graph in the PageHostRank algorithm. 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 that

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

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.

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

$$C_{I_a, j} = \min \left( \sum_{i \in \mathcal{I}_a} W_{i,j}, 1 \right)$$

$$C_{I_a, j} = 0 \text{ if } j \notin \mathcal{I}$$

$$C_{i, J_a} = \sum_{j \in \mathcal{J}_a \cup \mathcal{F}} W_{i,j}$$

$$C_{i, j} = W_{i,j} \text{ if } j \in \mathcal{J}_e.$$

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 weight between all agglomerated nodes in  $\mathcal{I}_a$  without the nodes with artificial in-links. 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 webgraph that has a summation property with the combination graph  $C$ . Let  $\tilde{C}$  be a webgraph with  $|\mathcal{V}| + |\mathcal{B}|$  nodes (recall 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 webgraph  $\tilde{C}$  is the original webgraph  $W$  with a 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 in  $j \in \mathcal{J}_a$ , instead of linking to  $i \in \mathcal{I}_e$  as in the original webgraph, 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  $\tilde{C}$  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 for  $\tilde{C}$  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 graph  $W$  modified with the extra host node as above<sup>1</sup>, i.e.

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

The following theorem now follows.

**Theorem 3.3.** *If  $x$  is a PageHostRank vector, then there exists a webgraph  $\tilde{W}$  with  $|\mathcal{V}| + |\mathcal{B}|$  vertices, such that the personalized PageRank vector on  $\tilde{W}$ ,  $\tilde{x}$  satisfies*

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

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

---

<sup>1</sup>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.

*Proof.* The modified webgraph  $\tilde{W} = \tilde{C}$  as defined above. The summation property follows because we defined  $\tilde{C}$  such that all nodes within a host are lumpable [21]. Thus, when we aggregate them together the corresponding stationary probability vectors have the summation property.  $\square$

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 step move gives more intuition about what occurs.

Using the previous theorem, we can conclude that PageHostRank gives us an approximation to personal PageRank.

**Theorem 3.4.** *Suppose that  $\tilde{x}$  is the solution from the full PageHostRank model,  $x$  is the solution from the PageHostRank model including the host nodes, and  $x^*$  is the exact PageRank vector extended with  $|\mathcal{B}|$  extra nodes with 0 rank. Let  $\delta = \tilde{x} - x^*$  and*

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

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

$$\|\delta\|_1 \leq \frac{2\alpha}{1-\alpha} \kappa.$$

*Proof.* By the summation property from the previous theory, 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

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

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 personal PageRank on a webgraph 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,

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

Hence,

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

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.

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

□

## 4 Experimentation on web graphs

We used our approximate PageRank algorithms on a few publicly available data-sets. The corresponding webgraphs had variable size from 10000 to 118 million vertices. We report on the experiments with the largest data-set, the WebBase graph of approximate 118 million vertices. Sources of our data sets were [2] and [7, 3].

We performed the experiments on an Intel<sup>TM</sup> Itanium<sup>TM</sup> 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 established 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 webgraph, then computed restricted personal PageRank (RPPR, section 2.1), boundary restricted personal PageRank (BRPPR, section 2.2), PageHostRank (PHRank, section 2.3), and boundary PageHostRank (BPHRank, 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 blue box has lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. Outliers (red crosses) are data with values beyond the ends of the whiskers.

In figure 5 we report the results relative to evaluation of corresponding rankings. The Kendall  $\tau$  between the exact and approximate ranking for the first 10, 100, and 1000 pages according to each ranking method are calculated. Figure 6 presents on the 1- and  $\infty$ -norm distance between PageRank vectors. We calculate these distances on the whole vector, as well as the subset of entries corresponding to active and frontier pages. Active pages correspond to the set of pages with known outlinks. Frontier pages correspond to the set of pages known from outlinks but unexplored.

In aggregate, RPPR is the worst algorithm. In contrast, BPHRank 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.

