Scaling Personalized Web Search

Glen Jeh
glenj@db.stanford.edu

Jennifer Widom
widom@db.stanford.edu

Abstract

Recent web search techniques augment traditional text matching with a global notion of "importance" based on the linkage structure of the web, such as in Google's *PageRank* algorithm. For more refined searches, this global notion of importance can be specialized to create personalized views of importance—for example, importance scores can be biased according to a user-specified set of initially-interesting pages. Computing and storing all possible personalized views in advance is impractical, as is computing personalized views at query time, since the computation of each view requires an iterative computation over the web graph. We present new graph-theoretical results, and a new technique based on these results, that encode personalized views as *partial vectors*. Partial vectors are shared across multiple personalized views, and their computation and storage costs scale well with the number of views. Our approach enables incremental computation, so that the construction of personalized views from partial vectors is practical at query time. We present efficient dynamic programming algorithms for computing partial vectors, an algorithm for constructing personalized views from partial vectors, and experimental results demonstrating the effectiveness and scalability of our techniques.

1 Introduction and Motivation

General web search is performed predominantly through text queries to search engines. Because of the enormous size of the web, text alone is usually not selective enough to limit the number of query results to a manageable size. The *PageRank* algorithm [10], among others [8], has been proposed (and implemented in *Google* [1]) to exploit the linkage structure of the web to compute global "importance" scores that can be used to influence the ranking of search results. To encompass different notions of importance for different users and queries, the basic PageRank algorithm can be modified to create "personalized views" of the web, redefining importance according to user preference. For example, a user may wish to specify his bookmarks as a set of preferred pages, so that any query results that are important with respect to his bookmarked pages would be ranked higher. While experimentation with the use of personalized PageRank has shown its utility and

This work was supported by the National Science Foundation under grant IIS-9817799.

promise [5, 10], the size of the web makes its practical realization extremely difficult. To see why, let us review the intuition behind the PageRank algorithm and its extension for personalization.

The fundamental motivation underlying PageRank is the recursive notion that important pages are those linked-to by many important pages. A page with only two in-links, for example, may seem unlikely to be an important page, but it may be important if the two referencing pages are *Yahoo!* and *Netscape*, which themselves are important pages because they have numerous in-links. One way to formalize this recursive notion is to use the "random surfer" model introduced in [10]. Imagine that trillions of *random surfers* are browsing the web: if at a certain time step a surfer is looking at page p, at the next time step he looks at a random out-neighbor of p. As time goes on, the expected percentage of surfers at each page p converges (under certain conditions) to a limit $r(p)$ that is independent of the distribution of starting points. Intuitively, this limit is the PageRank of p, and is taken to be an importance score for p, since it reflects the number of people expected to be looking at p at any one time.

The PageRank score $r(p)$ reflects a "democratic" importance that has no preference for any particular pages. In reality, a user may have a set P of preferred pages (such as his bookmarks) which he considers more interesting. We can account for preferred pages in the random surfer model by introducing a "teleportation" probability c: at each step, a surfer jumps back to a random page in P with probability c, and with probability $1 - c$ continues forth along a hyperlink. The limit distribution of surfers in this model would favor pages in P, pages linked-to by P, pages linked-to in turn, etc. We represent this distribution as a *personalized PageRank vector* (*PPV*) personalized on the set P. Informally, a PPV is a personalized view of the importance of pages on the web. Rankings of a user's text-based query results can be biased according to a PPV instead of the global importance distribution.

Each PPV is of length n, where n is the number of pages on the web. Computing a PPV naively using a fixed-point iteration requires multiple scans of the web graph [10], which makes it impossible to carry out online in response to a user query. On the other hand, PPV's for all preference sets, of which there are 2^n, is far too large to compute and store offline. We present a method for encoding PPV's as partially-computed, shared vectors that are practical to compute and store offline, and from which PPV's can be computed quickly at query time.

In our approach we restrict preference sets P to subsets of a set of *hub* pages H, selected as those of greater interest for personalization. In practice, we expect H to be a set of pages with high PageRank ("important pages"), pages in a human-constructed directory such as *Yahoo!* or *Open Directory* [2], or pages important to a particular enterprise or application. The size of H can be thought of as the available degree of personalization. We present algorithms that, unlike previous work [5, 10], scale well with the size of H. Moreover, the same techniques we introduce can yield approximations on the much broader set of all PPV's, allowing at least some level of personalization on arbitrary preference sets.

The main contributions of this paper are as follows.

- A method, based on new graph-theoretical results (listed next), of encoding PPV's as *partial quantities*, enabling an efficient, scalable computation that can be divided between precomputation time and query time, in a customized fashion according to available resources and application requirements.

- Three main theorems: The *Linearity Theorem* allows every PPV to be represented as a linear combination of *basis vectors*, yielding a natural way to construct PPV's from shared components. The *Hubs Theorem* allows basis vectors to be encoded as *partial vectors* and a *hubs skeleton*, enabling basis vectors themselves to be constructed from common components. The *Decomposition Theorem* establishes a linear relationship among basis vectors, which is exploited to minimize redundant computation.

- Several algorithms for computing basis vectors, specializations of these algorithms for computing partial vectors and the hubs skeleton, and an algorithm for constructing PPV's from partial vectors using the hubs skeleton.

- Experimental results on real web data demonstrating the effectiveness and scalability of our techniques.

In Section 2 we introduce the notation used in this paper and formalize personalized PageRank mathematically. Section 3 presents basis vectors, the first step towards encoding PPV's as shared components. The full encoding is presented in Section 4. Section 5 discusses the computation of partial quantities. Experimental results are presented in Section 6. Related work is discussed in Section 7. Section 8 summarizes the contributions of this paper. Additional material, primarily proofs of theorems, appears in a set of appendices.

2 Preliminaries

Let $G = (V, E)$ denote the *web graph*, where V is the set of all web pages and E contains a directed edge $\langle p, q \rangle$ iff page p links to page q. For a page p, we denote by $I(p)$ and $O(p)$ the set of in-neighbors and out-neighbors of p, respectively. Individual in-neighbors are denoted as $I_i(p)$ ($1 \leq i \leq |I(p)|$), and individual out-neighbors are denoted analogously. For convenience, pages are numbered from 1 to n, and we refer to a page p and its associated number i interchangeably. For a vector $\boldsymbol{v}$, $v(p)$ denotes *entry* p, the p-th component of $\boldsymbol{v}$. We always typeset vectors in boldface and scalars (e.g., $v(p)$) in normal font. All vectors in this paper are n-dimensional and have nonnegative entries. They should be thought of as distributions rather than arrows. The *magnitude* of a vector $\boldsymbol{v}$ is defined to be $\sum_{i=1}^{n} v(i)$ and is written $|\boldsymbol{v}|$. In this paper, vector magnitudes are always in $[0, 1]$.

In an implementation, a vector may be represented as a list of its nonzero entries, so another useful measure is the *size* of $\boldsymbol{v}$, the number of nonzero entries in $\boldsymbol{v}$.

We generalize the preference set P discussed in Section 1 to a *preference vector* $\boldsymbol{u}$, where $|\boldsymbol{u}| = 1$ and $u(p)$ denotes the amount of preference for page p. For example, a user who wants to personalize on his bookmarked pages P uniformly would have a $\boldsymbol{u}$ where $u(p) = \frac{1}{|P|}$ if $p \in P$, and $u(p) = 0$ if $p \notin P$. We formalize personalized PageRank scoring using matrix-vector equations. Let $\boldsymbol{A}$ be the matrix corresponding to the web graph G, where $A_{ij} = \frac{1}{|O(j)|}$ if page j links to page i, and $A_{ij} = 0$ otherwise. For simplicity of presentation, we assume that every page has at least one out-neighbor, as can be enforced by adding self-links to pages without out-links. The resulting scores can be adjusted to account for the (minor) effects of this modification, as specified in Appendix C.2.

For a given $\boldsymbol{u}$, the personalized PageRank equation can be written as

$$\boldsymbol{v} = (1-c)\boldsymbol{A}\boldsymbol{v} + c\boldsymbol{u} \tag{1}$$

where $c \in (0, 1)$ is the "teleportation" constant discussed in Section 1. Typically $c \approx 0.15$, and experiments have shown that small changes in c have little effect in practice [10]. A solution $\boldsymbol{v}$ to equation (1) is a steady-state distribution of random surfers under the model discussed in Section 1, where at each step a surfer teleports to page p with probability $c \cdot u(p)$, or moves to a random out-neighbor otherwise [10]. By a theorem of Markov Theory, a solution $\boldsymbol{v}$ with $|\boldsymbol{v}| = 1$ always exists and is unique [9].[1] The solution $\boldsymbol{v}$ is the *personalized PageRank vector* (*PPV*) for preference vector $\boldsymbol{u}$. If $\boldsymbol{u}$ is the uniform distribution vector $\boldsymbol{u} = [1/n, \ldots, 1/n]$, then the corresponding solution $\boldsymbol{v}$ is the *global PageRank vector* [10], which gives no preference to any pages.

For the reader's convenience, Table 1 on the next page lists terminology that will be used extensively in the coming sections.

3 Basis Vectors

We present the first step towards encoding PPV's as shared components. The motivation behind the encoding is a simple observation about the linearity[2] of PPV's, formalized by the following theorem.

Theorem (Linearity). *For any preference vectors $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$, if $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are the two corresponding PPV's, then for any constants $\alpha_1, \alpha_2 \geq 0$ such that $\alpha_1 + \alpha_2 = 1$,*

$$\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} = (1-c)\boldsymbol{A}(\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2}) + c(\alpha_1 \boldsymbol{u_1} + \alpha_2 \boldsymbol{u_2}) \tag{2}$$

[1] Specifically, $\boldsymbol{v}$ corresponds to the steady-state distribution of an *ergodic*, *aperiodic Markov chain*.

[2] More precisely, the transformation from personalization vectors $\boldsymbol{u}$ to their corresponding solution vectors $\boldsymbol{v}$ is linear.

Term	Description	Section
Hub Set H	A subset of web pages.	1
Preference Set P	Set of pages on which to personalize (restricted in this paper to subsets of H).	1
Preference Vector $\boldsymbol{u}$	Preference set with weights.	2
Personalized PageRank Vector (PPV)	Importance distribution induced by a preference vector.	2
Basis Vector $\boldsymbol{r_p}$ (or $\boldsymbol{r_i}$)	PPV for a preference vector with a single nonzero entry at p (or i).	3
Hub Vector $\boldsymbol{r_p}$	Basis vector for a hub page $p \in H$.	3
Partial Vector $(\boldsymbol{r_p} - \boldsymbol{r_p^H})$	Used with the hubs skeleton to construct a hub vector.	4.2
Hubs Skeleton S	Used with partial vectors to construct a hub vector.	4.3
Web Skeleton	Extension of the hubs skeleton to include pages not in H.	4.4.3
Partial Quantities	Partial vectors and the hubs, web skeletons.	
Intermediate Results	Maintained during iterative computations.	5.2

Table 1: Summary of terms.

Informally, the Linearity Theorem says that the solution to a linear combination of preference vectors $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ is the same linear combination of the corresponding PPV's $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$. The proof is in Appendix A.

Let $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ be the unit vectors in each dimension, so that for each i, $\boldsymbol{x_i}$ has value 1 at entry i and 0 everywhere else. Let $\boldsymbol{r_i}$ be the PPV corresponding to $\boldsymbol{x_i}$. Each *basis vector* $\boldsymbol{r_i}$ gives the distribution of random surfers under the model that at each step, surfers teleport back to page i with probability c. It can be thought of as representing page i's view of the web, where entry j of $\boldsymbol{r_i}$ is j's importance in i's view. Note that the global PageRank vector is $\frac{1}{n}(\boldsymbol{r_1} + \cdots + \boldsymbol{r_n})$, the average of every page's view.

