AggregateRank: Bringing Order to Web Sites

Guang Feng[1,2*], Tie-Yan Liu[1], Ying Wang[3], Ying Bao[1,3*], Zhiming Ma[3], Xu-Dong Zhang[2] and Wei-Ying Ma[1]

[1]Microsoft Research Asia
5F, Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing, 100080, P. R. China
{tyliu, wyma}@microsoft.com

[2]Department of Electronic Engineering, Tsinghua University, Beijing 100084, P.R. china
{fengguang03@mails, zhangxd@}tsinghua.edu.cn

[3]Academy of Math and Systems Science, Chinese Academy of Science, Beijing 100080, China
{wangying, ybao}@amss.ac.cn; mazm@amt.ac.cn

ABSTRACT

Since the website is one of the most important organizational structures of the Web, how to effectively rank websites has been essential to many Web applications, such as Web search and crawling. In order to get the ranks of websites, researchers used to describe the inter-connectivity among websites with a so-called HostGraph in which the nodes denote websites and the edges denote linkages between websites (if and only if there are hyperlinks from the pages in one website to the pages in the other, there will be an edge between these two websites), and then adopted the random walk model in the HostGraph. However, as pointed in this paper, the random walk over such a HostGraph is not reasonable because it is not in accordance with the browsing behavior of web surfers. Therefore, the derivate rank cannot represent the true probability of visiting the corresponding website.

In this work, we mathematically proved that the probability of visiting a website by the random web surfer should be equal to the sum of the PageRank values of the pages inside that website. Nevertheless, since the number of web pages is much larger than that of websites, it is not feasible to base the calculation of the ranks of websites on the calculation of PageRank. To tackle this problem, we proposed a novel method named AggregateRank rooted in the theory of stochastic complement, which cannot only approximate the sum of PageRank accurately, but also have a lower computational complexity than PageRank. Both theoretical analysis and experimental evaluation show that AggregateRank is a better method for ranking websites than previous methods.

Categories and Subject Descriptors
H.3.3 [Information Search and Retrieval]: Search process

General Terms
Algorithms, Performance, Theory, Experimentation

Keywords:
AggregateRank, Stochastic Complement Theory, Coupling Matrix

*This work was done when Guang Feng and Ying Bao were interns at Microsoft Research Asia.

1. INTRODUCTION

In the traditional view, there are two kinds of essential elements that make up of the Web. One is the web page and the other is the hyperlink, corresponding to the content and structure of the Web respectively. In recent years, many researchers have realized that the website, another element of the Web, has played a more and more important role in the Web search and mining applications [1][2][3][4][5]. As compared to the individual web page, the website can sometimes provide plenty of semantic information in the following aspects. First, pages in the same website may share the same IP, run on the same web server and database server, and be authored / maintained by the same person or organization. Second, there might be high correlations between pages in the same website, in terms of content, page layout and hyperlinks. Third, from the topological point of view, websites contain higher density of hyperlinks inside them (about 75% according to [7]) and lower density of edges in between [6]. These properties make websites semantically important to understand the whole picture of the Web.

Actually, the ranking of websites have been a key technical component in many commercial search engines. On the one hand, in the service part, it can help define a more reasonable rank for web pages because those pages from important websites tend to be important as well. On the other hand, in the crawler part, it can help crawl pages from those important websites first, or help determine a quota (number of pages) for each website in the index according to its rank. In this way, with the same size of index in total, search engines can achieve the best tradeoff between index coverage and index quality.

In the literature of website ranking, people used to apply those technologies proposed for ranking web pages to the ranking of websites. For example, the famous PageRank algorithm, which was proposed by Brin and Page in 1998 [8] was used to rank websites in [9] and [10]. As we know, PageRank uses a random walk model in the Web graph to describe the behavior of Web surfers. That is, a random surfer is supposed to jump from a page to another following the hyperlinks with a probability of α, or jumps to a random page with a probability of $1-\alpha$. In order to apply PageRank to the ranking of websites, a HostGraph was constructed in these works, in which the nodes denote websites and the edges denote linkages between websites (if and only if there are hyperlinks from the pages in one website to the pages in the other, there will be an edge between these two websites). According to different definitions of the edge weights, two categories of HostGraphs were used in the literature. In the first category, the weight of an edge between two websites was defined by the number of hyperlinks between the two sets of web pages in these sites [11]. In the second category, the weight of any edge was simply set to 1 [12]. For the sake of clarity, we refer to the two categories as *weighted HostGraph* and *naïve HostGraph* respectively. Figure 1 and 2 show how these two categories of HostGraphs can be constructed.

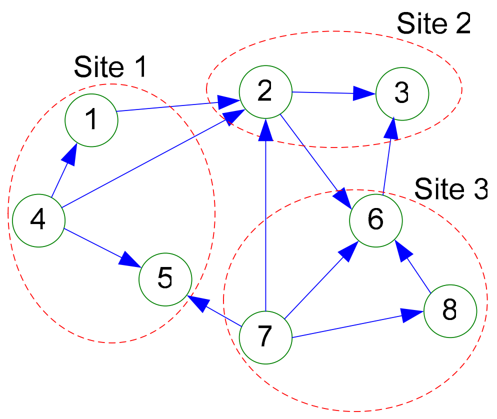

Figure 1. Illustration for web graph and website.

