Link Analysis Ranking: Algorithms, Theory, and Experiments

ALLAN BORODIN
University of Toronto
GARETH O. ROBERTS
Lancaster University
JEFFREY S. ROSENTHAL
University of Toronto
and
PANAYIOTIS TSAPARAS
University of Helsinki

The explosive growth and the widespread accessibility of the Web has led to a surge of research activity in the area of information retrieval on the World Wide Web. The seminal papers of Kleinberg [1998, 1999] and Brin and Page [1998] introduced *Link Analysis Ranking*, where hyperlink structures are used to determine the relative *authority* of a Web page and produce improved algorithms for the ranking of Web search results. In this article we work within the hubs and authorities framework defined by Kleinberg and we propose new families of algorithms. Two of the algorithms we propose use a Bayesian approach, as opposed to the usual algebraic and graph theoretic approaches. We also introduce a theoretical framework for the study of Link Analysis Ranking algorithms. The framework allows for the definition of specific properties of Link Analysis Ranking algorithms, as well as for comparing different algorithms. We study the properties of the algorithms that we define, and we provide an axiomatic characterization of the InDegree heuristic which ranks each node according to the number of incoming links. We conclude the article with an extensive experimental evaluation. We study the quality of the algorithms, and we examine how different structures in the graphs affect their performance.

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

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Bayesian, HITS, link analysis, ranking, Web search

Authors' addresses: A. Borodin, J. S. Rosenthal, University of Toronto, Toronto, Canada; email: bor@cs.toronto.edu; G. O. Roberts, Lancaster University; P. Tsaparas, University of Helsinki, Helsinki, Finland; email: tsaparas@cs.helsinki.fi.

1. INTRODUCTION

Ranking is an integral component of any information retrieval system. In the case of Web search, because of the size of the Web and the special nature of the Web users, the role of ranking becomes critical. It is common for Web search queries to have thousands or millions of results. On the other hand, Web users do not have the time and patience to go through them to find the ones they are interested in. It has actually been documented [Broder 2002; Silverstein et al. 1998; Jansen et al. 1998] that most Web users do not look beyond the first page of results. Therefore, it is important for the ranking function to output the desired results within the top few pages, otherwise the search engine is rendered useless.

Furthermore, the needs of the users when querying the Web are different from traditional information retrieval. For example, a user that poses the query "microsoft" to a Web search engine is most likely looking for the home page of Microsoft Corporation, rather than the page of some random user that complains about the Microsoft products. In a traditional information retrieval sense, the page of the random user may be highly relevant to the query. However, Web users are most interested in pages that are not only relevant, but also *authoritative*, that is, *trusted* sources of correct information that have a strong *presence* in the Web. In Web search, the focus shifts from *relevance* to *authoritativeness*. The task of the ranking function is to identify and rank highly the authoritative documents within a collection of Web pages.

To this end, the Web offers a rich context of information which is expressed through the hyperlinks. The hyperlinks define the "context" in which a Web page appears. Intuitively, a link from page p to page q denotes an endorsement for the quality of page q. We can think of the Web as a network of recommendations which contains information about the authoritativeness of the pages. The task of the ranking function is to extract this latent information and produce a ranking that reflects the relative authority of Web pages. Building upon this idea, the seminal papers of Kleinberg [1998], and Brin and Page [1998] introduced the area of *Link Analysis Ranking*, where hyperlink structures are used to rank Web pages.

In this article, we work within the hubs and authorities framework defined by Kleinberg [1998]. Our contributions are three-fold.

(1) We identify some potential weaknesses of the HITS algorithm, proposed by Kleinberg [1998], and we propose new algorithms that use alternative methods for computing hub and authority weights. Two of our new algorithms are based on a Bayesian statistical approach as opposed to the more common algebraic/graph theoretic approach.

(2) We define a theoretical framework for the study of Link Analysis Ranking algorithms. Within this framework, we define properties that characterize the algorithms, such as monotonicity, stability, locality, label independence. We also define various notions of similarity between different Link Analysis Ranking algorithms. The properties we define allow us to provide an axiomatic characterization of the INDEGREE algorithm which ranks nodes according to the number of incoming links.

(3) We perform an extensive experimental evaluation of the algorithms on multiple queries. We observe that no method is completely safe from "topic drift", but some methods seem to be more resistant than others. In order to better understand the behavior of the algorithms, we study the graph structures that they tend to favor. The study offers valuable insight into the reasons that cause the topic drift and poses interesting questions for future research.

The rest of the article is structured as follows. In Section 2, we review some of the related literature and we introduce the notation we use in the article. In Section 3, we define the new Link Analysis Ranking algorithms. In Section 4, we define the theoretical framework for the study of Link Analysis Ranking algorithms, and we provide some preliminary results. Section 5 presents the experimental study of the Link Analysis Ranking algorithms, and Section 6 concludes the article.

2. BACKGROUND AND PREVIOUS WORK

In this section, we present the necessary background for the rest of the article. We also review the literature in the area of link analysis ranking, upon which this work builds.

2.1 Preliminaries

A link analysis ranking algorithm starts with a set of Web pages. Depending on how this set of pages is obtained, we distinguish between *query independent* algorithms, and *query dependent* algorithms. In the former case, the algorithm ranks the whole Web. The PAGERANK algorithm by Brin and Page [1998] was proposed as a query independent algorithm that produces a *PageRank* value for all Web pages. In the latter case, the algorithm ranks a subset of Web pages that is associated with the query at hand. Kleinberg [1998] describes how to obtain such a query dependent subset. Using a text-based Web search engine, a *Root Set* is retrieved consisting of a short list of Web pages relevant to a given query. Then, the Root Set is augmented by pages which point to pages in the Root Set, and also pages which are pointed to by pages in the Root Set, to obtain a larger *Base Set* of Web pages. This is the query dependent subset of Web pages on which the algorithm operates.

Given the set of Web pages, the next step is to construct the underlying hyperlink graph. A node is created for every Web page, and a *directed* edge is placed between two nodes if there is a hyperlink between the corresponding Web pages. The graph is *simple*. Even if there are multiple links between two pages, only a single edge is placed. No self-loops are allowed. The edges could be weighted using, for example, content analysis of the Web pages, similar to the spirit of the work of Bharat and Henzinger [1998]. In our work, we will assume that no weights are associated with the edges of the graph. Usually links within the same Web site are removed since they do not convey an endorsement; they serve the purpose of navigation. Isolated nodes are removed from the graph.

Let P denote the resulting set of nodes, and let n be the size of the set P. Let $G = (P, E)$ denote the underlying graph. The input to the link analysis algorithm is the adjacency matrix W of the graph G, where $W[i, j] = 1$ if there is a link from node i to node j, and zero otherwise. The output of the algorithm is an n-dimensional vector $\boldsymbol{a}$, where a_i, the i-th coordinate of the vector $\boldsymbol{a}$, is the *authority weight* of node i in the graph. These weights are used to rank the pages.

We also introduce the following notation. For some node i, we denote by $B(i) = \{j : W[j, i] = 1\}$ the set of nodes that point to node i (Backwards links), and by $F(i) = \{j : W[i, j] = 1\}$ the set of nodes that are pointed to by node i (Forward links). Furthermore, we define an *authority node* in the graph G to be a node with nonzero in-degree, and a *hub node* in the graph G to be a node with nonzero out-degree. We use A to denote the set of authority nodes, and H to denote the set of hub nodes. We have that $P = A \cup H$. We define the undirected *authority* graph $G_a = (A, E_a)$ on the set of authorities A, where we place an edge between two authorities i and j, if $B(i) \cap B(j) \neq \emptyset$. This corresponds to the (unweighted) graph defined by the matrix $W^T W$.

2.2 Previous Algorithms

We now describe some of the previous link analysis ranking algorithms that we will consider in this work.

2.2.1 *The* InDegree *Algorithm.* A simple heuristic that can be viewed as the predecessor of all Link Analysis Ranking algorithms is to rank the pages according to their *popularity* (also referred to as *visibility* [Marchiori 1997]). The popularity of a page is measured by the number of pages that link to this page. We refer to this algorithm as the InDegree algorithm, since it ranks pages according to their in-degree in the graph G. That is, for every node i, $a_i = |B(i)|$. This simple heuristic was applied by several search engines in the early days of Web search [Marchiori 1997]. Kleinberg [1998] makes a convincing argument that the InDegree algorithm is not sophisticated enough to capture the authoritativeness of a node, even when restricted to a query dependent subset of the Web.

2.2.2 *The* PageRank *Algorithm.* The intuition underlying the InDegree algorithm is that a good authority is a page that is pointed to by many nodes in the graph G. Brin and Page [1998] extended this idea further by observing that not all links carry the same weight. Links from pages of high quality should confer more authority. It is not only important to know how many pages point to a page, but also whether the quality of these pages is high or low. Therefore, they propose a one-level weight propagation scheme, where a good authority is one that is pointed to by many good authorities. They employ this idea in the PageRank algorithm. The PageRank algorithm performs a random walk on the graph G that simulates the behavior of a "random surfer". The surfer starts from some node chosen according to some distribution $\mathcal{D}$ (usually assumed to be the uniform distribution). At each step, the surfer proceeds as follows: with probability $1 - \epsilon$, an outgoing link is picked uniformly at random and the surfer moves to a new page, and with probability ϵ, the surfer jumps to a random page chosen according to distribution $\mathcal{D}$. The "jump probability" ϵ is passed as a parameter to the algorithm. The authority weight a_i of node i (called the PageRank of node i) is the fraction of time that the surfer spends at node i, that is, it is proportional to the number of visits to node i during the random walk. The authority vector $\boldsymbol{a}$ output by the algorithm is the stationary distribution of the Markov chain associated with the random walk.

2.2.3 *The* HITS *Algorithm.* Independent of Brin and Page [1998], Kleinberg [1998] proposed a different definition of the importance of Web pages. Kleinberg argued that it is not necessary that good authorities point to other good authorities. Instead, there are special nodes that act as hubs that contain collections of links to good authorities. He proposed a two-level weight propagation scheme where endorsement is conferred on authorities through hubs, rather than directly between authorities. In his framework, every page can be thought of as having two identities. The *hub* identity captures the quality of the page as a pointer to useful resources, and the *authority* identity captures the quality of the page as a resource itself. If we make two copies of each page, we can visualize graph G as a bipartite graph where hubs point to authorities. There is a mutual reinforcing relationship between the two. A good hub is a page that points to good authorities, while a good authority is a page pointed to by good hubs. In order to quantify the quality of a page as a hub and an authority, Kleinberg associated with every page a hub and an authority weight. Following the mutual reinforcing relationship between hubs and authorities, Kleinberg defined the hub weight to be the sum of the authority weights of the nodes that are pointed to by the hub, and the authority weight to be the sum of the hub weights that point to this authority. Let $\boldsymbol{h}$ denote the n-dimensional vector of the hub weights, where h_i, the i-th coordinate of vector $\boldsymbol{h}$, is the hub weight of node i. We have that

$$a_i = \sum_{j \in B(i)} h_j \qquad \text{and} \qquad h_j = \sum_{i \in F(j)} a_i \; . \tag{1}$$

In matrix-vector terms,

$$\boldsymbol{a} = W^T \boldsymbol{h} \qquad \text{and} \qquad \boldsymbol{h} = W \boldsymbol{a} \, .$$

Kleinberg [1998] proposed the following iterative algorithm for computing the hub and authority weights. Initially all authority and hub weights are set to 1. At each iteration, the operations $\mathcal{O}$ ("out") and $\mathcal{I}$ ("in") are performed. The $\mathcal{O}$ operation updates the authority weights, and the $\mathcal{I}$ operation updates the hub weights, both using Equation 1. A normalization step is then applied, so that the vectors $\boldsymbol{a}$ and $\boldsymbol{h}$ become unit vectors in some norm. The algorithm iterates until the vectors converge. This idea was later implemented as the HITS (Hyperlink Induced Topic Distillation) algorithm [Gibson et al. 1998]. The algorithm is summarized in Figure 1.

Kleinberg [1998] proves that the algorithm computes the principal left and right *singular* vectors of the adjacency matrix W. That is, the vectors $\boldsymbol{a}$ and $\boldsymbol{h}$ converge to the principal right eigenvectors of the matrices $M_H = W^T W$ and $M_H^T = WW^T$, respectively. The convergence of HITS to the singular vectors of matrix W is subject to the condition that the initial authority and hub vectors are not orthogonal to the principal eigenvectors of matrices M_H and M_H^T,

```
HITS

Initialize all weights to 1
Repeat until the weights converge:
    For every hub $i \in H$
        $h_i = \sum_{j \in F(i)} a_j$
    For every authority $i \in A$
        $a_i = \sum_{j \in B(i)} h_j$
    Normalize
```

Fig. 1. The HITS algorithm.

respectively. Since these eigenvectors have nonnegative values, it suffices to initialize all weights to positive values greater than zero. The convergence of the HITS algorithm does not depend on the normalization. Indeed, for different normalization norms, the authority weights are the same, up to a constant scaling factor. The relative order of the nodes in the ranking is also independent of the normalization.

There is an interesting observation about the weights assigned by the HITS algorithm after n steps. We first introduce the following notation. We say that we follow a B path if we follow a link backwards, and we say we follow an F path if we follow a link forward. We can combine these to obtain longer paths. For example, a $(BF)^n$ path is a path that alternates between backward and forward links n times. Now, let $(BF)^n(i, j)$ denote the set of $(BF)^n$ paths that go from i to j, $(BF)^n(i)$ the set of $(BF)^n$ paths that leave node i, and $(BF)^n$ the set of all possible $(BF)^n$ paths. We can define similar sets for the $(FB)^n$ paths.

By definition of the $(W^T W)^n$ and $(W W^T)^n$ matrices, we have that $|(BF)^n(i, j)| = (W^T W)^n(i, j)$, and $|(FB)^n(i, j)| = (WW^T)^n(i, j)$. Also, $|(BF)^n(i)| = \sum_j (W^T W)^n(i, j)$, and $|(FB)^n(i)| = \sum_j (WW^T)^n(i, j)$. Let $\boldsymbol{u}$ denote the vector of all ones. After the n-th operation of the HITS algorithm, the authority vector $\boldsymbol{a}$ and hub vector $\boldsymbol{h}$ are the unit vectors in the direction of $(W^T W)^n \boldsymbol{u}$ and $(W W^T)^n \boldsymbol{u}$, respectively. If we take the unit vectors under the L_1 norm, then we have

$$a_i = \frac{|(BF)^n(i)|}{|(BF)^n|} \quad \text{and} \quad h_i = \frac{|(FB)^n(i)|}{|(FB)^n|} .$$

Thus, the authority weight assigned by the HITS algorithm to node i after n iterations is proportional to the number of (BF) paths of length n that leave node i.

2.2.4 *The* SALSA *Algorithm.* An alternative algorithm, SALSA, that combines ideas from both HITS and PAGERANK was proposed by Lempel and Moran [2000]. As in the case of HITS, visualize the graph G as a bipartite graph where hubs point to authorities. The SALSA algorithm performs a random walk on the bipartite hubs and authorities graph, alternating between the hub and authority sides. The random walk starts from some authority node selected uniformly at random. The random walk then proceeds by alternating between backward and forward steps. When at a node on the authority side of the bipartite graph, the algorithm selects one of the incoming links uniformly at random and moves to a hub node on the hub side. When at a node on the hub side, the algorithm selects one of the outgoing links uniformly at random and moves to an authority. The authority weights are defined to be the stationary distribution of this random walk. Formally, the Markov Chain of the random walk has transition probabilities

$$P_a(i, j) = \sum_{k\,:\,k \in B(i) \cap B(j)} \frac{1}{|B(i)|} \frac{1}{|F(k)|}.$$

Recall that $G_a = (A, E_a)$ denotes the authority graph where there is an (undirected) edge between two authorities if they share a hub. This Markov Chain corresponds to a random walk on the authority graph G_a where we move from authority i to authority j with probability $P_a(i, j)$. Let W_r denote the matrix derived from matrix W by normalizing the entries such that, for each row, the sum of the entries is 1, and let W_c denote the matrix derived from matrix W by normalizing the entries such that, for each column, the sum of the entries is 1. Then the stationary distribution of the SALSA algorithm is the principal left eigenvector of the matrix $M_S = W_c^T W_r$.

If the underlying authority graph G_a consists of more than one component, then the SALSA algorithm selects a starting point uniformly at random and performs a random walk within the connected component that contains that node. Let j be a component that contains node i, let A_j denote the set of authorities in the component j, and E_j the set of links in component j. Then the weight of authority i in component j is

$$a_i = \frac{|A_j|}{|A|} \frac{|B(i)|}{|E_j|} .$$

If the graph G_a consists of a single component (we refer to such graphs as *authority connected* graphs), that is, the underlying Markov Chain is *irreducible*, then the algorithm reduces to the INDEGREE algorithm. Furthermore, even when the graph G_a is not connected, if the starting point of the random walk is selected with probability proportional to the "popularity" (in-degree) of the node in the graph G, then the algorithm again reduces to the INDEGREE algorithm. This algorithm was referred to as pSALSA (popularity-SALSA) by Borodin et al. [2001].

