Nonlinear Algorithms for Static-Rank Computation

Bin Lu[1*], Shuming Shi[2], Yunxiao Ma[2], Ji-Rong Wen[2]
[1] Peking University
[2] Microsoft Research Asia
[1]melonskin@live.com, [2]{shumings, yunxiaom, jrwen}@microsoft.com

ABSTRACT
Mainstream link-based static-rank algorithms (e.g. PageRank and its variants) express the importance of a page as the linear combination of its in-links and compute page importance scores by solving a linear system in an iterative way. Such linear algorithms, however, may give apparently unreasonable static-rank results for some link structures. In this paper, we examine the static-rank computation problem from the viewpoint of evidence combination and build a probabilistic model for it. Based on the model, we argue that a nonlinear formula should be adopted, due to the correlation or dependence between links. We focus on examining some simple formulas which only consider the correlation between links in the same domain. Experiments conducted on 100 million web pages (with multiple static-rank quality evaluation metrics) show that higher quality static-rank could be yielded by the new nonlinear algorithms. The convergence of the new algorithms is also proved in this paper by nonlinear functional analysis.

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

General Terms
Algorithms, Experimentation

Keywords
Nonlinear static rank, Link aggregation, Probabilistic model

1. INTRODUCTION
Link-based static ranking, the activity of calculating importance scores for web documents from a link graph, has shown to be very important for web search. Among all existing static ranking algorithms, PageRank [15] could be the most popular one. Given a link graph, the PageRank of any page *v* is calculated as follows,

$$R(v) = (1-c) + c\sum_{u\in B_v}\frac{R(u)}{d(u)} \qquad (1.1)$$

where $d(u)$ is the out-degree (i.e. number of out-links) of page *u*, B_v is the set of pages linking to *v*, and *c* is a constant between 0 and 1. An alternative formula is to have the (1-*c*) term above divided by *N*, the total number of pages in the link graph. The PageRank of all pages can be computed via an iterative process (with possible normalization after every iteration to have the sum of PageRank values to be 1.0 or *N*). According to the above formula, it is clear that the PageRank of a page is a *linear combination* of the PageRank values of all its in-links.

The PageRank algorithm may give apparently unreasonable static-rank results for some link structures, as illustrated in Figure 1. In the figure, page D_1 has 50,000 in-links, with all of them coming from the same site domain2.com; while page D_2 has only 1,000 in-links, coming from 100 sites. Assume all the in-links of D_1 and D_2 have the same static-rank value and the same number of out-links. According to the standard static-rank methodology (Formula 1.1), the PageRank of D_1 would be much larger than that of D_2. This is contrary to our intuition. Intuitively, if a page is linked to by a lot of domains, it should have, with very high probability, larger static-rank than another page which is linked to by numerous pages from few domains. Web pages in one domain are typically maintained by one editor or a small group of editors. It is easy for the editors to build (intentionally or unintentionally) many pages in the domain linking to one destination page. While for domains controlled by different owners, it is much harder or less likely for them to add links pointing to the same page. It is therefore "safer" to assign high static-rank to pages linked to by many domains than pages linked to by only few domains.

The above problem cannot be solved by differentiating inter-domain links from intra-domain links (one straight-forward optimization upon the classical PageRank algorithm), because all links in the figure are inter-domain ones. Other PageRank enhancement techniques (e.g. biased surfer or dynamic damping factor) are also not able to entirely solve the problem (referring to Section 2 for details).

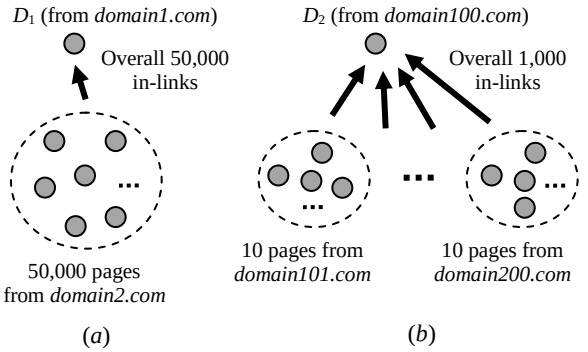

Figure 1. The in-link information of two pages (D_1 and D_2)

In this paper, we examine the static-rank computation problem from the viewpoint of evidence combination and build a probabilistic model to address the problem. Given a web page, we treat each of its back links as a piece of *evidence* indicating the importance of the page. The static-rank of a page is defined by the probability of the page being important, which is determined by the pieces of evidence corresponding to its back links. The problem then becomes how to combine the pieces of linking evidence to get the static-rank score of a page. We demonstrate by probability theory that, when all links pointing to a destination

* This work was done when the author was an intern at Microsoft Research Asia

Microsoft technical report: **MSR-TR-2009-55**

page are independent, the static-rank of the page can be computed via the linear combination of its back links (just like in Formula 1.1). However, also according to our model, if the links are correlated, the linear combination function does not hold anymore. The higher the correlation, the smaller the static-rank score is relative to the linear combination of its in-links. In the extreme case that other links are fully determined by one link, the static-rank can be computed as other links do not exist. Observing that many links are highly correlated in reality, we argue that nonlinear formulas should be utilized in static-rank computation, in order to consider the correlations among links.

The correlations between links are quite complex in reality, resulting in complex rank combination formulas. In this paper, we examine a simple formula which only covers, in a simple way, the correlation between links coming from one domain. With the nonlinear formulas being adopted in static-rank computation, it is natural to ask the following questions,

- ✓ *Can the new algorithms generate better static-rank scores?* Yes. Experimental results (with two metrics) show that our new algorithms generate better static-rank.
- ✓ *Do the new algorithms converge?* Existing ways for proving the convergence of the standard PageRank algorithm cannot be applied to nonlinear systems. In this paper, we prove (with the help of nonlinear functional analysis) that our new algorithms *do* converge.
- ✓ *What's the convergence speed of the new algorithms?* Experiments conducted on 100 million web pages show that it takes significantly fewer iterations for some of our nonlinear algorithms to converge than the standard PageRank algorithm.

The rest of the paper is organized as follows: In Section 2, we review related work. The probabilistic model for rank aggregation is proposed and discussed in Section 3. In Section 4, we describe the algorithms which consider the dependence between links from the same domain. Experiments are conducted on a real dataset in Section 5 to verify the effectiveness of our new algorithms.

2. RELATED WORK

PageRank [15] and HITS [12], both proposed in 1998, may be two of the earliest static ranking algorithms for web pages. In HITS, each page has a "hub" score and an "authority" score. The hub score of a page is determined by the authority values of its forward links; while the authority score of a page depends on the hub values of its back links.

