ISSN:0975-9646

An Overview of Efficient Computation of PageRank

Madhu Bala, Simple Sharma
Computer Science & Engineering,Manav Rachna ,International University,Faridabad, India

Abstract— **With the rapid growth of the Web, users get easily lost in the rich hyper structure. Providing relevant information to the users to cater to their needs is the primary goal of website owners. Therefore, finding the content of the Web and retrieving the users' interests and needs from their behavior have become increasingly important. Web mining is used to categorize users and pages by analyzing the users' behavior, the content of the pages, and the order of the URLs that tend to be accessed in order. Web structure mining plays an important role in this approach. Two page ranking algorithms, HITS and PageRank, are commonly used in web structure mining. Both algorithms treat all links equally when distributing rank scores. Several algorithms have been developed to improve the performance of these methods. This paper discusses efficient techniques for computing PageRank. How to rank Web resources is critical to Web Resource Discovery (Search Engine). This paper points out the weakness of current approaches, we discuss several methods for analyzing the convergence of PageRank based on the induced ordering of the pages**

Keywords- HITS, Web search, web graph, link analysis, PageRank, search in context, personalized search, ranking algorithm, Weighted PageRank, Adaptive PageRank.

I. INTRODUCTION

The World Wide Web is rapidly emerging as an important medium for the dissemination of information related to a wide range of topics [1]. There are about 300 million pages on the Web today with about 1 million being added daily. According to most predictions, the majority of human information will be available on the Web in 10 years. But, it is widely believed that 99% of the information on the Web is of no interest to 99% of the people. Looking for something valuable in this tremendous amount of information is as difficult as looking for a needle in a haystack.

Traditional information retrieval techniques can give poor results on the Web, with its vast scale and highly variable content quality. Recently, however, it was found that Web search results can be much improved by using the information contained in the link structure between pages. The two best-known algorithms which do this are HITS [4] and PageRank [2]. The latter is used in the highly successful Google search engine [3]. The heuristic underlying both of these approaches is that pages with many inlinks are more likely to be of high quality than pages with few inlinks, given that the author of a page will presumably include in it links to pages that s/he believes are of high quality. Given a query (set of words or other query terms), HITS invokes a traditional search engine to obtain a set of pages relevant to it, expands this set with its inlinks and outlinks, and then attempts to find two types of pages, *hubs* (pages that point to many pages of high quality) and *authorities* (pages of high quality). Because this computation is carried out at query time, it is not feasible for today's search engines, which need to handle tens of millions of queries per day. In contrast, PageRank computes a single measure of quality for a page at crawl time. This measure is then combined with a traditional information retrieval score at query time. Compared with HITS, this has the advantage of much greater efficiency, but the disadvantage that the PageRank score of a page ignores whether or not the page is relevant to the query at hand.

The PageRank algorithm for determining the "importance" of Web pages has become a central technique in Web search [5]. The core of the PageRank algorithm involves computing the principal eigenvector of the Markov matrix representing the hyperlink structure of the Web. As the Web graph is very large, containing over a billion nodes, the PageRank vector is generally computed offline, during the preprocessing of the Web crawl, before any queries have been issued. The development of techniques for computing PageRank efficiently for Web-scale graphs is important for a number of reasons.

For Web graphs containing a billion nodes, computing a PageRank vector can take several days. Computing PageRank quickly is necessary to reduce the lag time from when a new crawl is completed to when that crawl can be made available for searching. Furthermore, recent approaches to personalized and topic-sensitive Page- Rank schemes [6, 7, 8] require computing *many* PageRank vectors, each biased towards certain types of pages. These approaches intensify the need for faster methods for computing PageRank.

This paper is structured as follows: in section II, we have discussed the background of algorithms for computing PageRank. In section III, IV and V PageRank, Adaptive, Weighted algorithms have been discussed. Amongst other things, we will investigate whether weighted and adaptive algorithm perform better than PageRank algorithm. Noise reduction is an important issue in image processing. Several algorithms for computing PageRank have already been developed; cfr.[1] and [2] for an extensive overview.