The SALSA algorithm can be thought of as a variation of the HITS algorithm. In the $\mathcal{I}$ operation of the HITS algorithm, the hubs *broadcast* their weights to the authorities and the authorities sum up the weight of the hubs that point to them. The SALSA algorithm modifies the $\mathcal{I}$ operation so that, instead of broadcasting, each hub *divides* its weight equally among the authorities to which it points. Similarly, the SALSA algorithm modifies the $\mathcal{O}$ operation so that each authority divides its weight equally among the hubs that point to it. Therefore,

$$a_i = \sum_{j:j \in B(i)} \frac{1}{|F(j)|} h_j \qquad \text{and} \qquad h_i = \sum_{j:j \in F(i)} \frac{1}{|B(j)|} a_j.$$

However, the SALSA algorithm does not really have the same "mutually reinforcing structure" that Kleinberg's [1998] algorithm does. Indeed, $a_i = \frac{|A_j|}{|A|} \frac{|B(i)|}{|E_j|}$, the relative authority of page *i within a connected component* is determined from local links, not from the structure of the component.

Lempel and Moran [2000] define a similar Markov Chain for the hubs and a hub weight vector that is the stationary distribution of the corresponding random walk. The stationary distribution $\boldsymbol{h}$ is the left eigenvector of the matrix $W_r^T W_c$.

2.2.5 *Other Related Work.* The ground-breaking work of Kleinberg [1998, 1999], and Brin and Page [1998] was followed by a number of extensions and modifications. Bharat and Henzinger [1998] and Chakrabarti et al. [1998] consider improvements on the HITS algorithm by using textual information to weight the importance of nodes and links. [Rafiei and Mendelzon 2000; Mendelzon and Rafiei 2000] consider a variation of the HITS algorithm that uses random jumps, similar to SALSA. The same algorithm is also considered by Ng et al. [2001a, 2001b], termed *Randomized HITS*. Extensions of the HITS algorithm that use multiple eigenvectors were proposed by Ng et al. [2001b] and Achlioptas et al. [2001]. Tomlin [2003] proposes a generalization of the PAGERANK algorithm that computes flow values for the edges of the Web graph, and a *TrafficRank* value for each page. A large body of work also exists that deals with personalization of the PAGERANK algorithm [Page et al. 1998; Haveliwala 2002; Jen and Widom 2003; Richardson and Domingos 2002].

A different line of research exploits the application of probabilistic and statistical techniques for computing rankings. The PHITS algorithm by Cohn and Chang [2000] assumes a probabilistic model in which a link is caused by latent "factors" or "topics". They use the Expectation Maximization (EM) Algorithm of Dempster et al. [1977] to compute the authority weights of the pages. Their work is based on the *Probabilistic Latent Semantic Analysis* framework introduced by Hofmann [1999], who proposed a probabilistic alternative to Singular Value Decomposition. Hofmann [2000] proposes an algorithm similar to PHITS which also takes into account the text of the documents. These algorithms require specifying in advance the number of factors. Furthermore, it is possible that the EM Algorithm gets "stuck" in a local maximum, without converging to the true global maximum.

3. NEW LINK ANALYSIS RANKING ALGORITHMS

The idea underlying the HITS algorithm can be captured in the following recursive definition of quality: "A good authority is one that is pointed to by many good hubs, and a good hub is one that points to many good authorities". Therefore, the quality of some page p as an authority (captured by the authority weight of page p) depends on the quality of the pages that point to p as hubs (captured in the hub weight of the pages), and vice versa. Kleinberg [1998] proposes to associate the hub and authority weights through the addition operation. The authority weight of a page p is defined to be the sum of the hub weights of the pages that point to p, and the hub weight of the page p is defined to be the sum of the authority weights of the pages that are pointed to by p. This definition has the following two implicit properties. First, it is *symmetric* in the sense that both hub and authority weights are defined in the same way. If we reverse the orientation of the edges in the graph G, then authority and hub weights are swapped. Second, it is *egalitarian* in the sense that, when computing the hub weight of

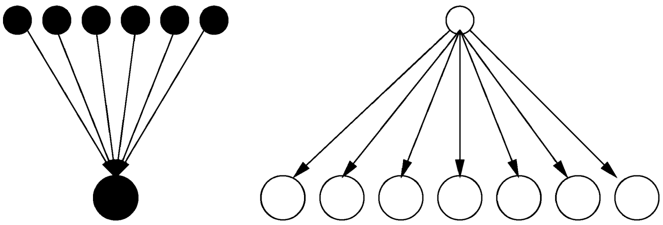


Fig. 2. A bad example for HITS algorithm.

some page p, the authority weights of the pages that are pointed to by page p are all treated equally (similarly when computing the authority weights).

These two properties may sometimes lead to nonintuitive results. Consider, for example, the graph in Figure 2. In this graph, there are two components. The black component consists of a single authority pointed to by a large number of hubs. The white component consists of a single hub that points to a large number of authorities. When the number of white authorities is larger than the number of black hubs, the HITS algorithm will allocate all authority weight to the white authorities, while giving zero weight to the black authority. The reason for this is that the white hub is deemed to be the best hub, thus causing the white authorities to receive more weight. However, intuition suggests that the black authority is better than the white authorities and should be ranked higher.

In this example, the two implicit properties of the HITS algorithm combine to produce the nonintuitive result. Equality means that all authority weights of the nodes that are pointed to by a hub contribute equally to the hub weight of the node. As a result, quantity becomes quality. The hub weight of the white hub increases inordinately simply because it points to many weak authorities. This leads us to question the definition of the hub weight and consequently, the other symmetric nature of HITS. Symmetry assumes that hubs and authorities are qualitatively the same. However, there is a difference between the two. For example, intuition suggests that a node with high in-degree is likely to be a good authority. On the other hand, a node with high out-degree is not necessarily a good hub. If this was the case, then it would be easy to increase the hub quality of a page, simply by adding links to random pages. It seems that we should treat hubs and authorities differently.

In this section, we challenge both implicit properties of HITS. We present different ways for breaking the symmetry and equality principles, and we study the ranking algorithms that emerge.

3.1 The Hub-Averaging (HUBAVG) Algorithm

In the example in Figure 2, the symmetric and egalitarian nature of the HITS algorithm produces the effect that the quality of the white hub is determined by the quantity of authorities it points to. Thus, the white hub is rewarded simply because it points to a large number of authorities, even though they are of low quality. We propose a modification of the HITS algorithm to help remedy this problem. The Hub-Averaging algorithm (HUBAVG) updates the authority weights like the HITS algorithm, but it sets the hub weight of some node i to the

```
HUBAVG

Initialize authority weights to 1
Repeat until the weights converge:
    For every hub $i \in H$
        $h_i = \frac{1}{|F(i)|} \sum_{j \in F(i)} a_j$
    For every authority $i \in A$
        $a_i = \sum_{j \in B(i)} h_j$
    Normalize
```

Fig. 3. The HUBAVG algorithm.

average authority weight of the authorities pointed to by hub i. Thus, for some node i, we have

$$a_i = \sum_{j \in B(i)} h_j \qquad \text{and} \qquad h_i = \frac{1}{|F(i)|} \sum_{j \in F(i)} a_j. \tag{2}$$

The intuition of the HUBAVG algorithm is that a good hub should point *only* (or at least mainly) to good authorities, rather than to both good and bad authorities. Note that in the example in Figure 2, HUBAVG assigns the same weight to both black and white hubs, and it identifies the black authority as the best authority. The HUBAVG algorithm is summarized in Figure 3.

The HUBAVG algorithm can be viewed as a "hybrid" of the HITS and SALSA algorithms. The operation of averaging the weights of the authorities pointed to by a hub is equivalent to dividing the weight of a hub among the authorities it points to. Therefore, the HUBAVG algorithm performs the $\mathcal{O}$ operation like the HITS algorithm (broadcasting the authority weights to the hubs), and the $\mathcal{I}$ operation like the SALSA algorithm (dividing the hub weights to the authorities). This lack of symmetry between the update of hubs and authorities is motivated by the qualitative difference between hubs and authorities discussed previously. The authority weights for the HUBAVG algorithm converge to the principal right eigenvector of the matrix $M_{HA} = W^T W_r$.

There is an interesting connection between HUBAVG and the Singular Value Decomposition. Note that the matrix M_{HA} can be expressed as $M_{HA} = W^T F W$, where F is a diagonal matrix with $F[i,i] = 1/|F(i)|$. We also have that

$$M_{HA} = (F^{1/2}W)^T(F^{1/2}W),$$

where $F^{1/2}$ is the square root of matrix F, that is, $F[i,i] = 1/\sqrt{|F(i)|}$. Let $W(i)$ denote the row vector that corresponds to the ith row of matrix W. Given that all entries of the matrix W take 0/1 values, we have that $\|W(i)\|_2 = \sqrt{|F(i)|}$. Thus, the matrix $F^{1/2}W$ is the matrix W, where each row is normalized to be a unit vector in the Euclidean norm. Let W_e denote this matrix. The hub and authority vectors computed by the HUBAVG algorithm are the principal left and right singular vectors of the matrix W_e.

3.2 The Authority Threshold (AT(k)) Family of Algorithms

The HUBAVG algorithm has its own shortcomings. Consider, for example, the graph in Figure 4. In this graph, there are again two components, one black

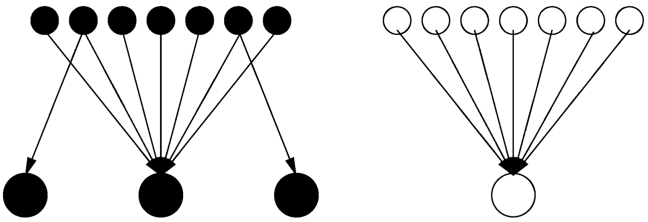


Fig. 4. A bad example for the HUBAVG algorithm.

and one white. They are completely identical, except for the fact that some of the hubs of the black component point to a few extra authorities. If we run the HUBAVG algorithm on this graph, then the white authority will receive higher authority weight than the black authority. This is due to the fact that the black hubs are *penalized* for pointing to these "weaker" authorities. The HUBAVG algorithm rewards hubs that point *only* (or mainly) to good authorities. Hubs that have links to a few poor authorities are penalized. However, the black authority seems to be at least as authoritative as the white authority. Although we would like the black hubs not to be rewarded for pointing to these weak authorities, we do not necessarily want them to be penalized either. Such situations may arise in practice when a node is simultaneously a strong hub on one topic and a weak hub on another topic. Such hubs are penalized by the HUBAVG algorithm.

We would like to reduce the effect of the weak authorities on the computation of the hub weight, while at the same time retaining the positive effect of the strong authorities. A simple solution is to apply a threshold operator that retains only the highest authority weights. We propose the *Authority-Threshold*, AT(k), algorithm which sets the hub weight of node i to be the sum of the k largest authority weights[1] of the authorities pointed to by node i. This is equivalent to saying that a node is a good hub if it points to *at least* k good authorities. The value of k is passed as a parameter to the algorithm.

Formally, given an authority weight vector $\boldsymbol{a}$, let $F_k(i)$ denote the subset of $F(i)$ that contains k nodes with the highest authority weights. That is, for any node $p \in F(i)$, such that $p \notin F_k(i)$, $a_p \leq a_q$, for all $q \in F_k(i)$. If $|F(i)| \leq k$, then $F_k(i) = F(i)$. The AT(k) algorithm computes the authority and hub weights as follows.

$$a_i = \sum_{j \in B(i)} h_j \qquad \text{and} \qquad h_i = \sum_{j \in F_k(i)} a_j .$$

The outline of the AT(k) algorithm is shown in Figure 5.

It is interesting to examine what happens at the extreme values of k. If d_{out} is the maximum out-degree of any node in the graph G, then for $k \geq d_{out}$, the AT(d_{out}) algorithm is the HITS algorithm. For $k = 1$, the threshold operator becomes the max operator. We discuss this case in detail in the following section.

[1] Other types of threshold are possible. For example, the threshold may depend on the largest difference between two weights.

```
AT($k$)

Initialize authority weights to 1
Repeat until the weights converge:
    For every hub $i \in H$
        $h_i = \sum_{j \in F_k(i)} a_j$
    For every authority $i \in A$
        $a_i = \sum_{j \in B(i)} h_j$
    Normalize
```

Fig. 5. The AT(k) algorithm.

3.3 The MAX Algorithm

The MAX algorithm is a special case of the AT(k) algorithm for the threshold value $k = 1$. The underlying intuition is that a hub node is as good as the best authority that it points to. That is, a good hub is one that points to at least one good authority.

The MAX algorithm has some interesting properties. First, it is not hard to see that the nodes that receive the highest weight are the nodes with the highest in-degree. We refer to these nodes as the *seed* nodes of the graph. Let d be the in-degree of the seed nodes. If we normalize in the L_∞ norm, then the normalization factor for the authority weights of the MAX algorithm is d. The seed nodes receives weight 1, the maximum weight. A detailed analysis shows that the rest of the nodes are ranked according to their relation to the seed nodes. The convergence and the combinatorial properties of the stationary weights of the MAX algorithm are discussed in detail in Tsaparas [2004b]. In the following paragraphs, we provide some of the intuition.

Let $f : H \rightarrow A$ denote a mapping between hubs and authorities, where the hub j is mapped to authority i, if authority i is the authority with the maximum weight among all the authorities pointed to by hub j. Define $H(i) = \{j \in H : f(j) = i\}$ to be the set of hubs that are mapped to authority i. Recall that the authority graph G_a, defined in Section 2.1, is an undirected graph, where we place an edge between two authorities if they share a hub. We now derive the *directed weighted* graph $G_A = (A, E_A)$ on the authority nodes A, from the authority graph G_a as follows. Let i and j be two nodes in A, such that there exists an edge (i, j) in the graph G_a, and $a_i \neq a_j$. Let $B(i, j) = B(i) \cap B(j)$ denote the set of hubs that point to both authorities i and j. Without loss of generality, assume that $a_i > a_j$. If $H(i) \cap B(i, j) \neq \emptyset$, that is, there exists at least one hub in $B(i, j)$ that is mapped to the authority i, then we place a directed edge from i to j. The weight $c(i, j)$ of the edge (i, j) is equal to the size of the set $H(i) \cap B(i, j)$, that is, it is equal to the number of hubs in $B(i, j)$ that are mapped to i. The intuition of the directed edge (i, j) is that there are $c(i, j)$ hubs that propagate the weight of node i to node j. The graph G_A captures the flow of authority weight between authorities.

Now, let $N(i)$ denote the set of nodes in G_A that point to node i. Also, let $c_i = \sum_{j \in N(i)} c(j, i)$ denote the total weight of the edges that point to i in the graph G_A. This is the number of hubs in the graph G that point to i, but are mapped to some node with weight greater than i. The remaining $d_i - c_i$ hubs (if any) are mapped to node i, or to some node with weight equal to the weight of i. We set $b_i = d_i - c_i$. The number b_i is also equal to the size of the set $H(i)$, the set of hubs that are mapped to node i when all ties are broken in favor of node i.

The graph G_A is a DAG, thus it must have at least one source. We can prove that the sources of G_A are the seed nodes of the graph G [Tsaparas 2004b]. We have already argued that the seed nodes receive maximum weight 1. Let s be some seed node, and let i denote a nonseed node. We define $dist(s, i)$ to be the distance of the longest path in G_A, from s to i. We define the distance of node i, $dist(i) = \max_{s \in S} dist(s, i)$, to be the maximum distance from a seed node to i, over all seed nodes. We note that the distance is well defined, since the graph G_A is a DAG. The following theorem gives a recursive formula for weight a_i, given the weights of the nodes in $N(i)$.

THEOREM 3.1. *Given a graph G, let $C_1, C_2, \dots C_k$ be the connected components of the graph G_a. For every component C_i, $1 \le i \le k$, if component C_i does not contain a seed node, then $a_x = 0$, for all x in C_i. If component C_i contains a seed node, then the weight of the seed node is 1, and for every nonseed node x in C_i, we can recursively compute the weight of node x at distance $\ell > 0$, using the equation*

$$a_x = \frac{1}{d - b_x} \sum_{j \in N(x)} c(j, x) a_j \ ,$$

where for all $j \in N(i)$, $dist(j) < \ell$.

3.4 The Breadth-First-Search (BFS) Algorithm

In this section, we introduce a Link Analysis Ranking algorithm that combines ideas from both the INDEGREE and the HITS algorithms. The INDEGREE algorithm computes the authority weight of a page, taking into account only the popularity of this page within its immediate neighborhood and disregarding the rest of the graph. On the other hand, the HITS algorithm considers the whole graph, taking into account the structure of the graph around the node, rather than just the popularity of that node in the graph.

We now describe the Breadth-First-Search (BFS) algorithm as a generalization of the INDEGREE algorithm inspired by the HITS algorithm. The BFS algorithm extends the idea of popularity that appears in the INDEGREE algorithm from a one-link neighborhood to an n-link neighborhood. The construction of the n-link neighborhood is inspired by the HITS algorithm. Recall from Section 2.2.3 that the weight assigned to node i by the HITS algorithm after n steps is proportional to the number of (BF) paths of length n that leave node i. For the BFS algorithm, instead of considering the number of $(BF)^n$ *paths* that leave i, it considers the number of $(BF)^n$ *neighbors* of node i. Overloading the notation, let $(BF)^n(i)$ denote the set of nodes that can be reached from i by following a $(BF)^n$ path. The contribution of node j to the weight of node i depends on the distance of the node j from i. We adopt an exponentially decreasing weighting scheme. Therefore, the weight of node i is determined as follows:

$$a_i = |B(i)| + \frac{1}{2}|BF(i)| + \frac{1}{2^2}|BFB(i)| + \ldots + \frac{1}{2^{2n-1}}|(BF)^n(i)|.$$

The algorithm starts from node i, and visits its neighbors in BFS order, alternating between backward and forward steps. Every time we move one link further from the starting node i, we update the weight factors accordingly. The algorithm stops either when n links have been traversed, or when the nodes that can be reached from node i are exhausted.