An arbitrary personalization vector $\boldsymbol{u}$ can be written as a weighted sum of the unit vectors $\boldsymbol{x_i}$:

$$\boldsymbol{u} = \sum_{i=1}^{n} \alpha_i \boldsymbol{x_i} \tag{3}$$

for some constants $\alpha_1, \ldots, \alpha_n$. By the Linearity Theorem,

$$\boldsymbol{v} = \sum_{i=1}^{n} \alpha_i \boldsymbol{r_i} \tag{4}$$

is the corresponding PPV, expressed as a linear combination of the basis vectors $\boldsymbol{r_i}$.

Recall from Section 1 that preference sets (now preference vectors) are restricted to subsets of a set of hub pages H. If a *basis hub vector* (or hereafter *hub vector*) for each $p \in H$ were computed and stored, then any PPV corresponding to a preference set P of size k (a preference vector with k nonzero entries) can be computed by adding up the k corresponding hub vectors $\boldsymbol{r_p}$ with the appropriate weights α_p.

Each hub vector can be computed naively using the fixed-point computation in [10]. However, each fixed-point computation is expensive, requiring multiple scans of the web graph, and the computation time (as well as storage cost) grows linearly with the number of hub vectors $|H|$. In the next section, we enable a more scalable computation by constructing hub vectors from shared components.

4 Decomposition of Basis Vectors

In Section 3 we represented PPV's as a linear combination of $|H|$ hub vectors $\boldsymbol{r_p}$, one for each $p \in H$. Any PPV based on hub pages can be constructed quickly from the set of precomputed hub vectors, but computing and storing all hub vectors is impractical. To compute a large number of hub vectors efficiently, we further decompose them into *partial vectors* and the *hubs skeleton*, components from which hub vectors can be constructed quickly at query time. The representation of hub vectors as partial vectors and the hubs skeleton saves both computation time and storage due to sharing of components among hub vectors. Note, however, that depending on available resources and application requirements, hub vectors can be constructed offline as well. Thus "query time" can be thought of more generally as "construction time".

We compute one partial vector for each hub page p, which essentially encodes the part of the hub vector $\boldsymbol{r_p}$ unique to p, so that components shared among hub vectors are not computed and stored redundantly. The complement to the partial vectors is the hubs skeleton, which succinctly captures the interrelationships among hub vectors. It is the "blueprint" by which partial vectors are assembled to form a hub vector, as we will see in Section 4.3.

The mathematical tools used in the formalization of this decomposition are presented next.[3]

4.1 Inverse P-distance

To formalize the relationship among hub vectors, we relate the personalized PageRank scores represented by PPV's to *inverse P-distances* in the web graph, a concept based on *expected-f distances* as introduced in [7].

[3]Note that while the mathematics and computation strategies in this paper are presented in the specific context of the web graph, they are general graph-theoretical results that may be applicable in other scenarios involving stochastic processes, of which PageRank is one example.

Let $p, q \in V$. We define the *inverse P-distance* $r'_p(q)$ from p to q as

$$r'_p(q) = \sum_{t:p \rightsquigarrow q} P[t]c(1-c)^{l(t)} \tag{5}$$

where the summation is taken over all *tours* t (paths that may contain cycles) starting at p and ending at q, possibly touching p or q multiple times. For a tour $t = \langle w_1, \ldots, w_k \rangle$, the length $l(t)$ is $k-1$, the number of edges in t. The term $P[t]$, which should be interpreted as "the probability of traveling t", is defined as $\prod_{i=1}^{k-1} \frac{1}{|O(w_i)|}$, or 1 if $l(t) = 0$. If there is no tour from p to q, the summation is taken to be 0.[4] Note that $r'_p(q)$ measures distances inversely: it is higher for nodes q "closer" to p. As suggested by the notation and proven in Appendix C, $r'_p(q) = r_p(q)$ for all $p, q \in V$, so we will use $r_p(q)$ to denote both the inverse P-distance and the personalized PageRank score. Thus PageRank scores can be viewed as an inverse measure of distance.

Let $H \subseteq V$ be some nonempty set of pages. For $p, q \in V$, we define $r_p^H(q)$ as a restriction of $r_p(q)$ that considers only tours which pass through some page $h \in H$ in equation (5). That is, a page $h \in H$ must occur on t somewhere other than the endpoints. Precisely, $r_p^H(q)$ is written as

$$r_p^H(q) = \sum_{t:p \rightsquigarrow H \rightsquigarrow q} P[t]c(1-c)^{l(t)} \tag{6}$$

where the notation $t : p \rightsquigarrow H \rightsquigarrow q$ reminds us that t passes through some page in H. Note that t must be of length at least 2. In this paper, H is always the set of hub pages, and p is usually a hub page (until we discuss the web skeleton in Section 4.4.3).

4.2 Partial Vectors

Intuitively, $r_p^H(q)$, defined in equation (6), is the influence of p on q through H. In particular, if all paths from p to q pass through a page in H, then H separates p and q, and $r_p^H(q) = r_p(q)$. For well-chosen sets H (discussed in Section 4.4.2), it will be true that $r_p(q) - r_p^H(q) = 0$ for many pages p, q. Our strategy is to take advantage of this property by breaking $\boldsymbol{r_p}$ into two components: $(\boldsymbol{r_p} - \boldsymbol{r_p^H})$ and $\boldsymbol{r_p^H}$, using the equation

$$\boldsymbol{r_p} = (\boldsymbol{r_p} - \boldsymbol{r_p^H}) + \boldsymbol{r_p^H} \tag{7}$$

We first precompute and store the *partial vector* $(\boldsymbol{r_p} - \boldsymbol{r_p^H})$ instead of the full hub vector $\boldsymbol{r_p}$. Partial vectors are cheaper to compute and store than full hub vectors, assuming they are represented as a list of their nonzero entries. Moreover, the size of each partial vector decreases as $|H|$ increases, making this approach particularly scalable. We then add $\boldsymbol{r_p^H}$ back at query time to compute the full hub vector. However, computing and storing $\boldsymbol{r_p^H}$ explicitly could be as expensive as $\boldsymbol{r_p}$ itself. In the next section we show how to encode $\boldsymbol{r_p^H}$ so it can be computed and stored efficiently.

[4]The definition here of inverse P-distance differs slightly from the concept of expected-f distance in [7], where tours are not allowed to visit q multiple times. Note that general expected-f distances have the form $\sum_t P[t]f(l(t))$; in our definition, $f(x) = c(1-c)^x$.

4.3 Hubs Skeleton

Let us briefly review where we are: In Section 3 we represented PPV's as linear combinations of hub vectors $\boldsymbol{r_p}$, one for each $p \in H$, so that we can construct PPV's quickly at query time if we have precomputed the hub vectors, a relatively small subset of PPV's. To encode hub vectors efficiently, in Section 4.2 we said that instead of full hub vectors $\boldsymbol{r_p}$, we first compute and store only partial vectors $(\boldsymbol{r_p} - \boldsymbol{r_p^H})$, which intuitively account only for paths that do not pass through a page of H (i.e., the distribution is "blocked" by H). Computing and storing the difference vector $\boldsymbol{r_p^H}$ efficiently is the topic of this section.

It turns out that the vector $\boldsymbol{r_p^H}$ can be be expressed in terms of the partial vectors $(\boldsymbol{r_h} - \boldsymbol{r_h^H})$, for $h \in H$, as shown by the following theorem. Recall from Section 3 that $\boldsymbol{x_h}$ has value 1 at h and 0 everywhere else.

Theorem (Hubs). *For any $p \in V$, $H \subseteq V$,*

$$\boldsymbol{r_p^H} = \frac{1}{c} \sum_{h \in H} (r_p(h) - c x_p(h)) \left(\boldsymbol{r_h} - \boldsymbol{r_h^H} - c\boldsymbol{x_h}\right) \tag{8}$$

In terms of inverse P-distances (Section 4.1), the Hubs Theorem says roughly that the distance from page p to any page $q \in V$ through H is the distance $r_p(h)$ from p to each $h \in H$ times the distance $r_h(q)$ from h to q, correcting for the paths among hubs by $r_h^H(q)$. The terms $cx_p(h)$ and $c\boldsymbol{x_h}$ deal with the special cases when p or q is itself in H. The proof, which is quite involved, is in Appendix D.

The quantity $\left(\boldsymbol{r_h} - \boldsymbol{r_h^H}\right)$ appearing on the right-hand side of (8) is exactly the partial vectors discussed in Section 4.2. Suppose we have computed $r_p(H) = \{(h, r_p(h)) \,|\, h \in H\}$ for a hub page p. Substituting the Hubs Theorem into equation 7, we have the following *Hubs Equation* for constructing the hub vector $\boldsymbol{r_p}$ from partial vectors:

$$\boldsymbol{r_p} = (\boldsymbol{r_p} - \boldsymbol{r_p^H}) + \frac{1}{c} \sum_{h \in H} (r_p(h) - c x_p(h)) \left[\left(\boldsymbol{r_h} - \boldsymbol{r_h^H}\right) - c\boldsymbol{x_h}\right] \tag{9}$$

This equation is central to the construction of hub vectors from partial vectors.

The set $r_p(H)$ has size at most $|H|$, much smaller than the full hub vector $\boldsymbol{r_p}$, which can have up to n nonzero entries. Furthermore, the contribution of each entry $r_p(h)$ to the sum is no greater than $r_p(h)$ (and usually much smaller), so that small values of $r_p(h)$ can be omitted with minimal loss of precision (Section 6). The set $S = \{r_p(H) \,|\, p \in H\}$ forms the *hubs skeleton*, giving the interrelationships among partial vectors.

An intuitive view of the encoding and construction suggested by the Hubs Equation (9) is shown in Figure 1. At the top, each partial vector $(\boldsymbol{r_h} - \boldsymbol{r_h^H})$, including $(\boldsymbol{r_p} - \boldsymbol{r_p^H})$, is depicted as a notched triangle labeled h at the tip. The triangle can be thought of as representing paths starting at h, although, more accurately, it represents the distribution of importance scores computed based

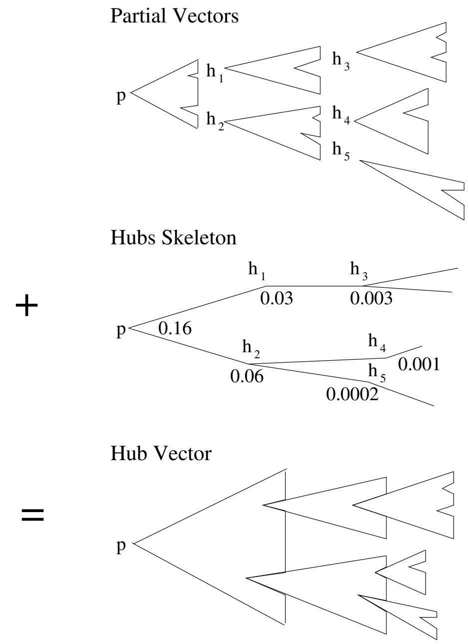

Figure 1: Intuitive view of the construction of hub vectors from partial vectors and the hubs skeleton.

on the paths, as discussed in Section 4.1. A notch in the triangle shows where the computation of a partial vector "stopped" at another hub page. At the center, a part $r_p(H)$ of the hubs skeleton is depicted as a tree so the "assembly" of the hub vector can be visualized. The hub vector is constructed by logically assembling the partial vectors using the corresponding weights in the hubs skeleton, as shown at the bottom.

4.4 Discussion

4.4.1 Summary

In summary, hub vectors are building blocks for PPV's corresponding to preference vectors based on hub pages. Partial vectors, together with the hubs skeleton, are building blocks for hub vectors. Transitively, partial vectors and the hubs skeleton are building blocks for PPV's: they can be used to construct PPV's without first materializing hub vectors as an intermediate step (Section 5.4). Note that for preference vectors based on multiple hub pages, constructing the corresponding PPV from partial vectors directly can result in significant savings versus constructing from hub vectors, since partial vectors are shared across multiple hub vectors.