A lot of research work has been done to study and improve the PageRank algorithm. A comprehensive mathematical analysis of choosing the damping factor is given by [5]. Topic-sensitive PageRank [7] is developed to provide a set of PageRank vectors biased on a set of topics. For a given topic, by pre-computing a non-uniform jumping vector which will contribute to the nodes in each iteration, a particular ranking vector could be got. If the query falls into one of the predefined categories, the corresponding ranking vector will be used. Query-dependent PageRank [1] adjusts the contribution given by one out-link by considering the relevance between a query and a page. An intelligent surfer model is proposed in [16] where several features (anchor text, relative position, HTML tags, etc) are used to assign different weights to different out-links of a page. To combat with web page spam, a method is introduced in [19] to discover link farm pages by identifying a spam seed set first and expanding it iteratively. TrustRank [6] also starts from a white list and propagates trust through links to other pages. Site structure is studied in [3, 20] to improve static-rank computation. All of the above existing efforts adopt the same linear method to combine contributions to a page, which is fundamentally different from our work. Based on probabilistic theory, this paper studies static-rank computation from the viewpoint of link aggregation, and proposes nonlinear formulas to address the problem.

The performance of static-rank calculation is important for large link graphs. Several methods to speed up the process are proposed in [9, 10]. Although the nonlinear algorithms we proposed are not designed to improve static-rank calculation speed, we observe from experimental results that the number of iterations is reduced with our nonlinear static-rank algorithms.

Tsaparas [18] studies using the MAX function (a nonlinear operator) to compute the hub weights of web pages in the HITS [12] algorithm.

Some theory work [14, 4] about static-rank computation has also been done. A survey on PageRank computation is given in [2].

3. PROBABILISTIC MODEL FOR LINK AGGREGATION

Here we treat the static-rank computation problem from the viewpoint of probabilistic evidence combination. Our goal is to estimate the static-rank of web pages using probability theory. We assume that for any page v, its static-rank is the probability (or a function of the probability) of the page being important, given the information of all its back-links. We define the following events related to page v and its back-links,

A: Page v is important

E_i: Page u_i links to v

The following probabilities are related to the above events,

$P(A)$: The *prior* probability of page v being important (without knowing its back-links)

$P(A|E_i)$: The probability of page v being important, given that u_i links to v (but without knowing other back-links)

$P(A|E_1, E_2 \ldots, E_m)$: The *posterior* probability of page v being important, given all the m back-links of it

Our problem is then to estimate $P(A|E_1, E_2 \ldots, E_m)$ using $P(A|E_i)$ and $P(A)$. Formally, we would like to find a function f to satisfy,

$$P(A|E_1,E_2\ldots,E_m) = f\,(\,P(A),\, P(A|E_1)\ldots,\, P(A|E_m)\,)$$

3.1 Independent Links

Now we start from the simple case that all back-links of a page are independent.

For simplicity of statement, we first consider the case of m=2, that is, two pages u_1 and u_2 link to page v. The problem is to estimate $P(A|E_1,E_2)$ with the following independence assumptions,

$$P(E_1, E_2) = P(E_1) \cdot P(E_2) \tag{3.1}$$

$$P(E_1, E_2|A) = P(E_1|A) \cdot P(E_2|A) \tag{3.2}$$

The first assumption means that E_1 and E_2 are independent; and the second one demonstrates that E_1 and E_2 are conditionally independent given A.

Given the above independence assumptions, by using Bayesian Formula, we have,

$$P(A|E_1,E_2) = \frac{P(E_1,E_2|A)\cdot P(A)}{P(E_1,E_2)} = \frac{P(E_1|A)\cdot P(A)}{P(E_1)} \cdot \frac{P(E_2|A)\cdot P(A)}{P(E_2)} \cdot \frac{1}{P(A)} = \frac{P(A|E_1)\cdot P(A|E_2)}{P(A)} \quad (3.3)$$

We define

$$LP(A) = \log(P(A)); \quad G(A|E) = LP(A|E) - LP(A)$$

Here $LP(A)$ denotes the *log-probability* of event A; and $G(A|E)$ represents the *log-probability-gain* of A given E, the meaning of which is the *increment* in the log-probability value of A after considering evidence E. Please note that $LP(A)$ is the *prior* log-probability of A before we know evidence E, and $LP(A|E)$ is the *posterior* log-probability of A after evidence E is considered. Therefore $G(A|E)$, the delta of log-probability, can be seen as a measure of the impact of evidence E to event A.

According to Formula 3.3, we have,

$$\begin{aligned} G(A|E_1,E_2) &= LP(A|E_1,E_2) - LP(A) \\ &= \log\big(P(A|E_1,E_2)\big) - \log(P(A)) \\ &= \log\left(\frac{P(A|E_1)\cdot P(A|E_2)}{P(A)}\right) - \log(P(A)) \\ &= \log(P(A|E_1))\text{-}\log(P(A)) + \log(P(A|E_2))\text{-}\log(P(A)) \\ &= G(A|E_1) + G(A|E_2) \end{aligned} \quad (3.4)$$

The above formula demonstrates that when the independence assumptions (in Formula 3.1 and 3.2) are satisfied, the log-probability-gain of A given both E_1 and E_2 is exactly the sum of the log-probability-gains of A given one single piece of evidence respectively.

For m>2, if the pieces of evidence are mutually independent and conditional independent given A, it is easy to prove (by following a similar procedure) that,

$$G(A|E_1,\dots,E_m) = \sum_{i=1}^{m} G(A|E_i) \quad (3.5)$$

The static-rank of a page could be the (linear) combination of its base-rank (the rank value when it does not have any back-links) and the rank contribution from all of its back-links. Thus the rank of page v can be expressed as,

$$R(v) = (1-c) + c \cdot G(A|E_1,E_2\dots,E_m) \quad (3.6)$$

where c is a parameter with its value in [0, 1]. For simplicity, we also have assumed that the base-rank of all pages is 1.0 in the above formula. By combining Formula 3.5 and 3.6, we get,

$$R(v) = (1-c) + c \cdot \sum_{1\le i\le m} G(A|E_i) \quad (3.7)$$

If we estimate $G(A|E_i)$ with $R(u_i)/d(u_i)$, then the above formula is the same as Formula 1.1.

Therefore, our model shows that when all back-links of a page satisfy the independence assumptions, the importance score of the page can be computed via the linear combination of (the scores of) the back-links.

3.2 Correlated Links

The independence of links is assumed in the previous section to simplify our analysis and to get an elegant formula for rank aggregation. The independence assumptions, however, do not hold in reality. For example, links in multiple pages generated by the same template are not independent. If we have already learned that one page u_1 links to a destination page v, then it is more likely that another page u_2 generated by the same template also links to v than if we do not know the linking information from u_1 to v. Another example, the links posted by the same person on different blog sites are not independent. Spam links intentionally created by the same spammer pointing to one specific page are also not independent one another. It is hence necessary to consider the correlation between links in our model.

Two correlated events or random variables E_1 and E_2 can be *positively* or *negatively* dependent. In the case of positive dependence, knowing that E_1 has occurred increases the chance of E_2, formally,

$$P(E_2|E_1) > P(E_2) \quad (3.8)$$

Oppositely, $P(E_2|E_1) < P(E_2)$ if the two events are negatively dependent.