The idea of applying an exponentially decreasing weighting scheme to *paths* that originate from a node has been previously considered by Katz [1953]. In the algorithm of Katz, for some fixed parameter $\alpha < 1$, the weight of node i is equal to

$$a_i = \sum_{k=1}^{\infty}\sum_{j=1}^{n} \alpha^k W^k[j,i],$$

where W^k is the kth power of the adjacency matrix W. The entry $W^k[j,i]$ is the number of paths in the graph G of length k from node j to node i. As we move further away from node i, the contribution of the paths decreases exponentially. There are two important differences between BFS and the algorithm of Katz. First, the way the paths are constructed is different, since the BFS algorithm alternates between backward and forward steps. More important, the BFS algorithm considers the *neighbors* at distance k. Every node j contributes to the weight of node i just once, and the contribution of node j is $1/2^k$ (or α^k if we select a different scaling factor), where k is the shortest path (which alternates between B and F steps) from j to i. In the algorithm of Katz, the same node j may contribute multiple times, and its contribution is the number of paths that connect j with i.

The BFS algorithm ranks the nodes according to their *reachability*, that is, the number of nodes reachable from each node. This property differentiates the BFS algorithm from the remaining algorithms, where *connectivity*, that is, the number of paths that leave each node, is the most important factor in the ranking of the nodes.

3.5 The BAYESIAN Algorithm

We now introduce a different type of algorithm that uses a fully Bayesian statistical approach to compute authority and hub weights. Let P be the set of nodes in the graph. We assume that each node i is associated with three parameters. An (unknown) real parameter e_i, corresponding to its "general tendency to have hypertext links", an (unknown) nonnegative parameter h_i, corresponding to its "tendency to have *intelligent* hypertext links to *authoritative* sites", and an (unknown) nonnegative parameter a_i corresponding to its level of authority.

Our statistical model is as follows. The *a priori* probability of a link from node i to node j is given by

$$\mathcal{P}(i \to j) = \frac{\exp(a_j h_i + e_i)}{1 + \exp(a_j h_i + e_i)}, \tag{3}$$

and the probability of no link from i to j given by

$$\mathcal{P}(i \not\to j) = \frac{1}{1 + \exp(a_j h_i + e_i)}. \tag{4}$$

This reflects the idea that a link is more likely if e_i is large (in which case hub i has a large tendency to link to *any* site), or if *both* h_i and a_j are large (in which case i is an intelligent hub, and j is a high-quality authority).

To complete the specification of the statistical model from a Bayesian point of view (see, e.g., Bernardo and Smith [1994]), we must assign *prior* distributions to the $3n$ unknown parameters e_i, h_i, and a_i. These priors should be general and uninformative, and should *not* depend on the observed data. For large graphs, the choice of priors should have only a small impact on the results. We let $\mu = -5.0$ and $\sigma = 0.1$ be fixed parameters, and let each e_i have prior distribution $N(\mu, \sigma^2)$, a normal distribution with mean μ and variance σ^2. We further let each h_i and a_j have prior distribution Exp(1) (since they have to be nonnegative), meaning that for $x \geq 0$, $\mathcal{P}(h_i \geq x) = \mathcal{P}(a_j \geq x) = \exp(-x)$.

The (standard) Bayesian inference method then proceeds from this fully-specified statistical model by *conditioning* on the observed data, which in this case is the matrix W of actual observed hypertext links in the Base Set. Specifically, when we condition on the data W we obtain a *posterior density* $\pi : \mathbb{R}^{3n} \to [0, \infty)$ for the parameters $(e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n)$. This density is defined so that

$$\begin{aligned}
&\mathcal{P}((e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n) \in S \mid \{W[i, j]\}) \\
&\quad = \textstyle\int_S \pi(e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n) de_1 \ldots de_n dh_1 \ldots dh_n da_1 \ldots da_n
\end{aligned} \tag{5}$$

for any (measurable) subset $S \subseteq \mathbb{R}^{3n}$, and also

$$\begin{aligned}
&\mathcal{E}(g(e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n) \mid \{W[i, j]\}) \\
&\quad = \textstyle\int_{\mathbb{R}^{3n}} g(e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n) \pi(e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n) \\
&\qquad de_1 \ldots de_n dh_1 \ldots dh_n da_1 \ldots da_n
\end{aligned}$$

for any (measurable) function $g : \mathbb{R}^{3n} \to \mathbb{R}$. An easy computation gives the following.

LEMMA 3.2. *For our model, the posterior density is given, up to a multiplicative constant, by*

$$\begin{aligned}
&\pi(e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n) \\
&\quad \propto \prod_{i=1}^{n} \exp(-h_i) \exp(-a_i) \exp[-(e_i - \mu)^2/(2\sigma^2)] \\
&\qquad \times \prod_{(i,j):W[i,j]=1} \exp(a_j h_i + e_i) \Bigg/ \prod_{all\ i,j} (1 + \exp(a_j h_i + e_i)).
\end{aligned}$$

PROOF. We compute that

$$\begin{aligned}
&\mathcal{P}(e_1 \in de_1, \ldots, e_n \in de_n, h_1 \in dh_1, \ldots, h_n \in dh_n, a_1 \in da_1, \ldots, a_n \\
&\quad \in da_n, \{W[i,j]\}) = \prod_{i=1}^{n}[\mathcal{P}(e_i \in de_i)\mathcal{P}(h_i \in dh_i)\mathcal{P}(a_i \in da_i)] \\
&\qquad \times \prod_{i,j:W[i,j]=1} \mathcal{P}(W[i,j]=1 \,|\, e_i, h_i, a_j) \times \prod_{i,j:W[i,j]=0} \mathcal{P}(W[i,j]=0 \,|\, e_i, h_i, a_j) \\
&\quad = \prod_{i=1}^{n}[\exp[-(e_i-\mu)^2/(2\sigma^2)]de_i \exp(-h_i)dh_i \exp(-a_i)da_i] \\
&\qquad \times \prod_{i,j:W[i,j]=1} \frac{\exp(a_j h_i + e_i)}{1+\exp(a_j h_i + e_i)} \prod_{i,j:W[i,j]=0} \frac{1}{1+\exp(a_j h_i + e_i)} \\
&\qquad = \prod_{i=1}^{n}[\exp[-(e_i-\mu)^2/(2\sigma^2)]de_i \exp(-h_i)dh_i \exp(-a_i)da_i] \\
&\qquad\quad \times \prod_{i,j:W[i,j]=1} \exp(a_j h_i + e_i) \Bigg/ \prod_{all\ i,j} (1+\exp(a_j h_i + e_i)).
\end{aligned}$$

The result now follows by inspection. □

Our Bayesian algorithm then reports the conditional means of the $3n$ parameters, according to the posterior density π. That is, it reports final values $\widehat{a}_j$, $\widehat{h}_i$, and $\widehat{e}_i$, where, for example,

$$\widehat{a}_j = \int_{\mathbb{R}^{3n}} a_j \pi(e_1, \ldots, e_n, h_1, \ldots, h_n, a_1, \ldots, a_n)\ de_1 \ldots de_n dh_1 \ldots dh_n da_1 \ldots da_n.$$

To actually compute these conditional means is nontrivial. To accomplish this, we used a *Metropolis Algorithm*. The Metropolis algorithm is an example of a *Markov Chain Monte Carlo Algorithm* (for background see, e.g., Smith and Roberts [1993]; Tierney [1994]; Gilks et al. [1996]; Roberts and Rosenthal [1998]). We denote this algorithm as BAYESIAN.

The Metropolis Algorithm proceeds by starting all the $3n$ parameter values at 1. It then attempts, for each parameter in turn, to add an independent $N(0, \xi^2)$ random variable to the parameter. It then "accepts" this new value with probability $\min(1, \pi(new)/\pi(old))$, otherwise it "rejects" it and leaves the parameter value the way it is. If this algorithm is iterated enough times, and the observed parameter values at each iteration are averaged, then the resulting averages will converge (see, e.g., Tierney [1994]) to the desired conditional means.

There is, of course, some arbitrariness in the specification of the BAYESIAN algorithm, for example, in the form of the prior distributions and in the precise formula for the probability of a link from i to j. However, the model appears to work well in practice as our experiments show. We note that it is possible that the priors for a new search query could instead depend on the performance of page i on different *previous* searches, though we do not pursue that direction here.

The BAYESIAN algorithm is similar in spirit to the PHITS algorithm of Cohn and Chang [2000] in that both use statistical modeling, and both use an iterative algorithm to converge to an answer. However, the algorithms differ substantially in their details. First, they use substantially different statistical models. Second, the PHITS algorithm uses a non-Bayesian (i.e., "classical" or "frequentist") statistical framework, as opposed to the Bayesian framework adopted here.

3.6 The Simplified Bayesian (SBAYESIAN) Algorithm

It is possible to simplify the above Bayesian model by replacing equation (3) with

$$\mathcal{P}(i \to j) = \frac{a_j h_i}{1 + a_j h_i},$$

and correspondingly, replace equation (4) with

$$\mathcal{P}(i \not\to j) = \frac{1}{1 + a_j h_i}.$$

This eliminates the parameters e_i entirely so that we no longer need the prior values μ and σ. A similar model for the generation of links was considered by Azar et al. [2001].

This leads to a slightly modified posterior density $\pi(\cdot)$, now given by $\pi : \mathbb{R}^{2n} \to \mathbb{R}^{\geq 0}$ where

$$\pi(h_1, \ldots, h_n, a_1, \ldots, a_n) \propto \prod_{i=1}^{n} \exp(-h_i)\exp(-a_i) \times \prod_{(i,j):W[i,j]=1} a_j h_i \Bigg/ \prod_{all\ i,j} (1{+}a_j h_i).$$

We denote this Simplified Bayesian algorithm as SBAYESIAN. The SBAYESIAN algorithm was designed to be to similar to the original BAYESIAN algorithm. Surprisingly, we observed that experimentally it performs very similarly to the INDEGREE algorithm.

4. A THEORETICAL FRAMEWORK FOR THE STUDY OF LINK ANALYSIS RANKING ALGORITHMS

The seminal work of Kleinberg [1998] and Brin and Page [1998] was followed by an avalanche of Link Analysis Ranking algorithms (hereinafter denoted *LAR algorithms*) [Borodin et al. 2001; Bharat and Henzinger 1998; Lempel and Moran 2000; Rafiei and Mendelzon 2000; Azar et al. 2001; Achlioptas et al. 2001; Ng et al. 2001b]. Faced with this wide range of choices for LAR algorithms, researchers usually resort to experiments to evaluate them and determine which one is more appropriate for the problem at hand. However, experiments are only indicative of the behavior of the LAR algorithm. In many cases, experimental studies are inconclusive. Furthermore, there are often cases where algorithms exhibit similar properties and ranking behavior. For example, in our experiments (Section 5), we observed a strong "similarity" between the SBAYESIAN and INDEGREE algorithms.

It seems that experimental evaluation of the performance of an LAR algorithm is not sufficient to fully understand its ranking behavior. We need a precise way to evaluate the properties of the LAR algorithms. We would like to be able to formally answer questions of the following type: "How similar are two

LAR algorithms?"; "On what kind of graphs do two LAR algorithms return similar rankings?"; "How does the ranking behavior of an LAR algorithm depend on the specific class of graphs?"; "How does the ranking of an LAR algorithm change as the underlying graph is modified?"; "Is there a set of properties that characterize an LAR algorithm?".

In this section we, describe a formal study of LAR algorithms. We introduce a theoretical framework that allows us to define properties of the LAR algorithms and compare their ranking behavior. We conclude with an axiomatic characterization of the InDegree algorithm.

4.1 Link Analysis Ranking Algorithms

We first need to formally define a Link Analysis Ranking algorithm. Let $\mathcal{G}_n$ denote the set of all possible graphs of size n. The size of the graph is the number of nodes in the graph. Let $\overline{\mathcal{G}}_n \subseteq \mathcal{G}_n$ denote a collection of graphs in $\mathcal{G}_n$. We define a link analysis algorithm $\mathcal{A}$ as a function $\mathcal{A} : \overline{\mathcal{G}}_n \to \mathbb{R}^n$ that maps a graph $G \in \overline{\mathcal{G}}_n$ to an n-dimensional real vector. The vector $\mathcal{A}(G)$ is the authority weight vector (or weight vector) produced by the algorithm $\mathcal{A}$ on graph G. We will refer to this vector as the *LAR vector*, or *authority vector*. The value of the entry $\mathcal{A}(G)[i]$ of the LAR vector $\mathcal{A}(G)$ denotes the authority weight assigned by the algorithm $\mathcal{A}$ to the node i. We will use $\boldsymbol{a}$ (or often $\boldsymbol{w}$) to denote the authority weight vector of algorithm $\mathcal{A}$. In this section, we will sometimes use $a(i)$ instead of a_i to denote the authority weight of node i. All algorithms that we consider are defined over $\mathcal{G}_n$, the class of all possible graphs. We will also consider another class of graphs, $\mathcal{G}_n^{AC}$, the class of *authority connected* graphs. Recall that a graph G is authority connected, if the corresponding authority graph G_a consists of a single component.

We will assume that the weight vector $\mathcal{A}(G)$ is normalized under some chosen norm. The choice of normalization affects the output of the algorithm, so we distinguish between algorithms that use different norms. For any norm L, we define an L-algorithm $\mathcal{A}$ to be an algorithm where the weight vector of $\mathcal{A}$ is normalized under L. That is, the algorithm maps the graphs in $\mathcal{G}_n$ onto the unit L-sphere. For the following, when not stated explicitly, we will assume that the weight vectors of the algorithms are normalized under the L_p norm for some $1 \leq p \leq \infty$.

4.2 Distance Measures Between LAR Vectors

We are interested in comparing different LAR algorithms as well as studying the ranking behavior of a specific LAR algorithm as we modify the underlying graph. To this end, we need to define a distance measure between the LAR vectors produced by the algorithms. Recall that an LAR algorithm $\mathcal{A}$ is a function that maps a graph $G = (P, E)$ from a class of graphs $\overline{\mathcal{G}}_n$ to an n-dimensional authority weight vector $\mathcal{A}(G)$. Let $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ be two LAR vectors defined over the same set of nodes P. We define the *distance* between the LAR vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ as $d(\boldsymbol{a}_1, \boldsymbol{a}_2)$, where $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is some function that maps two real n-dimensional vectors to a real number $d(\boldsymbol{a}_1, \boldsymbol{a}_2)$. We now examine different choices for the function d.

4.2.1 *Geometric Distance Measures.* The first distance functions we consider capture the closeness of the actual weights assigned to every node. The LAR vectors can be viewed as points in an n-dimensional space, thus we can use common geometric measures of distance. We consider the Manhattan distance, that is, the L_1 distance of the two vectors. Let $\boldsymbol{a}_1$, $\boldsymbol{a}_2$ be two LAR vectors. We define the d_1 distance measure between $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ as follows

$$d_1(\boldsymbol{a}_1, \boldsymbol{a}_2) = \min_{\gamma_1,\gamma_2 \geq 1} \sum_{i=1}^{n} |\gamma_1 a_1(i) - \gamma_2 a_2(i)|.$$

The constants γ_1 and γ_2 are meant to allow for an arbitrary scaling of the two vectors, thus eliminating large distances that are caused solely due to normalization factors. For example, let $\overline{\boldsymbol{w}} = (1, 1, \ldots, 1, 2)$ and $\overline{\boldsymbol{v}} = (1, 1, \ldots, 1)$ be two LAR vectors before any normalization is applied. These two vectors appear to be close. Suppose that we normalize the LAR vectors in the L_∞ norm, and let $\boldsymbol{w}_\infty$ and $\boldsymbol{v}_\infty$ denote the normalized vectors. Then $\sum_{i=1}^{n} |w_\infty(i) - v_\infty(i)| = (n-1)/2 = \Theta(n)$. The maximum L_1 distance between any two L_∞-unit vectors is $\Theta(n)$ [Tsaparas 2004a] and therefore, these two vectors appear to be far apart. Suppose now that we normalize in the L_1 norm, and let $\boldsymbol{w}_1$ and $\boldsymbol{v}_1$ denote the normalized vectors. Then $\sum_{i=1}^{n} |w_1(i) - v_1(i)| = \frac{2(n-1)}{n(n+1)} = \Theta(1/n)$. The maximum L_1 distance between any two L_1-unit vectors is $\Theta(1)$; therefore, the two vectors now appear to be close. We use the constants γ_1, γ_2 to avoid such discrepancies.

Instead of the L_1 distance, we may use other geometric distance measures, such as the Euclidean distance L_2. In general, we define the d_q distance, as the L_q distance of the weight vectors. Formally,

$$d_q(\boldsymbol{a}_1, \boldsymbol{a}_2) = \min_{\gamma_1,\gamma_2 \geq 1} \|\gamma_1 a_1(i) - \gamma_2 a_2(i)\|_q.$$

For the remainder of the article, we only consider the d_1 distance measure.

4.2.2 *Rank Distance Measures.* The next distance functions we consider capture the similarity between the *ordinal* rankings induced by the two LAR vectors. The motivation behind this definition is that the ordinal ranking is the usual end-product seen by the user. Let $\boldsymbol{a}$ be an LAR vector defined over a set P of n nodes. The vector $\boldsymbol{a}$ induces a *ranking* of the nodes in P, such that a node i is ranked above node j if $a_i > a_j$. If all weights are distinct, the authority weights induce a *total ranking* of the elements in P. If the weights are not all distinct, then we have a *partial ranking* of the elements in P. We will also refer to total rankings as *permutations*.