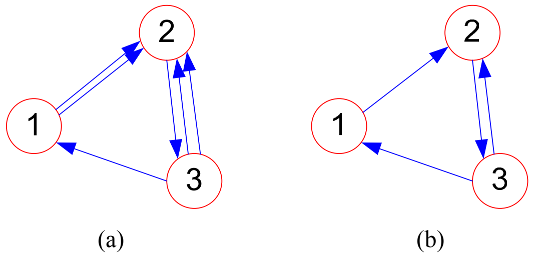

Figure 2. Illustration for (a) weighted HostGraph and (b) naïve HostGraph

After constructing the HostGraph, the similar random walk was conducted. That is, a random surfer was supposed to jump between websites following the edges with a probability of α, or jump to a random website with a probability of 1-α. In such a way, one can obtain the HostRank, which is the importance measure of websites.

At the first glance, the above random walk model over the HostGraph seems to be a natural extension of the PageRank algorithm. However, we want to point out that it is actually not as reasonable as PageRank because it is not in accordance with the browsing behavior of the Web surfers. As we know, real-world web surfers usually have two basic ways to access the web. One is to type URL in the address edit of the web browser. And the other is to click any hyperlink in the current loaded page. These two manners can be well described by the parameter α used in PageRank. That is, with a probability of 1-α, the web users visit a random web page by inputting its URL (using favorite folder can also be considered as a shortcut of typing URL), and with a probability of α, they visit a web page by clicking a hyperlink. Nevertheless, as for the random walk in the HostGraph, we can hardly find the same evident correlation between the random walk model and real-world user behaviors. For example, even if there is a edge between two websites *A* and *B* in the HostGraph, when a web surfer visits a page in website *A*, he may not be able to jump to website *B* because the hyperlink to website *B* may exist in another page in website *A* which is even unreachable from the page that he is currently visiting. In other words, the HostGraph is only a kind of approximation to the web graph: it loses much transition information, especially as for the *naïve HostGraph*. As a result, we argue that the rank values derived from the aforementioned HostGraph are not convincing enough, and a more reasonable way to define the ranks of a website should be the probability that the users visit a random page in that website. Actually, it can be proved that this probability is equal to the sum of the PageRanks of all the pages in that website (denoted by PageRankSum for ease of reference).

However, it is clear that using PageRankSum to calculate the ranks of websites is not yet a feasible solution, especially for those applications that only care about the websites (i.e. those site ranking services such as http://www.alaxa.com). The reason is that the number of web pages (usually measured in billion. Google announced that it indexed over 8 billions pages, and Yahoo! claimed an even larger index of 20 billions [13]) is much larger than the number of websites (usually measured in million. According to [14], there are around 70 million websites by the end of 2005). Therefore, it is much more complex to rank web pages than to rank websites, and it is almost impossible for small research groups or companies to afford such kind of expensive computations.

To tackle these aforementioned problems, we propose a novel method based on the theory of stochastic complement [15] to calculate the rank of a website that can approximate the PageRankSum accurately, while the corresponding computational complexity is lower than PageRankSum. We name this algorithm by AggregateRank. Experiments demonstrated the effectiveness and efficiency of this algorithm.

The rest of this paper is organized as follows. In Section 2, we give the specific probability explanation for AggregateRank with the random walk model, and then conduct convergence rate analysis and error bound analysis. Then in Section 3, some experiments with a real web page collection are shown in details. Conclusions and future work are discussed in Section 4.

2. RANK FOR WEBSITES

As aforementioned, website ranking is a very important technology for Web search. However, traditional approaches on calculating website ranks lost some transition information of the random surfer. To tackle this problem, we will propose our solution in this section. In particular, we will first discuss how to define a more reasonable website rank, and then propose a method for efficient calculation of this rank. After that, we will also discuss the error bound and convergence rate of the proposed method.

2.1 Probabilities of Visiting a Website

In this subsection, we will discuss what the probability of visiting a particular website is, under the framework of Marokv random walk. For this purpose, we need to define the transition probability matrix, called coupling matrix, between websites. And then use the stationary distribution of the coupling matrix to denote the probabilities that the random surfer visits each website.

For the sake of the convenience of conduction, we intend to describe the random surfer model in mathematics at the beginning. The random surfer model can be formally described with a Markov chain $\{X_k\}_{k\geq 0}$ whose state space is the set of the web pages. The current state of the Markov chain is P_l if and only if the random surfer is staying at the l^{th} page at current time. The evolution of the Markov chain represents the surfing behavior of the random surfer from one Web page to another. Suppose there are N sites, n pages, and r hyperlinks in total. Note that according to [11], $r \approx 10n$. If there is a hyperlink from page i to page j, the probability transiting from P_i to P_j is equal to

$$p_{ij} = \alpha \cdot \frac{1}{d_i} + (1-\alpha) \cdot \frac{1}{n} \tag{1}$$

otherwise,

$$p_{ij} = (1-\alpha) \cdot \frac{1}{n} \tag{2}$$

where d_i denotes the out-degree of page i. Usually, this transition probability matrix[1] of the surfing Markov chain is denoted by $P(\alpha)=(p_{ij}(\alpha))$. The parameter α, usually set to 0.85, is called the damping factor. The PageRank is the stationary distribution of this Markov chain $\{X_k\}_{k\geq 0}$, or the principle eigenvector of $P(\alpha)$ [16].

For the next step, we will discuss how to induce the transition probabilities between websites from the aforementioned page-level Markov chain $\{X_k\}_{k\geq 0}$. For this purpose, we proceed as follows.