By analyzing the above examples, we learned that the dependence between links to the same destination page tends to be positive. So in the following discussion, positive dependence is assumed, where Formula 3.1[1] will be replaced by Formula 3.8. Still by Bayesian theory, Formula 3.3 becomes,

$$P(A|E_1,E_2) < \frac{P(A|E_1)\cdot P(A|E_2)}{P(A)} \quad (3.9)$$

As a result, Formula 3.4 becomes accordingly,

$$G(A|E_1,E_2) < G(A|E_1) + G(A|E_2) \quad (3.10)$$

The above formula indicates that when events E_1 and E_2 are positively correlated, page v's static-rank *gain* given both the events is *smaller* than the sum of static-rank gains caused by E_1-only and E_2-only respectively. In the extreme case that E_2 fully depends on E_1 (i.e. $P(E_2|E_1)$=1), it is easy to prove that

$$G(A|E_1,E_2) = G(A|E_1) \quad (3.11)$$

The above formula means the static-rank gain of v given E_1 and E_2 is the same as the gain given E_1 only. It is reasonable, since event E_2 brings no extra information about the importance of v in addition to E_1.

In order to compute the static-rank of a page v when the back-links E_1, E_2…, E_m are not independent, it is required to find a nonlinear function h, such that

$$G(A|E_1,\dots,E_m) = h\big(G(A|E_1),\dots,G(A|E_m)\big) \quad (3.12)$$

The specific expression of function h directly depends on how these links are correlated. Strictly speaking, different pages may have different functions for aggregating their back-links, if the link correlations are different between them. In spite of this, some constraints or conditions are available which every reasonable link aggregation function h should satisfy. Some of them are listed as follows (where $m > 1$),

$$h(x) = x \quad (C1)$$

$$h(x_1\dots,x_m) \ge \max_{1\le i\le m} x_i \quad (C2)$$

$$h(x_1\dots,x_m) < \sum_{i=1}^{m} x_i \quad (C3)$$

When a page has only one back-link, we should have $G(A|E_1)=h(G(A|E_1))$ according to Formula 3.12. Thus condition C1 should be satisfied for a reasonable aggregation function. For any back-link E_i, we have already known the log-probability-gain

[1] Please note that an equivalent way of stating that E_1 and E_2 are independent is: $P(E_2|E_1) = P(E_2)$.

of A is $G(A|E_i)$ if E_i is the only link being discovered or considered. After other back-links (E_1…, E_{i-1}, E_{i+1}…, E_m) are discovered, the probability of page v being important should get larger than if we consider E_i only (or at least keep unchanged), because more links are added to demonstrate the importance of the page. Thus $G(A|E_1, \ldots, E_m) \geq G(A|E_i)$ should hold for every i. So condition C2 should be satisfied. For the case of m=2, condition C3 can be derived directly from Formula 3.10. Moreover, it is not hard to extend Formula 3.10 to more than two events by following a similar procedure. Thus C3 should be satisfied.

Please note that the three constraints above are necessary but may not be sufficient for becoming a reasonable link aggregation function. Symmetry should not be a constraint here. In fact, the aggregation function in Formula 4.6 (Section 4) is not symmetric.

At the first glance, it seems easy to find a function to satisfy all the above conditions (C1~C3). It is however not the case. For example, NONE of the functions listed in Table 1 is a reasonable link aggregation function, because, for each function, there is at least one constraint which the function does NOT satisfy.

Table 1. List of functions each of which does NOT satisfy all the constraints (C1 ~ C3)

Function $h(x_1 \ldots, x_m)$	*Dissatisfied Constraint(s)*
$\sum_{i=1}^{m} \alpha_i x_i$ (where $\sum_{i=1}^{m} \alpha_i = 1$)	C2
$\sqrt[m]{\prod_{i=1}^{m} x_i}$	C2
$1/(\sum_{i=1}^{m} 1/x_i)$	C2
$\ln(1 + \sum_{i=1}^{m} x_i)$	C1, C2
$\ln(\sum_{i=1}^{m} e^{x_i})$	C3

On the other hand, valid link aggregation functions do exist. In the next section, we will study some of them and illustrate how to build new static-rank algorithms by adopting the functions.

4. NONLINEAR ALGORITHMS

In this section, we first (in Subsection 4.1) analyze some formulas which satisfy all the three constraints and therefore can be treated as valid link aggregation functions. And then in Subsection 4.2, we build a nonlinear static-rank algorithm, with the simple assumption that links from the same domain are correlated and links from different domains are mutually independent.

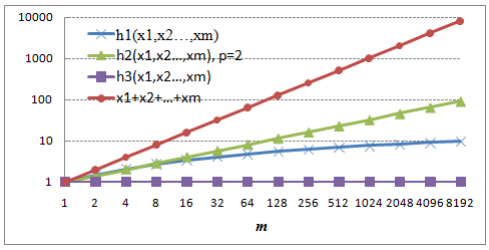

Figure 2. Comparing $\sum_{i=1}^{m} x_i$ with the nonlinear functions in Formulas 4.1 to 4.3 (x_1=x_2=…=x_m=1.0, m=1, 2, 4…, 8192)

4.1 Sample Nonlinear Functions

The following functions can be proved[2] to satisfy all the three constraints (C1~C3) in the previous section. Therefore we treat them as valid link aggregation functions in this paper.

$$h_1(x_1 \ldots, x_m) = \ln(1 + \sum_{i=1}^{m}(e^{x_i} - 1)) \quad (4.1)$$

$$h_2(x_1 \ldots, x_m) = \sqrt[p]{\sum_{i=1}^{m} x_i^p} \quad (p>1) \quad (4.2)$$

$$h_3(x_1 \ldots, x_m) = \max\{x_1 \ldots, x_m\} \quad (4.3)$$

Here h_3 can be treated as a special case of h_2 (with $p = +\infty$).

For a better understanding of the properties of the above three nonlinear functions, we plot them in Figure 2 (by varying m), with the special case of x_1=x_2=…=x_m=x. It is shown that, when m increases, the nonlinear functions increase much slower than function mx (corresponding to the case of mutual independence).

4.2 Nonlinear Static-Rank Algorithms

The core of a static-rank algorithm is the ranking formula where the static-rank of a page is expressed as a function of its back-links. In this section, we construct nonlinear ranking formulas by making specific assumptions about the correlation among links and then adopting appropriate nonlinear link-aggregation functions. Our primary assumption here is: Links coming from one domain are correlated while links from different domains are mutually independent.

Generally speaking, the forward links of pages in the same domain typically have higher correlation than the links in a group of random pages, because pages in the same domain are more likely to be maintained by the same group of editors or to be discussing the same topic. As a result, if it is already known that page u_1 links to page v, it is more likely that another page u_2 in the same domain also links to v than if we do not know the linking information from u_1 to v. Here we adopt the link aggregation function h_1 in Formula 4.1 to represent the positive correlation between links coming from one domain. That is,

$$h(x_1, x_2 \ldots, x_k) = \ln\left(1 + \sum_{i=1}^{k}(e^{x_i} - 1)\right) \quad (4.4)$$

where x_1, x_2…, and x_k are the static-rank gains related to the links from one single domain. Meanwhile, we ignore the correlation between links from different domains, that is,

$$h(x_1, x_2 \ldots, x_k) = \sum_{i=1}^{k} x_i \quad (4.5)$$

where x_1, x_2…, and x_k are static-rank gains related to the links from mutually different domains.