The problem of comparing permutations has been studied extensively [Kendall 1970; Diaconis and Graham 1977; Dwork et al. 2001]. One popular distance measure is the *Kendall's tau* distance which captures the number of disagreements between the rankings. Let $\mathcal{P}$ denote the set of all pairs of nodes. Let $\boldsymbol{a}_1, \boldsymbol{a}_2$ be two LAR vectors. We define the *violating set*, $\mathcal{V}(\boldsymbol{a}_1, \boldsymbol{a}_2) \subseteq \mathcal{P}$, as follows

$$\mathcal{V}(\boldsymbol{a}_1, \boldsymbol{a}_2) = \{(i, j)\, in\, \mathcal{P} : (a_1(i) < a_1(j) \wedge a_2(i) > a_2(j)) \vee (a_1(i) > a_1(j) \wedge a_2(i) < a_2(j))\}.$$

That is, the violating set contains the set of pairs of nodes that are ranked in a different order by the two permutations. We now define the indicator function

$\mathcal{I}_{\boldsymbol{a}_1\boldsymbol{a}_2}(i,j)$ as follows.

$$\mathcal{I}_{\boldsymbol{a}_1\boldsymbol{a}_2}(i,j) = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{V}(\boldsymbol{a}_1, \boldsymbol{a}_2) \\ 0 & \text{otherwise} \end{cases}.$$

Kendall's tau is defined as follows.

$$K(\boldsymbol{a}_1, \boldsymbol{a}_2) = \sum_{\{i,j\}\in\mathcal{P}} \mathcal{I}_{\boldsymbol{a}_1\boldsymbol{a}_2}(i,j).$$

Kendall's tau is equal to the number of bubble sort swaps that are necessary to convert one permutation to the other. The maximum value of Kendall's tau is $n(n-1)/2$, and it occurs when one ranking is the reverse of the other.

If the two LAR vectors take distinct values for each node in the set P, then we can compare rankings by directly applying Kendal's tau distance. However, it is often the case that LAR algorithms assigns equal weights to two different nodes. Thus, the algorithm produces a partial ranking of the nodes. In this case, there is an additional source of discrepancy between the two rankings; pairs of nodes that receive equal weight in one vector, but different weights in the other. We define the *weakly violating set*, $\mathcal{W}(\boldsymbol{a}_1, \boldsymbol{a}_2)$, as follows.

$$\mathcal{W}(\boldsymbol{a}_1, \boldsymbol{a}_2) = \{(i,j) : (a_1(i) = a_1(j) \wedge a_2(i) \neq a_2(j)) \vee (a_1(i) \neq a_1(j) \wedge a_2(i) = a_2(j))\}.$$

In order to define a distance measure between partial rankings, we need to address the pairs of nodes that belong to this set. Following the approach in Fagin et al. [2003], we penalize each such pair by a value p, where $p \in [0,1]$. We define the indicator function $\mathcal{I}^{(p)}_{\boldsymbol{a}_1\boldsymbol{a}_2}(i,j)$ as follows.

$$\mathcal{I}^{p}_{\boldsymbol{a}_1\boldsymbol{a}_2}(i,j) = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{V}(\boldsymbol{a}_1, \boldsymbol{a}_2) \\ p & \text{if } (i,j) \in \mathcal{W}(\boldsymbol{a}_1, \boldsymbol{a}_2). \\ 0 & \text{otherwise} \end{cases}$$

Kendall's tau with penalty p is defined as follows.

$$K^{(p)}(\boldsymbol{a}_1, \boldsymbol{a}_2) = \sum_{\{i,j\}\in\mathcal{P}} \mathcal{I}^{(p)}_{\boldsymbol{a}_1\boldsymbol{a}_2}(i,j).$$

The parameter p takes values in $[0,1]$. The value $p = 0$ gives a lenient approach where we penalize the algorithm only for pairs that are weighted so that they *force* an inconsistent ranking. The value $p = 1$ gives a strict approach where we penalize the algorithm for all pairs that are weighted so that they *allow* for an inconsistent ranking. Values of p in $(0,1)$ give a combined approach. We note that $K^{(0)}(\boldsymbol{a}_1, \boldsymbol{a}_2) \leq K^{(p)}(\boldsymbol{a}_1, \boldsymbol{a}_2) \leq K^{(1)}(\boldsymbol{a}_1, \boldsymbol{a}_2)$.

In this article, we only consider the extreme values of p. We define the *weak rank distance*, $d_r^{(0)}$, as follows.

$$d_r^{(0)}(\boldsymbol{a}_1, \boldsymbol{a}_2) = \frac{1}{n(n-1)/2} K^{(0)}(\boldsymbol{a}_1 \boldsymbol{a}_2).$$

We define the *strict rank distance*, $d_r^{(1)}$, as follows.

$$d_r^{(1)}(\boldsymbol{a}_1, \boldsymbol{a}_2) = \frac{1}{n(n-1)/2} K^{(1)}(\boldsymbol{a}_1 \boldsymbol{a}_2).$$

A p-rank distance $d^{(p)}$ can be defined similarly. The rank distance measures are normalized by $n(n-1)/2$, the maximum Kendal's tau distance, so that they takes values in [0, 1]. We will use the weak rank distance when we want to argue about two LAR vectors being far apart, and the strict rank distance when we want to argue about two LAR vectors being close.

The problem of comparing partial rankings is studied independently in Tsaparas [2004a] and Fagin et al. [2004] where they discuss the properties of Kendal tau distance on partial rankings. It is shown that the $d_r^{(1)}$ distance measure is a metric. Also, Fagin et al. [2004] generalize other distance measures for the case of partial rankings.

4.3 Similarity of LAR Algorithms

We now turn to the problem of comparing two LAR algorithms. We first give the following generic definition of *similarity* of two LAR algorithms for any distance function d, and any normalization norm $L = ||\cdot||$.

Definition 4.1. Two L-algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, are similar on the class of graph $\overline{\mathcal{G}}_n$ under distance d, if as $n \to \infty$

$$\max_{G \in \overline{\mathcal{G}}_n} d(\mathcal{A}_1(G), \mathcal{A}_2(G)) = o(M_n(d, L))$$

where $M_n(d, L) = \sup_{\|\boldsymbol{w}_1\|=\|\boldsymbol{w}_2\|=1} d(\boldsymbol{w}_1, \boldsymbol{w}_2)$ is the maximum distance between any two n-dimensional vectors with unit norm $L = ||\cdot||$.

In the definition of similarity, instead of taking the maximum over all $G \in \overline{\mathcal{G}}_n$, we may use some other operator. For example, if there exists some distribution over the graphs in $\overline{\mathcal{G}}_n$, we could replace max operator by the expectation of the distance between the algorithms. In this article, we only consider the max operator.

We now give the following definitions of similarity for the d_1, $d_r^{(0)}$, and $d_r^{(1)}$ distance measures. For the d_1 distance measure, the maximum d_1 distance between any two n-dimensional L_p unit vectors is $\Theta(n^{1-1/p})$ [Tsaparas 2004a].

Definition 4.2. Let $1 \leq p \leq \infty$. Two L_p-algorithms, $\mathcal{A}_1$, and $\mathcal{A}_2$, are d_1-*similar* (or, *similar*) on the class of graphs $\overline{\mathcal{G}}_n$, if as $n \to \infty$,

$$\max_{G \in \overline{\mathcal{G}}_n} d_1(\mathcal{A}_1(G), \mathcal{A}_2(G)) \;=\; o(n^{1-1/p}).$$

Definition 4.3. Two algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, are *weakly rank similar* on the class of graphs $\overline{\mathcal{G}}_n$, if as $n \to \infty$,

$$\max_{G \in \overline{\mathcal{G}}_n} d_r^{(0)}(\mathcal{A}_1(G), \mathcal{A}_2(G)) \;=\; o(1).$$

Definition 4.4. Two algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, are *strictly rank similar* on the class of graphs $\overline{\mathcal{G}}_n$, if as $n \to \infty$,

$$\max_{G \in \overline{\mathcal{G}}_n} d_r^{(1)}(\mathcal{A}_1(G), \mathcal{A}_2(G)) \;=\; o(1).$$

Definition 4.5. Two algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, are *rank consistent* on the class of graphs $\overline{\mathcal{G}}_n$, if for every graph $G \in \overline{\mathcal{G}}_n$,

$$d_r^{(0)}(\mathcal{A}_1(G), \mathcal{A}_2(G)) = 0.$$

Definition 4.6. Two algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, are *rank equivalent* on the class of graphs $\overline{\mathcal{G}}_n$, if for every graph $G \in \overline{\mathcal{G}}_n$,

$$d_r^{(1)}(\mathcal{A}_1(G), \mathcal{A}_2(G)) = 0.$$

We note that, according to the above definition, every algorithm is rank consistent with the trivial algorithm that gives the same weight to all authorities. Although this may seem somewhat bizarre, it does have an intuitive justification. For an algorithm whose goal is to produce an *ordinal* ranking, the weight vector with all weights equal conveys no information; therefore, it lends itself to all possible ordinal rankings. The weak rank distance counts only the pairs that are weighted inconsistently, and in this case there are none. If a stronger notion of similarity is needed, the $d_r^{(1)}$ distance measure can be used where all such pairs contribute to the distance.

From the definitions in Section 4.2.2, it is obvious that if two algorithms are strictly rank similar, then they are similar under p-rank distance $d_r^{(p)}$ for every $p < 1$. Equivalently, if two algorithms are not weakly rank similar, then they are not similar under p-rank distance for every $p > 0$.

The definition of similarity depends on the normalization of the algorithms. In the following, we show that, for the d_1 distance, similarity in the L_p norm implies similarity in the L_q norm, for any $q > p$.

THEOREM 4.7. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two algorithms, and let $1 \leq p \leq q \leq \infty$. If the L_p-algorithm $\mathcal{A}_1$ and the L_p-algorithm $\mathcal{A}_2$ are similar, then the L_q-algorithm $\mathcal{A}_1$ and the L_q-algorithm $\mathcal{A}_2$ are also similar.*

PROOF. Let G be a graph of size n, and let $\boldsymbol{u} = \mathcal{A}_1(G)$, and $\boldsymbol{v} = \mathcal{A}_2(G)$ be the weight vectors of the two algorithms. Let $\boldsymbol{v}_p$ and $\boldsymbol{u}_p$ denote the weight vectors, normalized in the L_p norm, and let $\boldsymbol{v}_q$ and $\boldsymbol{u}_q$ denote the weight vectors, normalized in the L_q norm. Since the L_p-algorithm $\mathcal{A}_1$ and the L_p-algorithm $\mathcal{A}_2$ are similar, there exist $\gamma_1, \gamma_2 \geq 1$ such that

$$d_1(\boldsymbol{v}_p, \boldsymbol{u}_p) = \sum_{i=1}^{n} |\gamma_1 v_p(i) - \gamma_2 u_p(i)| = o(n^{1-1/p}).$$

Now, $\boldsymbol{v}_q = \boldsymbol{v}_p / \|\boldsymbol{v}_p\|_q$, and $\boldsymbol{u}_q = \boldsymbol{u}_p / \|\boldsymbol{u}_p\|_q$. Therefore, $\sum_{i=1}^{n} |\gamma_1 \|\boldsymbol{v}_p\|_q v_q(i) - \gamma_2 \|\boldsymbol{u}_p\|_q u_q(i)| = o(n^{1-1/p})$. Without loss of generality, assume that $\|\boldsymbol{u}_p\|_q \geq \|\boldsymbol{v}_p\|_q$. Then

$$\|\boldsymbol{v}_p\|_q \sum_{i=1}^{n} \left| \gamma_1 v_q(i) - \gamma_2 \frac{\|\boldsymbol{u}_p\|_q}{\|\boldsymbol{v}_p\|_q} u_q(i) \right| = o(n^{1-1/p}).$$

We set $\gamma_1' = \gamma_1$ and $\gamma_2' = \gamma_2 \frac{\|\boldsymbol{u}_p\|_q}{\|\boldsymbol{v}_p\|_q}$. Then we have that

$$d_1(\boldsymbol{v}_q, \boldsymbol{u}_q) \leq \sum_{i=1}^{n} |\gamma_1' v_q(i) - \gamma_2' u_q(i)| = o\left(\frac{n^{1-1/p}}{\|\boldsymbol{v}_p\|_q}\right).$$

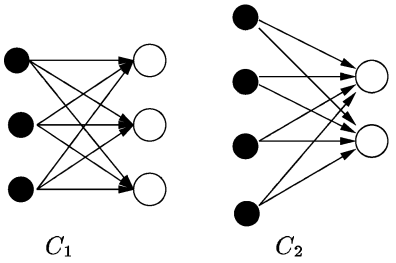


Fig. 6. Dissimilarity of HITS and INDEGREE. The graph G for $r = 2$.

It is known [Tsaparas 2004a] that $\|\boldsymbol{v}_p\|_q \geq \|\boldsymbol{v}_p\|_p n^{1/q-1/p} = n^{1/q-1/p}$. Hence, $\frac{n^{1-1/p}}{\|\boldsymbol{v}_p\|_q} \leq \frac{n^{1-1/p}}{n^{1/q-1/p}} = n^{1-1/q}$. Therefore, $d_1(\boldsymbol{v}_q, \boldsymbol{u}_q) = o(n^{1-1/q})$, and thus L_q-algorithm $\mathcal{A}_1$, and L_q-algorithm $\mathcal{A}_2$ are similar. □

Theorem 4.7 implies that if two L_1-algorithms are similar, then the corresponding L_p-algorithms are also similar, for any $1 \leq p \leq \infty$. Consequently, if two L_∞-algorithms are dissimilar, then the corresponding L_p-algorithms are also dissimilar, for any $1 \leq p \leq \infty$. Therefore, all dissimilarity results proven for the L_∞ norm hold for any L_p norm, for $1 \leq p \leq \infty$.

4.3.1 *Similarity Results.* We now consider the similarity of the HITS, INDEGREE, SALSA, HUBAVG, and MAX algorithms. We will show that no pair of algorithms are similar, or rank similar, in the class $\mathcal{G}_n$ of all possible graphs of size n. For the dissimilarity results under the d_1 distance measure, we will assume that the weight vectors are normalized under the L_∞ norm. Dissimilarity between two L_∞-algorithms implies dissimilarity in L_p norm, for $p < \infty$. Also, for rank similarity, we will use the strict rank distance, $d_r^{(1)}$, while for rank dissimilarity we will use the weak rank distance, $d_r^{(0)}$.

4.3.1.1 *The* HITS *and the* INDEGREE *Algorithms.*

PROPOSITION 4.8. *The* HITS *and the* INDEGREE *algorithms are neither similar, nor weakly rank similar on $\mathcal{G}_n$.*

PROOF. Consider a graph G on $n = 7r - 2$ nodes that consists of two disconnected components. The first component C_1 consists of a complete bipartite graph with $2r-1$ hubs and $2r-1$ authorities. The second component C_2 consists of a bipartite graph with $2r$ hubs and r authorities. The graph G for $r = 2$ is shown in Figure 6.

Let $\boldsymbol{a}$ and $\boldsymbol{w}$ denote the weight vectors of the HITS and the INDEGREE algorithm, respectively, on graph G. The HITS algorithm allocates all the weight to the nodes in C_1. After normalization, for all $i \in C_1, a_i = 1$, while for all $j \in C_2, a_j = 0$. On the other hand, the INDEGREE algorithm distributes the weight to both components, allocating more weight to the nodes in C_2. After the normalization step, for all $j \in C_2$, $w_j = 1$, while for all $i \in C_1$, $w_i = \frac{2r-1}{2r}$.

There are r nodes in C_2 for which $w_i = 1$ and $a_i = 0$. For all $\gamma_1, \gamma_2 \geq 1$, $\sum_{i \in C_2} |\gamma_1 w_i - \gamma_2 a_i| \geq r$. Therefore, $d_1(\boldsymbol{w}, \boldsymbol{a}) = \Omega(r) = \Omega(n)$, which proves that the algorithms are not similar.

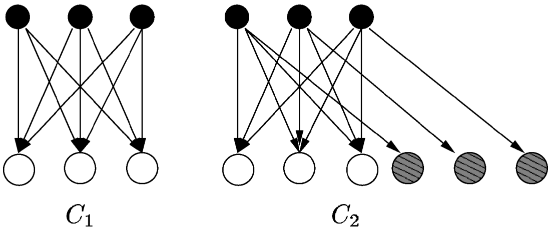


Fig. 7. Dissimilarity of HubAvg and Hits. The graph G for $r = 3$.

The proof for weak rank dissimilarity follows immediately from the above. For every pair of nodes $\{i, j\}$ such that $i \in C_1$ and $j \in C_2$, $a_i > a_j$ and $w_i < w_j$. There are $\Theta(n^2)$ such pairs, therefore, $d_r^{(0)}(\boldsymbol{a}, \boldsymbol{w}) = \Theta(1)$. Thus, the two algorithms are not weakly rank similar. □

4.3.1.2 *The* HubAvg *and* Hits *Algorithms.*

Proposition 4.9. *The* HubAvg *and* Hits *algorithms are neither similar, nor weakly rank similar on $\mathcal{G}_n$.*

Proof. Consider a graph G on $n = 5r$ nodes that consists of two disconnected components. The first component C_1 consists of a complete bipartite graph with r hub and r authority nodes. The second component C_2 consists of a complete bipartite graph C with r hub and r authority nodes, and a set of r "external" authority nodes E, such that each hub node in C points to a node in E, and no two hub nodes in C point to the same "external" node. Figure 7 shows the graph G for $r = 3$.