4.4.2 Choice of H

So far we have made no assumptions about the set of hub pages H. Not surprisingly, the choice of hub pages can have a significant impact on performance, depending on the location of hub pages within the overall graph structure. In particular, the size of partial vectors is smaller when pages in H have higher PageRank, since high-PageRank pages are on average close to other pages in terms of inverse P-distance (Section 4.1), and the size of the partial vectors is related to the inverse P-distance between hub pages and other pages according to the Hubs Theorem. Our intuition is that high-PageRank pages are generally more interesting for personalization anyway, but in cases where the intended hub pages do not have high PageRank, it may be beneficial to include some high-PageRank pages in H to improve performance. We ran experiments confirming that the size of partial vectors is much smaller using high-PageRank pages as hubs than using random pages.

4.4.3 Web Skeleton

The techniques used in the construction of hub vectors can be extended to enable at least approximate personalization on arbitrary preference vectors that are not necessarily based on H. Suppose we want to personalize on a page $p \notin H$. The Hubs Equation can be used to construct $\boldsymbol{r_p^H}$ from partial vectors, given that we have computed $r_p(H)$. As discussed in Section 4.3, the cost of computing and storing $r_p(H)$ is orders of magnitude less than $\boldsymbol{r_p}$. Though $\boldsymbol{r_p^H}$ is only an approximation to $\boldsymbol{r_p}$, it may still capture significant personalization information for a properly-chosen hub set H, as $\boldsymbol{r_p^H}$ can be thought of as a "projection" of $\boldsymbol{r_p}$ onto H. For example, if H contains pages from *Open Directory*, $\boldsymbol{r_p^H}$ can capture information about the broad topic of $\boldsymbol{r_p}$. Exploring the utility of the *web skeleton* $W = \{r_p(H) \,|\, p \in V\}$ is an area of future work.

5 Computation

In Section 4 we presented a way to construct hub vectors from partial vectors ($\boldsymbol{r_p} - \boldsymbol{r_p^H}$), for $p \in H$, and the hubs skeleton $S = \{r_p(H) \,|\, p \in H\}$. We also discussed the web skeleton $W = \{r_p(H) \,|\, p \in V\}$. Computing these *partial quantities* naively using a fixed-point iteration [10] for each p would scale poorly with the number of hub pages. Here we present scalable algorithms that compute these quantities efficiently by using dynamic programming to leverage the interrelationships among them. We also show how PPV's can be constructed from partial vectors and the hubs skeleton at query time. All of our algorithms have the property that they can be stopped at any time (e.g., when resources are depleted), so that the current "best results" can be used as an approximation, or the computation can be resumed later for increased precision if resources permit.

We begin in Section 5.1 by presenting a theorem underlying all of the algorithms presented (as well as the connection between PageRank and inverse P-distance, as shown in Appendix C). In Section 5.2, we present three algorithms, based on this theorem, for computing general basis vectors. The algorithms in Section 5.2 are not meant to be deployed, but are used as foundations for the algorithms in Section 5.3 for computing partial quantities. Section 5.4 discusses the construction of PPV's from partial vectors and the hubs skeleton.

5.1 Decomposition Theorem

Recall the random surfer model of Section 1, instantiated for preference vector $\boldsymbol{u} = \boldsymbol{x_p}$ (for page p's view of the web). At each step, a surfer s teleports to page p with some probability c. If s is at p, then at the next step, s with probability $1-c$ will be at a random out-neighbor of p. That is, a fraction $(1-c)\frac{1}{|O(p)|}$ of the time, surfer s will be at any given out-neighbor of p one step after teleporting to p. This behavior is strikingly similar to the model instantiated for preference vector $\boldsymbol{u'} = \frac{1}{|O(p)|}\sum_{i=1}^{|O(p)|} \boldsymbol{x_{O_i(p)}}$, where surfers teleport directly to each $O_i(p)$ with equal probability $\frac{1}{|O(p)|}$. The similarity is formalized by the following theorem.

Theorem (Decomposition). *For any $p \in V$,*

$$\boldsymbol{r_p} = \frac{(1-c)}{|O(p)|}\sum_{i=1}^{|O(p)|} \boldsymbol{r_{O_i(p)}} + c\boldsymbol{x_p} \tag{10}$$

The Decomposition Theorem says that the basis vector $\boldsymbol{r_p}$ for p is an average of the basis vectors $\boldsymbol{r_{O_i(p)}}$ for its out-neighbors, plus a compensation factor $c\boldsymbol{x_p}$. The proof is in Appendix B.

The Decomposition Theorem gives another way to think about PPV's. It says that p's view of the web ($\boldsymbol{r_p}$) is the average of the views of its out-neighbors, but with extra importance given to p itself. That is, pages important in p's view are either p itself, or pages important in the view of p's out-neighbors, which are themselves "endorsed" by p. In fact, this recursive intuition yields an equivalent way of formalizing personalized PageRank scoring: basis vectors can be defined as vectors satisfying the Decomposition Theorem.

While the Decomposition Theorem identifies relationships among basis vectors, a division of the computation of a basis vector $\boldsymbol{r_p}$ into related subproblems for dynamic programming is not inherent in the relationships. For example, it is possible to compute some basis vectors first and then to compute the rest using the former as solved subproblems. However, the presence of cycles in the graph makes this approach ineffective. Instead, our approach is to consider as a subproblem the computation of a vector to less precision. For example, having computed $\boldsymbol{r_{O_i(p)}}$ to a certain precision, we can use the Decomposition Theorem to combine the $\boldsymbol{r_{O_i(p)}}$'s to compute $\boldsymbol{r_p}$ to greater precision. This approach has the advantage that precision needs not be fixed in advance: the process can be stopped at any time for the current best answer.

5.2 Algorithms for Computing Basis Vectors

We present three algorithms in the general context of computing full basis vectors. These algorithms are presented primarily to develop our algorithms for computing partial quantities, presented in Section 5.3. All three algorithms are iterative fixed-point computations that maintain a set of *intermediate results* $(\boldsymbol{D_k}[*], \boldsymbol{E_k}[*])$. For each p, $\boldsymbol{D_k}[\boldsymbol{p}]$ is a lower-approximation of $\boldsymbol{r_p}$ on iteration k, i.e., $D_k[p](q) \leq r_p(q)$ for all $q \in V$. We build solutions $\boldsymbol{D_k}[\boldsymbol{p}]$ $(k = 0, 1, 2, \ldots)$ that are successively better approximations to $\boldsymbol{r_p}$, and simultaneously compute the error components $\boldsymbol{E_k}[\boldsymbol{p}]$, where $\boldsymbol{E_k}[\boldsymbol{p}]$ is the "projection" of the vector $(\boldsymbol{r_p} - \boldsymbol{D_k}[\boldsymbol{p}])$ onto the (actual) basis vectors. That is, we maintain the invariant that for all $k \geq 0$ and all $p \in V$,

$$\boldsymbol{D_k}[\boldsymbol{p}] + \sum_{q \in V} E_k[p](q)\boldsymbol{r_q} = \boldsymbol{r_p} \tag{11}$$

Thus, $\boldsymbol{D_k}[\boldsymbol{p}]$ is a lower-approximation of $\boldsymbol{r_p}$ with error $\left|\sum_{q \in V} E_k[p](q)\boldsymbol{r_q}\right| = |\boldsymbol{E_k}[\boldsymbol{p}]|$. We begin with $\boldsymbol{D_0}[\boldsymbol{p}] = \boldsymbol{0}$ and $\boldsymbol{E_0}[\boldsymbol{p}] = \boldsymbol{x_p}$, so that logically, the approximation is initially $\boldsymbol{0}$ and the error is $\boldsymbol{r_p}$. To store $\boldsymbol{E_k}[\boldsymbol{p}]$ and $\boldsymbol{D_k}[\boldsymbol{p}]$ efficiently, we can represent them in an implementation as a list of their nonzero entries. While all three algorithms have in common the use of these intermediate results, they differ in how they use the Decomposition Theorem to refine intermediate results on successive iterations.

It is important to note that the algorithms presented in this section and their derivatives in Section 5.3 compute vectors to arbitrary precision; they are not approximations. In practice, the precision desired may vary depending on the application. Our focus is on algorithms that are efficient and scalable with the number of hub vectors, regardless of the precision to which vectors are computed.

5.2.1 Basic Dynamic Programming Algorithm

In the *basic dynamic programming algorithm*, a new basis vector for each page p is computed on each iteration using the vectors computed for p's out-neighbors on the previous iteration, via the Decomposition Theorem. On iteration k, we derive $(\boldsymbol{D_{k+1}}[\boldsymbol{p}], \boldsymbol{E_{k+1}}[\boldsymbol{p}])$ from $(\boldsymbol{D_k}[\boldsymbol{p}], \boldsymbol{E_k}[\boldsymbol{p}])$ using the equations:

$$\boldsymbol{D_{k+1}}[\boldsymbol{p}] = \frac{1-c}{|O(a)|} \sum_{i=1}^{|O(p)|} \boldsymbol{D_k}[\boldsymbol{O_i(p)}] + c\boldsymbol{x_p} \tag{12}$$

$$\boldsymbol{E_{k+1}}[\boldsymbol{p}] = \frac{1-c}{|O(a)|} \sum_{i=1}^{|O(p)|} \boldsymbol{E_k}[\boldsymbol{O_i(p)}] \tag{13}$$

A proof of the algorithm's correctness is given in Appendix E, where the error $|\boldsymbol{E_k}[\boldsymbol{p}]|$ is shown to be reduced by a factor of $1 - c$ on each iteration.

Note that although the $\boldsymbol{E_k}[*]$ values help us to see the correctness of the algorithm, they are not used here in the computation of $\boldsymbol{D_k}[*]$ and can be omitted in an implementation (although they will be used to compute partial quantities in Section 5.3). The sizes of $\boldsymbol{D_k}[\boldsymbol{p}]$ and $\boldsymbol{E_k}[\boldsymbol{p}]$ grow with the number of iterations, and in the limit they can be up to the size of $\boldsymbol{r_p}$, which is the number of pages reachable from p. Intermediate scores $(\boldsymbol{D_k}[*], \boldsymbol{E_k}[*])$ will likely be much larger than available main memory, and in an implementation $(\boldsymbol{D_k}[*], \boldsymbol{E_k}[*])$ could be read off disk and $(\boldsymbol{D_{k+1}}[*], \boldsymbol{E_{k+1}}[*])$ written to disk on each iteration. When the data for one iteration has been computed, data from the previous iteration may be deleted. Specific details of our implementation are discussed in Section 6.

5.2.2 Selective Expansion Algorithm

The *selective expansion algorithm* is essentially a version of the naive algorithm that can readily be modified to compute partial vectors, as we will see in Section 5.3.1.

We derive $(\boldsymbol{D_{k+1}}[\boldsymbol{p}], \boldsymbol{E_{k+1}}[\boldsymbol{p}])$ by "distributing" the error at each page q (that is, $E_k[p](q)$) to its out-neighbors via the Decomposition Theorem. Precisely, we compute results on iteration-k using the equations:

$$\boldsymbol{D_{k+1}}[\boldsymbol{p}] = \boldsymbol{D_k}[\boldsymbol{p}] + \sum_{q \in Q_k(p)} c E_k[p](q) \boldsymbol{x_q} \tag{14}$$

$$\boldsymbol{E_{k+1}}[\boldsymbol{p}] = \boldsymbol{E_k}[\boldsymbol{p}] - \sum_{q \in Q_k(p)} E_k[p](q) \boldsymbol{x_q} + \sum_{q \in Q_k(p)} \frac{1-c}{|O(q)|} \sum_{i=1}^{|O(q)|} E_k[p](q) \boldsymbol{x_{O_i(q)}} \tag{15}$$

for a subset $Q_k(p) \subseteq V$. If $Q_k(p) = V$ for all k, then the error is reduced by a factor of $1-c$ on each iteration, as in the basic dynamic programming algorithm. However, it is often useful to choose a selected subset of V as $Q_k(p)$. For example, if $Q_k(p)$ contains the m pages q for which the error $E_k[p](q)$ is highest, then this *top-m* scheme limits the number of expansions and delays the growth in size of the intermediate results while still reducing much of the error. In Section 5.3.1, we will compute the hub vectors by choosing $Q_k(p) = H$. The correctness of selective expansion is proven in Appendix F.