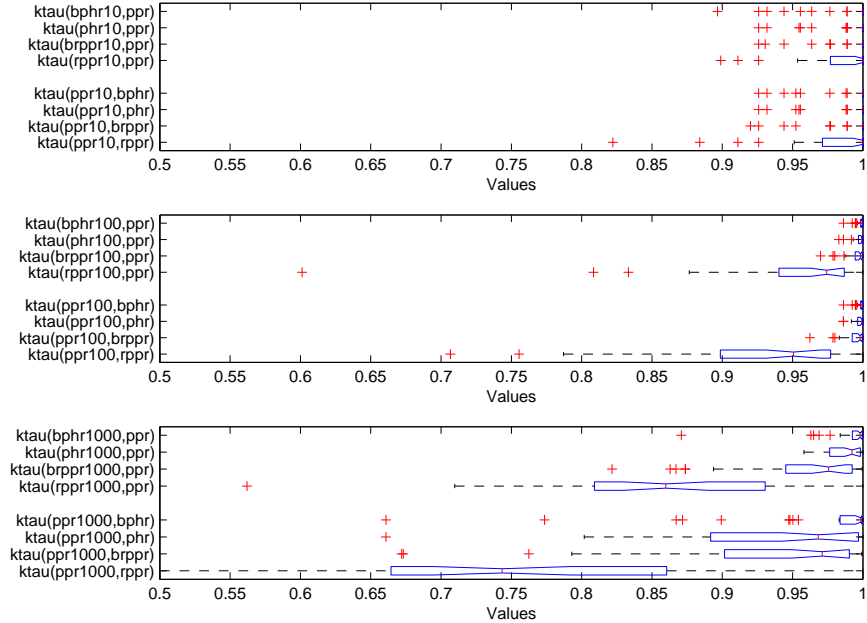


Figure 5: *Upper*: The first 4 rows are related to the first 10 pages ranked according to each of the 4 approximation algorithms. The following 4 rows are related to the first 10 pages ranked according to the exact algorithm. The values are the Kendall’s tau’s between the approximate and the exact ranking. *Middle and Lower*: Same representation, but with the first 100 and 1000 pages.

However, using that coarse information is expensive. In table 2 we summarize on the time necessary to the experiments. While the median time for RPPR and BRPPR was below 2 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

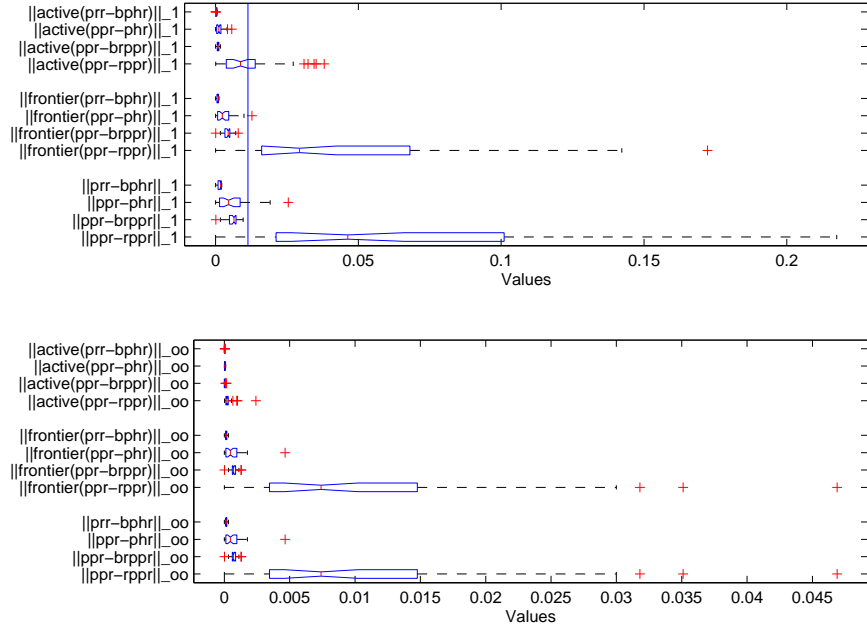


Figure 6: *Upper*: The first three rows are related to the set of active pages, the second three rows to the set of frontier pages, and the third set of three rows to the complete set of all pages. The values are the 1-norm distances between the approximate and the exact PageRank vectors. Each row refers to a different approximation algorithm. The vertical line shows the theoretical bound. *Lower*: Same representation, but with the  $\infty$ -norm distance.