According to the above assumptions, given a web page v and its back links, the link aggregation function is as follows,

$$G(A|E_{1,1}\ldots, E_{n,m_n}) = \sum_{i=1}^{n} \ln\left(1 + \sum_{j=1}^{m_i}\left(e^{G(A|E_{i,j})} - 1\right)\right) \quad (4.6)$$

where n is the number of domains linking to page v, m_i is the number of pages linking to v in domain i, $u_{i,j}$ is the j^{th} page in domain i linking to v, and $E_{i,j}$ is the event that $u_{i,j}$ links to v.

By estimating $G(A|E_{i,j}) = \ln(1 + R(u_{i,j})/d(u_{i,j}))$ and substituting Formula 4.6 into Formula 3.6, we get our nonlinear static-rank formula for page v,

$$R(v) = (1 - c) + c \sum_{i=1}^{n} \ln\left(1 + \sum_{j=1}^{m_i} \frac{R(u_{i,j})}{d(u_{i,j})}\right) \quad (4.7)$$

[2] The proof is omitted due to space limitations.

According to the above formula, the static-rank of a page is the *nonlinear* combination of its back-links. A new static-rank algorithm can be acquired if the above formula is exploited to replace the linear ranking formula (Formula 1.1) in the standard PageRank algorithm. We call the new algorithm NL-LOG (which means that the algorithm adopts a nonlinear ranking formula containing logarithm operations). Given the new nonlinear algorithm, we hope the static-rank of all pages can still be computed via an iterative process, just as in the standard PageRank algorithm. To do so, we should first demonstrate that our new nonlinear algorithm converges.

The convergence of the standard PageRank algorithm is determined by the convergence of iteratively solving the following linear system,

$$\boldsymbol{x} = \boldsymbol{A} \cdot \boldsymbol{x} + \boldsymbol{b}$$

where $\boldsymbol{x}$ is a vector representing the static-rank of all pages, and A is a matrix. It can be proved by linear algebra or the theory of Markov chain that the above linear system can be solved in an iterative way (with convergence guarantee).

The convergence of our new algorithm, however, cannot be proved this way, because we are solving a nonlinear system. In Appendix A, we give a proof of the convergence of our new algorithm, which is based on *nonlinear functional analysis*, an area of mathematics for solving nonlinear problems.

Till now we suppose the function h_1 (in Formula 4.1) is adopted to aggregate the links from the same domain. Now we consider the case that the function h_2 in Formula 4.2 (assuming p=2 here) is adopted to take the role of h_1. By following the similar procedure as above and assuming $G(A|E_{i,j}) = R(u_{i,j})/d(u_{i,j})$, we get,

$$R(v) = (1-c) + c\sum_{i=1}^{n}\sqrt{\sum_{j=1}^{m_i}\frac{R(u_{i,j})^2}{d(u_{i,j})^2}} \tag{4.8}$$

We call a static-rank algorithm NL-SQRT-1 if it adopts the above ranking formula.

If we make the estimation that $G(A|E_{i,j}) = \sqrt{R(u_{i,j})}/d(u_{i,j})$, we get the following nonlinear ranking formula,

$$R(v) = (1-c) + c\sum_{i=1}^{n}\sqrt{\sum_{j=1}^{m_i}\frac{R(u_{i,j})}{d(u_{i,j})^2}} \tag{4.9}$$

Ranking algorithms corresponding to the above formula are called NL-SQRT-2.

Similarly, for nonlinear function h_3 (Formula 4.3), we have (assuming $G(A|E_{i,j}) = R(u_{i,j})/d(u_{i,j})$),

$$R(v) = (1-c) + c\sum_{i=1}^{n}\max_{j=1}^{m_i}\frac{R(u_{i,j})}{d(u_{i,j})} \tag{4.10}$$

Please refer to Appendix B for the proof of the convergence of Formulas 4.8 to 4.10. Formulas 4.7 to 4.10 are our resultant nonlinear static-rank formulas acquired by exploiting the correlation among links coming from the same domain.

5. EXPERIMENTS

In this section, we conduct experiments on a large-scale realistic dataset to test the performance of our nonlinear algorithms.

5.1 Experimental Setup

Dataset The link graph for the experiments consists of around 100 million Chinese web pages (and 1.3 billion links) which was crawled from the web from September 2006 to March 2007. To crawl pages, several hundred popular home pages were chosen as seeds for our crawler, which then conducted a breadth-first traversal over the Web. The crawled pages are distributed in about 535000 domains (or web sites). We choose such a dataset for experiments because query logs and user labeling information are available for this dataset for facilitating the evaluation of static-rank quality (referring to Section 5.2).

System Environments and Data Processing The pages are distributed (via URL hashing) and processed in a small cluster consisting of 10 workstations. By parsing the pages in a distributed way, a link graph is constructed and stored in the cluster. Slightly different from ordinary web graphs, domain information of all vertices is recorded in order to implement our nonlinear algorithms. A parallel version of each static-rank algorithm is built and run on the cluster. In each iteration, every workstation calculates static-rank for its local pages and propagates static-rank contribution information to other rankers.

Algorithms and Parameters The following four static-rank algorithms will be studied and compared in experiments,

BASIC: The basic static-rank algorithm (with the static-rank of every page being calculated via Formula 1.1).
NL-LOG: Formula 4.7 is utilized as the static-rank formula.
NL-SQRT-1: Formula 4.8 is adopted in the algorithm.
NL-SQRT-2: Formula 4.9 is adopted in the algorithm.
NL-MAX: Formula 4.10 is adopted in the algorithm.

In all the algorithms, parameter c is fixed to be 0.85, a value suggested in [15].

5.2 Evaluation Criteria

Three criteria are adopted to evaluate and compare different static-rank algorithms: click-through correlation, search results quality, and the speed of convergence.

5.2.1 Click-through correlation

Since it is impossible in reality to know the "ideal" page quality values, we adopt user access information obtained from toolbar software as a rough indication of page importance. Intuitively, the more important a page is, the more times of user access would be. We used two types of user access information collected by the search toolbar of a commercial web search engine: *user-click-through*, and *user-input*. Every entry in our aggregated click-through/user-input data contains a URL and the overall number of times that users click/input the URL. There are respectively 345 million and 1.26 billion distinct URLs contained in our user-input data and click-through data. For a given collection of URLs, assume L_1 is the URL ordering according to user access information, and L_2 is the URL ordering according to the static-rank values generated by a static ranking algorithm. We measure static-rank quality of the algorithm by computing the Kendall tau correlation [11] of L_1 and L_2.

The Kendall tau rank correlation coefficient evaluates the degree of similarity between two sets of ranks given to a same set of objects. Kendall's tau is defined as

$$\tau = \frac{n_c - n_d}{\binom{L}{2}}$$

where n_c is the number of concordant pairs, n_d is the number of discordant pairs in the ranking lists, and L is the number of elements in every set. The coefficient has values in [-1,1] for all orders, and a greater value indicates a higher positive correlation between the two lists.