As each page belongs to some determinate site, we rearrange the transition probability matrix $P(\alpha)$ to another $n\times n$ transition probability matrix $Q(\alpha)$, which can be partitioned into $N\times N$ blocks according to the N sites and has the following form,

$$Q(\alpha) = \begin{pmatrix} Q_{11}(\alpha) & Q_{12}(\alpha) & \cdots & Q_{1N}(\alpha) \\ Q_{21}(\alpha) & Q_{22}(\alpha) & \cdots & Q_{2N}(\alpha) \\ \vdots & \vdots & \ddots & \vdots \\ Q_{N1}(\alpha) & Q_{N2}(\alpha) & \cdots & Q_{NN}(\alpha) \end{pmatrix} \tag{3}$$

where elements in each diagonal block denote the transition probabilities between pages in the same website, and elements in each off-diagonal block denote the transition probabilities between pages in different websites. The diagonal blocks $Q_{ii}(\alpha)$ are square and of order n_i (n_i is the number of pages in website i), and $n = \sum_{i=1}^{N} n_i$. The stationary distribution $\phi(\alpha)$ of $Q(\alpha)$, i.e. the PageRank, is given by

$$\phi(\alpha)Q(\alpha) = \phi(\alpha) \text{ with } \phi(\alpha)e = 1 \tag{4}$$

where e is the column vector with all elements equal to 1. Partitioning $\phi(\alpha)$ conformably with $Q(\alpha)$, we get $\phi(\alpha)=\{\phi_1(\alpha), \phi_2(\alpha), \ldots, \phi_N(\alpha)\}$, in which $\phi_i(\alpha)$ is a row vector of length n_i, for i=1, …, N.

Till now, we just get a *rearranged* PageRank vector. To move forward, we will turn our attention to the probability that the random surfer visits site i (denoted by S_i for ease of reference, i=1, …, N), when the page-level surfing Markov chain attains equilibrium after long-time evolution. We assume that the Markov chain $\{X_k\}_{k\geq 0}$ is started with the stationary probability vector $\phi(\alpha)$ [17]. Then, the one-step transition probability from S_i to S_j is defined by

$$c_{ij}(\alpha) = \Pr_{\phi(\alpha)}\{X_{t+1} \in S_j \mid X_t \in S_i\}. \tag{5}$$

By the properties of conditional probability and the definition of $c_{ij}(\alpha)$, we can get

$$c_{ij}(\alpha) = \frac{\phi_i(\alpha)}{\|\phi_i(\alpha)\|_1} Q_{ij}(\alpha) e \tag{6}$$

where $Q_{ij}(\alpha)$ is the ijth block of the one-step transition matrix $Q(\alpha)$.

The $N\times N$ matrix $C(\alpha)=(c_{ij}(\alpha))$ is referred to as the coupling matrix, whose elements represent the transition probabilities between websites. It can be proved that $C(\alpha)$ is an irreducible stochastic matrix [15], so that it possesses a unique stationary probability vector. We use $\xi(\alpha)$ to denote this stationary probability, which can be got from

$$\xi(\alpha)C(\alpha) = \xi(\alpha) \text{ with } \xi(\alpha)e = 1 \cdot \tag{7}$$

One may have realized that the above computation can also be regarded as being carried out with a certain HostGraph. However, the edge weight of this new HostGraph is not decided heuristically as in previous works [9][10], but determined by a sophisticated formulation (6). Besides, the transition probability from S_i to S_j actually summarizes all the cases that the random surfer jumps from any page in S_i to any page in S_j within any transition steps. Therefore, the transition in this new HostGraph is in accordance with the real behavior of the Web surfers. In this regard, the so-calculated rank from the coupling matrix $C(\alpha)$ will be more reasonable than those previous works.

After describing how the probability of visiting a website can be defined, we will try to get its solution. Actually it is easy to validate that $\xi(\alpha)=\{\|\phi_1(\alpha)\|_1, \|\phi_2(\alpha)\|_1, \ldots, \|\phi_N(\alpha)\|_1\}$ is a solution to (7)[2]. Because the coupling matrix is an irreducible stochastic matrix, its stationary distribution is unique. So, $\xi(\alpha)=\{\|\phi_1(\alpha)\|_1, \|\phi_2(\alpha)\|_1, \ldots, \|\phi_N(\alpha)\|_1\}$ is the unique solution to (7). That is, the probability of visiting a website is equal to the sum of PageRanks of all the pages in that website. This conclusion is consistent to our intuition.

Based on the above discussions, the direct approach of computing the ranks of websites is to accumulate PageRank values (denoted by PageRankSum). However, this approach is unfeasible because the computation of PageRank is not a trivial task when the number of web pages is as large as several billions. Therefore, efficient computation becomes a significant problem . In the next subsection, we will propose an approximate algorithm for this purpose, which can be more efficient than PageRankSum with very little accuracy loss.

2.2 The AggregateRank Algorithm

In this section, we will discuss how to compute the aforementioned probabilities of visiting a website in an efficient manner. First of all, we will discuss how to use the theory of stochastic complement to speed up the calculation of PageRankSum, and then we will discuss the convergence speed and the error bound of the proposed algorithm.

[1] In fact, there are some details in PageRank computation [16], such as how to handle dangling nodes.

[2] $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$, or the sum of the absolute value of each element in the row vector $\boldsymbol{x}$.

2.2.1 Computation Issues and Solutions

As aforementioned, the coupling matrix $C(\alpha)$, with $c_{ij}(\alpha) = \frac{\phi_i(\alpha)}{\|\phi_i(\alpha)\|_1} Q_{ij}(\alpha)e$, represents the transition probability from one site to the other in terms of random surfer model; and its stationary distribution $\xi(\alpha)$, which is equal to PageRankSum, is regarded as a reasonable rank of websites.