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 web pages that have a personalized PageRank score larger than a fixed threshold  $\epsilon$ , 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 [16].

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	23730.21

Table 2: Each row represents the execution times of all experiments, for each one of the approximation algorithms. All values are in seconds.

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 web pages with non-negligible personalized PageRank scores. Thus, we anticipate that the Local Web will not exceed 0.1 – 1% of total pages.

The current estimate of the size of the crawled web is around 20 billion pages [1]; this yields a Local Web of 20 – 200 million pages. We estimate the storage required for just the link structure of such a web is less than 20GB.

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 pre-ranked.

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 of 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 which 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.

### Personalized PageRank

Jeh and Widom [16] created a hub and skeleton system to efficiently compute a large number of personalized PageRank vectors. In this system, each user has a unique personalized PageRank vector that is biased towards 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 appropriately to perform on the client. Other work on PageRank acceleration [20, 23, 24] focuses on using the full webgraph data in the most effective manner for global PageRank. Our ideas focus on using less webgraph data to quickly compute an approximation to personal PageRank.

An alternative approach to compute an approximation of personalized PageRank is found in [13]. To compute personalized PageRank scores, they use a Monte Carlo approximation

to precompute a manageable size index. This index provides personalization scores for on-line queries. Their idea differs from previous approaches and yields personalized PageRank scores for any personalization vector, instead of personalization vectors which 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 webgraph data and our framework does not involve any precomputation or a central server. Thus, our work is decentralized and allows total privacy of the personalization parameters.

We also found that Chien et al. used a similar algorithm to our restricted personal PageRank algorithm to compute the change in PageRank following small changes in webgraph structure in [10].

### **Focused Crawling**

One possible interpretation of the goal of our approximate algorithms is to build a focused crawler for the user’s personalization interests. In [11], Cho et al. use an algorithm similar to our restricted personal PageRank algorithm to guide a crawler to find all the *hot* pages on the web (pages with high PageRank). Our work differs in that we are concerned only with personalized PageRank, which represents a smaller and more tractable goal from a crawling perspective. In [9], the authors use a classification process to guide the crawler to pages that are likely to be related.

### **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 is related to a random-walk on the underlying webgraph, 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. In [12], Flake et al. used graph cuts to infer community structure in the webgraph, and hence, we can view our approximate algorithms as personalization community search.

### **Personalization Privacy**

The concern of protecting user’s privacy in personalizing web search results was expressed by Teevan et al. [18], as well as proposing a solution that would involve the computational potential of the client. There, the focus is on extracting personalization information from web data already present on the desktop and on re-ranking on the client side the results of web search. Pitkow et al. [17] propose a similar method as well as a query modification method based on personalization data. 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.

## Decentralized Search

Recently, several models and prototypes have been proposed for peer-to-peer (web)search systems (see e.g. [5, 26, 25]). Those systems do not rely on a central server to provide the user with web search results, but rather explore distributed indexing and peer to peer communication techniques. In each of these models, peers contain collections of web pages. The research into these models is analyzing how to search different peer collections. However, there is no indication on what a single collection should contain. The experiments presented in these papers use thematic collections of web pages 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 personal PageRank.

## 7 Conclusion and future work

In this paper we proposed efficient algorithms that allow to compute accurate approximations of Personalized PageRank score, by utilizing only the strictly necessary information out of the webgraph data.

We evaluated these algorithms on various webgraphs, of size up to 118 million nodes. One of our algorithms (Boundary Restricted Personal PageRank) showed very good performance in terms of approximation and can almost be run in real time to update scores following an update of the personalization parameters. Another (Boundary PageHostRank) demonstrated excellent performance. It requires a larger, yet manageable, amount of data from the webgraph 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.