Let $\boldsymbol{a}$ and $\boldsymbol{w}$ denote the weight vectors of the Hits and the HubAvg algorithm, respectively, on graph G. It is not hard to see that the Hits algorithm allocates all the weight to the authority nodes in C_2. After normalization, for all authority nodes $i \in C$, $a_i = 1$, for all $j \in E$, $a_j = \frac{1}{r-1}$, and for all $k \in C_1$, $a_k = 0$. On the other hand, the HubAvg algorithm allocates all the weight to the nodes in C_1. After normalization, for all authority nodes $k \in C_1$, $w_k = 1$, and for all $j \in C_2$, $w_j = 0$.

Let $U = C_1 \cup C$. The set U contains $2r$ authority nodes. For every authority $i \in U$, either $a_i = 1$ and $w_i = 0$, or $a_i = 0$ and $w_i = 1$. Therefore, for all $\gamma_1, \gamma_2 \geq 1$, $\sum_{i \in U} |\gamma_1 a_i - \gamma_2 w_i| \geq 2r$. Thus, $d_1(\boldsymbol{a}, \boldsymbol{w}) = \Omega(r) = \Omega(n)$, which proves that the algorithms are not similar.

The proof for weak rank dissimilarity follows immediately from the above. For every pair of authority nodes (i, j) such that $i \in C_1$ and $j \in C_2$, $a_i < w_j$, and $a_i > w_j$. There are $\Theta(n^2)$ such pairs, therefore, $d_r^{(0)}(\boldsymbol{a}, \boldsymbol{w}) = \Theta(1)$. Thus, the two algorithms are not weakly rank similar. □

4.3.1.3 *The* HubAvg *and* InDegree *Algorithms.*

Proposition 4.10. *The* HubAvg *algorithm and the* InDegree *algorithm are neither similar, nor weakly rank similar on G_n.*

Proof. Consider a graph G with $n = 15r$ nodes. The graph G consists of r copies of a subgraph G_s on 15 nodes. The subgraph G_s contains two components,

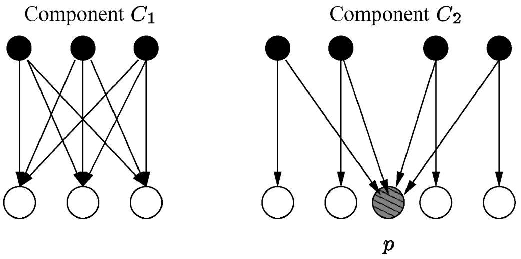


Fig. 8. Dissimilarity of HubAvg and InDegree. The G_s graph.

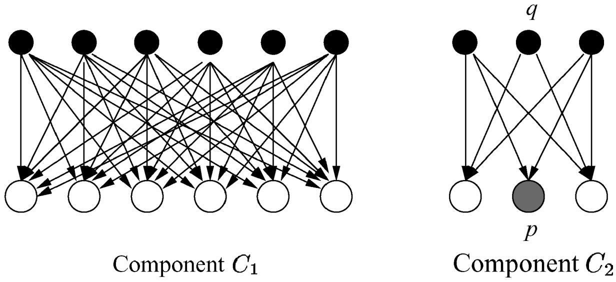


Fig. 9. Dissimilarity of Salsa with Hits,InDegree and HubAvg. The graph G for $r = 3$.

C_1 and C_2. The component C_1 is a complete bipartite graph with 3 hubs and 3 authorities. The component C_2 consists of 4 hubs that all point to an authority node p. Furthermore, each hub points to one more authority, a different one for each hub. The graph G_s is shown in Figure 8.

Let $\boldsymbol{a}$ denote the authority weight vector of HubAvg algorithm, and let $\boldsymbol{w}$ denote the authority weight of the InDegree algorithm on graph G. It is not hard to see that for every copy of the subgraph G_s, the HubAvg algorithm assigns weight 1 to the authorities in component C_1 and zero weight to component C_2. On the other hand, the InDegree algorithm assigns weight 1 to all nodes with in-degree 4, and weight 3/4 to the authorities in the C_1 components of the copies of the G_s subgraph. Since the graph G contains r copies of the graph G_s, it follows that there are $r = \Theta(n)$ nodes for which $a_i = 0$ and $w_i = 1$. Therefore, $d_r(\boldsymbol{a}, \boldsymbol{w}) = \Theta(1)$. Furthermore, for all $\gamma_1, \gamma \geq 1$, $\|\gamma_1 \boldsymbol{a} - \gamma_2 \boldsymbol{w}\|_1 = \Theta(n)$. Thus, HubAvg and InDegree are neither similar, nor weakly rank similar. □

4.3.1.4 *The* Salsa *Algorithm.*

Proposition 4.11. *The* Salsa *algorithm is neither similar, nor weakly rank similar, to the* InDegree, HubAvg, *or* Hits *algorithms.*

Proof. Consider a graph G on $n = 6r$ nodes that consists of two components, C_1 and C_2. The component C_1 is a complete bipartite graph with $2r$ hubs and $2r$ authorities. The component C_2 is a complete bipartite graph with r hubs and r authorities, with one link (q, p) removed. Figure 9 shows the graph G for $r = 3$.

Let $\boldsymbol{u}$, $\boldsymbol{a}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ denote the normalized weight vectors for Salsa, Hits, HubAvg, and InDegree algorithms, respectively. Also, let $\boldsymbol{u}_1$ denote the Salsa weight vector normalized in the L_1 norm (i.e., as it is computed by the random walk of the SALSA algorithm). The SALSA algorithm allocates weight $u_1(i) = 1/3r$ for all authority nodes $i \in C_1$, and weight $u_1(j) = (r-1)/3(r^2-1)$ for all authority node $j \in C_2 \setminus \{p\}$. Hub nodes receive weight zero for all algorithms. It is interesting to note that the removal of the link (q, p) increases the weight of the rest of the nodes in C_2. Since $(r-1)/3(r^2-1) > 1/3r$, after normalization in the L_∞ norm, we have that $u_i = 1-\frac{1}{r^2}$ for all $i \in C_1$, and $u_j = 1$ for all $j \in C_2 \setminus \{p\}$. On the other hand, both the HITS and HUBAVG algorithms distribute all the weight equally to the authorities in the C_1 component and allocate zero weight to the nodes in the C_2 component. Therefore, after normalization, $a_i = v_i = 1$ for all nodes $i \in C_1$, and $a_j = v_j = 0$ for all nodes $j \in C_2$. The INDEGREE algorithm allocates weight proportionally to the in-degree of the nodes, therefore, after normalization, $w_i = 1$ for all nodes in C_1, while $w_j = \frac{1}{2}$ for all nodes $j \in C_2 \setminus \{p\}$.

Let $\|\cdot\|$ denote the L_1 norm. For the HITS and HUBAVG algorithm, there are r entries in $C_2 \setminus \{p\}$, for which $a_i = v_i = 0$ and $u_i = 1$. Therefore, for all of $\gamma_1, \gamma_2 \geq 1$, $\|\gamma_1 \boldsymbol{u} - \gamma_2 \boldsymbol{a}\| = \Omega(r) = \Omega(n)$, and $\|\gamma_1 \boldsymbol{u} - \gamma_2 \boldsymbol{a}\| = \Omega(r) = \Omega(n)$. From this, we have that $d_r^{(0)}(\boldsymbol{u}, \boldsymbol{a}) = \Theta(1)$, and $d_r^{(0)}(\boldsymbol{u}, \boldsymbol{v}) = \Theta(1)$.

The proof for the INDEGREE algorithm, is a little more involved. Let

$$S_1 = \sum_{i \in C_1} |\gamma_1 w_i - \gamma_2 u_i| = 2r \left| \gamma_1 - \gamma_2 - \frac{\gamma_2}{r^2} \right|$$

$$S_2 = \sum_{i \in C_2 \setminus \{p\}} |\gamma_1 w_i - \gamma_2 u_i| = r \left| \gamma_1 \frac{1}{2} - \gamma_2 \right| .$$

We have that $\|\gamma_1 \boldsymbol{w} - \gamma_2 \boldsymbol{u}\| \geq S_1 + S_2$, unless $\frac{1}{2}\gamma_1 - \gamma_2 = o(1)$, then $S_2 = \Theta(r) = \Theta(n)$. If $\gamma_1 = 2\gamma_2 + o(1)$, since $\gamma_1, \gamma_2 \geq 1$, we have that $S_1 = \Theta(r) = \Theta(n)$. Therefore, $d_1(\boldsymbol{w}, \boldsymbol{u}) = \Omega(n)$. From this, $d_r^{(0)}(\boldsymbol{w}, \boldsymbol{u}) = \Theta(1)$.

Thus, SALSA is neither similar, nor weakly rank similar, to HITS, INDEGREE, and HUBAVG. □

4.3.2 *Other Results.* On the positive side, the following Lemma follows immediately from the definition of the SALSA algorithm and the definition of the authority-connected class of graphs.

LEMMA 4.12. *The* SALSA *algorithm is rank equivalent and similar to the* INDEGREE *algorithm on the class of authority connected graphs* $\mathcal{G}_n^{AC}$.

In a recent work, Lempel and Moran [2003] showed that the HITS, INDEGREE (SALSA), and PAGERANK algorithms are not weakly rank similar on the class of authority connected graphs, $\mathcal{G}_n^{AC}$. The similarity of the MAX algorithm with the rest of the algorithms is studied in Tsaparas [2004a]. It is shown that the MAX algorithm is neither similar, nor weakly rank similar, with the HITS, HUBAVG, INDEGREE, and SALSA algorithms.

4.4 Stability

In the previous section, we examined the similarity of two different algorithms on the same graph G. In this section, we are interested in how the output of a *fixed* algorithm changes as we alter the graph. We would like small changes in the graph to have a small effect on the weight vector of the algorithm. We capture this requirement by the definition of stability. The notion of stability has been independently considered (but not explicitly defined) in a number of different papers [Ng et al. 2001a, 2001b; Azar et al. 2001; Achlioptas et al. 2001]. For the definition of stability, we will use some of the terminology employed by Lempel and Moran [2001].

Let $\overline{\mathcal{G}}_n$ be a class of graphs, and let $G = (P, E)$ and $G' = (P, E')$ be two graphs in $\overline{\mathcal{G}}_n$. We define the *link distance* d_ℓ between graphs G and G' as follows.

$$d_\ell\left(G, G'\right) = \left|(E \cup E') \setminus (E \cap E')\right|.$$

That is, $d_\ell(G, G')$ is the minimum number of links that we need to add and/or remove so as to change one graph into the other. Generalizations and alternatives to d_ℓ are considered in Tsaparas [2004a].

Given a class of graphs $\overline{\mathcal{G}}_n$, we define a *change*, ∂, within class $\overline{\mathcal{G}}_n$ as a pair $\partial = \{G, G'\}$, where $G, G' \in \overline{\mathcal{G}}_n$. The size of the change is defined as $|\partial| = d_\ell(G, G')$. We say that a change ∂ *affects* node i if the links that point to node i are altered. In algebraic terms, the ith column vectors of the adjacency matrices W and W' are different. We define the *impact set* of a change ∂, $\{\partial\}$, to be the set of nodes affected by the change ∂.

For a graph $G \in \mathcal{G}_n$, we define the set $\mathcal{C}_k(G) = \{G' \in \overline{\mathcal{G}}_n : d_\ell(G, G') \leq k\}$. The set $C_k(G)$ contains all graphs that have a link distance of at most k from graph G, that is, all graphs G' that can be produced from G, with a change of size of at most k.

We are now ready to define stability. The definition of stability depends upon the normalization norm and the distance measure.

Definition 4.13. An L-algorithm $\mathcal{A}$ is stable on the class of graphs $\overline{\mathcal{G}}_n$ under distance d, if for every fixed positive integer k, we have as $n \to \infty$

$$\max_{G \in \overline{\mathcal{G}}_n, G' \in \mathcal{C}_k(G)} d(\mathcal{A}(G), \mathcal{A}(G')) = o(M_n(d, L)),$$

where $M_n(d, L) = \sup_{\|\boldsymbol{w}_1\| = \|\boldsymbol{w}_2\| = 1} d(\boldsymbol{w}_1, \boldsymbol{w}_2)$ is the maximum distance between any two n-dimensional vectors with unit norm $L = || \cdot ||$.

We now give definitions for stability for the specific distance measures we consider.

Definition 4.14. An L_p-algorithm $\mathcal{A}$ is d_1-stable (or, stable) on the class of graphs $\overline{\mathcal{G}}_n$, if for every fixed positive integer k, we have as $n \to \infty$

$$\max_{G \in \overline{\mathcal{G}}_n, G' \in \mathcal{C}_k(\overline{\mathcal{G}}_n)} d_1(\mathcal{A}(G), \mathcal{A}(G')) = o(n^{1-1/p}).$$

Definition 4.15. An algorithm $\mathcal{A}$ is *weakly rank stable* on the class of graphs $\overline{\mathcal{G}}_n$, if for every fixed positive integer k, we have as $n \to \infty$

$$\max_{G \in \overline{\mathcal{G}}_n, G' \in \mathcal{C}_k(G)} d_r^{(0)}(\mathcal{A}(G), \mathcal{A}(G')) = o(1).$$

Definition 4.16. An algorithm $\mathcal{A}$ is *strictly rank stable* on the class of graphs $\overline{\mathcal{G}}_n$, if for every fixed positive integer k, we have as $n \to \infty$

$$\max_{G \in \overline{\mathcal{G}}_n, G' \in \mathcal{C}_k(G)} d_r^{(1)}(\mathcal{A}(G), \mathcal{A}(G')) = o(1).$$

As in the case of similarity, strict rank stability implies stability for all p-rank distance measures $d_r^{(p)}$, while weak rank instability implies instability for all p-rank distance measures.

Stability seems to be a desirable property. If an algorithm is not stable, then slight changes in the link structure of the Base Set may lead to large changes in the rankings produced by the algorithm. Given the rapid evolution of the Web, stability is necessary to guarantee consistent behavior of the algorithm. Furthermore, stability may provide some "protection" against malicious spammers.

The following theorem is the analogue of Theorem 4.7 for stability.

THEOREM 4.17. *Let $\mathcal{A}$ be an algorithm, and let $1 \leq p \leq q \leq \infty$. If the L_p-algorithm $\mathcal{A}$ is stable on class $\overline{\mathcal{G}}_n$, then the L_q-algorithm $\mathcal{A}$ is also stable on $\overline{\mathcal{G}}_n$.*

PROOF. Let $\partial = \{G, G'\}$ be a change within $\overline{\mathcal{G}}_n$ of 3 size of at most k, for a fixed constant k. Set $\boldsymbol{v} = \mathcal{A}(G)$, and $\boldsymbol{u} = \mathcal{A}(G')$, and then the rest of the proof is identical to the proof of Theorem 4.7. □

Theorem 4.17 implies that, if an L_1-algorithm $\mathcal{A}$ is stable then the L_p-algorithm $\mathcal{A}$ is also stable for any $1 \leq p \leq \infty$. Consequently, if the L_∞-algorithm $\mathcal{A}$ is unstable then the L_p-algorithm $\mathcal{A}$ is also unstable for any $1 \leq p \leq \infty$. Therefore, instability results, proven for the L_∞ norm, hold for any L_p norm for $1 \leq p \leq \infty$.

4.4.1 *Stability and Similarity.* We now prove an interesting connection between stability and similarity.

THEOREM 4.18. *Let d be a distance function that is a metric, or a near metric[2] If two L-algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, are similar under d on the class $\overline{\mathcal{G}}_n$, and the algorithm $\mathcal{A}_1$ is stable under d on the lass $\overline{\mathcal{G}}_n$, then $\mathcal{A}_2$ is also stable under d on the class $\overline{\mathcal{G}}_n$.*

PROOF. Let $\partial = \{G, G'\}$ be a change in $\overline{\mathcal{G}}_n$ of size k, where k is some fixed constant independent of n. Now let $\boldsymbol{w}_1 = \mathcal{A}_1(G)$, $\boldsymbol{w}_2 = \mathcal{A}_2(G)$, $\boldsymbol{w}'_1 = A(G')$, and $\boldsymbol{w}'_2 = A(G')$. Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are similar, we have that $d(\boldsymbol{w}_1, \boldsymbol{w}_2) = o(M_n(d, L))$, and $d(\boldsymbol{w}'_1, \boldsymbol{w}'_2) = o(M_n(d, L))$. Since $\mathcal{A}_1$ is stable, we have that $d(\boldsymbol{w}_1, \boldsymbol{w}'_1) = o(M_n(d, L))$. Since the distance measure d is a metric, or a near metric, we have that

$$d(\boldsymbol{w}_2, \boldsymbol{w}'_2) = O(d(\boldsymbol{w}_1, \boldsymbol{w}_2) + d(\boldsymbol{w}'_1, \boldsymbol{w}'_2) + d(\boldsymbol{w}_1, \boldsymbol{w}'_1)) = o(M_n(d, L)).$$

Therefore, $\mathcal{A}_2$ is stable on $\overline{\mathcal{G}}_n$. □

4.4.2 *Stability Results*

PROPOSITION 4.19. *The* HITS *and* HUBAVG *algorithms are neither stable, nor weakly rank stable, on class $\mathcal{G}_n$.*

[2] A near metric [Fagin et al. 2003] is a distance function that is reflexive and symmetric, and satisfies the following relaxed polygonal inequality. There is a constant c, independent of n, such that for all $k > 0$, and all vectors $\boldsymbol{u}, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_k, \boldsymbol{v}$, $d(\boldsymbol{u}, \boldsymbol{v}) \leq c(d(\boldsymbol{u}, \boldsymbol{w}_1) + d(\boldsymbol{w}_1, \boldsymbol{w}_2) + \cdots + d(\boldsymbol{w}_k, \boldsymbol{v}))$.