It is clear that the construction of the coupling matrix asks for the calculation of PageRank over the whole Web graph. To avoid this time-consuming step, it is necessary to invent a new method to construct the coupling matrix $C(\alpha)$. Fortunately, the theory of stochastic complement [15] gives a good solution to approximate $C(\alpha)$ without PageRank values. To illustrate this, we take a simple web graph for example. Suppose the web graph only contains two websites, and the transition probability matrix of this Web graph (rearranged according to website information) can be denoted by

$$Q(\alpha) = \begin{pmatrix} Q_{11}(\alpha) & Q_{12}(\alpha) \\ Q_{21}(\alpha) & Q_{22}(\alpha) \end{pmatrix} \tag{8}$$

And its stationary distribution is $\phi(\alpha)=\{\phi_1(\alpha), \phi_2(\alpha)\}$. For any diagonal block in $Q(\alpha)$, we can calculate its stochastic complement. For example, the stochastic complement of $Q_{11}(\alpha)$ is defined as below,

$$S_{11}(\alpha) = Q_{11}(\alpha) + Q_{12}(\alpha)\left(I - Q_{22}(\alpha)\right)^{-1} Q_{21}(\alpha) \tag{9}$$

The stochastic complement is also a stochastic matrix, each row of which is summed up to 1. It can be proved that $\phi_1(\alpha)/\|\phi_1(\alpha)\|_1$ is the unique stationary probability vector for the stochastic complement $S_{11}(\alpha)$, i.e.

$$\frac{\phi_1(\alpha)}{\|\phi_1(\alpha)\|_1} S_{11}(\alpha) = \frac{\phi_1(\alpha)}{\|\phi_1(\alpha)\|_1} \quad \text{with} \ . \ \frac{\phi_1(\alpha)}{\|\phi_1(\alpha)\|_1} e = 1 \tag{10}$$

Generally, $\phi_i(\alpha)/\|\phi_i(\alpha)\|_1$ is the unique stationary distribution for the stochastic complement $S_{ii}(\alpha)$, i.e.

$$\frac{\phi_i(\alpha)}{\|\phi_i(\alpha)\|_1} S_{ii}(\alpha) = \frac{\phi_i(\alpha)}{\|\phi_i(\alpha)\|_1} \quad \text{with} \quad \frac{\phi_i(\alpha)}{\|\phi_i(\alpha)\|_1} e = 1 \tag{11}$$

Intuitively, the computation of the stationary distribution of each $S_{ii}(\alpha)$ will be cheaper than that of PageRank directly because the dimension of each $S_{ii}(\alpha)$ is very small and equal to the page number in this site (we will discuss this in more detail in the next subsection). However, it is time consuming to compute the exact stochastic complement since we should compute a contradict matrix for I- $Q_{ii}(\alpha)$. As we know, the computation for contradict matrix is very expensive. Thus, we prefer an approximate approach to get the stationary distribution of each stochastic complement instead. According to [15], there is such an efficient method. It does not use (9) to aggregate the stochastic complement directly. Instead, it only modifies each diagonal block $Q_{ii}(\alpha)$ by a little to get a new matrix with the same dimension as $S_{ii}(\alpha)$. The details are given as follows.

For the first step, we modify the original diagonal block $Q_{ii}(\alpha)$ to be a transition probability matrix. It is clear that the sum of each row in the original diagonal block $Q_{ii}(\alpha)$ is always less than 1. To make it a transition probability matrix, we simple adjust the diagonal elements of $Q_{ii}(\alpha)$ (added or subtracted by a small value) to make the sum of each row equal to 1. Letting $Q^*_{ii}(\alpha)$ denote the matrix after adjustment, we can calculate its stationary distribution $u_i(\alpha)$ as follows,

$$u_i(\alpha)Q^*_{ii}(\alpha) = u_i(\alpha) \text{ with } u_i(\alpha)e = 1 \tag{12}$$

According to [15], we can prove that

$$\frac{\phi_i(\alpha)}{\|\phi_i(\alpha)\|_1} \approx \frac{u_i(\alpha)}{\|u_i(\alpha)\|_1} . \tag{13}$$

From the description above, $Q^*_{ii}(\alpha)$ is very easy to get from $Q_{ii}(\alpha)$. Moreover, it can even be stored sparsely like original $Q(\alpha)$. Thus, formulation (13) means that we can get each $\phi_i(\alpha)/\|\phi_i(\alpha)\|_1$ very efficiently.

Utilizing the result of (13), we can obtain an approximate coupling matrix $C^*(\alpha)$ as below,

$$\left(C^*(\alpha)\right)_{ij} = \frac{u_i(\alpha)}{\|u_i(\alpha)\|_1} Q_{ij}(\alpha)e \tag{14}$$

Consequently, the stationary distribution $\xi^*(\alpha)$ of the approximate coupling matrix can be regarded as a good approximation to $\xi(\alpha)$. We name the aforementioned algorithm by AggregateRank, which detailed algorithm flow is shown as in Figure 3.

1. Divide the $n \times n$ matrix $Q(\alpha)$ into $N \times N$ blocks according to the N websites.
2. Construct the stochastic matrix $Q^*_{ii}(\alpha)$ for $Q_{ii}(\alpha)$ by changing the diagonal elements of $Q_{ii}(\alpha)$ to make each row of $Q^*_{ii}(\alpha)$ summed up to 1.
3. Use power method to calculate $u_i(\alpha)$ from
$$u_i(\alpha)Q^*_{ii}(\alpha) = u_i(\alpha) \text{ with } u_i(\alpha)e = 1 \tag{15}$$
4. Construct the approximation coupling matrix $C^*(\alpha)$ as follows
$$\left(C^*(\alpha)\right)_{ij} = u_i(\alpha)Q_{ij}(\alpha)e \tag{16}$$
5. Use power method to get $\xi^*(\alpha)$
$$\xi^*(\alpha)C^*(\alpha) = \xi^*(\alpha) \text{ with } \xi^*(\alpha)e = 1 \tag{17}$$

Figure 3. Algorithm Description of AggregateRank.