Kendall's tau evaluation is performed over the intersections of the user access information and the link graph in order to measure the

correlation between each ranking scheme and user preferences. We believe that the results are convincing since most of the high quality pages in the link graph fall into the intersections.

5.2.2 Indirect evaluation by search results quality

We believe good static-rank values, when properly combined with dynamic ranking scores, should be helpful to yield good search results. Thus another way of evaluating static-rank quality is through the quality of the final search results.

To generate search results, we built an inverted index for the web pages we crawled. And 1000 queries were selected as our query set from the query log of a commercial web search engine. The ranking function used for computing page scores is the linear combination of static-rank and the dynamic ranking scores computed via the BM25 formula [17]. For each static-rank algorithm, top 10 results were returned for each query. We then collected the search results and had them judged by labelers, with each result page marked with a relevance level from 1 (meaning "bad") to 5 (meaning "perfect"). We adopted nDCG (normalized discounted cumulative gain [8]) to evaluate the quality of search results. The nDCG metric has two parameters: discount factor *b*, and gains for all labeled relevance levels. In our experiments, the value of the discount factor *b* is fixed to be 2. And the gain value for the 5 relevance levels (from 1 to 5) are 0.01, 1, 3, 7 and 15, respectively.

5.2.3 Speed of convergence

We measure the speed of convergence with the number of iterations before the algorithm terminates with the observation that the time cost per iteration for different algorithms are quite close to each other. Different terminating conditions (measured by the L1 residual between the static-rank vectors of adjacent iterations) are applied in our experiments.

5.3 Experimental Results

5.3.1 Static-rank quality comparison

Figure 3 shows the results of Kendall tau correlation for the static-rank algorithms we assessed. According to the figure, the four nonlinear algorithms have higher Kendall tau correlations with both user clickthrough and user-input than the standard PageRank algorithm. For the click-through and user-input data, the correlations improve by 19.5% and 30.4% respectively, by taking into account the correlation between back-links. The four nonlinear algorithms perform comparably.

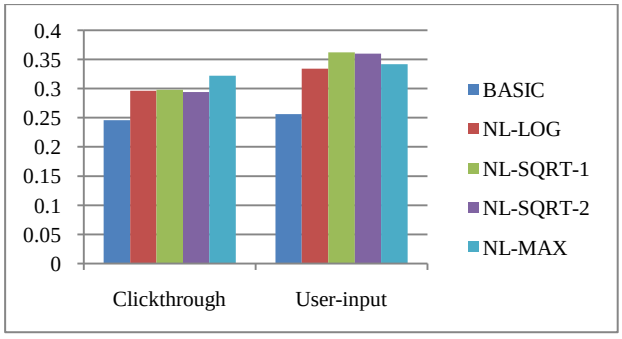

Figure 3. Static-rank quality comparison among various algorithms (metric: Kendall tau correlation)

The results of nDCG@n evaluation are presented in Table 2. It is observed that, like in the previous evaluation, the nonlinear algorithms perform quite close in quality again (from around 0.50 for nDCG@1 to around 0.42 for nDCG@10) and together outperform the BASIC algorithm (from 0.47 for nDCG@1 to 0.40 for nDCG@10). In general, the nonlinear algorithms we adopted boost nDCG up by more than 4%.

Table 2. Search quality measured by nDCG@n for various algorithms (the best results are in bold)

	nDCG@1	nDCG@3	nDCG@5	nDCG@10
BASIC	0.472	0.449	0.427	0.399
NL-LOG	0.499 (↑ 5.7%)	0.472 (↑ 5.1%)	0.452 (↑ 5.9%)	0.422 (↑ 5.8%)
NL-SQRT-1	0.496 (↑ 5.1%)	0.470 (↑ 4.7%)	0.450 (↑ 5.4%)	0.420 (↑ 5.3%)
NL-SQRT-2	**0.500 (↑ 5.9%)**	**0.476 (↑ 6.0%)**	**0.456 (↑ 6.8%)**	**0.427 (↑ 7.0%)**
NL-MAX	0.493 (4.4%)	0.463 (3.1%)	0.446 (4.4%)	0.415 (4.0%)

Table 3. Sign test results for comparing various static-rank algorithms (Search quality metric: nDCG@3)

	BASIC	NL-LOG	NL-SQRT-1	NL-SQRT-2
NL-LOG	>>			
NL-SQRT-1	>>	≈		
NL-SQRT-2	>>	≈	≈	
NL-MAX	>>	≈	≈	≈

We use the sign test to determine whether the search quality improvement is significant after the new nonlinear static-rank algorithms are applied. Table 3 displays the sign test results for comparing every pair of static-rank algorithms. In the table, ">>" indicates the P value is less than 0.01 and reaches the level of significant difference. And "≈" means the P value is larger than 0.05, and therefore we cannot conclude the search quality of one algorithm is better than another. We can see from the table that the nonlinear static-rank algorithms lead to significantly better search results (in terms of nDCG@3) than the BASIC algorithm. Also from the table, the differences between the nonlinear algorithms seem not significant. Similar observations can be made for nDCG@1, nDCG@5, and nDCG@10.

5.3.2 Speed of convergence

Table 4 shows the convergence speed (measured by the number of iterations and the average time per-iteration) of various algorithms. It can be seen that some of the nonlinear algorithms takes significantly fewer iterations to converge than the basic (linear) algorithm. For example, the NL-LOG algorithm takes about 85% fewer iterations (from 111 to 17) to converge with a residual of 0.001, compared with the basic PageRank algorithm. Even the NL-MAX algorithm, which has the lowest convergence speed among the four nonlinear algorithms, reduces about 66.7% of iterations with a residual of 0.001. We can see that, among the nonlinear algorithms, algorithm NL-SQRT-2 converges fastest, followed by NL-LOG. It can also be observed from Table 4 that the nonlinear algorithms only take a bit more time (about 5%) per-iteration than the BASIC algorithm. Thus the overall computation time is reduced by utilizing the nonlinear algorithms. Therefore the time of static-rank computation can be significantly reduced if we choose a proper aggregation function.

The huge differences of convergence speed among the algorithms are somewhat unanticipated, because the algorithms only differ in the way of expressing the static-rank of a page by the rank of its

back-links. We leave it for future work the analysis of underlying reason for this phenomenon.

Table 4. Convergence speed comparison among algorithms (metric: number of iterations; the best results are in bold)

L1 Residual / *Algorithm*	*0.01 (10^{-2})*	*10^{-3}*	*10^{-4}*	*10^{-5}*	*Avg. Time Per-Iter.*
BASIC	96	111	125	139	412 s
NL-LOG	12	17	23	28	434 s
NL-SQRT-1	23	37	51	65	430 s
NL-SQRT-2	**9**	**11**	**14**	**17**	431 s
NL-MAX	23	37	51	66	426 s

5.3.3 Overall comparison

In summary, the above experimental results demonstrate that our nonlinear algorithms outperform the classic PageRank algorithm both in static-rank quality (measured by two distinct criteria) and in convergence speed. This demonstrates that it is reasonable and necessary to consider the correlation among links in static-rank computation. All the four nonlinear algorithms generate static-rank values of very similar quality. While in terms of convergence speed, the NL-LOG and NL-SQRT-2 seem to be more effective than other algorithms.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed a probabilistic link-aggregation model for static-rank computation, based on which some nonlinear algorithms were built. Most existing work for improving PageRank quality focuses on changing the contribution of out-links (by dynamically setting the damping factor or adjusting the weight of links). Differently, our work focuses on link-aggregation. By this way, we provide another angle to optimize static-rank computation.