II. BACKGROUND

With the rapid growth of the Web, providing relevant pages of the highest quality to the users based on their queries becomes increasingly difficult. The reasons are that some web pages are not self-descriptive and that some links exist purely for navigational purposes. Therefore, finding appropriate pages through a search engine that relies on web contents or makes use of hyperlink information is very difficult. To address the problems mentioned above, several algorithms have been

proposed. Among them are PageRank [4] and *Hypertext Induced Topic Selection* (HITS) [2, 9] algorithms.
PageRank is a commonly used algorithm in Web Structure Mining. It measures the importance of the pages by analyzing the links [1, 8]. PageRank has been developed by Google and is named after Larry Page, Google's co-founder and president[4]. PageRank ranks pages based on the web structure. Google first retrieves a list of relevant pages to a given query based on factors such as title tags and keywords. Then it uses PageRank to adjust the results so that more "important" pages are provided at the top of the page list [4].
The Pagerank algorithm is described in detail in the next section. HITS ranks webpages by analyzing their inlinks and outlinks. In this algorithm, webpages pointed to by many hyperlinks are called *authorities* whereas webpages that point to many hyperlinks are called *hubs* [9, 10, 6]. Authorities and hubs are illustrated in Figure 1.

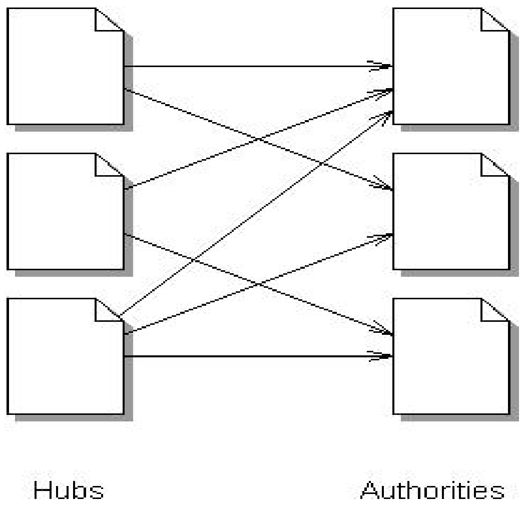

Figure 1. Hubs and authorities[2]

Hubs and authorities are assigned respective scores. Scores are computed in a mutually reinforcing way: an authority pointed to by several highly scored hubs should be a strong authority while a hub that points to several highly scored authorities should be a popular hub [9, 10]. Let ap and hp represent the authority and hub scores of page p, respectively. B(p) and I(p) denote the set of referrer and reference pages of page p, respectively. The scores of hubs and authorities are calculated as follows [2, 9, 10]:

$$a_p = \sum_{q \in B(p)} h_q$$

$$h_p = \sum_{q \in I(p)} a_q$$

Figure 2 shows an example of the calculation of authority and hub scores.