5.2.3 Repeated Squaring Algorithm

The *repeated squaring algorithm* is similar to the selective expansion algorithm, except that instead of extending $(\boldsymbol{D_{k+1}}[*], \boldsymbol{E_{k+1}}[*])$ one step using equations (14) and (15), we compute what are essentially iteration-$2k$ results using the equations

$$\boldsymbol{D_{2k}[p]} = \boldsymbol{D_k[p]} + \sum_{q \in Q_k(p)} E_k[p](q)\boldsymbol{D_k[q]} \tag{16}$$

$$\boldsymbol{E_{2k}[p]} = \boldsymbol{E_k[p]} - \sum_{q \in Q_k(p)} E_k[p](q)\boldsymbol{x_q} + \sum_{q \in Q_k(p)} E_k[p](q)\boldsymbol{E_k[q]} \tag{17}$$

where $Q_k(p) \subseteq V$. For now we can assume that $Q_k(p) = V$ for all p; we will set $Q_k(p) = H$ to compute the hubs skeleton in Section 5.3.2. The correctness of these equations is proven in Appendix G, where it is shown that repeated squaring reduces the error much faster than the basic dynamic programming or selective expansion algorithms. If $Q_k(p) = V$, the error is squared on each iteration, as equation (17) reduces to:

$$\boldsymbol{E_{2k}[p]} = \sum_{q \in V} E_k[p](q)\boldsymbol{E_k[q]} \tag{18}$$

As an alternative to taking $Q_k(p) = V$, we can also use the top-m scheme of Section 5.2.2.

Note that while all three algorithms presented can be used to compute the set of all basis vectors, they differ in their requirements on the computation of other vectors when computing $\boldsymbol{r_p}$: the basic dynamic programming algorithm requires the vectors of out-neighbors of p to be computed as well, repeated squaring requires results $(\boldsymbol{D_k[q]}, \boldsymbol{E_k[q]})$ to be computed for q such that $E_k[p](q) > 0$, and selective expansion computes $\boldsymbol{r_p}$ independently.

5.3 Computing Partial Quantities

In Section 5.2 we presented iterative algorithms for computing full basis vectors to arbitrary precision. Here we present modifications to these algorithms to compute the partial quantities:

- Partial vectors $(\boldsymbol{r_p} - \boldsymbol{r_p^H})$, $p \in H$.
- The hubs skeleton $S = \{r_p(H) \mid p \in H\}$ (which can be computed more efficiently by itself than as part of the entire web skeleton).
- The web skeleton $W = \{r_p(H) \mid p \in V\}$.

Each partial quantity can be computed in time no greater than its size, which is far less than the size of the hub vectors.

5.3.1 Partial Vectors

Partial vectors can be computed using a simple specialization of the selective expansion algorithm (Section 5.2.2): we take $Q_0(p) = V$ and $Q_k(p) = V - H$ for $k > 0$, for all $p \in V$. That is, we never "expand" hub pages after the first step, so tours passing through a hub page H are never considered. Under this choice of $Q_k(p)$, $\boldsymbol{D_k[p]} + c\boldsymbol{E_k[p]}$ converges to $(\boldsymbol{r_p} - \boldsymbol{r_p^H})$ for all $p \in V$.

Of course, only the intermediate results $(\boldsymbol{D_k}[\boldsymbol{p}], \boldsymbol{E_k}[\boldsymbol{p}])$ for $p \in H$ should be computed. A proof is presented in Appendix H.

This algorithm makes it clear why using high-PageRank pages as hub pages improves performance: from a page p we expect to reach a high-PageRank page q sooner than a random page, so the expansion from p will stop sooner and result in a shorter partial vector.

5.3.2 Hubs Skeleton

While the hubs skeleton is a subset of the complete web skeleton and can be computed as such using the technique to be presented in Section 5.3.3, it can be computed much faster by itself if we are not interested in the entire web skeleton, or if higher precision is desired for the hubs skeleton than can be computed for the entire web skeleton.

We use a specialization of the repeated squaring algorithm (Section 5.2.3) to compute the hubs skeleton, using the intermediate results from the computation of partial vectors. Suppose $(\boldsymbol{D_k}[\boldsymbol{p}], \boldsymbol{E_k}[\boldsymbol{p}])$, for $k \geq 1$, have been computed by the algorithm of Section 5.3.1, so that $\sum_{q \notin H} E_k[p](q) < \epsilon$, for some error ϵ. We apply the repeated squaring algorithm on these results using $Q_k(p) = H$ for all successive iterations. As shown in Appendix I, after i iterations of repeated squaring, the total error $|\boldsymbol{E_i}[\boldsymbol{p}]|$ is bounded by $(1-c)^{2^i} + \epsilon/c$. Thus, by varying k and i, $r_p(H)$ can be computed to arbitrary precision.

Notice that only the intermediate results $(\boldsymbol{D_k}[\boldsymbol{h}], \boldsymbol{E_k}[\boldsymbol{h}])$ for $h \in H$ are ever needed to update scores for $\boldsymbol{D_k}[\boldsymbol{p}]$, and of the former, only the entries $D_k[h](q), E_k[h](q)$, for $q \in H$, are used to compute $D_k[p](q)$. Since we are only interested in the hub scores $D_k[p](q)$, we can simply drop all non-hub entries from the intermediate results. The running time and storage would then depend only on the size of $r_p(H)$ and not on the length of the entire hub vectors $\boldsymbol{r_p}$. If the restricted intermediate results fit in main memory, it is possible to defer the computation of the hubs skeleton to query time.

5.3.3 Web Skeleton

To compute the entire web skeleton, we modify the basic dynamic programming algorithm (Section 5.2.1) to compute only the hub scores $r_p(H)$, with corresponding savings in time and memory usage. We restrict the computation by eliminating entries $q \notin H$ from the intermediate results $(\boldsymbol{D_k}[\boldsymbol{p}], \boldsymbol{E_k}[\boldsymbol{p}])$, similar to the technique used in computing the hubs skeleton.

The justification for this modification is that the hub score $D_{k+1}[p](h)$ is affected only by the hub scores $D_k[*](h)$ of the previous iteration, so that $D_{k+1}[p](h)$ in the modified algorithm is equal to that in the basic algorithm. Since $|H|$ is likely to be orders of magnitude less than n, the size of the intermediate results is reduced significantly.

5.4 Construction of PPV's

Finally, let us see how a PPV for preference vector $\boldsymbol{u}$ can be constructed directly from partial vectors and the hubs skeleton using the Hubs Equation. (Construction of a single hub vector is a specialization of the algorithm outlined here.) Let $\boldsymbol{u} = \alpha_1 p_1 + \cdots + \alpha_z p_z$ be a preference vector, where $p_i \in H$ for $1 \leq i \leq z$. Let $Q \subseteq H$, and let

$$r_u(h) = \sum_{i=1}^{z} \alpha_i \left(r_{p_i}(h) - c x_{p_i}(h) \right) \tag{19}$$

which can be computed from the hubs skeleton. Then the PPV $\boldsymbol{v}$ for $\boldsymbol{u}$ can be constructed as

$$\boldsymbol{v} = \sum_{i=1}^{z} \alpha_i (\boldsymbol{r_{p_i}} - \boldsymbol{r_{p_i}^H}) + \frac{1}{c} \sum_{\substack{h \in Q \\ r_u(h) > 0}} r_u(h) \left[(\boldsymbol{r_h} - \boldsymbol{r_h^H}) - c \boldsymbol{x_h} \right] \tag{20}$$

Both the terms $(\boldsymbol{r_{p_i}} - \boldsymbol{r_{p_i}^H})$ and $(\boldsymbol{r_h} - \boldsymbol{r_h^H})$ are partial vectors, which we assume have been precomputed. The term $c\boldsymbol{x_h}$ represents a simple subtraction from $(\boldsymbol{r_h} - \boldsymbol{r_h^H})$. If $Q = H$, then (20) represents a full construction of $\boldsymbol{v}$. However, for some applications, it may suffice to use only parts of the hubs skeleton to compute $\boldsymbol{v}$ to less precision. For example, we can take Q to be the m hubs h for which $r_u(h)$ is highest. Experimentation with this scheme is discussed in Section 6.3. Alternatively, the result can be improved incrementally (e.g., as time permits) by using a small subset Q each time and accumulating the results.

6 Experiments

We performed experiments using real web data from Stanford's *WebBase* [6], a crawl of the web containing 120 million pages. Since the iterative computation of PageRank is unaffected by *leaf pages* (i.e., those with no out-neighbors), they can be removed from the graph and added back in after the computation [10]. After removing leaf pages, the graph consisted of 80 million pages

Both the web graph and the intermediate results $(\boldsymbol{D_k}[*], \boldsymbol{E_k}[*])$ were too large to fit in main memory, and a partitioning strategy, based on that presented in [4], was used to divide the computation into portions that can be carried out in memory. Specifically, the set of pages V was partitioned into k arbitrary sets $P_1, \ldots, P_k$ of equal size ($k = 10$ in our experiments). The web graph, represented as an edge-list E, is partitioned into k chunks E_i ($1 \leq i \leq k$), where E_i contains all edges $\langle p, q \rangle$ for which $p \in P_i$. Intermediate results $\boldsymbol{D_k}[\boldsymbol{p}]$ and $\boldsymbol{E_k}[\boldsymbol{p}]$ were represented together as a list $\boldsymbol{L_k}[\boldsymbol{p}] = \langle (q_1, d_1, e_1), (q_2, d_2, e_2), \ldots \rangle$ where $D_k[p](q_z) = d_z$ and $E_k[p](q_z) = e_z$, for $z = 1, 2, \ldots$. Only pages q_z for which either $d_z > 0$ or $e_z > 0$ were included. The set of intermediate results $\boldsymbol{L_k}[*]$ was partitioned into k^2 chunks $\boldsymbol{L_k^{i,j}}[*]$, so that $\boldsymbol{L_k^{i,j}}[\boldsymbol{p}]$ contains triples (q_z, d_z, e_z) of $\boldsymbol{L_k}[\boldsymbol{p}]$ for which $p \in P_i$ and $q_z \in P_j$. In each of the algorithms for computing partial quantities, only a

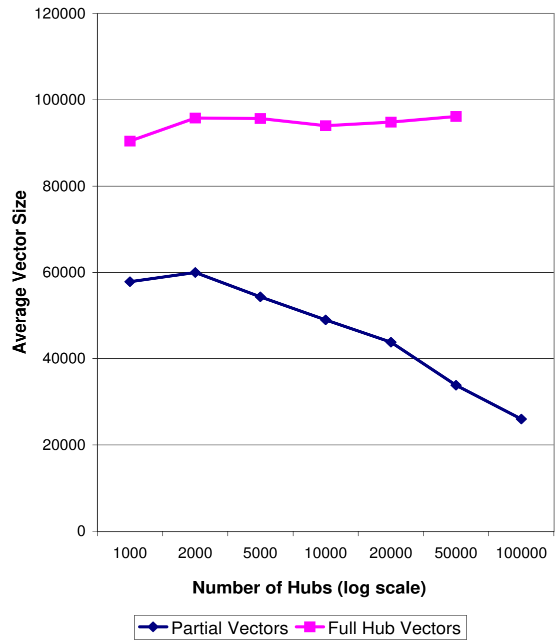

Figure 2: Average Vector Size vs. Number of Hubs

single column $\boldsymbol{L}_{\boldsymbol{k}}^{*,\boldsymbol{j}}[*]$ was kept in memory at any one time, and part of the next-iteration results $\boldsymbol{L}_{\boldsymbol{k+1}}[*]$ were computed by successively reading in individual blocks of the graph or intermediate results as appropriate. Each iteration requires only one linear scan of the intermediate results and web graph, except for repeated squaring, which does not use the web graph explicitly.

6.1 Computing Partial Vectors

For comparison, we computed both (full) hub vectors and partial vectors for various sizes of H, using the selective expansion algorithm with $Q_k(p) = V$ (full hub vectors) and $Q_k(p) = V - H$ (partial vectors). As discussed in Section 4.4.2, we found the partial vectors approach to be much more effective when H contains high-PageRank pages rather than random pages. In our experiments H ranged from the top 1000 to top $100,000$ pages with the highest PageRank. The constant c was set to 0.15.