To sum up, the proposed algorithm improves the efficiency in the following ways. First, it uses a much smaller matrix, i.e. the stochastic complement, to compute each $\phi_i(\alpha)/\|\phi_i(\alpha)\|_1$ instead of the whole transition probability matrix. Second, it uses a easy-to-construct sparse matrix to replace the stochastic complement to approximate $\phi_i(\alpha)/\|\phi_i(\alpha)\|_1$. Third, this algorithm is much easier to be implemented in parallel than PageRankSum. The reason is that for PageRank, if we want to implement it in parallel, we must take care of the information exchange between different servers since there are hyperlinks which sources and destinations are not in the same server. While, for our method, we do not need the exchange of information if we put a website at most in one sever since the computations over $Q^*_{ii}(\alpha)$ is done for each particular website and is independent of other part of the whole web graph.

2.2.2 Complexity Analysis

As can be seen in the previous sections, in our proposed AggregateRank algorithm, we divide the Web graph into websites, and conduct power-method iterations within each website. After that, we apply power method once again to the coupling matrix $C^*(\alpha)$. It is easy to understand that, in this way, we can save some memory and the corresponding algorithm is easier to be implemented in parallel. When we deal with the Web graph with billions of pages, this advantage will become very meaningful.

However for the computational complexity, it is not obvious whether the proposed method can be more efficient. The reason is that PageRank has a complexity of $O(r)$. Considering that 75% of the hyperlinks connect pages in the same website [7], dividing the Web graph into websites can only save 25% propagations along hyperlinks and thus the complexity is still around $O(r)$. Furthermore, for the computation in Step 5, it is not obvious whether $C^*(\alpha)$ is also a sparse matrix, thus its computational complexity might be as high as $O(N^2)$ in the worse case. All of this can be a big issue.

In this subsection, we will discuss the aforementioned problems in detail. Specifically, we will prove in Subsection 2.2.2.1 that although the overall complexity of one iteration of power method applied to all $Q^*_{ii}(\alpha)$ (i=1,…,N) is also $O(r)$, its convergence speed can be significantly faster than PageRank over the whole Web graph. And then we will prove in Subsection 2.2.2.2 that $C^*(\alpha)$ is actually also a sparse matrix, and there are only O(N) non-zero elements in this matrix. Therefore, the computation of calculating stationary distribution for this matrix will also be faster than that for the PageRank matrix.

2.2.2.1 Convergence Speed Analysis for Power Method Applied to $Q^*_{ii}(\alpha)$

In order to understand the convergence speed of the power method applied to $Q^*_{ii}(\alpha)$, we need to review the following Lemma at first. As we know, the convergence speed of the power method is determined by the magnitude of the subdominant eigenvalue of the transition probability matrix [18]. Lemma 1 just tells us the relationship between matrix $P(\alpha)$ and its eigenvalues [16].

Lemma 1 (Langville and Meyer)

Given the spectrum of the stochastic matrix P as $\{1, \lambda_2, \lambda_3, \ldots, \lambda_n\}$, *the spectrum of the primitive stochastic matrix* $P(\alpha)=\alpha P+(1-\alpha)ev^T$ *is* $\{1, \alpha\lambda_2, \alpha\lambda_3, \ldots, \alpha\lambda_n\}$, *where* v^T *is a probability vector.*

In order to use Lemma 1 to analyze the convergence speed of the power method applied to $Q^*_{ii}(\alpha)$, we transform $Q^*_{ii}(\alpha)$ into the following form

$$Q^*_{ii}(\alpha)=\alpha\overline{Q}^*_{ii}+(1-\alpha)\frac{1}{n_i}ee^T, \tag{18}$$

where $\overline{Q}^*_{ii}$ is a stochastic matrix and e^T/n_i is a probability vector.

Given the eigenvalues of $\overline{Q}^*_{ii}$ as $\{1,\tilde{\lambda}_2,\tilde{\lambda}_3,\cdots,\tilde{\lambda}_{n_i}\}$, by Lemma 1, we can get that the eigenvalues of $Q^*_{ii}(\alpha)$ is $\{1,\alpha\tilde{\lambda}_2,\alpha\tilde{\lambda}_3,\cdots,\alpha\tilde{\lambda}_{n_i}\}$. Since the convergence speed of the power method is determined by the magnitude of the subdominant eigenvalue of $Q^*_{ii}(\alpha)$, we can conclude that the convergence rate of the power method applied to $Q^*_{ii}(\alpha)$ is around the rate at which $(\alpha\tilde{\lambda}_2)^k\to 0$.

As we know, the convergence speed of PageRank is around the rate at which $\alpha^k\to 0$. So whether we can be more efficient than PageRank is determined by how small $\tilde{\lambda}_2$ could be. According to the following discussions, we know that $\tilde{\lambda}_2\ll 1$.

As we know, the web link graph has a natural block structure: the majority of hyperlinks are intra-site links [19]. Therefore, the random walk on the web with the transition matrix $Q(\alpha)$ can be viewed as a nearly completely decomposable Markov chain. According to [15], we know that when the states in a nearly completely decomposable Markov chain are naturally ordered, and when the transition matrix is partitioned into N closely coupled subclasses in the natural way, the underlying transition matrix of the Markov chain has exactly N-1 non-unit eigenvalues clustered near λ=1 (There are pathological cases, but they are rare in practical work [15]). Thus $Q(\alpha)$ has exactly N-1 non-unit eigenvalues clustered near λ=1.