As future work, it is desirable to conduct more experiments using larger and various kinds of datasets in order to further validate our conclusions in this paper. Although we observed, in experiments, the improvement of convergence speed (i.e. the reduction in the number of iterations) with our nonlinear algorithms, the reason for that remains unclear till now. It will be interesting to study in theory or at least in intuition why this happens. Other future work includes exploiting more types of dependence among links, studying the possibility of combining nonlinear static-rank with existing PageRank enhancement techniques, etc.

7. REFERENCES

[1] R. Baeza-Yates, E. Davis. Web Page Ranking using Link Attributes. In WWW 2004.

[2] P. Berkhin. A Survey on PageRank Computing. Internet Mathematics, 2(1):73-120, 2005.

[3] K. Bharat, B.-W. Chang, M. Henzinger and M. Ruhl. Who Links to Whom: Mining Linkage between Web Site. In ICDM 2001.

[4] M. Bianchini, M. Gori and F. Scarselli. Inside PageRank. ACM Transactions on Internet Technology, 5(1):92-128, 2005.

[5] P. Boldi, M. Santini and S. Vigna. PageRank as a Function of the Damping Factor. In WWW 2005.

[6] Z. Gyongyi, H. Garcie-Molina and J. Pedersen. Combating Web Spam with TrustRank. In VLDB 2004.

[7] T. H. Haveliwala. Topic-Sensitive PageRank. In WWW 2002.

[8] K. Jarvelin and J. Kekalainen. IR evaluation Methods for Retrieving Highly Relevant Documents. In SIGIR 2000.

[9] S. Kamvar, T. Haveliwala, C. Manning and G. Golub. Extrapolation Methods for Accelerating the Computation of PageRank. In WWW 2003.

[10] S. Kamvar, T. Haveliwala, C. Manning and G. Golub. Exploiting the Block Structure of the Web for Computing PageRank. Technical Report, Stanford University, 2003.

[11] M. G. Kendall. Rank Correlation Methods, 4th edition. Griffin, London, 1970.

[12] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. In Proceedings of ACM-SLAM Symposium on Discrete Algorithms, 1998.

[13] M.A. Krasnoselskii and P.P. Zabreiko. Geometric Methods in Nonlinear Analysis, Springer-Verlag, Berlin, 1984.

[14] A. Y. Ng, A. X. Zheng and M. I. Jordan. Stable Algorithms for Link Analysis. In SIGIR 2001.

[15] L. Page, S. Brin, R. Motwani and T. Winograd. The PageRank Citation Ranking: Bring Order to the Web. Technical report, Stanford University Database Group, 1998.

[16] M. Richardson and P. Domingos. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. In NIPS 2002.

[17] S. E. Robertson. Overview of Okapi Projects. Journal of Documentation, 53(1):3–7, 1997.

[18] P. Tsaparas. Using Non-Linear Dynamical Systems for Web Searching and Ranking. PODS, 2004.

[19] B. Wu and B. D. Davison. Identifying Link Farm Spam Pages. In WWW 2005.

[20] G.-R. Xue, Q. Yang, H.-J. Zeng, Y. Yu and Z. Chen. Exploiting the Hierarchical Structure for Link Analysis. In SIGIR, 2005.

8. APPENDIX

8.1 Appendix-A

Prove the convergence of our nonlinear algorithm NL-LOG (with ranking formula 4.7). Most of the following notations, definitions and theorems are from [13].

8.1.1 Notations and definitions

Banach space: A normed space S with the norm $\|\cdot\|$ is called a *Banach space* if S is a complete metric space for the metric d defined on S by the formula $d(\boldsymbol{x}, \boldsymbol{y})=\|\boldsymbol{x}-\boldsymbol{y}\|$.

Here we omit the detailed definitions for the terms appeared above for simplicity but present an example in most books about nonlinear analysis that the Euclidean spaces $\mathbb{R}^n$, where the norm of $\boldsymbol{x}=(x_1,\ldots,x_n)$ is given by $\|\boldsymbol{x}\|=(\sum_{i=1}^{n}|x_i|^2)^{1/2}$, are Banach spaces.

Cone: A closed convex set $\boldsymbol{K} \subset E$ (where E is a Banach space) is called a *cone* if $\boldsymbol{x} \in \boldsymbol{K}$ and $\boldsymbol{x} \neq 0$ implies $\alpha\boldsymbol{x} \in \boldsymbol{K}$ for $\alpha \geq 0$ and $\alpha\boldsymbol{x} \notin \boldsymbol{K}$ for $\alpha < 0$. A cone K defines a *partial ordering*: we write $\boldsymbol{x} \leq \boldsymbol{y}$ or $\boldsymbol{y} \geq \boldsymbol{x}$ if $\boldsymbol{y}-\boldsymbol{x} \in \boldsymbol{K}$, and $\boldsymbol{x}<\boldsymbol{y}$ or $\boldsymbol{y}>\boldsymbol{x}$ if $\boldsymbol{y}-\boldsymbol{x} \in K-\{\boldsymbol{0}\}$.

Solid cone: A cone K is called *solid* if there is at least one point x_0 in K such that a neighborhood of x_0 is contained in K.

Normal cone: A cone is called *normal* if there is an $N>0$ such that $0 \leq \boldsymbol{x} \leq \boldsymbol{y}$ implies $\|\boldsymbol{x}\| \leq N\|\boldsymbol{y}\|$ and N does not depend on $\boldsymbol{x}$ and $\boldsymbol{y}$.

Operator: Let $M \subset \mathbb{R}^n$. An n-dimensional *vector field* (a.k.a *map*, *operator*) A assigns to each $\boldsymbol{x} \in M$ a vector $A(\boldsymbol{x}) \in \mathbb{R}^n$; that is to say, a vector field on M is a mapping A: $M \to \mathbb{R}^n$. If we choose a coordinate system in $\mathbb{R}^n$, then A may be represented as

$$A(\boldsymbol{x}) = (A_1(x_1, \ldots, x_n), \ldots, A_n(x_1, \ldots, x_n))$$

where $x_1,\ldots,x_n$ are the coordinates of $\boldsymbol{x}$ and $A_1,\ldots,A_n$ are the components of A.

Positive operator: The operator A is called *positive* on a set $K \subset E$(where E is always a Banach space) if $A(K) \subset K$.

Monotone operator: The operator A is said to be *monotone* on a set $K \subset E$ if $\boldsymbol{x} \leq \boldsymbol{y}$ $(\boldsymbol{x}, \boldsymbol{y} \in K)$ implies $A(\boldsymbol{x}) \leq A(\boldsymbol{y})$.