To evaluate the performance and scalability of our strategy independently of implementation and platform, we focus on the size of the results rather than computation time, which is linear in the size of the results. Because of the number of trials we had to perform and limitations on resources, we computed results only up to 6 iterations, for $|H|$ up to $100,000$. Figure 2 plots the average size of (full) hub vectors and partial vectors (recall that size is the number of nonzero entries), as computed after 6 iterations of the selective expansion algorithm, which for computing full hub vectors is equivalent to the basic dynamic programming algorithm. Note that the x-axis plots $|H|$

in logarithmic scale.

Experiments were run using a 1.4 gigahertz CPU on a machine with 3.5 gigabytes of memory. For $|H| = 50,000$, the computation of full hub vectors took about 2.8 seconds per vector, and about 0.33 seconds for each partial vector. We were unable to compute full hub vectors for $|H| = 100,000$ due to the time required, although the average vector size is expected not to vary significantly with $|H|$ for full hub vectors. In Figure 2 we see that the reduction in size from using our technique becomes more significant as $|H|$ increases, suggesting that our technique scales well with $|H|$.

6.2 Computing the Hubs Skeleton

We computed the hubs skeleton for $|H| = 10,000$ by running the selective expansion algorithm for 6 iterations using $Q_k(p) = H$, and then running the repeated squaring algorithm for 10 iterations (Section 5.3.2), where $Q_k(p)$ is chosen to be the top 50 entries under the top-m scheme (Section 5.2.2). The average size of the hubs skeleton is 9021 entries. Each iteration of the repeated squaring algorithm took about an hour, a cost that depends only on $|H|$ and is constant with respect to the precision to which the partial vectors are computed.

6.3 Constructing Hub Vectors from Partial Vectors

Next we measured the construction of (full) hub vectors from partial vectors and the hubs skeleton. Note that in practice we may construct PPV's directly from partial vectors, as discussed in Section 5.4. However, performance of the construction would depend heavily on the user's preference vector. We consider hub vector computation because it better measures the performance benefits of our partial vectors approach.

As suggested in Section 4.3, the precision of the hub vectors constructed from partial vectors can be varied at query time according to application and performance demands. That is, instead of using the entire set $r_p(H)$ in the construction of $\boldsymbol{r_p}$, we can use only the highest m entries, for $m \leq |H|$. Figure 3 plots the average size and time required to construct a full hub vector from partial vectors in memory versus m, for $|H| = 10,000$. Results are averaged over 50 randomly-chosen hub vectors. Note that the x-axis is in logarithmic scale.

Recall from Section 6.1 that the partial vectors from which the hubs vector is constructed were computed using 6 iterations, limiting the precision. Thus, the error values in Figure 3 are roughly 16% (ranging from 0.166 for $m = 100$ to 0.163 for $m = 10,000$). Nonetheless, this error is much smaller than that of the iteration-6 full hub vectors computed in Section 6.1, which have error $(1-c)^6 = 38\%$. Note, however, that the size of a vector is a better indicator of precision than the magnitude, since we are usually most interested in the number of pages with nonzero entries in the distribution vector. An iteration-6 full hub vector (from Section 6.1) for page p contains nonzero

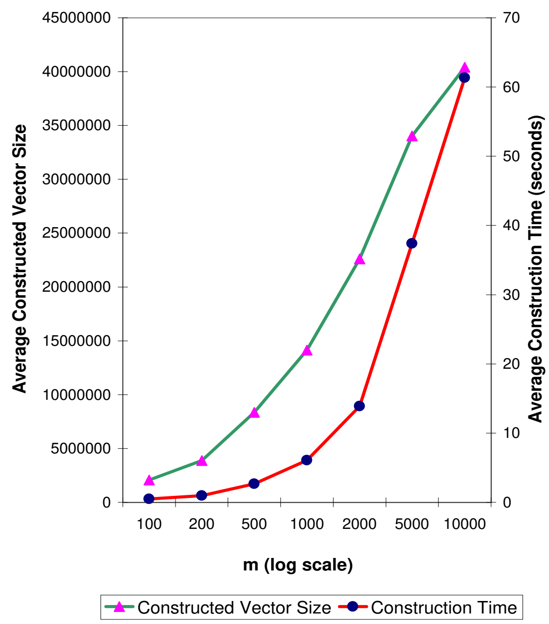

Figure 3: Construction Time and Size vs. Hubs Skeleton Portion (m)

entries for pages at most 6 links away from p, $93,993$ pages on average. In contrast, from Figure 3 we see that a hub vector containing 14 million nonzero entries can be constructed from partial vectors in 6 seconds.

7 Related Work

The use of personalized PageRank to enable personalized web search was first proposed in [10], where it was suggested as a modification of the global PageRank algorithm, which computes a universal notion of importance. The computation of (personalized) PageRank scores was not addressed beyond the naive algorithm.

In [5], personalized PageRank scores were used to enable "topic-sensitive" web search. Specifically, precomputed hub vectors corresponding to broad categories in *Open Directory* were used to bias importance scores, where the vectors and weights were selected according to the text query. Experiments in [5] concluded that the use of personalized PageRank scores can improve web search, but the number of hub vectors used was limited to 16 due to the computational requirements, which were not addressed in that work. Scaling the number of hub pages beyond 16 for finer-grained personalization is a direct application of our work.

Another technique for computing web-page importance, *HITS*, was presented in [8]. In HITS, an iterative computation similar in spirit to PageRank is applied at query time on a subgraph consisting of pages matching a text query and those "nearby". Personalizing based on user-specified web pages (and their linkage structure in the web graph) is not addressed by HITS. Moreover, the number of pages in the subgraphs used by HITS (order of thousands) is much smaller than that we consider in this paper (order of millions), and the computation from scratch at query time makes the HITS approach difficult to scale.

Another algorithm that uses query-dependent importance scores to improve upon a global version of importance was presented in [11]. Like HITS, it first restricts the computation to a subgraph derived from text matching. (Personalizing based on user-specified web pages is not addressed.) Unlike HITS, [11] suggested that importance scores be precomputed offline for every possible text query, but the enormous number of possibilities makes this approach difficult to scale.

The concept of using "hub nodes" in a graph to enable partial computation of solutions to the shortest-path problem was used in [3] in the context of database search. That work deals with searches within databases, and on a scale far smaller than that of the web.

Some system aspects of (global) PageRank computation were addressed in [4]. The disk-based data-partitioning strategy used in the implementation of our algorithm is adopted from that presented therein.

Finally, the concept of inverse P-distance used in this paper is based on the concept of expected-f distance introduced in [7], where it was presented as an intuitive model for a similarity measure in graph structures.

8 Summary

We have addressed the problem of scaling personalized web search:

- We started by identifying a linear relationship that allows personalized PageRank vectors to be expressed as a linear combination of *basis vectors*. Personalized vectors corresponding to arbitrary preference sets drawn from a *hub set* H can be constructed quickly from the set of precomputed basis *hub vectors*, one for each hub $h \in H$.

- We laid the mathematical foundations for constructing hub vectors efficiently by relating personalized PageRank scores to *inverse P-distances*, an intuitive notion of distance in arbitrary directed graphs. We used this notion of distance to identify interrelationships among basis vectors.

- We presented a method of encoding hub vectors as *partial vectors* and the *hubs skeleton*. Redundancy is minimized under this representation: each partial vector for a hub page p represents the part of p's hub vector unique to itself, while the skeleton specifies how partial vectors are assembled into full vectors.

- We presented algorithms for computing basis vectors, and showed how they can be modified to compute partial vectors and the hubs skeleton efficiently.
- We ran experiments on real web data showing the effectiveness of our approach. Results showed that our strategy results in significant resource reduction over full vectors, and scales well with $|H|$, the degree of personalization.

9 Acknowledgment

The authors thank Taher Haveliwala for many useful discussions and extensive help with implementation.

References

[1] `http://www.google.com`.

[2] `http://dmoz.org`.

[3] Roy Goldman, Narayanan Shivakumar, Suresh Venkatasubramanian, and Hector Garcia-Molina. Proximity search in databases. In *Proceedings of the Twenty-Fourth International Conference on Very Large Databases*, New York, New York, August 1998.

[4] Taher H. Haveliwala. Efficient computation of PageRank. Technical report, Stanford University Database Group, 1999. `http://dbpubs.stanford.edu/pub/1999-31`.

[5] Taher H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*, Honolulu, Hawaii, May 2002.

[6] Jun Hirai, Sriram Raghavan, Andreas Paepcke, and Hector Garcia-Molina. WebBase: A repository of web pages. In *Proceedings of the Ninth International World Wide Web Conference*, Amsterdam, Netherlands, May 2000. `http://www-diglib.stanford.edu/~testbed/doc2/WebBase/`.

[7] Glen Jeh and Jennifer Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 2002.

[8] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, California, January 1998.

[9] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, United Kingdom, 1995.

[10] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University Database Group, 1998. http://citeseer.nj.nec.com/368196.html.

[11] Matthew Richardson and Pedro Domingos. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In *Proceedings of Advances in Neural Information Processing Systems 14*, Cambridge, Massachusetts, December 2002.

APPENDIX

A Proof: Linearity Theorem

Theorem (Linearity). *For any preference vectors $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$, if $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ are the two corresponding PPV's, then for any constants $\alpha_1, \alpha_2 \geq 0$ such that $\alpha_1 + \alpha_2 = 1$,*

$$\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} = (1-c)\boldsymbol{A}(\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2}) + c(\alpha_1 \boldsymbol{u_1} + \alpha_2 \boldsymbol{u_2})$$

Proof:

$$\begin{aligned}
\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} &= \alpha_1((1-c)\boldsymbol{A}\boldsymbol{v_1} + c\boldsymbol{u_1}) + \alpha_2((1-c)\boldsymbol{A}\boldsymbol{v_2} + c\boldsymbol{u_2}) \\
&= \alpha_1(1-c)\boldsymbol{A}\boldsymbol{v_1} + \alpha_1 c\boldsymbol{u_1} + \alpha_2(1-c)\boldsymbol{A}\boldsymbol{v_2} + \alpha_2 c\boldsymbol{u_2} \\
&= (1-c)\boldsymbol{A}(\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2}) + c(\alpha_1 \boldsymbol{u_1} + \alpha_2 \boldsymbol{u_2}) \qquad \square
\end{aligned}$$

B Proof: Decomposition Theorem

Theorem (Decomposition). *For any $p \in V$,*

$$\boldsymbol{r_p} = \frac{(1-c)}{|O(p)|} \sum_{i=1}^{|O(p)|} \boldsymbol{r_{O_i(p)}} + c\boldsymbol{x_p}$$

Proof: First we rewrite equation (1) in an equivalent form. For a given preference vector $\boldsymbol{u}$, we define the *derived matrix* $\boldsymbol{A_u}$ as

$$\boldsymbol{A_u} = (1-c)\boldsymbol{A} + c\boldsymbol{U} \tag{21}$$

where $\boldsymbol{U}$ is the $n \times n$ matrix with $U_{ij} = u_i$ for all i, j. If we require that $|\boldsymbol{v}| = 1$, we can write equation (1) as

$$\boldsymbol{v} = \boldsymbol{A_u}\boldsymbol{v}$$

Without loss of generality, let the out-neighbors of p be $1, \ldots, k$. Let $\boldsymbol{A_p}$ be the derived matrix corresponding to $\boldsymbol{x_p}$, and let $\boldsymbol{A_1}, \ldots, \boldsymbol{A_k}$ be the derived matrices for $\boldsymbol{u} = \boldsymbol{x_1}, \ldots, \boldsymbol{x_k}$, respectively. Let $\boldsymbol{U_p}$ and $\boldsymbol{U_1}, \ldots, \boldsymbol{U_k}$ be the corresponding $\boldsymbol{U}$'s in equation (21).