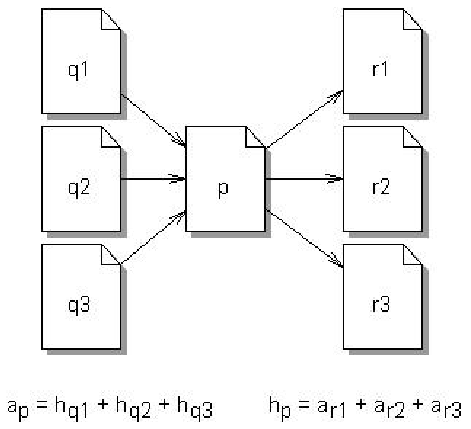

Figure 2. An example of HITS operations

HITS is a purely link-based algorithm. It is used to rank pages that are retrieved from the Web, based on their textual contents to a given query. Once these pages have been assembled, the HITS algorithm ignores textual content and focuses itself on the structure of the Web only. Some difficulties arise from this feature [2]:
• HITS frequently returns more general webpages on an otherwise narrowly focused topic because the web does not contain many resources for the topic,
• Topic drift occurs while the hub has multiple topics because all of the outlinks of a hub page get equivalent weights, and
• Some popular sites that are not highly relevant to the given query gain overhead weight values.
The CLEVER algorithm is an extension of standard HITS and provides an appropriate solution to the problems that result from standard HITS [2]. CLEVER assigns a weight to each link based on the terms of the queries and end-points of the link. It combines anchor text to set weights to the links as well. Moreover, it breaks large hub pages into smaller units so that each hub page is focused on as a single topic. Finally, in the case of a large number of pages from a single domain,
it scales down the weights of pages to reduce the probabilities of overhead weights [2]. Another major shortcoming of standard HITS is that it assumes that all links pointing to a page are of equal weight and fails to recognize that some links might be more important than others. A *Probabilistic analogue of the HITS Algorithm*(PHITS) has been developed to solve this problem[3]. PHITS provides a probabilistic interpretation of term-document relationships and identifies authoritative documents. In the experiment on a set of hyperlinked documents, PHITS demonstrates better results compared to those obtained by standard HITS. The most important feature of the PHITS algorithm is its ability to estimate the actual probabilities of authorities compared to the scalar magnitudes of authority that are provided by standard HITS[3].

III. **Review of PageRank Algorithm**

The First let us review the motivation behind PageRank [10]. The essential idea is that if page u has a link to page v, then the author of u is implicitly conferring some importance to page v. How much importance does u confer? Let N_u be the outdegree of page u, and let Rank(p) represent the importance (i.e., PageRank) of page p. Then the link (u, v) confers Rank(u)/N_u units of rank to v. This simple idea leads to the following fixpoint computation that yields the rank vector Rank* over all of the pages on the web. If N is the number of pages, assign all pages the initial value 1/N. Let B_v represent the set of pages pointing to v. In each iteration, propagate the ranks as follows:[1]

$$\vartheta_v Rank_{i+1}(v) = \sum_{u \in B_v} Rank_i(u)/_u$$

We continue the iterations until Rank stabilizes to within some threshold. The final vector Rank* contains the PageRank vector over the web. This vector is computed only once after each crawl of the web; the values can then be used to influence the ranking of search results [2].

The process can also be expressed as the following eigenvector calculation, pro- viding useful insight into PageRank. Let M be the square, stochastic matrix corresponding to the directed graph G of the web, assuming all nodes in G have at least one outgoing edge. If there is a link from page j to page i, then let the matrix entry mij have the value 1/Nj . Let all other entries have the value 0. One iteration of the previous fix point computation corresponds to the matrix-vector multiplication M × Rank. Repeatedly multiplying Rank by M yields the dominant eigenvector Rank* of the matrix M. Because M corresponds to the stochastic transition matrix over the graph G, PageRank can be viewed as the stationary probability distribution over pages induced by a random walk on the web.

We can measure the convergence of the iterations using the Residual vector. Because M is stochastic (i.e., the entries in each column sum to 1), the dominant eigenvalue of M is 1. Thus the PageRank vector Rank*, the dominant eigenvector of M, has eigenvalue 1, leading to the equality M × Rank* = Rank*. We can use the deviation from this equality for some other vector as an error estimate. For some intermediate vector Ranki, let Residuali = $M \times Rank_i - Rank_i$. Equivalently, because multiplication by M is an iteration step, we have $Residual_i = Rank_{i+1} - Rank_i$. We can treat $\|Residual_i\|$ as an indicator for how well Ranki approximates Rank*. We expect $\|Residual_i\|$ to tend to zero after an adequate number of iterations.