$\boldsymbol{u_0}$-concave operator: We consider a nonlinear operator A which is positive on a cone K in a Banach space E. Fix a nonzero element $\boldsymbol{u}_0$ in K. The operator A is called $\boldsymbol{u}_0$*-concave* on K if for each nonzero element $\boldsymbol{x} \in K$ there are $\alpha(\boldsymbol{x}), \beta(\boldsymbol{x}) > \boldsymbol{0}$ (here $\alpha(\boldsymbol{x})$ means the choice of α depends on $\boldsymbol{x}$) such that

$$\alpha(\boldsymbol{x})\boldsymbol{u}_0 \leq A(\boldsymbol{x}) \leq \beta(\boldsymbol{x})\boldsymbol{u}_0$$

and if for each $\boldsymbol{x} \in K$ with $\alpha_1(\boldsymbol{x})\boldsymbol{u}_0 \leq A(\boldsymbol{x}) \leq \beta_1(\boldsymbol{x})\boldsymbol{u}_0$ $(\alpha_1(\boldsymbol{x}),\ \beta_1(\boldsymbol{x}) > \boldsymbol{0})$ we have that

$$A(t\boldsymbol{x}) \geq [1 + \eta(\boldsymbol{x}, t)]tA(\boldsymbol{x}) \qquad (0 < t < 1)$$

where $\eta(\boldsymbol{x}, t) > 0$.

Fixed point: A vector $\boldsymbol{x}^*$ is called a *fixed point* of operator A if it is a solution to the equation $\boldsymbol{x} = A(\boldsymbol{x})$.

8.1.2 Theorems

Theorem 1 (Schauder's Principle): Let E be a Banach space and T a nonempty closed convex subset of E. If A is a continuous operator mapping T into a compact (closed and bounded) subset of T, then A has at least one fixed point.

Theorem 2: Let K be a solid normal cone and $\boldsymbol{u}_0$ an element in the interior of K. If $\boldsymbol{A}$ is a $\boldsymbol{u}_0$-concave monotone operator and $\boldsymbol{x}^* \in K$ is a nonzero solution to equation $\boldsymbol{x} = A(\boldsymbol{x})$, then for each initial approximation $\boldsymbol{x}_0 \in K$ the approximating sequence $\boldsymbol{x}_n = A(\boldsymbol{x}_{n-1})$ converges to $\boldsymbol{x}^*$. $(n = 1,2, \ldots)$

Please refer to [13] for the proof of the above theorems.

8.1.3 Proof of convergence

8.1.3.1 Underlying vector space

We define a vector set $\boldsymbol{K}$ as $\{\boldsymbol{x} \in \mathbb{R}^n | 0 \leq x_i,\ i{=}1\ldots,\mathrm{n}\}$, where n is the number of pages in the link graph. $\boldsymbol{K}$ is closed since boundary is included, and is convex since for each $\boldsymbol{x}, \boldsymbol{y}$ in K and any $t \in (0,1)$, $t\boldsymbol{x}$+(1-t)$\boldsymbol{y}$ is also in K. It is also clear that $\alpha\boldsymbol{x} \in K$ for $\alpha \geq 0$ and $\alpha\boldsymbol{x} \notin K$ for $\alpha < 0$ if $\boldsymbol{x} \neq \boldsymbol{0}$. Therefore $\boldsymbol{K}$ **is a cone**.

Thus we derive a **partial ordering** on $\boldsymbol{K}$: $\boldsymbol{x} \leq \boldsymbol{y}$ if $\boldsymbol{y} - \boldsymbol{x} \in K$, which means $\boldsymbol{x} \leq \boldsymbol{y}$ iff every coordinate of $\boldsymbol{x}$ is less than or equal to the corresponding coordinate of $\boldsymbol{y}$.

$\boldsymbol{K}$ is **solid** since any $\boldsymbol{x}$>0 has a neighborhood in $\boldsymbol{K}$. And $\boldsymbol{K}$ is **normal** because for N=1, $0 \leq \boldsymbol{x} \leq \boldsymbol{y} \Rightarrow \|x\| \leq N\|y\|$.

8.1.3.2 Operator $\boldsymbol{A}$

As discussed in the previous sections, we are calculating the rank score for each page by iteratively solving the equation (4.7):

$$R(v) = (1-c) + c\sum_{i=1}^{n} \ln\left(1 + \sum_{j=1}^{m_i} \frac{R(u_{i,j})}{d(u_{i,j})}\right)$$

Here we rewrite this equation in an equivalent nonlinear operator style. We declare operator A: $K \longrightarrow K$ and define A for any $\boldsymbol{x} \in K$,

$$A(\boldsymbol{x})_i = (1-c) + c\sum_S \ln\left(1 + \sum_{j \in S} w_{ij}\, x_j\right) \qquad \text{(A-1)}$$

where S represents a domain, j represents a page, $\boldsymbol{x}$ is the rank vector, and w_{ij} is the weight of link j to i (usually $1/d(j)$ if there exists a hyperlink from j to i or 0 otherwise).

For the above operator $\boldsymbol{A}$, we will prove that: 1) $\boldsymbol{A}$ has at least one fixed point $\boldsymbol{x}$*; and 2) for each initial value $\boldsymbol{x}_0 \in K$, the sequence $\boldsymbol{x}_n$=$A(\boldsymbol{x}_{n-1})$ converges to $\boldsymbol{x}^*$ (i.e. $\boldsymbol{x}^*$ is the *only* fixed point).

8.1.3.3 Fixed point exists

We define a set $T \in K$ as follows,

$$T = \{\boldsymbol{x} \in \mathbb{R}^n | 0 \leq x_i \leq \mathrm{e}^{2n} - 1, i = 1, \ldots, n\} \qquad \text{(A-2)}$$

For a vector $\boldsymbol{x} \in T$, since $\ln(1{+}a{+}b) \leq \ln(1{+}a){+}\ln(1{+}b)$ for any a, b $\geq$0, we have

$$\begin{aligned} A(\boldsymbol{x})_i &\leq 1 - c + c\sum_S \ln\left(1 + \sum_{j \in S} w_{ij}\,(\mathrm{e}^{2n} - 1)\right) \\ &\leq 1 - c + c\sum_{j=1}^{n} \ln\left(1 + w_{ij}\,(\mathrm{e}^{2n} - 1)\right) \\ &< 1 - c + cn\ln(1 + (\mathrm{e}^{2n} - 1)) \\ &= 1 - c + 2cn^2 < 2n^2 < \mathrm{e}^{2n} - 1 \end{aligned} \qquad \text{(A-3)}$$

It is clear that A maps T into a compact (closed and bounded) subset of T. According to Theorem 1 we know that A has at least one fixed point $\boldsymbol{a}^* \in K$.

8.1.3.4 Exactly one fixed point

To adopt Theorem 2, here we first prove that the operation $\boldsymbol{A}$ defined in Section 8.1.3.2 is a $\boldsymbol{u}_0$-concave monotone operator.