Since $Q^*_{ii}(\alpha)$ is an irreducible stochastic matrix, the Perron-Frobenius theorem [20] guarantees that the unit eigenvalue of each $Q^*_{ii}(\alpha)$ is simple. Because $Q^*_{ii}(\alpha)\approx Q_{ii}(\alpha)$, by the continuity of the eigenvalues, the non-unit eigenvalues of $Q^*_{ii}(\alpha)$ must be rather far from the unit eigenvalue of $Q^*_{ii}(\alpha)$. Otherwise the spectrum of $Q(\alpha)$ would contain a cluster of at least N non-unit eigenvalues positioned near λ=1. As a result, we can come to the conclusion that $\alpha\tilde{\lambda}_2\ll 1$, and then $\tilde{\lambda}_2\ll 1$. That is, the convergence speed of power method applied to $Q^*_{ii}(\alpha)$ is much faster than that of PageRank.

2.2.2.2 Complexity of Power Method Applied to $C^*(\alpha)$

As mentioned at the beginning of this section, the sparseness of the matrix $C^*(\alpha)$ is a critical factor that influences the computational complexity of our proposed AggregateRank algorithm. To understand this, we conduct the following discussions.

First of all, we transform $Q(\alpha)$ into the following form

$$\begin{aligned}Q(\alpha)&=\alpha\overline{Q}+(1-\alpha)\frac{1}{n}ee^T\\&=\alpha(Q+a\frac{1}{n}e^T)+(1-\alpha)\frac{1}{n}ee^T\\&=\alpha Q+(\alpha a+(1-\alpha)e)\frac{1}{n}e^T,\end{aligned} \tag{19}$$

where Q is the transition matrix whose element q_{ij} is the probability of moving from webpage i to webpage j in one step following the hyperlink structure of the Web Graph, a is a vector whose element a_i=1 if row i of Q corresponds to a dangling node, and 0, otherwise, α is damping factor, and e is a column vector of all ones. Then we investigate the construction process of $C^*(\alpha)$ as follows,

$$\begin{aligned}C^*(\alpha)&=U(\alpha)Q(\alpha)V\\&=U(\alpha)(\alpha Q+(\alpha a+(1-\alpha)e)\frac{1}{n}e^T)V\\&=\alpha(U(\alpha)QV)+(\alpha U(\alpha)a+(1-\alpha)e)v^T,\end{aligned} \tag{20}$$

where

$$U(\alpha)=\begin{pmatrix}u_1(\alpha)&&&\\&u_2(\alpha)&&\\&&\ddots&\\&&&u_N(\alpha)\end{pmatrix}_{N\times n},\quad V=\begin{pmatrix}e&&&\\&e&&\\&&\ddots&\\&&&e\end{pmatrix}_{n\times N},$$

and $v^T = (\frac{n_1}{n}, \frac{n_2}{n}, \cdots, \frac{n_N}{n})$ is a probability vector.

According to this decomposition, in Step 4, we actually only need to compute $A=:U(\alpha)QV$. The corresponding count of multiplications is $O(r)$. Note that we do not need any iteration here, so the complexity of Step 4 is actually much lower than PageRank that will take tens or even hundreds of iterations to converge.

In Step 5, for any starting vector $\xi_i^{(0)}(\alpha)$,

$$\begin{aligned}\xi^{*(k)}(\alpha) &= \xi^{*(k-1)}(\alpha)C^*(\alpha)\\ &= \xi^{*(k-1)}(\alpha)U(\alpha)Q(\alpha)V\\ &= \xi^{*(k-1)}(\alpha)U(\alpha)(\alpha(Q + a\frac{1}{n}e^T) + (1-\alpha)\frac{1}{n}ee^T)V\\ &= \alpha\xi^{*(k-1)}(\alpha)A + (\alpha\xi^{*(k-1)}(\alpha)U(\alpha)a + (1-\alpha))v^T,\end{aligned} \tag{21}$$

Then it is clear that the computational complex of each iteration in Step 5 depends on the number of non-zeroes in A. Because a_{ij} equals to the linear combination of the elements in block Q_{ij}, we have a_{ij}=0 when every element is 0 in Q_{ij}. Suppose that the average number of sites that a particular website links to is μ, then A has μ non-zeroes in each row and the number of non-zeroes in A is μN. Considering that for the Web, μ is almost a constant which is tens or so [11], we can come to the conclusion that the computational complex of one iteration in Step 5 is $O(N) << O(r)$.

2.2.3 *Error Analysis*

As one can see, the proposed AggregateRank algorithm is an approximation to PageRankSum. In this subsection, we will discuss the error bound of this approximation.

According to the theory of stochastic complement, the approximation as shown in (13) requires the matrix to be nearly completely decomposable. This condition is well satisfied in real Web applications, because about 75% of the hyperlinks connect pages in the same website [7], and it is reasonable to treat the transition probability matrix $Q(\alpha)$ as a nearly completely decomposable matrix.

According to the discussions in Section 2.2.2, $Q(\alpha)$ has exactly N-1 non-unit eigenvalues that are very close to the unit eigenvalue. Thus, the approximation in (13) has an upper error bound according to [21], which is determined by the number of pages n, the number of sites N, the size of each site n_i, and the condition number of the coupling matrix $\kappa(C(\alpha))$, the deviation from complete reducibility δ[3] and the eigen structure of the probability transition matrix $Q(\alpha)$.