PROOF. Consider the graph G of size $n = 2r + 1$ that consists of two disjoint components, C_1 and C_2, each a complete graph on r nodes. There is also an extra hub node h that points to some node in C_1. For both HITS and HUBAVG, in the corresponding matrices M_H and M_{HA}, the singular value of the component C_1 is (slightly) larger than that of C_2. Therefore, both algorithms will allocate all the weight to the nodes of C_1 and zero weight to C_2. Now, construct the graph G' by removing the link from h to C_1 and adding a link to some node in C_2. In G', the singular value of C_2 becomes larger than that of C_1, causing all the weight to shift from C_1 to C_2, and leaving the nodes in C_1 with zero weight. It follows that the two algorithms are neither stable, nor weakly rank stable. □

PROPOSITION 4.20. *The* SALSA *algorithm, is neither stable, nor weakly rank stable, on the class $\mathcal{G}_n$.*

PROOF. We first establish the rank instability of the SALSA algorithm. The example is similar to that used in the previous proof. Consider a graph G of size $n = 2r + 1$ which consists of two disjoint components. The first component consists of a complete graph C_1 on r nodes and an extra authority p that is pointed to by a single node of the complete graph C_1. The second component consists of a complete graph C_2 on r nodes.

Let $\boldsymbol{a}$ denote the weight vector of the SALSA algorithm on the graph G. Then for every node $i \in C_1$, $a_i = \frac{r+1}{2r+1}\frac{r-1}{r(r-1)+1}$. For every node $j \in C_2$, $a_j = \frac{1}{2r+1}$. If $r > 2$, then the SALSA algorithm ranks the r authorities in C_1 higher than those in C_2. We now remove the link from the node in C_1 to node p, and we add a link from a node in C_2 to p. Now, the nodes in C_2 are ranked higher than the nodes in C_1. There are $\Theta(n^2)$ pairs of nodes whose relative order is changed; therefore, SALSA is weakly rank unstable.

The proof of instability is a little more involved. Consider again the graph G that consists of two complete graphs, C_1 and C_2, of size n_1 and n_2, respectively, such that $n_2 = cn_1$, where $c < 1$ is a fixed constant. There exists also an extra hub h that points to two authorities p and q from the components, C_1 and C_2, respectively. The graph has $n = n_1 + n_2 + 1$ nodes, and $n_a = n_1 + n_2$ authorities.

The authority Markov chain defined by the SALSA algorithm is irreducible; therefore, the weight of authority i is proportional to the in-degree of node i. Let $\boldsymbol{a}$ be the weight vector of the SALSA algorithm. Node p is the node with the highest in-degree, $B(p) = n_1$, and therefore, after normalizing in the L_∞ norm, $a_p = 1$, $a_i = 1 - 1/n_1$ for all $i \in C_1 \setminus \{p\}$, $a_q = c$, and $a_j = c - 1/n_1$ for all $j \in C_2 \setminus \{q\}$.

Now let G' be the graph G after we remove the two links from hub h to authorities p and q. Let $\boldsymbol{a}'$ denote the weight vector of the SALSA algorithm on graph G'. It is not hard to see that all authorities receive the same weight $1/n_a$ by the SALSA algorithm. After normalization, $a'_i = 1$ for all authorities i in G'.

Consider now the difference $\|\gamma_1\boldsymbol{a} - \gamma_2\boldsymbol{a}'\|_1$. Let

$$S_1 = \sum_{C_1\setminus\{p\}} |\gamma_1 a_i - \gamma_2 a'_i| = (n_1 - 1)\left|\gamma_1 - \gamma_2 - \frac{\gamma_1}{n_1}\right|$$

$$S_2 = \sum_{C_2 \setminus \{q\}} |\gamma_1 a_i - \gamma_2 a'_i| = (n_2 - 1) \left| c\gamma_1 - \gamma_2 - \frac{\gamma_1}{n_1} \right| .$$

It holds that $\|\gamma_1 \boldsymbol{a} - \gamma_2 \boldsymbol{a}'\|_1 \geq S_1 + S_2$. It is not hard to see that unless $\gamma_1 = \frac{1}{c}\gamma_2 + o(1)$, then $S_2 = \Theta(n_2) = \Theta(n)$. If $\gamma_1 = \frac{1}{c}\gamma_2 + o(1)$, then $S_1 = \Theta(n_1) = \Theta(n)$. Therefore, $d_1(\boldsymbol{a}, \boldsymbol{a}') = \Omega(n)$. Thus, the SALSA algorithm is unstable. □

On the positive side, we can prove that the INDEGREE algorithm is stable.

THEOREM 4.21. *The* INDEGREE *algorithm is stable on the class $\mathcal{G}_n$.*

PROOF. Let $\partial = \{G, G'\}$ be a change within $\mathcal{G}_n$ of size k. Let m be the size of the impact set $\{\partial\}$ where $m \leq k$. Without loss of generality, assume that $\{\partial\} = \{1, 2, \ldots, m\}$. Let $\boldsymbol{u}$ be the weight vector that assigns to node i weight equal to $|B(i)|$, the in-degree of i. Let $\boldsymbol{w}$ be the weight of the L_1-INDEGREE algorithm. Then $\boldsymbol{w} = \boldsymbol{u}/\|\boldsymbol{u}\|$ where $\|\cdot\|$ is the L_1 norm. Let $\boldsymbol{u}'$ and $\boldsymbol{w}'$ denote the corresponding weight vectors for the graph G'. For all $i \notin \{1, 2, \ldots, m\}$ $u'_i = u_i$. Furthermore, $\sum_{i=1}^{m} |u_i - u'_i| \leq k$. Set $\gamma_1 = 1$ and $\gamma_2 = \frac{\|\boldsymbol{u}'\|}{\|\boldsymbol{u}\|}$. Then

$$\|\gamma_1 \boldsymbol{w} - \gamma_2 \boldsymbol{w}'\|_1 = \frac{1}{\|\boldsymbol{u}\|} \sum_{i=1}^{n} |u_i - u'_i| \leq \frac{k}{\|\boldsymbol{u}\|} .$$

We note that $\|\boldsymbol{u}\|$ is equal to the sum of the links in the graph; therefore, $\|\boldsymbol{u}\| = \Omega(1)$. Thus, $d_1(\boldsymbol{w}, \boldsymbol{w}') = o(1)$ which proves that, L_1-INDEGREE, and consequently, INDEGREE is stable. □

We examine the rank stability of INDEGREE in Section 4.5 where we discuss *locality*.

4.4.3 *Other Results.* Following the work of Borodin et al. [2001], Lempel and Moran [2003] proved that the HITS and PAGERANK algorithms are not weakly rank stable on the class $\mathcal{G}_n^{AC}$ of authority connected graphs. Recently, Lee and Borodin [2003] considered a different definition of stability where, given a change $\partial = \{G, G'\}$, the distance between the weight vectors of an LAR algorithm on graphs G and G' may depend on the weights of the nodes whose in and out links were affected. The intuition is that, if a change is performed on a highly authoritative node, then we expect a large change in the weights. They prove that, under their definition, the PAGERANK algorithm is stable. They also prove the stability of a randomized version of SALSA where, similar to PAGERANK, at each iteration a random jump may be performed. On the negative side, they prove that HITS and SALSA remain unstable. The stability of the MAX algorithm is studied in Tsaparas [2004a] where it is shown that the MAX algorithm is neither stable nor weakly rank stable, even when restricted to authority connected graphs.

4.5 Locality

The properties we have considered so far are either of practical interest (similarity) or desirable (stability). In the following, we consider properties that, although of no immediate practical use, help to *characterize* an LAR algorithm. The importance of these properties will become clear in Section 4.8 where we use them to provide an axiomatic characterization of the InDegree algorithm.

We now introduce the concept of "locality". The idea behind locality is that for a local algorithm, a change should not affect the relative order of the nodes that are not affected by the change.

Definition 4.22. An algorithm $\mathcal{A}$ is local if for every change $\partial = \{G, G'\}$, there exists $\lambda > 0$ such that $\mathcal{A}(G')[i] = \lambda \mathcal{A}(G)[i]$ for all $i \notin \{\partial\}$.

Definition 4.23. An algorithm $\mathcal{A}$ is *weakly rank local* if for every change $\partial = \{G, G'\}$, if $\boldsymbol{a} = \mathcal{A}(G)$ and $\boldsymbol{a}' = \mathcal{A}(G')$, then for all $i, j \notin \{\partial\}$, $a_i > a_j \Rightarrow a'_i \geq a'_j$, or $a_i < a_j \Rightarrow a'_i \leq a'_j$. (equivalently, $\mathcal{I}^{(0)}_{\boldsymbol{aa}'}(i, j) = 0$). The algorithm is *strictly rank local* if for all $i, j \notin \{\partial\}$, $a_i > a_j \Leftrightarrow a'_i > a'_j$ (equivalently, $\mathcal{I}^{(1)}_{\boldsymbol{aa}'}(i, j) = 0$).

We note that locality and rank locality do not depend upon the normalization used by the algorithm. From the definitions, one can observe that if an algorithm is local, then it is also strictly rank local. If it is strictly rank local, then it is obviously weakly rank local.

We have the following.

Theorem 4.24. *If an algorithm $\mathcal{A}$ is weakly rank local on the class $\overline{\mathcal{G}}_n$, then it is weakly rank stable on the class $\overline{\mathcal{G}}_n$. If $\mathcal{A}$ is strictly rank local on $\overline{\mathcal{G}}_n$, then it is strictly rank stable on $\overline{\mathcal{G}}_n$.*

Proof. Let $\partial = \{G, G'\}$ be a change within the class $\overline{\mathcal{G}}_n$ of ∂ size of at most k. Let $\mathcal{A}$ be an algorithm defined on $\mathcal{G}_n$, let $\boldsymbol{a}$ be the weight vector of $\mathcal{A}$ on graph G, and $\boldsymbol{a}'$ be the weight vector of $\mathcal{A}$ on graph G'. Let $T = \{\partial\}$ be the impact set of change ∂, and let m be the size of the set T, where $m \leq k$. If the algorithm $\mathcal{A}$ is weakly rank local, then $\mathcal{I}^{(0)}_{\boldsymbol{aa}'}(i, j) = 0$ for all $i, j \notin T$. Therefore,

$$\begin{aligned} d_r^{(0)}(\boldsymbol{a}, \boldsymbol{a}') &= \frac{1}{n(n-1)/2} \sum_{i=1}^{n} \sum_{j \in T} \mathcal{I}^{(0)}_{\boldsymbol{aa}'}(i, j) \\ &\leq \frac{nm}{n(n-1)/2} \leq 2k/(n-1) \\ &= o(1). \end{aligned}$$

Similarly, if the algorithm $\mathcal{A}$ is strictly rank local, $\mathcal{I}^{(1)}_{\boldsymbol{aa}'}(i, j) = 0$ for all $i, j \notin T$, and

$$\begin{aligned} d_r^{(1)}(\boldsymbol{a}, \boldsymbol{a}') &= \frac{1}{n(n-1)/2} \sum_{i=1}^{n} \sum_{j \in T} \mathcal{I}^{(1)}_{\boldsymbol{aa}'}(i, j) \\ &\leq 2k/(n-1) = o(1) \end{aligned}$$

which concludes the proof of the theorem. □

Therefore, locality implies rank stability. It is not necessarily the case that it also implies stability. For example, consider the algorithm $\mathcal{A}$, which for a graph G on n nodes assigns weight $n^{|B(i)|}$ to node i. This algorithm is local, but it is not stable. Consider, for example, a graph where every node has in-degree 1. Adding one extra link to one of the nodes causes a large amount of weight to be transfered to the node pointed to by the new link, thus causing instability.

THEOREM 4.25. *The* InDEGREE *algorithm is local and consequently, strictly rank local and rank local.*

PROOF. Given a graph G, let $\boldsymbol{u}$ be the weight vector that assigns to node i weight equal to $|B(i)|$, the in-degree of i. Let $\boldsymbol{w}$ be the weight vector of the InDEGREE algorithm; then $w_i = u_i/\|\boldsymbol{u}\| = |B(i)|/\|\boldsymbol{u}\|$, where $\|\cdot\|$ is any norm.

Let $\partial = \{G, G'\}$ be a change within $\mathcal{G}_n$, and let $\boldsymbol{u}'$ and $\boldsymbol{w}'$ denote the corresponding weight vectors on graph G'. For every $i \not\in \{\partial\}$, the number of links to i remains unaffected by the change ∂; therefore $u'_i = u_i$. For the InDEGREE algorithm, $w'_i = u'_i/\|\boldsymbol{u}'\| = u_i/\|\boldsymbol{u}'\|$. For $\lambda = \frac{\|\boldsymbol{u}\|}{\|\boldsymbol{u}'\|}$, it holds that $w'_i = \lambda w_i)$ for all $i \not\in \{\partial\}$. Thus, InDEGREE is local, and consequently strictly rank local and rank local. □

The following is a direct corollary of the locality of the InDEGREE algorithm.

COROLLARY 4.26. *The* InDEGREE *algorithm is strictly rank stable.*

The following corollary follows from the fact that the SALSA and InDEGREE algorithms are equivalent on the class $\mathcal{G}_n^{AC}$ of authority connected graphs.

COROLLARY 4.27. *The* SALSA *algorithm is stable and strictly rank stable on* $\mathcal{G}_n^{AC}$.

We originally thought that the BAYESIAN and SBAYESIAN algorithms were also local. However, it turns out that they are neither local, nor rank local. Indeed, it is true that *conditional* on the values of h_i, e_i, and a_j, the conditional distribution of a_k for $k \neq j$ is unchanged upon removing a link from i to j. However, the *unconditional* marginal distribution of a_k, and hence also its posterior mean $\widehat{a}_k$ (or even ratios $\widehat{a}_k/\widehat{a}_q$ for $q \neq j$), may still be changed upon removing a link from i to j. Indeed, we have computed experimentally that, even for a simple example with just four nodes, the ratio of the authority weights may change upon removing a single link. In the next section, we show that the BAYESIAN and the SBAYESIAN algorithms cannot be rank local, since (as shown experimentally) they are not rank-matching with the InDEGREE algorithm.

4.6 Monotonicity

We now define the property of *monotonicity*. Monotonicity requires that, if all hubs that point to node j also point to node k, then node k should receive authority weight at least as high as that of node j. Formally, we define monotonicity as follows.

Definition 4.28. An LAR algorithm $\mathcal{A}$ is *monotone* on the class of graphs $\overline{\mathcal{G}}_n$ if it has the following property. For every graph $G \in \overline{\mathcal{G}}_n$, and for every pair of nodes j and k in the graph G, if $B(j) \subseteq B(k)$, then $\mathcal{A}(G)[j] \leq \mathcal{A}(G)[k]$.

Monotonicity appears to be a "reasonable" property but one can define "reasonable" algorithms that are not monotone. For example, consider the

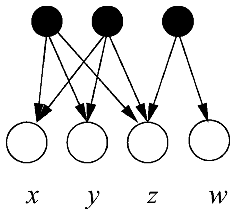


Fig. 10. The nonmonotonicity of AuthorityAvg.

AuthorityAvg algorithm, the authority analogue of the HubAvg algorithm, where the authority weight of a node is defined to be the average of the hub weights of the nodes that point to this node. Consider now the graph in Figure 10. In this graph, we have that $B(x) \subset B(z)$ and $B(y) \subset B(z)$, but AuthorityAvg assigns higher weight to nodes x and y than to z. An idea similar to that of the AuthorityAvg algorithm is suggested by Bharat and Henzinger [1998]. When computing the authority weight of node i, they average the weights of hubs that belong to the same domain. Another example of a nonmonotone algorithm is the HubTheshold algorithm defined by Borodin et al. [2001].

Theorem 4.29. *The algorithms* InDegree*,* Hits*,* PageRank*,* Salsa*,* HubAvg*,* AT*(k), and* BFS *are all monotone.*

Proof. Let j and k be two nodes in a graph G, such that $B(j) \subseteq B(k)$. For the InDegree algorithm, monotonicity is obvious since the authority weights are proportional to the in-degrees of the nodes, and the in-degree of j is less than, or equal to, the in-degree of k. The same holds for the Salsa algorithm within each authority connected component which is sufficient to prove the monotonicity of the algorithm.

For the PageRank algorithm, if a_j and a_k are the weights of nodes j and k, then we have that

$$a_j = \sum_{i=1}^{n} M_{PR}[i, j]a_i \qquad \text{and} \qquad a_k = \sum_{i=1}^{n} M_{PR}[i, k]a_i,$$

where M_{PR} is the matrix for the PageRank algorithm [Brin and Page 1998]. For all i, $M_{PR}[i, j] \leq M_{PR}[i, k]$. Therefore, $a_j \leq a_k$.

For the Hits, HubAvg, and AT(k) algorithms, it suffices to observe that, at every iteration t,

$$\overline{a}_j^t = \sum_{i \in B(j)} h_i \leq \sum_{i \in B(k)} h_i = \overline{a}_k^t,$$

where $\overline{a}_j^t$ and $\overline{a}_k^t$ are the weights of nodes j and k at iteration t before applying the normalization step. Normalization may result in both weights converging to zero, as $t \to \infty$, but it cannot be the case that in the limit $a_j > a_k$.

For the BFS algorithm, it suffices to observe that, since $B(j) \subseteq B(k)$, every node that is reachable from node j is also reachable from node k. Thus $a_j \leq a_k$. □

We also define the following stronger notion of monotonicity.

Definition 4.30. An LAR algorithm $\mathcal{A}$ is *strictly monotone* on the class of graphs $\overline{\mathcal{G}}_n$ if it has the following property. For every graph $G \in \overline{\mathcal{G}}_n$, and for every pair of nodes j and k in the graph G, $B(j) \subset B(k)$, if and only if $\mathcal{A}(G)[j] < \mathcal{A}(G)[k]$.