Let

$$\boldsymbol{v_p} = \frac{(1-c)}{k} \sum_{i=1}^{k} \boldsymbol{r_i} + c\boldsymbol{x_p}$$

Clearly, $|\boldsymbol{v_p}| = 1$. We need to show that $\boldsymbol{A_p}\boldsymbol{v_p} = \boldsymbol{v_p}$, in which case $\boldsymbol{v_p} = \boldsymbol{r_p}$, since PPV's are unique (Section 1). First we have that:

$$\begin{aligned}
\boldsymbol{A_p}\boldsymbol{v_p} &= \boldsymbol{A_p}\left(\frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{r_i} + c\boldsymbol{x_p}\right) \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{A_p}\boldsymbol{r_i} + c\boldsymbol{A_p}\boldsymbol{x_p}
\end{aligned}$$

Using the identity

$$\boldsymbol{A_p} = \boldsymbol{A_i} - c\boldsymbol{U_i} + c\boldsymbol{U_p}$$

we have:

$$\begin{aligned}
\boldsymbol{A_p v_p} &= \frac{1-c}{k}\sum_{i=1}^{k}(\boldsymbol{A_i} - c\boldsymbol{U_i} + c\boldsymbol{U_p})\boldsymbol{r_i} + c\boldsymbol{A_p x_p} \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{A_i r_i} - \frac{1-c}{k}c\sum_{i=1}^{k}\boldsymbol{U_i r_i} + \frac{1-c}{k}c\sum_{i=1}^{k}\boldsymbol{U_p r_i} + c\boldsymbol{A_p x_p} \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{r_i} - \frac{1-c}{k}c\sum_{i=1}^{k}\boldsymbol{x_i} + \frac{1-c}{k}c\sum_{i=1}^{k}\boldsymbol{x_p} + c\boldsymbol{A_p x_p} \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{r_i} - \frac{1-c}{k}c\sum_{i=1}^{k}\boldsymbol{x_i} + (1-c)c\boldsymbol{x_p} + c((1-c)\boldsymbol{A} + c\boldsymbol{U_p})\boldsymbol{x_p} \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{r_i} - \frac{1-c}{k}c\sum_{i=1}^{k}\boldsymbol{x_i} + (1-c)c\boldsymbol{x_p} + (1-c)c\boldsymbol{A x_p} + c^2\boldsymbol{x_p} \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{r_i} + (1-c)c\boldsymbol{x_p} + c^2\boldsymbol{x_p} + (1-c)c\left(\boldsymbol{A x_p} - \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{x_i}\right) \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{r_i} + (1-c)c\boldsymbol{x_p} + c^2\boldsymbol{x_p} \\
&= \frac{1-c}{k}\sum_{i=1}^{k}\boldsymbol{r_i} + c\boldsymbol{x_p} \\
&= \boldsymbol{v_p} \qquad \square
\end{aligned}$$

C Inverse P-distance

C.1 Relation to Personalized PageRank

The relationship between inverse P-distances and personalized PageRank scores is given by the following theorem.

Theorem. *For all* $p, q \in V$,

$$r_p(q) = r'_p(q)$$

Proof: Writing the Decomposition Theorem in scalar form for page p, we get a set of n equations, one for each $q \in V$, of the form

$$r_p(q) = \begin{cases} (1-c)\sum_{i=1}^{|O(p)|} r_{O_i(p)}(q) & \text{(if } p \neq q\text{)} \\ (1-c)\sum_{i=1}^{|O(p)|} r_{O_i(p)}(q) + c & \text{(if } p = q\text{)} \end{cases}$$

Let us now fix q, and consider the set of n equations, one for each $p \in V$, in the above form. By a proof very similar to that given in [7], it can be shown these equations have a unique solution, so we need only show that $r'_p(q)$ satisfies these equations as well.

Clearly, if there is no path from p to q, then $r_p(q) = r'_p(q) = 0$, so suppose q can be reached from p. Consider the tours t starting at p and ending at q in which the first step is to the out-neighbor $O_z(p)$. If $p \neq q$, there is a one-to-one correspondence between such t and tours t' from $O_z(p)$ to q: for each t' we may derive a corresponding t by appending the edge $\langle p, O_z(p) \rangle$ at the beginning. Let T be the bijection that takes each t' to the corresponding t. If the length of t' is l, then the length of $t = T(t')$ is $l + 1$. Moreover, the probability of traveling t is $P[t] = \frac{1}{|O(p)|} P[t']$. We can now split the sum in (5) according to the first step of the tour t to write

$$\begin{aligned} r'_p(q) &= \sum_{z=1}^{|O(p)|} \sum_{t' :\ O_z(p) \rightsquigarrow q} P[T(t')] c (1-c)^{l(T(t'))} \\ &= \frac{1-c}{|O(p)|} \sum_{z=1}^{|O(p)|} \sum_{t' :\ O_z(p) \rightsquigarrow b} P[t'] c (1-c)^{l(t)} \\ &= \frac{1-c}{|O(p)|} \sum_{i=1}^{|O(p)|} r'_p(q) \end{aligned}$$

If $p = q$, then the same correspondence holds except that there is an extra tour t from p to $q = p$ which does not correspond to any tour t' starting from an $O_z(p)$: the zero length tour $t' = \langle p \rangle$. The length of this tour is 0, and in this case $P[t]c(1-c)^{l(t)} = c$. Thus

$$r'_p(q) = \frac{1-c}{|O(p)|} \sum_{i=1}^{|O(p)|} r'_{O_i(p)}(q) + c$$

when $p = q$. $\square$

C.2 Loop Factor

The use of inverse P-distances yields further insight into the fairness of PageRank scoring. Since the global PageRank for a page q is just the uniform sum $\sum_{p=1}^{n} r_p(q)/n$, we see that the PageRank of a page q is the average, over all pages p, of the inverse P-distance from p to q. The intuition is that high-PageRank pages are on average "close" to other pages under this distance measure. However, note that the summation in (5) is taken over tours that may touch q multiple times. The effect is that a page q can influence its own PageRank (by a factor less than $1/c$) simply by changing its out-links. In particular, if a page q with PageRank $\mathtt{PR}(q)$ links to every page p for which there is a path to q (as are logically created for pages without out-links in [5, 10]), then its PageRank would be a factor $c + (1-c)\mathtt{PR}(q)$ less than if it had linked to itself and no other page. This "loop factor" can be quantified as $r_q(q)$: under the definition that tours t from p to q may touch q only once, $r_p(q)$ can be written as

$$r_p(q) = r_q(q) \sum_{t:p \rightsquigarrow q} P[t](1-c)^{l(t)}$$

where the summation is independent of q's out-links. This is the expected-f distance [7] from p to q, for $f(x) = (1-c)^x$. Thus eliminating the loop factor (dividing by $r_q(q)$ to get the expected-f distance) may result in a fairer scoring.

D Proof: Hubs Theorem

Theorem (Hubs). *For any $p \in V$, $H \subseteq V$,*

a) $\boldsymbol{r_p^H} = \frac{1}{c} \sum_{h \in H} (r_p(h) - cx_p(h)) \left(\boldsymbol{r_h} - \boldsymbol{r_h^H} - c\boldsymbol{x_h}\right)$

b) $\boldsymbol{r_p^H} = \frac{1}{c} \sum_{h \in H} \left(r_p(h) - r_p^H(h) - cx_p(h)\right) (\boldsymbol{r_h} - c\boldsymbol{x_h})$

Proof of (a): The idea is to separate tours t going through H into two parts, everything up to the last occurrence of a page $h \in H$, and the rest. Let $\beta(t)$, for tours $t : p \rightsquigarrow H \rightsquigarrow q$, denote the beginning of t to the last occurrence of a page $h \in H$ which t passes through, so $\beta(t) = \langle p, \ldots, h \rangle$. Let $\gamma(t)$ be the rest, so $\gamma(t) = \langle h, \ldots, q \rangle$. Let $\pi(t) = P[t]c(1-c)^{l(t)}$ for short. Let $s(t)$ be the set of pages that t passes through, so that $r_p^H(q)$ can be written as

$$r_p^H(q) = \sum_{\substack{t:p \rightsquigarrow q \\ s(t) \cap H \neq \emptyset}} P[t]c(1-c)^{l(t)}$$

Let us first partition the summation in (6) according to $\beta(t)$:

$$r_p^H(q) = \sum_{\substack{t_1 | t_1 = \beta(t) \\ t:p \rightsquigarrow H \rightsquigarrow q}} \sum_{\substack{t:p \rightsquigarrow H \rightsquigarrow q \\ \beta(t) = t_1}} P[t]c(1-c)^{l(t)} \tag{22}$$

For each t, $\beta(t)$ is itself a tour $t' : p \rightsquigarrow h$; conversely, each $t' : p \rightsquigarrow h$ is a $\beta(t)$ for some t, with the exception of the zero-length tour $t' = \langle p \rangle$ in the special case where $p \in H$. Thus we can group the tours t by h and $\beta(t)$ ending at h to rewrite (22) as:

$$r_p^H(q) = \sum_{h \in H} \sum_{\substack{t_1:p \rightsquigarrow h \\ l(t_1) > 0}} \sum_{\substack{t:p \rightsquigarrow H \rightsquigarrow q \\ \beta(t) = t_1}} P[t]c(1-c)^{l(t)}$$

But $P[t] = P[\beta(t)]P[\gamma(t)]$, and $l(t) = l(\beta(t)) + l(\gamma(t))$, so

$$\begin{aligned}
r_p^H(q) &= \sum_{h \in H} \sum_{\substack{t_1 : p \rightsquigarrow h \\ l(t_1) > 0}} \sum_{\substack{t : p \rightsquigarrow H \rightsquigarrow q \\ \beta(t) = t_1}} P[\beta(t)]P[\gamma(t)]c(1-c)^{l(\beta(t)) + l(\gamma(t))} \\
&= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_1 : a \rightsquigarrow h \\ l(t_1) > 0}} \pi(t_1) \sum_{\substack{t : p \rightsquigarrow H \rightsquigarrow q \\ \beta(t) = t_1}} \pi(\gamma(t))
\end{aligned}$$

There is a canonical bijection γ_{t_1} between tours $t : p \rightsquigarrow H \rightsquigarrow q$ with $\beta(t) = t_1$ and tours $t' : h \rightsquigarrow q$ which do not pass through H (for which $s(t') \cap H = \emptyset$), with the exception of the zero-length tour $\langle q \rangle$ when $q \in H$. That is, $\gamma_{t_1}(t) = \gamma(t) = t'$, so we can write each tour t as $t = \gamma_{t_1}^{-1}(t')$. Replacing $\gamma(t)$ in the previous equation with $\gamma(t) = \gamma(\gamma_{t_1}^{-1}(t')) = t'$ and accounting for the possible zero-length tour, we have

$$\begin{aligned}
r_p^H(q) &= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_1 : p \rightsquigarrow h \\ l(t_1) > 0}} \pi(t_1) \left(\sum_{\substack{t' : h \rightsquigarrow q \\ s(t') \cap H = \emptyset}} \pi(t') - x_h(q) \sum_{t' = \langle q \rangle} \pi(t') \right) \\
&= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_1 : p \rightsquigarrow h \\ l(t_1) > 0}} \pi(t_1) \left(\sum_{\substack{t' : h \rightsquigarrow q \\ s(t') \cap H = \emptyset}} \pi(t') - x_h(q) c \right)
\end{aligned}$$

But the set of tours t from h to q which do not pass through H is the set of tours from h to q minus the set of tours from h to q which pass through H. Thus,

$$\begin{aligned}
r_p^H(q) &= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_1 : p \rightsquigarrow h \\ l(t_1) > 0}} \pi(t_1) \left(\sum_{t' : h \rightsquigarrow q} \pi(t') - \sum_{t' : h \rightsquigarrow H \rightsquigarrow q} \pi(t') - c x_h(q) \right) \\
&= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_1 : p \rightsquigarrow h \\ l(t_1) > 0}} \pi(t_1) \left(r_h(q) - r_h^H(q) - c x_h(q) \right)
\end{aligned}$$

Finally,

$$\sum_{\substack{t_1 : p \rightsquigarrow h \\ l(t_1) > 0}} \pi(t_1) = r_p(h) - c x_p(h)$$

where $c x_p(h)$ accounts for the possible tour $t_1 = \langle p \rangle$ when $p = h$, for which $P[t_1]c(1-c)^{l(t_1)} = c$, and we have

$$r_p^H(q) = \frac{1}{c} \sum_{h \in H} \left(r_p(h) - c x_p(h) \right) \left(r_h(q) - r_h^H(q) - c x_h(q) \right)$$

This equation written in vector form is the Hubs Theorem (a). $\square$

Proof of (b): The idea is to separate tours t going through H differently: everything up to the first (instead of last) occurrence of a page $h \in H$, and the rest. Let $\beta(t)$, for tours $t : p \rightsquigarrow H \rightsquigarrow q$, denote the beginning of t to the first occurrence of a page $h \in H$ which t passes through, so $\beta(t) = \langle p, \ldots, h\rangle$. Let $\gamma(t)$ be the rest, so $\gamma(t) = \langle h, \ldots, q\rangle$.