Theorem 1

If $Q(\alpha)$ has exactly N-1 non-unit eigenvalues close to the unit eigenvalue, then there are the following error bounds:

$$\left\| \frac{\phi_i(\alpha)}{\|\phi_i(\alpha)\|} - \frac{u_i(\alpha)}{\|u_i(\alpha)\|_1} \right\|_1 \le \delta(n_i - 1) \tag{22}$$

$$\left\| C(\alpha) - C^*(\alpha) \right\|_\infty \le \delta^2 \left(\max_i n_i - 1 \right) \tag{23}$$

[3] Not as popular as other concepts, the deviation from complete reducibility is defined in [21].

$$\left\| \xi(\alpha) - \xi^*(\alpha) \right\|_1 \le \delta^2 \kappa(C(\alpha)) \left(\max_i m_i - 1 \right) \tag{24}$$

From Theorem 1, we can see that the upper error bound of the AggregateRank algorithm principally depends on the scale of the largest website. Evidently, the number of pages in the largest website is much smaller than the size of the Web, i.e. m=$o(n)$. In this regard, we can say that the corresponding error bound is well controlled.

3. EXPERIMENTS

In our experiments, the data corpus is the benchmark data for the Web track of TREC 2003 and 2004, which was crawled from the .gov domain in the year of 2002. It contains 1,247,753 pages in total.

Before testing our proposed method, we need to partition the Web graph into websites. For this purpose, we follow the rules as below. Because the URLs in the .gov domain are very regular, we can easily decide the site that a page belongs to. After removing the *http://* or *https://* from the URL, the rest part before the first slash can be considered as the site name. However, because there maybe some subsites, we only use the adjacent word before .gov as the identifier for the site. For example, http://aaa.bbb.gov/xxxx belongs to the website *bbb*.

After this preprocess, we get 731 sites in the .gov dataset. The largest website contains 137,103 web pages while the smallest one contains only 1 page. The distribution of the sizes of all the websites is shown in Figure 4. It nearly follows a power law and is consistent with previous research on the sizes of websites [22].

In our experiment, we validated whether the proposed AggregateRank algorithm can well approximate PageRankSum. For comparison, we also investigated other two HostRank algorithms, which work on the weighted HostGraph and naïve HostGraph respectively [9]. The differences between PageRankSum and the algorithms under investigation are shown in Table 1, in terms of Euclidean distance between the rank vectors.

From Table 1, we can see that the AggregateRank algorithm has the best performance: its Euclidean distance from PageRankSum is only 0.0057, while the ranking results produced by the other two algorithms are farther from PageRankSum, with Euclidean distances of 0.1125 and 0.1601 respectively.

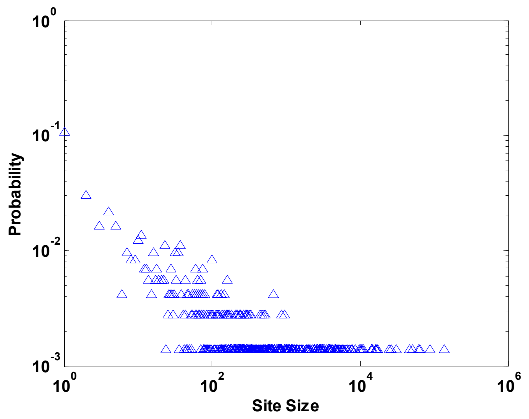

Figure 4. The distribution of the sizes of websites.

Table 1. Performance Evaluation based on Euclidean Distance

Ranking algorithms	*Euclidean distance*	*Max distance in one single dimension*	*Min distance in one single dimension*
PageRankSum	0.0	0.0	0.0
AggregateRank	0.0057	0.0029	0.000000
Weighted HostRank	0.1125	0.0805	0.000020
NaïveHostRank	0.1601	0.1098	0.000007

In addition to the Euclid distance, we also used another similarity measure based on the Kendall's τ distance [23] to evaluate the performance of the ranking results. This measure ignores the absolute values of the ranking scores and only counts the partial-order preferences. Therefore, it can better reflect the true ranking performance in real applications. This similarity between two ranking lists *s* and *t* is defined as follows.

$$Sim(s,t) = 1 - \frac{K(s,t)}{C_n^2} \tag{25}$$

where *K*(*s*,*t*) is the *Kendall's τ distance*, which counts the number of pair-wise disagreements between *s* and *t*, and is defined as below.

$$K(s,t) = \left|\{(i,j) \mid i < j, s(i) < s(j), t(i) > t(j)\}\right| \tag{26}$$

According to the definition, the larger *Sim*(*s*,*t*) is, the more similar two lists are. If the two ranking lists are consistent with each other, their *Sim* measure is equal to 1.

We list the performance evaluation of the aforementioned algorithm based on Kendall's τ distance in Table 2. From this table, once again we can see that the AggregateRank algorithm is the best approximation to PageRankSum. Furthermore, the advantage of the AggregateRank algorithm over the reference algorithms becomes even more obvious if we look at the top-*k* ranking results. Actually, we got the top *k* sites in terms of their PageRankSum, and obtained their order O_p. Then, we got their relative orders according to other ranking algorithms, i.e. O_a, O_w and O_n which correspond to the AggregateRank algorithm, the weighted HostRank algorithm and the naïve HostRank algorithm respectively. We plot the similarity based on the Kendall's τ distance between these top-*k* ranking results in Figure 5.

Table 2. Performance Evaluation of Ranking Algorithms based on Kendall's τ distance

Ranking Algorithms	**Sim**
PageRankSum	1
AggregateRank	0.9826
Weighted HostRank	0.8428
Naïve HostRank	0.8889

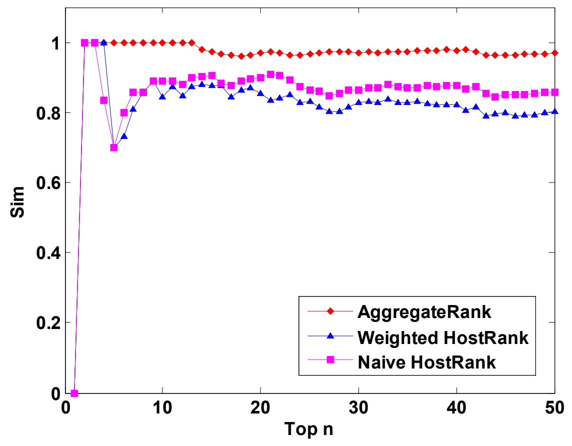

Figure 5. Similarity between PageRankSum and other three ranking results.