We now address several issues regarding the computation. We previously made the assumption that each node in G has at least one outgoing edge. To enforce this assumption, we can iteratively remove nodes in G that have outdegree 0. Alternatively, we can conceptually add a complete set of outgoing edges to any node with outdegree 0. Another caveat is that the convergence of PageRank is guaranteed only if M is irreducible (i.e., G is strongly connected) and aperiodic [12]. The latter is guaranteed in practice for the web, while the former is true if we add a dampening factor to the rank propagation. We can define a new matrix M′ in which we add transition edges of probability $\frac{1-C}{N}$ between every pair of nodes in G:

$$M' = cM + (1 - c) \times [1/N]_{N \times N}$$

This modification improves the quality of PageRank by introducing a decay factor which limits the effect of rank sinks [4], in addition to guaranteeing convergence to a unique rank vector. For the above modification to M, an iteration of PageRank can be expressed as follows:[2]

$$M' \times Rank = cM \times Rank + (1 - c) \times [1/N]_{N \times 1}$$

We can bias PageRank to weight certain categories of pages by replacing the uniform vector [1 N]N×1 with the nonuniform N ×1 personalization vector ~p as discussed in [4]. In terms of the random-walk model of PageRank, the personalization vector represents the addition of a complete set of transition edges where the probability of edge (u, v) is given by $(1 - c) \times p_v$. Although the matrix M′ that results from the modifications discussed above is not sparse, we never need to store it explicitly. We need only the ability to evaluate M′ × Rank efficiently.

```
function pageRank(A,$x^{(0)}$,v) {
repeat
$x^{(k+1)} = Ax^{(k)}$;
        $\delta = \|x^{(k+1)} - x^k\|_1$;
until $\delta < \epsilon$;
return $x^{(k+1)}$;
}
```

Algorithm 2: PageRank

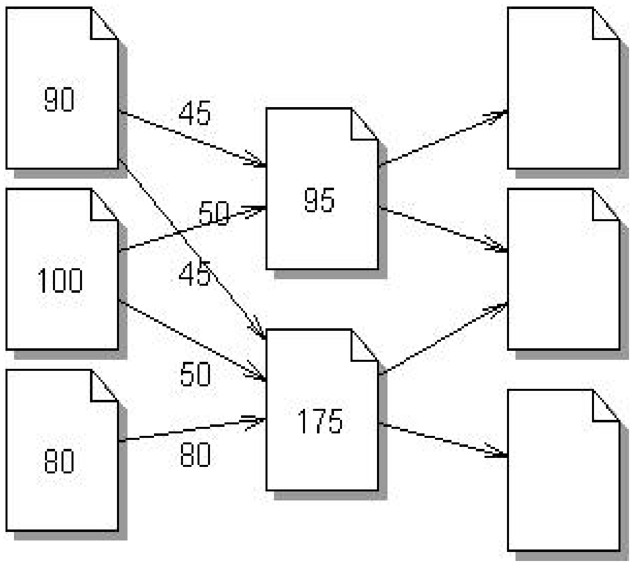

Figure 3. An example of simplified version of PageRank[5]

IV. Adaptive Method for the Computation of PageRank

We observe that the convergence patterns of pages in the PageRank algorithm have a nonuniform distribution. Specifically, many pages converge to their true PageRank quickly, while relatively few pages take a much longer time to converge. Furthermore, we observe that these slow-converging pages are generally those pages with high PageRank.We use this observation to devise a simple algorithm to speed up the computation of PageRank, in which the PageRank of pages that have converged are not recomputed at each iteration after convergence. This algorithm, which we call Adaptive PageRank[11], speeds up the computation of PageRank by nearly 30%.

In particular, we do not need to recompute the Page- Ranks of the pages that have already converged, and we do not need to recompute the contribution of PageRank from pages that have converged to other pages.We discuss in this section how each of these redundancies can be eliminated.

A. Algorithm Intuition

We begin by describing the intuition behind the Adaptive PageRank algorithm. We consider next a single iteration of the Power Method, and show how we can reduce the cost.

Consider that we have completed k iterations of the powermethod. Using the iterate $x^{(k)}$, we now wish to generate the iterate $x^{(k+1)}$. Let C be set of pages that have converged to a given tolerance, and N be the set of pages that have not yet converged,

We can split the matrix A defined in Section 2 into two submatrices. Let A_N be the m×n submatrix corresponding to the inlinks of those m pages whose PageRanks have not yet converged, and AC be the $(n - m) \times n$ submatrix corresponding to the inlinks of those pages that have already converged.