We can now prove the following theorem.

THEOREM 4.31. *The algorithms* INDEGREE, PAGERANK, SALSA, *and* BFS *are strictly monotone on the class* $\mathcal{G}_n$*, while the algorithms* HITS, HUBAVG, *and* MAX *are not strictly monotone on the class* $\mathcal{G}_n$*. The algorithms* INDEGREE, HITS, PAGERANK, SALSA, HUBAVG, AT*(k), and* BFS *are all strictly monotone on the class of authority connected graphs* $\mathcal{G}_n^{AC}$.

PROOF. The strict monotonicity on the class of the authority connected graphs $\mathcal{G}_n^{AC}$ follows directly from the proof of Theorem 4.29 for the monotonicity of the algorithms. Similarly, for the strict monotonicity of INDEGREE, PAGERANK, SALSA, and BFS on the class $\mathcal{G}_n$.

For the HITS and HUBAVG algorithms, consider a graph $G \in \mathcal{G}_n$ consisting of two disconnected components. If the components are chosen appropriately, the HITS and HUBAVG algorithms will allocate all the weight to one of the components, and zero weight to the nodes of the other component. Furthermore, if one of the two components does not contain a seed node, then the MAX algorithm will allocate zero weight to the nodes of that component. If chosen appropriately, the nodes in the component that receive zero weight violate the strict monotonicity property. □

A different notion of monotonicity is considered by Chien et al. [2002]. In their paper, they study how the weight of a node changes as new links are added to the node. In this setting, an algorithm is monotone if the weight of a node increases as its in-degree increases.

4.7 Label-Independence

We now introduce the property of *label-independence*. This property is needed in the axiomatic characterization of the INDEGREE algorithm.

Definition 4.32. Let $G \in \mathcal{G}_n$ be a graph of size n, and let $\{1, 2, \ldots, n\}$ denote a labeling of the nodes of G. Let $\mathcal{A}$ be a LAR algorithm, and let $\boldsymbol{a} = \mathcal{A}(G)$ denote the weight vector of $\mathcal{A}$ on a graph $G \in \mathcal{G}_n$. Let π denote a permutation of the labels of the nodes of G, and let $\boldsymbol{a}'$ denote the weight vector of $\mathcal{A}$ on the graph with the permuted labels. The algorithm $\mathcal{A}$ is *label-independent* if $a'(\pi(i)) = a(i)$.

All the algorithms we considered in this article (INDEGREE, PAGERANK, HITS, SALSA, HUBAVG, AT(k), BAYESIAN, SBAYESIAN BFS) are clearly label-independent. Label-independence is a reasonable property, but one can define reasonable algorithms that are not label-independent. For example, an algorithm may choose to give more weight to a link from a node with a specific label. The algorithm defined by Bharat and Henzinger [1998], when computing the authority weight of a node i, averages the hub weights of the nodes that belong to the same domain.

This algorithm is not label-independent since it takes into account the “label” of the node when computing the authority weights.

4.8 An Axiomatic Characterization of the InDegree Algorithm

We will now prove that there is a set of properties that characterize the InDegree algorithm.

Theorem 4.33. *An algorithm $\mathcal{A}$ that is strictly rank local, monotone, and label-independent is rank consistent with the* InDegree *algorithm on the class $\mathcal{G}_n$ for any $n \geq 3$. If $\mathcal{A}$ is strictly rank local, strictly monotone, and label-independent, then it is rank equivalent to the* InDegree *algorithm on the class $\mathcal{G}_n$ for any $n \geq 3$.*

Proof. Let G be a graph of size $n \geq 3$, and let $\boldsymbol{a} = \mathcal{A}(G)$ be the LAR vector of algorithm $\mathcal{A}$ on graph G, and $\boldsymbol{w}$ be the LAR vector of InDegree. We will modify G to form graphs G_1, and G_2, and we use $\boldsymbol{a}_1$, and $\boldsymbol{a}_2$ to denote (respectively) the weight vector of algorithm $\mathcal{A}$ on these graphs.

Let i and j be two nodes in G. First, we consider the case that at least one of i, j has zero in-degree. If $w_i = w_j = 0$, that is $B(i) = B(j) = \emptyset$, then from the monotonicity of $\mathcal{A}$, we have that $a_i = a_j$. Therefore, $\mathcal{I}^{(0)}_{\boldsymbol{aw}}(i, j) = 0$, and $\mathcal{I}^{(1)}_{\boldsymbol{aw}}(i, j) = 0$. If $w_i > w_j = 0$, that is, $B(i) = \emptyset$, and $B(j) \neq \emptyset$, then $B(i) \subset B(j)$. If $\mathcal{A}$ is monotone, $a_i \leq a_j$, thus $\mathcal{I}^{(0)}_{\boldsymbol{aw}}(i, j) = 0$. If $\mathcal{A}$ is strictly monotone, $a_i < a_j$, thus $\mathcal{I}^{(1)}_{\boldsymbol{aw}}(i, j) = 0$.

Consider now the case that both i, j have nonzero in-degree. Without loss of generality, assume that $w_i \geq w_j > 0$, or equivalently that node i has at least as many in-links as node j. The set $B(i) \cup B(j)$ of nodes that point to i or j is decomposed as follows.

—The set $C = B(i) \cap B(j)$ contains the nodes that point to both i and j.

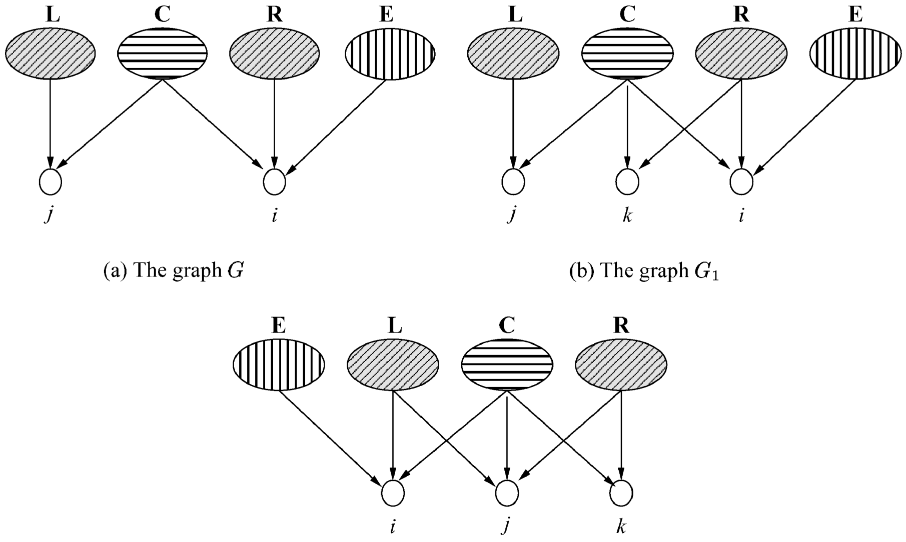
—The set $L = B(j) \setminus C$ contains the nodes that point to node j, but not to node i.

—The set $V = B(i) \setminus C$ contains the nodes that point to node i, but not to node j. The set V is further decomposed into the sets R and E. The set R is an arbitrary subset of the set V with cardinality equal to that of L. Since the in-degree of node i is at least as large as that of node j, the set R is well defined. We also have that, $E = V \setminus R$.

Note that some of these sets may be empty, but not all of them can be empty. Specifically, $V \cup C \neq \emptyset$, and $L \cup C \neq \emptyset$. The set E is empty if and only if nodes i and j have equal in-degrees. The decomposition of the set $B(i) \cup B(j)$ is shown in Figure 11(a). The elliptic shapes denote sets of nodes, while the arrows represent the links from a set of nodes to a single node.

Let $k \neq i, j$ be an arbitrary node in the graph. We now perform the following change to graph G. We remove all links that do not point to i or j, and add links from the nodes in R and C to node k. Let G_1 denote the resulting graph. The graph G_1 is shown in Figure 11(b). Since $\mathcal{A}$ is strictly rank local, and the links to nodes i and j were not affected by the change, we have that

$$a(i) < a(j) \Leftrightarrow a_1(i) < a_1(j). \tag{6}$$



(a) The graph G

(b) The graph G_1

(c) The graph G_2

Fig. 11. Axiomatic characterization of InDegree.

We will now prove that $a_1(k) = a_1(j)$. Assume that $a_1(k) < a_1(j)$. Let G_2 denote the graph that we obtain by removing all the links from set R to node i, and adding links from set L to node i. The graph G_2 is shown in Figure 11(c). We observe that the graphs G_1 and G_2 are the same up to a label permutation that swaps the labels of nodes j and k, and the labels of the nodes in L with the labels of the nodes in R. Thus, $a_2(j) = a_1(k)$, and $a_2(k) = a_1(j)$. Therefore, from our assumption that $a_1(k) < a_1(j)$, we have $a_2(j) < a_2(k)$. However, graph G_2 was obtained from graph G_1 by performing a local change to node i. Given the strict locality assumption, the relative order of the weights of nodes j and k should remain unaffected, that is, $a_2(j) > a_2(k)$, thus reaching a contradiction. We reach the same contradiction if we assume that $a_1(k) > a_1(j)$. Therefore, it must be that $a_1(k) = a_1(j)$.

In the graph G_1, we have that $B(k) \subseteq B(i)$. We distinguish two cases. If $B(k) = B(i)$, then the set E is empty. Therefore, $w_i = w_j$ since i and j have the same number of in-links. Furthermore, from the monotonicity property (weak or strict) of the algorithm $\mathcal{A}$, we have that $a_1(i) = a_1(k) = a_1(j)$. From Equation (6), it follows that $a(i) = a(j)$.

If $B(k) \subset B(i)$, then the set E is not empty, and $w_i > w_j$ since i has more links than j. If $\mathcal{A}$ is monotone, then $a_1(i) \geq a_1(k) = a_1(j)$. From Equation (6), we have that $a(i) \geq a(j)$. Therefore, for all i, j $w_i > w_j \Rightarrow a_i \geq a_j$. Thus, $d_r^{(0)}(\boldsymbol{w}, \boldsymbol{a}) = 0$, and $\mathcal{A}$ and InDegree are rank consistent. If $\mathcal{A}$ is strictly monotone, then $a_1(i) > a_1(k) = a_1(j)$. From Equation (6), we have that $a(i) > a(j)$. Therefore, for all i, j $w_i > w_j \Leftrightarrow a_i > a_j$. Thus, $d_r^{(1)}(\boldsymbol{w}, \boldsymbol{a}) = 0$, and $\mathcal{A}$ and InDegree are rank equivalent. □

The conditions of Theorem 4.33 characterize InDegree. All three conditions, label-independence, (strict) monotonicity, and strict rank locality are necessary for the proof of the theorem. Assume that we discard the label-independence condition. Now, define algorithm $\mathcal{A}$ that assigns to each link a weight that depends on the label of the node from which the link originates. The algorithm sets the authority weight of each node to be the sum of the link weights that point to this node. This algorithm is clearly monotone and local, however, if the link weights are chosen appropriately, it will not be rank consistent with InDegree. Assume now that we discard the monotonicity condition. Define an algorithm $\mathcal{A}$, that assigns weight 1 to each node with odd in-degree, and weight 0 to each node with even in-degree. This algorithm is local and label-independent, but it is clearly not rank consistent with InDegree. Monotonicity and label-independence are clearly not sufficient for proving the theorem; we have provided examples of algorithms that are monotone and label-independent, but not rank consistent with the InDegree (e.g., the Hits algorithm). Strict monotonicity is necessary for proving rank equivalence. The algorithm that assigns equal weight to all nodes is monotone, label-independent, and strictly rank local. It is rank consistent with InDegree, but not rank equivalent.

5. EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation of the algorithms that we consider in this article. We study the rankings they produce, and how they relate to each other. The goal of this experimental study is to assess the quality of the algorithms and, more importantly, to understand how theoretically predicted properties manifest themselves in a practical setting.

5.1 The Experimental Set-Up

5.1.1 *The Queries.* We experiment with our algorithms on the following 34 different queries.

```
abortion, affirmative action, alcohol, amusement parks,
architecture, armstrong, automobile industries, basketball,
blues, cheese, classical guitar, complexity, computational
complexity, computational geometry, death penalty,
genetic, geometry, globalization, gun control, iraq war,
jaguar, jordan, moon landing, movies, national parks, net
censorship, randomized algorithms, recipes, roswell, search
engines, shakespeare, table tennis, weather, vintage cars
```

Many of these queries have already appeared in previous works [Dwork et al. 2001; Lempel and Moran 2000; Kleinberg 1998]. For the remaining queries, we selected queries that correspond to topics for which there are opposing communities, such as, “death penalty”, “gun control”, “iraq war”, “globalization”, “moon landing”, or queries that are of interest to different communities, depending on the interpretation of the word (for example, “jordan”, “complexity”, “armstrong”). Our objective is to observe how the different algorithms represent these different (usually unrelated, and sometimes opposing) communities in the top positions of the ranking. We are also interested in observing the behavior of the algorithms when we move from a broad topic (“geometry”, “complexity”) to a more specific subset of this topic (“computational complexity”, “computational geometry”). We also selected some queries for which we expect the most relevant results not to contain the query words. For example, most search engines do not contain the words “search engine”. The same for the query “automobile industries” which is a variation of the query “automobile manufacturers” that appears in the work of Kleinberg [1998]. The remaining queries were selected as interesting queries on broad topics.

The Base Sets for these queries are constructed in the fashion described by Kleinberg [1998]. We start with a Root Set of pages related to the query. This Root Set is obtained by querying the Google[3] search engine. The Root Set consists of the first 200 pages returned by the search engine. This set is then expanded to the Base Set by including nodes that point to, or are pointed to, by the pages in the Root Set. Following the guidelines of Kleinberg [1998], for every page in the Root Set, we include only the first 50 pages that point to this page in the order that they are returned by the Google search engine. We then extract the links between the pages of the Base Set, and we construct the hyperlink graph.

The next step is to eliminate the *navigational* links. These are links that solely serve the purpose of navigating within a Web site, and they do not convey an endorsement for the contents of the target page. Finding navigational links is a nontrivial problem that has received some attention in the literature [Davison 2000; Bharat and Mihaila 2001]. We adopt the following heuristics for identifying navigational links. First, we compare the IP addresses of the two links. If the first three bytes are the same, then we label the link as navigational. If not, we look at the actual URL. This is of the form “$\texttt{http://string}_1\texttt{/string}_2\texttt{/}\cdots$”. The domain identifier is $\texttt{string}_1$. This is of the form “ $\texttt{x}_1.\texttt{x}_2.\cdots.\texttt{x}_k$”. If $k \geq 3$, then we use $\texttt{x}_2.\cdots.\texttt{x}_{k-1}$ as the domain identifier. If $k = 2$, we use $\texttt{x}_1$. If the domain identifiers are the same for the source and target pages of the link, then the link is labeled as navigational, and it is discarded. After the navigational links are removed, we remove any isolated nodes, and we produce the Base Set P and the graph $G = (P, E)$. Unfortunately, our heuristics do not eliminate all possible navigational links which in some cases results in introducing clusters of pages from the same domain.

Table I presents statistics for our graphs. The “med out” is the median out-degree, the “avg-out” is the average out-degree, where median and average are taken over all hub nodes. The “ACC size” is the size of the largest authority connected component, that is, the size of the largest connected component in the authority graph G_a. Recall that the graph G_a is a graph defined on the authority nodes where there exists an edge between two authorities if they have a hub in common.

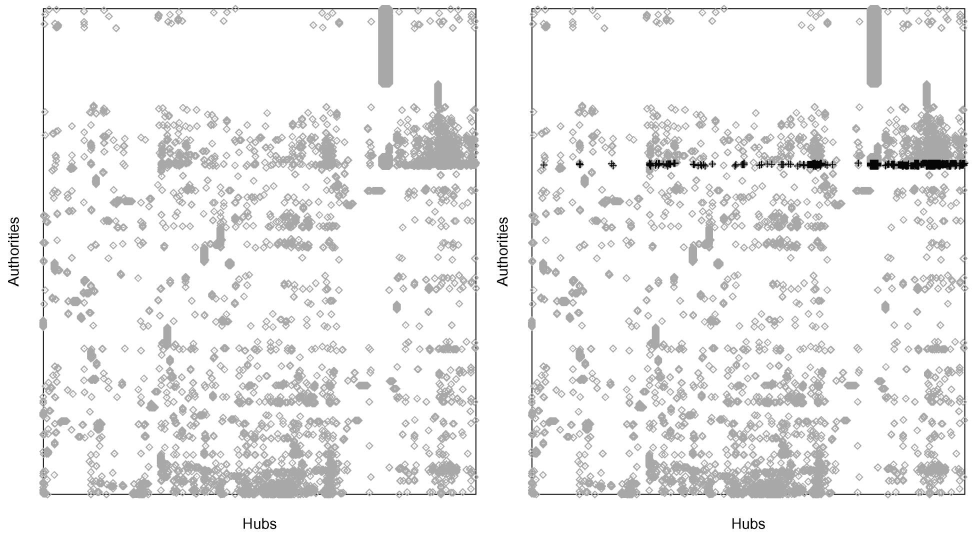
Matrix Plots: For the purpose of exhibition, we will often present the graph for a query as a *matrix plot*. A matrix plot, is a bitmap picture of the (transposed)

[3] http://www.google.com.