For any $\boldsymbol{x}$ in K, $\boldsymbol{y} = A(\boldsymbol{x})$ satisfies that $y_i > 0$ for $i = 1, \ldots, n$, so $A(\boldsymbol{x}) \in K$, $A(K) \subset K$. Therefore A is **positive**. For arbitrary $\boldsymbol{x}, \boldsymbol{y} \in K$, $\boldsymbol{x} \leq \boldsymbol{y}$, it is easy to prove that $A(\boldsymbol{x})_i \leq A(\boldsymbol{y})_i$ for each i. Therefore A is **monotone**.

For a fixed nonzero element $\boldsymbol{u}_0$ in K and any given $\boldsymbol{x} \in K - \{\boldsymbol{0}\}$, we define $\boldsymbol{y} = A(\boldsymbol{x})$, and define $\alpha = \min_{1 \leq i \leq n} y_i/(\boldsymbol{u}_0)_i$ and $\beta = \max_{1 \leq i \leq n} y_i/(\boldsymbol{u}_0)_i$, then $\alpha(\boldsymbol{x})\boldsymbol{u}_0 \leq A(\boldsymbol{x}) \leq \beta(\boldsymbol{x})\boldsymbol{u}_0$ is satisfied.

Each non-constant component of operator A is a composition of $\ln(1 + \sum x)$ pattern. For any $\boldsymbol{x} \in K$ and $t \in (0,1)$ we define

$$p(\boldsymbol{x}, t) = \min_{\text{each page } i \text{ and site } S \text{ to } i} \frac{\ln(1 + t\sum_{j \in S} w_{ij}\, x_j)}{t\ln(1 + \sum_{j \in S} w_{ij}\, x_j)} \qquad \text{(A-4)}$$

We can prove that $p(\boldsymbol{x}, t) > 1$, because $\ln(1 + tx) > t\ln(1 + x)$ for any $x \in \mathbb{R}^+$ and $t \in (0,1)$. Define $\eta(\boldsymbol{x}, t) = p(\boldsymbol{x}, t) - 1 > 0$. It is also clear that $p(\boldsymbol{x}, t)t < 1$. Therefore we have

$$\begin{aligned} &(1 + \eta(\boldsymbol{x}, t)) \cdot t \cdot A(\boldsymbol{x})_i \\ &= p(\boldsymbol{x}, t)t\left(1 - c + c\sum_S \ln(1 + \sum_{j \in S} w_{ij}\, \boldsymbol{x}_j)\right) \\ &= p(\boldsymbol{x},t)t(1\text{-}c) + c\sum_S p(\boldsymbol{x}, t)t\ln\left(1 + \sum_{j \in S} w_{ij}\, \boldsymbol{x}_j\right) \\ &\leq (1 - c) + c\sum_S \ln\left(1 + \sum_{j \in S} w_{ij}\, t\boldsymbol{x}_j\right) \\ &= A(t\boldsymbol{x})_i \end{aligned} \qquad \text{(A-5)}$$

Thus A is a $\boldsymbol{u_0}$**-concave** operator.

According to Theorem 2, given *any* initial value $\boldsymbol{x}_0 \in K$ the approximating sequence $\boldsymbol{x}_n = A(\boldsymbol{x}_{n-1})$ converges to $\boldsymbol{a}^*$. This also means $\boldsymbol{a}^*$ is the only fixed point of A (otherwise there is another fixed point $\boldsymbol{b}^* \neq \boldsymbol{a}^*$. Taking $\boldsymbol{b}^*$ as the initial value, the sequence $\boldsymbol{x}_n = A(\boldsymbol{x}_{n-1})$ clearly converges to $\boldsymbol{b}^*$. But according to Theorem 2, it should converges to $\boldsymbol{a}^*$. Therefore $\boldsymbol{b}^* = \boldsymbol{a}^*$). □

8.2 Appendix-B

Prove the convergence of NL-SQRT-1, NL-SQRT-2, and NL-MAX. Since the proof is quite similar with the case of NL-LOG, we only list the key differences here to save space.

First, the operator A is changed from Formula A-1 to,

$$A(\boldsymbol{x})_i = (1-c) + c\sum_S \sqrt[p]{\sum_{j \in S} w_{ij}^p\, x_j^q} \qquad \text{(B-1)}$$

where p, q are integers satisfying p>1 and $q \leq p$. For NL-SQRT-1, p=q=2; for NL-SQRT-2, p=2, q=1; while for NL-MAX, p=q= ∞.

Second, we define the following set $T \in K$ to replace Formula A-2,

$$T = \{\boldsymbol{x} \in \mathbb{R}^n | 0 \leq x_i \leq z_i, i = 1, \ldots, n\} \qquad \text{(B-2)}$$

where z_i is the static-rank of page i computed by applying the standard PageRank algorithm to the link graph (without domain

information utilized). For a vector $\boldsymbol{x} \in T$, since $\sqrt[p]{a} + \sqrt[p]{b} \geq \sqrt[p]{a+b}$ for any a, b ≥0, we have

$$\begin{aligned} A(\boldsymbol{x})_i &\leq 1 - c + c\sum_S \sqrt[p]{\sum_{j\in S} w_{ij}^p \cdot z_j^q} \\ &\leq 1 - c + c\sum_{j=1}^n \sqrt[p]{w_{ij}^p \cdot z_j^q} \\ &\leq 1 - c + c\sum_{j=1}^n (w_{ij} \cdot z_j) = z_i \end{aligned} \tag{B-3}$$

Thus A maps T into T itself (which is compact).

Third, for any $\boldsymbol{x} \in K$ and $t \in (0,1)$, we give the following definition of $p(x,t)$ to replace the one in Formula A-4,

$$p(\boldsymbol{x}, t) = \min_i \frac{A(t\boldsymbol{x})_i}{t(1-c) + (A(t\boldsymbol{x})_i - 1 + c)} \tag{B-4}$$

It is clear that $p(\boldsymbol{x}, t) > 1$ and $p(\boldsymbol{x}, t)t < 1$ (please note that $A(x)_i>0$ for each i, because of $c<1$). By defining $\eta(\boldsymbol{x}, t) = p(\boldsymbol{x}, t) - 1 > 0$, we have

$$\begin{aligned} &(1 + \eta(\boldsymbol{x}, t)) \cdot t \cdot A(\boldsymbol{x})_i \\ &= p(\boldsymbol{x}, t) \cdot t \cdot \left(1 - c + c\sum_S \sqrt[p]{\sum_{j\in S} w_{ij}^p x_j^q}\right) \\ &\leq p(\boldsymbol{x}, t) \cdot \left(t(1-c) + c\sum_S \sqrt[p]{\sum_{j\in S} w_{ij}^p \cdot (t \cdot x_j)^q}\right) \\ &\leq p(\boldsymbol{x}, t) \cdot \left(t(1-c) + (A(t\boldsymbol{x})_i - 1 + c)\right) \\ &\leq A(t\boldsymbol{x})_i \end{aligned} \tag{B-5}$$

Thus A is a $\boldsymbol{u_0}$**-concave** operator.

Therefore, by replacing the Formulas A-1 to A-5 with B-1 to B-5, we can prove the convergence of NL-SQRT-1, NL-SQRT-2, and NL-MAX by using the similar procedure with that in Section 8.1.3.