Let us likewise split the current iterate of the PageRank vector $x^{(k)}$ into the m- vector corresponding to the components of $x^{(k)}$ that have not yet converged, and the $(m - n)$-vector corresponding to the components of $x^{(k)}$ that have not yet converged that have already converged. We may order A and $x^{(k)}$ as follows:

$$x^{(k)}=\begin{pmatrix} x_N^{(k)} \\ x_C^{(k)} \end{pmatrix} \quad (3)$$

$$A= \begin{pmatrix} A_N \\ A_C \end{pmatrix} \quad (4)$$

$$\begin{pmatrix} x_N^{(k+1)} \\ x_C^{(k+1)} \end{pmatrix} = \begin{pmatrix} A_N \\ A_C \end{pmatrix} \cdot \begin{pmatrix} x_N^{(k)} \\ x_C^{(k)} \end{pmatrix}$$

However, since the elements of have already converged, we do not need to recompute. Therefore, we may simplify each iteration of the computation to be:

$$x_N^{(k+1)} = A_N x^{(k)}$$

$$x_C^{(k+1)} = x_C^{(k)}$$

The basic Adaptive PageRank algorithm is given in Algorithm 3

```
function adaptivePR(A, $x^{(0)}$ , v) {
repeat
  $x_N^{(k+1)} = A_N x^{(k)}$;
  $x_C^{(k+1)} = x_C^{(k)}$ ;
[N,C] = detectConverged($x^{(k)}$ , $x^{(k+1)}$, €);
periodically, $\delta = ||Ax^{(k)} - x^k||_1$;
until $\delta$ <€;
return $x^{(k+1)}$;
}
```

Algorithm 3: Adaptive PageRank[11]

Identifying pages in each iteration that have converged is inexpensive. However, reordering the matrix A at each iteration is expensive. Therefore, we exploit the idea given above by periodically identifying converged pages and constructing AN without explicitly reordering identifiers. Since AN is smaller than A, the iteration cost for future iterations is reduced.

IV. Weighted PageRank Algorithm

The Weighted PageRank algorithm (WPR)[12], an extension to the standard PageRank algorithm. WPR takes into account the importance of both the inlinks and the outlinks of the pages and distributes rank scores based on the popularity of the pages. The results of our simulation studies show that WPR performs better than the conventional PageRank algorithm in terms of returning larger number of relevant pages to a given query. The more popular webpages are, the more linkages that other webpages tend to have to them or are linked to by them. The proposed extended PageRank algorithm–a Weighted PageRank Algorithm–assigns larger rank values to more important (popular) pages instead of dividing the rank value of a page evenly among its outlink pages. Each outlink page gets a value proportional to its popularity (its number of inlinks and outlinks). The popularity from the number of inlinks and outlinks is recorded as and respectively is the weight of link(v, u) calculated based on the number of inlinks of page u and the number of inlinks of all reference pages.

$$W_{v,u}^{in} = \frac{I_u}{\sum_{p \in R(v)} I_p}$$

where I_u and I_p represent the number of inlinks of page u and page p, respectively. R(v) denotes the reference page list of page v is the weight of link(v, u) calculated based on the number of outlinks of page u and the number of outlinks of all reference pages of page v.

$$W_{v,u}^{out} = \frac{O_u}{\sum_{p \in R(v)} O_p}$$

where O_u and O_p represent the number of outlinks of page u and page p, respectively. R(v) denotes the reference page list of page v.

Figure 4 shows an example of some links of a hypothetical website.