Let us first partition the summation in (6) according to $\gamma(t)$:

$$r_p^H(q) = \sum_{\substack{t_2 | t_2 = \gamma(t) \\ t : p \rightsquigarrow H \rightsquigarrow q}} \sum_{\substack{t : p \rightsquigarrow H \rightsquigarrow q \\ \gamma(t) = t_2}} P[t] c (1-c)^{l(t)} \tag{23}$$

For each t, $\gamma(t)$ is itself a tour $t' : h \rightsquigarrow q$; conversely, each $t' : h \rightsquigarrow q$ is a $\gamma(t)$ for some t, with the exception of the zero-length tour $t' = \langle q\rangle$ in the special case where $q \in H$. Thus we can group the tours t by h and $\gamma(t)$ beginning at h to rewrite (23) as:

$$r_p^H(q) = \sum_{h \in H} \sum_{\substack{t_2 : h \rightsquigarrow q \\ l(t_2) > 0}} \sum_{\substack{t : p \rightsquigarrow H \rightsquigarrow q \\ \gamma(t) = t_2}} P[t] c (1-c)^{l(t)}$$

But $P[t] = P[\beta(t)]P[\gamma(t)]$, and $l(t) = l(\beta(t)) + l(\gamma(t))$, so

$$\begin{aligned} r_p^H(q) &= \sum_{h \in H} \sum_{\substack{t_2 : h \rightsquigarrow q \\ l(t_2) > 0}} \sum_{\substack{t : p \rightsquigarrow H \rightsquigarrow q \\ \gamma(t) = t_2}} P[\beta(t)]P[\gamma(t)] c (1-c)^{l(\beta(t)) + l(\gamma(t))} \\ &= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_2 : h \rightsquigarrow q \\ l(t_2) > 0}} \pi(t_2) \sum_{\substack{t : p \rightsquigarrow H \rightsquigarrow q \\ \gamma(t) = t_2}} \pi(\beta(t)) \end{aligned}$$

There is a canonical bijection β_{t_2} between tours $t : p \rightsquigarrow H \rightsquigarrow q$ with $\gamma(t) = t_2$ and tours $t' : a \rightsquigarrow h$ which do not pass through H (for which $s(t') \cap H = \emptyset$), with the exception of the zero-length tour $\langle p\rangle$ when $p \in H$. That is, $\beta_{t_2}(t) = \beta(t) = t'$, so we can write each tour t as $t = \beta_{t_2}^{-1}(t')$. Replacing $\beta(t)$ in the previous equation with $\beta(t) = \beta(\beta_{t_1}^{-1}(t')) = t'$ and accounting for the possible zero-length tour, we have

$$\begin{aligned} r_p^H(q) &= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_2 : h \rightsquigarrow q \\ l(t_2) > 0}} \pi(t_2) \left(\sum_{\substack{t' : p \rightsquigarrow h \\ s(t') \cap H = \emptyset}} \pi(t') - x_p(h) \sum_{t' = \langle p\rangle} \pi(t') \right) \\ &= \frac{1}{c} \sum_{h \in H} \sum_{\substack{t_2 : h \rightsquigarrow q \\ l(t_2) > 0}} \pi(t_2) \left(\sum_{\substack{t' : p \rightsquigarrow h \\ s(t') \cap H = \emptyset}} \pi(t') - x_p(h) c \right) \end{aligned}$$

But the set of tours t from p to h which do not pass through H is the set of tours from p to h minus the set of tours from p to h which pass through H. Thus,

$$
\begin{aligned}
r_p^H(q) &= \frac{1}{c}\sum_{h\in H}\sum_{\substack{t_2:h\rightsquigarrow q\\ l(t_2)>0}} \pi(t_2)\left(\sum_{t':p\rightsquigarrow h}\pi(t') - \sum_{t':p\rightsquigarrow H\rightsquigarrow h}\pi(t') - cx_p(h)\right)\\
&= \frac{1}{c}\sum_{h\in H}\sum_{\substack{t_2:h\rightsquigarrow q\\ l(t_2)>0}} \pi(t_2)\left(r_p(h) - r_p^H(h) - cx_p(h)\right)
\end{aligned}
$$

Finally,

$$
\sum_{\substack{t_2:h\rightsquigarrow q\\ l(t_2)>0}} \pi(t_2) = r_h(q) - cx_h(q)
$$

where $cx_h(q)$ accounts for the possible tour $t_2 = \langle q\rangle$ when $q = h$, for which $P[t_2]c(1-c)^{l(t_2)} = c$, and we have

$$
r_p^H(q) = \frac{1}{c}\sum_{h\in H}\left(r_p(h) - r_p^H(h) - cx_p(h)\right)(r_h(q) - cx_h(q))
$$

This equation written in vector form is the Hubs Theorem (b). $\square$

E Proof: Basic Dynamic Programming Algorithm

To prove correctness of the basic dynamic programming algorithm, we need to show that for all $k \geq 0$ and $p \in V$, $\boldsymbol{D_k[p]} + \sum_{q\in V} E_{k+1}[p](q)\boldsymbol{r_q} = \boldsymbol{r_p}$, and that the sequence $\{\boldsymbol{E_k[p]}\}$ converges to $\mathbf{0}$ as k tends towards infinity, which implies that $\boldsymbol{D_k[p]}$ converges to $\boldsymbol{r_p}$. In particular, $|\boldsymbol{E_k[p]}| = (1-c)^k$. The proof is by induction on k. The case for $k = 0$ is obvious, so suppose the claim is true for k, for some $k \geq 0$. First we show that $\boldsymbol{D_{k+1}[p]} + \sum_{q\in V} E_{k+1}[p](q)\boldsymbol{r_q} = \boldsymbol{r_p}$:

$$
\begin{aligned}
\boldsymbol{D_{k+1}[p]} + \sum_{q\in V} E_{k+1}[p](q)\boldsymbol{r_q} &= \frac{1-c}{|O(p)|}\sum_{i=1}^{|O(p)|} D_k[O_i(p)] + c\boldsymbol{x_p} + \sum_{q\in V}\frac{1-c}{|O(p)|}\sum_{i=1}^{|O(p)|} E_k[O_i(p)](q)\boldsymbol{r_q}\\
&= c\boldsymbol{x_p} + \frac{1-c}{|O(p)|}\sum_{i=1}^{|O(p)|}\left(D_k[O_i(p)] + \sum_{q\in V} E_k[O_i(p)](q)\boldsymbol{r_q}\right)\\
&= \frac{1-c}{|O(p)|}\sum_{i=1}^{|O(p)|} \boldsymbol{r_{O_i(p)}} + c\boldsymbol{x_p}\\
&= \boldsymbol{r_p}
\end{aligned}
$$

where the last step is justified by the Decomposition Theorem. Now we show that $|\boldsymbol{E_{k+1}[p]}| = (1-c)^{k+1}$:

$$\begin{aligned}
|\boldsymbol{E_{k+1}[p]}| &= \left| \frac{1-c}{|O(p)|} \sum_{i=1}^{|O(p)|} \boldsymbol{E_k[O_i(p)]} \right| \\
&= \frac{1-c}{|O(p)|} \sum_{i=1}^{|O(p)|} |\boldsymbol{E_k[O_i(p)]}| \\
&= \frac{1-c}{|O(p)|} \sum_{i=1}^{|O(p)|} (1-c)^k \\
&= \frac{1-c}{|O(p)|} |O(p)| (1-c)^k \\
&= (1-c)^{k+1} \qquad \square
\end{aligned}$$

F Proof: Selective Expansion Algorithm

As in the proof of the basic dynamic programming algorithm, we first show that $\boldsymbol{D_{k+1}[p]} + \sum_{q \in V} E_{k+1}[p](q)\boldsymbol{r_q} = \boldsymbol{r_p}$ for an arbitrary $Q_k(p) \subseteq V$:

$$\begin{aligned}
&\boldsymbol{D_{k+1}[p]} + \sum_{q \in V} E_{k+1}[p](q)\boldsymbol{r_q} \\
&\quad = \left(\boldsymbol{D_k[p]} + \sum_{q \in Q_k(p)} cE_k[p](q)\boldsymbol{x_q} \right) \\
&\qquad + \sum_{q' \in V} \left(\boldsymbol{E_k[p]} - \sum_{q \in Q_k(p)} E_k[p](q)\boldsymbol{x_q} + \sum_{q \in Q_k(p)} \frac{1-c}{|O(q)|} \sum_{i=1}^{|O(q)|} E_k[p](q)\boldsymbol{x_{O_i(q)}} \right)(q')\boldsymbol{r_{q'}} \\
&\quad = \left(\boldsymbol{D_k[p]} + \sum_{q' \in V} E_k[p](q')\boldsymbol{r_{q'}} \right) + \sum_{q \in Q_k(p)} cE_k[p](q)\boldsymbol{x_q} \\
&\qquad - \sum_{q' \in V} \sum_{q \in Q_k(p)} E_k[p](q)x_q(q')\boldsymbol{r_{q'}} + \sum_{q' \in V} \sum_{q \in Q_k(p)} \frac{1-c}{|O(q)|} \sum_{i=1}^{|O(q)|} E_k[p](q)x_{O_i(q)}(q')\boldsymbol{r_{q'}}
\end{aligned}$$

By the inductive hypothesis,

$$\boldsymbol{D_k[p]} + \sum_{q' \in V} E_k[p](q')\boldsymbol{r_{q'}} = \boldsymbol{r_p}$$

so we need only show that the latter terms cancel. Since $x_q(q') = 1$ if $q = q'$ and 0 otherwise, and similarly for $x_{O_i(q)}(q')$, we have

$$\sum_{q' \in V} \sum_{q \in Q_k(p)} E_k[p](q)x_q(q')\boldsymbol{r_{q'}} = \sum_{q \in Q_k(p)} E_k[p](q)\boldsymbol{r_q}$$

and

$$\sum_{q'\in V}\sum_{q\in Q_k(p)}\frac{1-c}{|O(q)|}\sum_{i=1}^{|O(q)|}E_k[p](q)x_{O_i(q)}(q')\boldsymbol{r_{q'}}=\sum_{q\in Q_k(p)}\frac{1-c}{|O(q)|}\sum_{i=1}^{|O(q)|}E_k[p](q)\boldsymbol{r_{O_i(q)}}$$

By the Decomposition Theorem,

$$cE_k[p](q)\boldsymbol{x_q}+\frac{1-c}{|O(q)|}\sum_{i=1}^{|O(q)|}E_k[p](q)\boldsymbol{r_{O_i(q)}}=E_k[p](q)\boldsymbol{r_b}$$

for all $q\in Q_k(p)$, which shows that the terms indeed cancel.