### 7.1 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 on varying the tolerance parameter. In particular, we need to perform a comparison between 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.

A specific crawling system needs to be build, 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.

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 [22] 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 computed and stored an inverted index along with the link structure. We need to perform user-evaluation experiments with suitable evaluation criteria.

The very coarse level vision 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 [5, 26, 25]). The concepts and algorithms presented in this paper need to to be integrated in a larger system, where peers voluntarily share (partial) information of each one’s local web and personalized ranking.

## 8 Acknowledgments

We are grateful to Ara Nefian and Carole Dulong for the numerous discussions and inputs in the definition and development stages of this project.

## References

- [1] Our blog is growing up and so has our index. <http://www.ysearchblog.com/archives/000172.html>.
- [2] Stanford and stanford-berkeley data sets. <http://www.stanford.edu/~sdkamvar/research.html>.
- [3] Webgraph data sets. <http://webgraph.dsi.unimi.it/>.
- [4] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. PageRank computation and the structure of the web: Experiments and algorithms. *11th international conference on World Wide Web*, 2002.
- [5] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: Collaborative p2p search. *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 1263–1266, 2005.

- [6] M. Bianchini, M. Gori, and F. Scarselli. Inside pagerank. *ACM Trans. Inter. Tech.*, 5(1):92–128, 2005.
- [7] P. Boldi and S. Vigna. The Webgraph Framework I: Compression techniques. *Proceedings of the 13th international conference on World Wide Web*, pages 595–602, 2004.
- [8] S. Brin and L. Page. The anatomy of a large-scale hypertextual (web) search engine. *Computer Networks*, 30:107–117, 1998.
- [9] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1623–1640, 1999.
- [10] S. Chien, C. Dwork, R. Kumar, D.R. Simon, and D. Sivakumar. Link evolution: Analysis and algorithms. *Internet Mathematics*, 1(3):277–304, 2004.
- [11] J. Cho, H. García-Molina, and L. Page. Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.
- [12] G. Flake, S. Lawrence, and C. Lee Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160, Boston, MA, August 20–23 2000.
- [13] D. Fogaras, B. Rácz, Károly Csalogány, and Tamás Sarlós”. Towards scaling fully personalized pagerank: algorithms, lower bounds, and experiments. *Internet Mathematics*, 2005. To appear, preprint accessed from [http://webgui.jol.hu/~bracz/p/pubs/fogaras05ppr\\_ext.pdf](http://webgui.jol.hu/~bracz/p/pubs/fogaras05ppr_ext.pdf).
- [14] T. Haveliwala. Topic-sensitive PageRank. *Proceedings of the 11th international conference on World Wide Web*, pages 517–526, 2002.
- [15] G. Horton. On the multilevel solution algorithm for markov chains. Technical Report NASA CR-201671 ICASE Report No. 97-17, NASA Langley Research Center, March 1997.
- [16] G. Jeh and J. Widom. Scaling personalized web search. *Proceedings of the 12th international conference on World Wide Web*, pages 271–279, 2003.
- [17] 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):50–55, 2002.
- [18] J.Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. *Proceedings of SIGIR*, 2005.
- [19] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. *Stanford University Technical Report*, 2003.

- [20] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Extrapolation methods for accelerating pagerank computations. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 261–270, New York, NY, USA, 2003. ACM Press.
- [21] J. Kemeny and J.L. Snell. *Finite Markov Chains*. Springer Verlag, New York Berlin Heidelberg Tokyo, 1983.
- [22] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
- [23] A. Langville and C. Meyer. Updating pagerank with iterative aggregation. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 392–393, New York, NY, USA, 2004. ACM Press.
- [24] F. McSherry. A uniform approach to accelerated pagerank computation. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 575–582, New York, NY, USA, 2005. ACM Press.
- [25] P. Reynolds and A. Vadhat. Efficient peer-to-peer keyword searching. *Middleware*, 2003.
- [26] 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 retrieval. *International Workshop on the Web and Databases (WebDB)*, 2003.