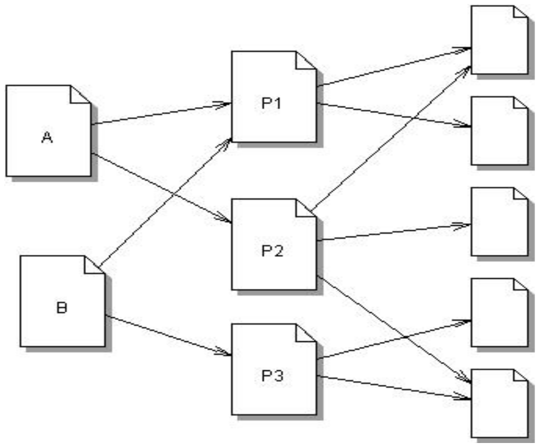

Figure 4. Links of a website[12]

In this example, Page A has two reference pages: p1 and p2. The inlinks and outlinks of these two pages are $I_{p1} = 2$, $I_{p2} = 1$, $O_{p1} = 2$, and $O_{p2} = 3$. Therefore,

$$W^{in}_{v,u} = \frac{I_u}{\sum_{p \in R(v)} I_p}$$

$$= I_{p1}/(I_{p1} + I_{p2}) = 2/3$$

and

$$W^{out}_{v,u} = \frac{O_u}{\sum_{p \in R(v)} O_p}$$

$$= O_{p1}/(O_{p1} + O_{p2}) = 2/5$$

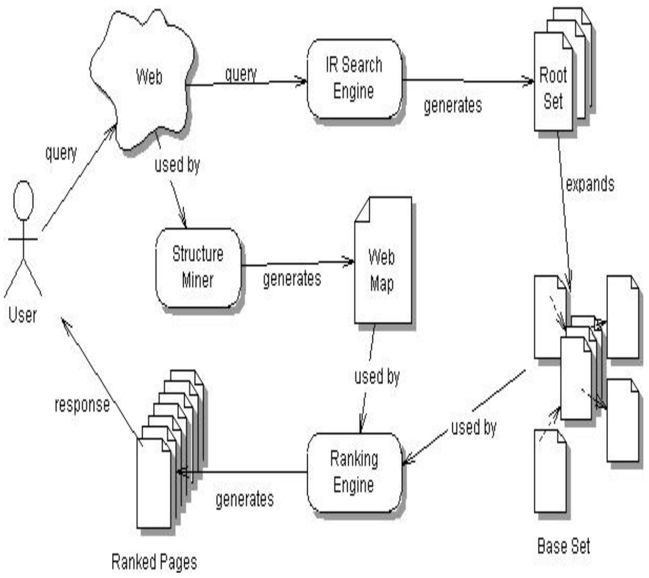

Figure 5. Architectural components of the system used to implement and evaluate the WPR algorithm[12]

References

[1]. P. Bernstein, M. Brodie, S. Ceri, et al., The Asilomar Report on Database Research, Technical Report MSTR-TR- 98-57, Microsoft Research, Microsoft Corporation, September 1998.

[2]. The Google Search Engine: Commercial search engine founded by the originators of PageRank. Located at http://www.google.com/.

[3]. S. Brin and L. Page (1998). The anatomy of a large-scale hypertextual Web search engine. *Proceedings of the Seventh International World Wide Web Conference*.

[4]. J. M. Kleinberg (1998). Authoritative sources in a hyperlinked environment. *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*.

[5]. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, 1998.

[6]. T. H. Haveliwala. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*, 2002.

[7]. M. Richardson and P. Domingos. *The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank*, volume 14. MIT Press, Cambridge, MA, 2002.

[8]. G. Jeh and J. Widom. Scaling personalized web search. *Stanford University Technical Report*, 2002.

[9]. G. Salton and M. J. McGill (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY.

[10]. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proceedings of the Seventh International World Wide Web Conference, 1998.

[11]. Sepandar Kamvar, Taher Haveliwala, and Gene Golub. Adaptive *Methods for the Computation of PageRank. Stanford University Technical Report*,1999.

[12]. Wenpu Xing and Ali Ghorbani *Weighted PageRank Algorithm ,University of New Brunswick*

[13]. S. Pal, V. Talwar, and P. Mitra. Web mining in soft computing framework : Relevance, state of the art and future directions. *IEEE Trans. Neural Networks*, 13(5):1163–1177, 2002.