After the comparison on similarity, we compare these ranking algorithms on complexity as well. As discussed in Section 2, the AggregateRank algorithm can converge faster than PageRank. To verify this, we use the L_1-norm of the difference between the current ranking list and the last one to measure whether the power method converges. When this difference is less than 10^{-3}, we regard the computation as converged and the computation process is terminated. The running time of each algorithm is shown in Table 3.

Table 3. Comparison of Running Time

Ranking Algorithms	*Running Time (s)*
PageRankSum	116.23
AggregateRank	29.83
Weighted HostRank	0.22
Naïve HostRank	0.10

From Table 3, we can see that the proposed AggregateRank method is faster than PageRankSum, while a little more complex than the HostRank methods. This is consistent with the theoretical analysis in the previous section. The fast speed of AggregateRank mainly comes from the fast convergence speed. And the fast speed of HostRank comes from the low dimension of the HostGraph.

In summary, by taking the effectiveness and efficiency into consideration at the same time, we consider the proposed AggregateRank algorithm as a better solution to website ranking.

4. CONCLUSIONS AND FUTURE WORKS

Considering that website is an important organizational structure of the Web, we discussed how to rank websites in this paper. In particular, we pointed out that the widely-used approach which used a simple HostGraph to represent the inter-connectivity between websites and adopt the random walk model to get the rank of the websites is not in accordance with the browsing behavior of real Web surfers. As a result, the so-calculated rank of the websites cannot reflect the probability of visiting the corresponding website. In order to solve this problem, we proposed a novel algorithm named AggregateRank, and proved mathematically that it can well approximate the probability of visiting a website. We also investigate the convergence speed and error bound of this proposed method, which indicates that it is not only effective but also very efficient. Experiments on the benchmark dataset for TREC 2003 Web track verified the aforementioned theoretical findings.

5. REFERENCES

[1] Despeyroux, T. Practical Semantic Analysis of Web Sites and Documents. In *Proceedings of the Thirteenth International World Wide Web Conference*, New York, USA, May 2004.

[2] Hasan Davulcu, Srinivas Vadrevu, Saravanakumar Nagarajan. OntoMiner: Bootstrapping Ontologies From Overlapping Domain Specific Web site. In *Proceedings of the Thirteenth International World Wide Web Conference*, New York, USA, May 2004.

[3] Kristina Lerman, Lise Getoor, Steven Minton, Craig Knoblock. Using the Structure of Web Sites for Automatic Segmentation of Tables. In *Proceedings of the ACM International Conference on Management of Data*, Paris, France, June 2004.

[4] Yuangui Lei, Enrico Motta, Domingue. Modelling Data-Intensive Web Sites with OntoWeaver. In *Proceedings of International Workshop on Web Information Systems Modeling*, Riga, Latvia, June 2004.

[5] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, Guang Feng, Wei-Ying Ma, Subsite Retrieval: A Novel Concept for Topic Distillation, In Proceedings of 2nd Asia Information Retrieval Symposium, Jeju Island, Korea, October 2005

[6] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA, pages 7821--7826, 2002.

[7] Henzinger, M.R., Motwani, R., Silverstein, C. Challenges in Web Search Engines. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (2003) 1573-1579

[8] L. Page, S. Brin, R. Motwani, and T. Winograd. The Pagerank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Libraries, 1998.

[9] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceedings of the 13th International World Wide Web Conference* (*WWW*), pages 309–318, New York, NY, USA, 2004. ACM Press.

[10] Jie Wu, Karl Aberer. Using SiteRank for P2P Web Retrieval. EPFL Technical Report ID: IC/2004/31, 2004

[11] Krishna Bharat, Bay-Wei Chang, Monika Henzinger, and Matthias Ruhl. Who links to whom: Mining linkage between web sites. In *Proceedings of the IEEE International Conference on Data Mining (ICDM '01)*, San Jose, USA, November 2001.

[12] S. Dill, R. Kumar, K. McCurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the Web. In *Proceedings of International Conference on Very Large Data Bases*, pages 69--78, Rome, 2001.

[13] http://www.researchbuzz.com/2005/09/google_celebrates_7_where_did.shtml

[14] http://www.netcraft.com

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

[16] Amy N. Langville and Carl D. Meyer. Deeper inside PageRank. Technical report, NCSU Center for Res. Sci Comp., 2003.

[17] John G Kemeny and J.Laurie Snell, Finite Markov Chains Springer-Verlag, New York Berlin Herdelberg Tokyo, 1976.

[18] William J. Stewart. Introduction to the Numerical Solution of Markov Chains. Princeton : Princeton University Press, 1994

[19] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford Univ., 2003.

[20] F. R. Gantmacher, *Matrix Theory* (Chelsea, 1959) Vol. II, Chapter XIII.

[21] G. E. Cho and C. D. Meyer. Aggregation/Disaggregation Methods of Nearly Uncoupled Markov Chains. http://meyer.math.ncsu.edu/Meyer/PS_Files/Numcad.ps

[22] R. Albert and A.-L. Barabasi. Statistical mechanics of complex networks. Rev. Mod. Phys. Vol. 74, January 2002.

[23] M. Kendall and J. Gibbons. Rank Correlation Methods. Edward Arnold, London, 5 edition, 1990.