Since $|\boldsymbol{D_k[p]}|$ increases by $c\sum_{q\in Q_k(p)}E_k[p](q)$ each iteration, the error decreases by $c\sum_{q\in Q_k(p)}E_k[p](q)$ each iteration. Thus, any choice of $Q_k(p)$ containing a maximal page q such that $E_k[p](q)=\max\{E_k[p](q)\,|\,q\in V\}$ ensures that the error tends towards 0. In particular, such is the case if $Q_k(p)=V$ or $Q_k(p)$ is the top $m>0$ pages q with the highest $E_k[p](q)$. $\square$

G Proof: Repeated Squaring Algorithm

To verify the correctness of the repeated squaring algorithm, we show that

$$\boldsymbol{D_{2k}[p]}+\sum_{q\in Q_k(p)}E_{2k}[p](q)\boldsymbol{r_q}=\boldsymbol{r_q}$$

for an arbitrary $Q_k(p)\subseteq V$:

$$\begin{aligned}
&\boldsymbol{D_{2k}[p]}+\sum_{q\in V}E_{2k}[p](q)\boldsymbol{r_q}\\
&=\boldsymbol{D_k[p]}+\sum_{q\in Q_k(p)}E_k[p](q)\boldsymbol{D_k[q]}+\\
&\qquad\sum_{q'\in V}\left(E_k[p](q')-\sum_{q\in Q_k(p)}E_k[p](q)x_q(q')+\sum_{q\in Q_k(p)}E_k[p](q)E_k[q](q')\right)\boldsymbol{r_{q'}}\\
&=\boldsymbol{D_k[p]}+\sum_{q'\in V}E_k[p](q')\boldsymbol{r_{q'}}+\sum_{q\in Q_k(p)}E_k[p](q)\left(\boldsymbol{D_k[q]}-\sum_{q'\in V}x_q(q')\boldsymbol{r_{q'}}+\sum_{q'\in V}E_k[q](q')\boldsymbol{r_{q'}}\right)\\
&=\boldsymbol{r_p}+\sum_{q\in Q_k(p)}E_k[p](q)\left[\left(\boldsymbol{D_k[q]}+\sum_{q'\in V}E_k[q](q')\boldsymbol{r_{q'}}\right)-\boldsymbol{r_q}\right]\\
&=\boldsymbol{r_p}+\sum_{q\in Q_k(p)}[0]\\
&=\boldsymbol{r_p}
\end{aligned}$$

As in the proof of the selective expansion algorithm, the error tends towards 0 if $Q_k(p)$ contains the top $m > 0$ pages q with the highest $E_k[p](q)$. If $Q_k(p) = V$, the error is squared on each iteration, for if $|\boldsymbol{E_k}[*]| = \epsilon$, using equation 18 we have:

$$\begin{aligned}
|\boldsymbol{E_{2k}[p]}| &= \left|\sum_{q\in V} E_k[p](q)\boldsymbol{E_k[q]}\right| \\
&= \sum_{q\in V} E_k[p](q)|\boldsymbol{E_k[q]}| \\
&= \sum_{q\in V} E_k[p](q)\epsilon \\
&= \epsilon|\boldsymbol{E_k[p]}| \\
&= \epsilon^2
\end{aligned}$$

Clearly, for all but the first two iterations, repeated squaring reduces error much faster than the decay factor of $1-c$ (for both the basic dynamic programming and selective expansion algorithms) when $Q_k(p) = V$. □

H Proof: Computation of Partial Vectors

We first show that the following hold for all $k \geq 1$ and $p, q \in V$:

$$D_k[p](q) = \sum_{\substack{t:p\rightsquigarrow q\\ l(t)<k\\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} \tag{24}$$

$$E_k[p](q) = \begin{cases} \displaystyle\sum_{\substack{t:p\rightsquigarrow q\\ l(t)=k\\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)} & (\text{if } q \notin H) \\ \displaystyle\sum_{\substack{t:p\rightsquigarrow q\\ 1\leq l(t)\leq k\\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)} & (\text{if } q \in H) \end{cases} \tag{25}$$

where $s(t)$ is the set of pages appearing on t other than at the endpoints (i.e., pages which t passes through), and $s'(t)$ is the set of pages appearing on t other than at the beginning. Consider the case for $k = 1$ (recall that all pages are expanded on iteration 0). The only tours in (24) are the zero-length tours $t = \langle p\rangle$ when $p = q$ (which pass through no hubs), for which $P[t]c(1-c)^{l(t)} = c = D_1[p](q)$. The only tours in (25) are $t = \langle p, q\rangle$ when q is an out-neighbor of p, for which $P[t](1-c)^{l(t)} = \frac{1-c}{|O(p)|} = E_k[p](q)$.

Now suppose for induction that equations (24) and (25) hold for some $k \geq 1$. By equation (14) with $Q_k(p) = V - H$, the difference between $D_{k+1}[p](q)$ and $D_k[p](q)$, for $q \notin H$, is $D_{k+1}[p](q) -$

$D_k[p](q) = cE_1[p](q)$. By the inductive hypothesis, this difference can be written as

$$cE_k[p](q) = \sum_{\substack{t:p \rightsquigarrow q \\ l(t)=k \\ s(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)}$$

Since $q \notin H$, the restriction $s(t) \cap H = \emptyset$ is equivalent to $s'(t) \cap H = \emptyset$, so that

$$\begin{aligned} D_{k+1}[p](q) &= D_k[p](q) + cE_k[p](q) \\ &= \sum_{\substack{t:p \rightsquigarrow q \\ l(t)<k \\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} + \sum_{\substack{t:p \rightsquigarrow q \\ l(t)=k \\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} \\ &= \sum_{\substack{t:p \rightsquigarrow q \\ l(t)<k+1 \\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} \end{aligned}$$

If $q \in H$,

$$D_{k+1}[p](q) = D_k[p](q) = \sum_{\substack{t:p \rightsquigarrow q \\ l(t)<k \\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} = \sum_{\substack{t:p \rightsquigarrow q \\ l(t)<k+1 \\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)}$$

since there is no tour $t : p \rightsquigarrow q$ with $l(t) > 0$ for which $s'(t) \cap H = \emptyset$.

Next we show that

$$E_{k+1}[p](q) = \sum_{\substack{t:p \rightsquigarrow q \\ l(t)=k+1 \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)}$$

for $q \notin H$. By equation (15) with $Q_k(p) = V - H$, we have

$$E_{k+1}[p](q) = \sum_{q' \in (V-H)\cap I(q)} \frac{1-c}{|O(q')|} E_k[p](q') \tag{26}$$

since only the expansion of the in-neighbors of q can contribute to $E_{k+1}[p](q)$, and of these, only the ones not in H are expanded. Expanding $E_k[p](q')$ using the inductive hypothesis, (26) becomes

$$E_{k+1}[p](q) = \sum_{q' \in (V-H)\cap I(q)} \frac{1-c}{|O(q')|} \sum_{\substack{t':p \rightsquigarrow q' \\ l(t')=k \\ s'(t')\cap H=\emptyset}} P[t'](1-c)^{l(t')}$$

where we have replaced $s(t')$ in the summation with $s'(t')$, since $q' \notin H$. We want to show that this is equal to

$$\sum_{\substack{t:p \rightsquigarrow q \\ l(t)=k+1 \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)} \tag{27}$$

Consider the set of tours $t : p \rightsquigarrow q$, with $l(t) = k + 1$ and $s(t) \cap H = \emptyset$, for which the last step is from $q' \in (V - H) \cap I(q)$ to q. There is a one-to-one correspondence between such t and tours $t' : p \rightsquigarrow q'$ of length k with $s'(t) \cap H = \emptyset$: for each t' we may derive a corresponding t by appending the edge $\langle q', q \rangle$ at the end. Let T be the bijection that takes each t' to the corresponding t. If the length of t' is l, then the length of $t = T(t')$ is $l + 1$. Moreover, the probability of traveling t is $P[t] = \frac{1}{|O(q')|} P[t']$. Thus we can split the summation in (27) according to q' to rewrite it as

$$\begin{aligned}
\sum_{\substack{t:p\rightsquigarrow q \\ l(t)=k+1 \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)} &= \sum_{q' \in (V-H)\cap I(q)} \sum_{\substack{t':p\rightsquigarrow q' \\ l(t')=k \\ s'(t')\cap H=\emptyset}} P[T(t')](1-c)^{l(T(t'))} \\
&= \sum_{q' \in (V-H)\cap I(q)} \frac{1-c}{|O(q')|} \sum_{\substack{t':p\rightsquigarrow q' \\ l(t')=k \\ s'(t')\cap H=\emptyset}} P[t'](1-c)^{l(t')}
\end{aligned} \tag{28}$$

which is what we wanted to show.

Now we show that

$$E_{k+1}[p](q) = \sum_{\substack{t:p\rightsquigarrow q \\ 1\le l(t)\le k+1 \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)}$$

for $q \in H$. By equation (15) with $Q_k(p) = V - H$,

$$\begin{aligned}
E_{k+1}[p](q) &= E_k[p](q) + \sum_{q' \in (V-H)\cap I(q)} \frac{1-c}{|O(q')|} E_k[p](q') \\
&= E_k[p](q) + \sum_{q' \in (V-H)\cap I(q)} \frac{1-c}{|O(q')|} \sum_{\substack{t':p\rightsquigarrow q' \\ l(t')=k \\ s(t')\cap H=\emptyset}} P[t'](1-c)^{l(t')}
\end{aligned}$$

Equation (28) still applies, and we have

$$\begin{aligned}
E_{k+1}[p](q) &= E_k[p](q) + \sum_{\substack{t:p\rightsquigarrow q \\ l(t)=k+1 \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)} \\
&= \sum_{\substack{t:p\rightsquigarrow q \\ 1\le l(t)\le k \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)} + \sum_{\substack{t:p\rightsquigarrow q \\ l(t)=k+1 \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)} \\
&= \sum_{\substack{t:p\rightsquigarrow q \\ 1\le l(t)\le k+1 \\ s(t)\cap H=\emptyset}} P[t](1-c)^{l(t)}
\end{aligned}$$

which completes the proof of equations (24) and (25).

Finally, we show that for all $q \in V$, $D_k[p](q)+cE_k[p](q)$ converges to $r_p(q)-r_p^H(q)$ as $k \to \infty$. If $q \notin H$, then $E_k[p](q) \to 0$ as $k \to \infty$, and

$$D_k[p](q) + cE_k[p](q) = D_k[p](q) + \sum_{\substack{t:p \rightsquigarrow q \\ l(t)<k \\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} \to r_p(q) - r_p^H(q)$$

since $s'(t) \cap H = s(t) \cap H$ when $q \notin H$. If $q \in H$, then

$$D_k[p](q) + cE_k[p](q) = \sum_{\substack{t:p \rightsquigarrow q \\ l(t)<k \\ s'(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} + \sum_{\substack{t:p \rightsquigarrow q \\ 1\le l(t)\le k \\ s(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)}$$

When $q \in H$, $s'(t) \cap H \neq \emptyset$ unless $p = q$ and $t = \langle p \rangle$. Thus,

$$\begin{aligned} D_k[p](q) + cE_k[p](q) &= cx_p(q) + \sum_{\substack{t:p \rightsquigarrow q \\ 1\le l(t)\le k \\ s(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} \\ &= \sum_{\substack{t:p \rightsquigarrow q \\ 0\le l(t)\le k \\ s(t)\cap H=\emptyset}} P[t]c(1-c)^{l(t)} \end{aligned}$$

which converges to $r_p(q) - r_p^H(q)$ as $k \to \infty$. $\square$

I Proof: Computation of the Hubs Skeleton

Let $(\boldsymbol{D_i[p]}, \boldsymbol{E_i[p]})$ denote the results after i iterations of repeated squaring, so that the intermediate results left by selective expansion correspond to $i = 0$.

The error initially associated with hub pages, $\sum_{h\notin H} E_0[p](h)$, is bounded by $1 - c$ because the first step of selective expansion expands all pages (Section 5.2.2). By equation (17) with $Q_i(p) = H$, the error associated with hub pages on iteration $i \geq 1$ of repeated squaring, $\sum_{q\in H} E_i[p](q)$, is bounded by $(1-c)^{2^i}$. Moreover, the error associated with non-hub pages, $\sum_{q\notin H} E_i[p](q)$, increases by at most $(1-c)^{2^i} \sum_{q\notin H} E_{i-1}[p](q)$ compared to the previous iteration. Using a geometric series to bound $\sum_{q\notin H} E_i[p](q)$, the total error $|\boldsymbol{E_i[p]}|$ of iteration i is bounded by $(1-c)^{2^i} + \epsilon/c$. $\square$