Table I. Query Statistics

Query	Nodes	Hubs	Authorities	Links	Med out	Avg out	ACC size	Users
abortion	3340	2299	1666	22287	3	9.69	1583	22
affirmative action	2523	1954	4657	866	1	2.38	752	7
alcohol	4594	3918	1183	16671	2	4.25	1124	8
amusement parks	3410	1893	1925	10580	2	5.58	1756	8
architecture	7399	5302	3035	36121	3	6.81	3003	8
armstrong	3225	2684	889	8159	2	9.17	806	8
automobile industries	1196	785	561	3057	2	3.89	443	7
basketball	6049	5033	1989	24409	3	4.84	1941	12
blues	5354	4241	1891	24389	2	5.75	1838	8
cheese	3266	2700	1164	11660	2	4.31	1113	5
classical guitar	3150	2318	1350	12044	3	5.19	1309	8
complexity	3564	2306	1951	13481	2	5.84	1860	4
computational complexity	1075	674	591	2181	2	3.23	497	4
computational geometry	2292	1500	1294	8189	3	5.45	1246	3
death penalty	4298	2659	2401	21956	3	8.25	2330	9
genetic	5298	4293	1732	19261	2	4.48	1696	7
geometry	4326	3164	1815	13363	2	4.22	1742	7
globalization	4334	2809	2135	17424	2	8.16	1965	5
gun control	2955	2011	1455	11738	3	5.83	1334	7
iraq war	3782	2604	1860	15373	3	5.90	1738	8
jaguar	2820	2268	936	8392	2	3.70	846	5
jordan	4009	3355	1061	10937	2	3.25	991	4
moon landing	2188	1316	1179	5597	2	4.25	623	8
movies	7967	6624	2573	28814	2	4.34	2409	10
national parks	4757	3968	1260	14156	2	3.56	1112	6
net censorship	2598	1618	1474	7888	2	4.87	1375	4
randomized algorithms	742	502	341	1205	1	2.40	259	5
recipes	5243	4375	1508	18152	2	4.14	1412	10
roswell	2790	1973	1303	8487	2	4.30	1186	4
search engines	11659	7577	6209	292236	5	38.56	6157	5
shakespeare	4383	3660	1247	13575	2	3.70	1199	6
table tennis	1948	1489	803	5465	2	3.67	745	6
weather	8011	6464	2852	34672	3	5.36	2775	9
vintage cars	3460	2044	1920	12796	3	6.26	1580	5

adjacency matrix of the graph where every nonzero entry of the matrix is represented by a dot. Figure 12(a) shows the matrix plot for the query “abortion”. In this plot, the rows of the matrix correspond to the authority nodes in the graph, and the columns to the hub nodes. Every point on the matrix plot corresponds to an edge in the graph. That is, there exists a dot in the matrix plot in the position (i, j) if the hub that corresponds to the column j points to the authority that corresponds to the row i. The points in the ith row can be thought of as a representation of the corresponding authority node. The ith row of the plot is the plot of the vector of hub nodes that point to the correspnding authority node.



(a) Matrix Plot for the “abortion” graph

(b) Matrix plot with the top 10 results of Hits

Fig. 12. Matrix plots for the query “abortion”.

In order to reveal some of the structure of the graph, the rows and columns are permuted so that similar nodes are brought together. For this, we used LIMBO, an agglomerative hierarchical clustering algorithm [Andritsos et al. 2003], which is based on the Information Bottleneck method [Tishby et al. 1999; Slonim and Tishby 1999]. The distance between two vectors is measured by normalizing the vectors so that the sum of their entries is 1, and then taking the Jensen-Shannon divergence [Lin 1991] of the two normzalized vectors. Any other hierarchical algorithm for clustering binary vectors would also be applicable. Executing the algorithm on the rows of the matrix produces a tree where each node in the tree corresponds to the merge of two clusters. The leaves of this tree are the rows of the matrix (the authority nodes). If we perform a depth-first traversal of this tree and we output the leaves in the order in which we visit them, then we expect this permutation to bring similar rows together. We perform the same operation for the columns of the graph. We do not claim that these permutations of rows and columns are optimal in any sense. The purpose of the permutations is to enhance the visualization of the graph by grouping together some of the similar rows and columns.

The matrix plot representation of the graph is helpful in identifying the parts of the graph on which the various algorithms focus in the top 10 results by highlighting the corresponding rows. For example, Figure 12(b) shows again the plot of the matrix for the “abortion” dataset. The rows in darker color correspond to the top 10 authority nodes of the Hits algorithm. These matrix plots allow us to inspect how the top 10 results of the different LAR algorithms are interconnected with each other and with the rest of the graph, and they yield significant insight in the behavior of the algorithms.

5.1.2 *Algorithms.* We implemented the new algorithms we introduced in Section 3, namely HubAvg, AT(k), Max, BFS, Bayesian, and SBayesian. For the AT(k) family of algorithms, given a graph G, we compute the distribution of the out-degrees in the graph and we experiment with k being the median, and the average out-degree where median and average are taken over all the hub nodes. We denote these algorithms as AT-med and AT-avg, respectively.

We also implemented the Hits, PageRank, InDegree, and Salsa algorithms. For the PageRank algorithm, the jump probability ϵ usually takes values in the interval $[0.1, 0.2]$ [Brin and Page 1998; Ng et al. 2001a]. We observed that the performance of the PageRank algorithm usually improves as we increase the value of ϵ. We set the jump probability ϵ to be 0.2, a value that is sufficiently low and produces satisfactory results.

For all algorithms, we iterate until the L_1 difference of the authority weight vectors in two successive iterations becomes less than $\delta = 10^{-7}$, or until 1000 iterations have been completed. Although there are more sophisticated methods for testing for convergence, we chose this one for the sake of simplicity. In most cases, the algorithms converge in no more than a hundred iterations.

5.1.3 *Measures.* The measure that we will use for the evaluation of the quality rankings is *precision over top 10*. This is the fraction of documents in the top 10 positions of the ranking that are relevant to the query. Given the impatient nature of the Web users, we believe that this is an appropriate measure for the evaluation of Web searching and ranking algorithms. Indeed, a search engine is often judged by the first page of results it returns. We will also refer to this fraction as the *relevance ratio* of the algorithm. Similar quality measures are used in the TREC conferences for evaluating Web search algorithms.[4]

We also use a more refined notion of relevance. Given a query, we classify a document as nonrelevant, relevant, or highly relevant to the topic of the query. High relevance is meant to capture the notion of authoritativeness. A highly relevant document is one that should definitely be in the few first page of the results of a search engine. For example, for the query "movies", the official site for the movie "A Beautiful Mind"[5] is relevant to the topic of movies, but it cannot be thought of as highly relevant. However, the Internet Movie Data Base (IMDB) site[6] that contains movie information and reviews is a page that is highly relevant to the topic. This is a result that a Web user would most likely want to retrieve when posing the query. The notion of high relevance is also employed in the TREC conference for topic distillation queries where the objective is to find the most authoritative pages for a specific topic. For each algorithm, we want to estimate the *high relevance ratio*, the fraction of the top 10 results that are highly relevant. Note that every highly relevant page is, of course, relevant so the high relevance ratio is always less or equal to the relevance ratio.

[4] For TREC data, relevance and high relevance is usually predefined by a set of experts.

[5] `http://abeautifulmind.com/`.

[6] `http://www.imdb.com`.

We are also interested in studying how the algorithms relate to each other. For the comparison of two rankings, we will use the geometric distance measure, d_1, defined in Section 4.2.1, and the strict rank distance, $d_r^{(1)}$, defined in Section 4.2.2. For brevity, we will refer to the strict rank distance as rank distance, and we will denote it by d_r. When computing the d_1 distance, the vectors are normalized so that the entries sum to 1. Thus, the maximum d_1 distance is 2. We will also consider the following two similarity measures for comparing the top-k results of two algorithms.

—*Intersection over top k, $I(k)$*: The number of documents that the two rankings have in common in the top k results.

—*Weighted Intersection over top k, $WI(k)$*: This is the average intersection over the top k results, where the average is taken over the intersection over the top 1, top 2, up to top k. The weighted intersection is given by the following formula.

$$WI(k) = \frac{1}{k} \sum_{i=1}^{k} I(i).$$

In our study, we measure the similarity over the top 10 results.

5.1.4 *User Study.* In order to assess the relevance of the documents, we performed a user study. The study was performed online.[7] The introductory page contained the queries with links to the results, together with some instructions. By clicking on a query, the union of the top 10 results of all algorithms was presented to the user. The results were permuted so that they appeared in a random order and no information was revealed about the algorithm(s) that introduced each result in the collection. The users were then asked to rate each document as “Highly Relevant”, “Relevant”, or “NonRelevant”. They were instructed to mark a page as “Highly Relevant” if they would definitely like to see if within the top positions of a search engine. An option “Don’t Know” (chosen as default) was also given in the case that the user could not assess the relevance of the result. When pressing the submit button, their feedback was stored into a file. No information was recorded about the users, respecting their anonymity. No queries were assigned to any users, they were free to do whichever ones, and however many they wanted. On average seven users rated each query. The maximum was twenty two, for the query “abortion”, and the minimum three, for the query “computational geometry”. The number of users per query are shown in Table I. The study was conducted mostly among friends and fellow grad students. Although they are not all experts on all of the topics, we believe that their feedback gives a useful indication about the actual relevance of the documents.[8]

We utilize the user’s feedback in the following way. Given the users’ feedback for a specific document, we rate the document as “Relevant” if the “Relevant” and “Highly Relevant” votes are more than the “Non-Relevant” votes (ties are resolved in favor of “Non-Relevant”). Among the documents that are deemed as

[7] The URL for the study is `http://www.cs.toronto.edu/~tsap/cgi-bin/entry.cgi`.

[8] The results of the study may be slightly biased towards the opinion of people of Greek origin.

"Relevant", we rate as "Highly Relevant" the ones for which the "Highly Relevant" votes are more than the "Relevant" ones. We can now compute the relevance ratios for the algorithms by using the relevance ratings of the documents.

5.2 Evaluation of the LAR Algorithms

In this section, we study the aggregate behavior of the algorithms. The results for all queries are posted on the Web[9] in a format that is easy to understand and navigate. We strongly encourage the reader to browse through the results while reading this part of the article. Appendix A contains tables with the top 10 results for three sample queries ("abortion", "amusement parks", "classical guitar") that are referenced often in the next sections. In these Tables, the results that are labeled highly relevant appear in boldface, the relevant ones appear in italics, and the nonrelevant appear in regular font. Due to space constraints, we omit the results of the SALSA and SBAYESIAN algorithms. In almost all queries, they are identical to those of INDEGREE. Tables II, III report the quality ratios of each algorithm for each query. In each row, the highlighted value is the best ratio for this query. For each algorithm, we also compute the average, the standard deviation, the minimum, and the maximum values for all quality ratios. Recall, that for the relevance ratio, we consider documents that are marked either "Relevant" or "Highly Relevant", so the relevance ratio is always greater or equal to the high relevance ratio. For the purpose of comparing algorithms, we also report the average values of all our similarity and distance measures in Appendix A.

The qualitative evaluation reveals that all algorithms fall, to some extent, victim to *topic drift*. That is, they promote pages that are not related to the topic of the query. In terms of average high relevance, more than half (and as many as 8 out of 10 for the case of HITS) of the results in the top 10 are not highly relevant. This is significant for the quality of the algorithms since the highly relevant documents represent the ones that the users would actually want to see in the top positions of the ranking, as opposed to the ones that they found just relevant to the topic. The performance improves when we consider relevance instead of high relevance. Still, the average relevance ratio is never more than 78%, that is, even for the best algorithm, on average 2 out of the top 10 documents are irrelevant to the query. Furthermore, there exist queries such as "armstrong" and "jaguar" for which no algorithm was able to produce satisfactory results.

The algorithm that emerges as the "best" is the BFS algorithm. It exhibits the best high relevance and relevance ratio on average, and it is the algorithm that most often achieves the maximum relevance ratio (for the high relevance ratio the record is held by the MAX algorithm). Furthermore, as the low standard deviation indicates, and the study of the results reveals, it exhibits a robust and consistent behavior across queries.

At the other end of the spectrum, the HITS and the PAGERANK algorithms emerge as the "worst" algorithms, exhibiting the lowest ratios, with HITS being slightly worse. However, although they have similar (poor) performance on

[9] http://www.cs.toronto.edu/~tsap/experiments/journal/.

Table II. High Relevance Ratio

Query	Hits	PageRank	InDegree	Salsa	HubAvg	Max	AT-med	AT-avg	BFS	Bayesian	SBayesian
abortion	30%	10%	**40%**	**40%**	30%	**40%**	30%	30%	30%	30%	**40%**
affirmative action	30%	0%	40%	40%	0%	0%	0%	0%	**60%**	30%	40%
alcohol	60%	30%	60%	60%	60%	50%	50%	50%	**70%**	50%	60%
amusement parks	50%	10%	30%	40%	0%	70%	10%	0%	40%	**90%**	30%
architecture	0%	30%	**70%**	**70%**	0%	60%	60%	0%	50%	0%	**70%**
armstrong	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
automobile industries	0%	10%	10%	20%	0%	0%	0%	0%	**40%**	0%	10%
basketball	0%	**60%**	20%	20%	0%	10%	10%	10%	**60%**	10%	20%
blues	**60%**	40%	40%	40%	50%	50%	50%	50%	20%	40%	40%
cheese	0%	0%	0%	0%	0%	0%	0%	0%	**10%**	0%	0%
classical guitar	**40%**	30%	30%	30%	0%	**40%**	0%	0%	**40%**	30%	30%
complexity	0%	30%	20%	20%	0%	**70%**	50%	0%	50%	0%	20%
computational complexity	30%	30%	30%	30%	**40%**	30%	30%	30%	20%	30%	30%
computational geometry	40%	20%	40%	40%	40%	40%	**50%**	40%	40%	30%	40%
death penalty	70%	30%	70%	70%	50%	**80%**	**80%**	**80%**	**80%**	**80%**	60%
genetic	**80%**	20%	70%	70%	60%	70%	70%	70%	60%	**80%**	70%
geometry	60%	10%	50%	50%	40%	**70%**	**70%**	60%	60%	60%	50%
globalization	0%	**30%**	20%	20%	0%	0%	0%	0%	**30%**	0%	20%
gun control	0%	50%	**70%**	**70%**	60%	60%	60%	60%	60%	50%	**70%**
iraq war	0%	10%	10%	10%	0%	10%	0%	0%	**40%**	10%	10%
jaguar	0%	**20%**	0%	0%	0%	0%	0%	0%	10%	0%	0%
jordan	0%	10%	20%	20%	20%	**40%**	**40%**	**40%**	30%	20%	20%
moon landing	0%	20%	10%	10%	0%	0%	0%	0%	**80%**	0%	10%
movies	10%	10%	30%	30%	40%	**70%**	**70%**	**70%**	40%	50%	30%
national parks	0%	50%	10%	10%	**60%**	**60%**	**60%**	0%	50%	0%	10%
net censorship	0%	20%	**80%**	**80%**	60%	**80%**	**80%**	**80%**	**80%**	70%	**80%**
randomized algorithms	0%	**40%**	10%	10%	0%	0%	0%	0%	10%	10%	10%
recipes	0%	10%	60%	60%	10%	60%	60%	**70%**	50%	50%	60%
roswell	0%	0%	0%	0%	0%	**10%**	**10%**	0%	**10%**	0%	0%
search engines	60%	70%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	90%	50%	**100%**
shakespeare	0%	20%	50%	50%	50%	**70%**	**70%**	**70%**	60%	50%	50%
table tennis	**50%**	20%	**50%**	**50%**	**50%**	**50%**	**50%**	**50%**	40%	40%	**50%**
weather	60%	20%	60%	60%	30%	60%	50%	50%	**70%**	60%	60%
vintage cars	0%	0%	**40%**	**40%**	0%	**40%**	**40%**	0%	30%	10%	**40%**
avg	21%	22%	36%	37%	25%	41%	37%	30%	**44%**	30%	36%
max	80%	70%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	90%	90%	**100%**
min	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
stdev	27%	17%	26%	26%	28%	30%	31%	32%	23%	28%	26%

average, the two algorithms exhibit completely different behaviors. The behavior of the Hits algorithm is erratic. As the standard deviation indicates and the query results reveal, the performance of Hits exhibits large variance. There are queries (such as “amusement parks”, “genetic”, “classical guitar”, “table tennis”) for which the Hits algorithm exhibits good relevance and high relevance ratios but, at the same time, there are many queries (such as, “basketball”, “gun control”, “moon landing”,“recipes”) for which Hits has a 0% relevance ratio. This is related to the *Tightly Knit Community*(TKC) effect. Hits is known to favor nodes that belong to the most tightly interconnected component in the graph.

Table III. Relevance Ratio

Query	HITS	PAGERANK	INDEGREE	SALSA	HUBAVG	MAX	AT-MED	AT-AVG	BFS	BAYESIAN	SBAYESIAN
abortion	90%	70%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	90%	**100%**
affirmative action	70%	50%	50%	50%	10%	10%	10%	10%	**80%**	40%	50%
alcohol	**90%**	60%	**90%**	**90%**	**90%**	80%	80%	80%	**90%**	80%	**90%**
amusement parks	**100%**	30%	30%	50%	0%	90%	10%	0%	80%	**100%**	30%
architecture	10%	**70%**	**70%**	**70%**	10%	60%	**70%**	10%	60%	10%	**70%**
armstrong	20%	**50%**	20%	20%	20%	20%	20%	20%	**50%**	20%	20%
automobile industries	10%	10%	20%	30%	10%	10%	10%	10%	**60%**	20%	20%
basketball	0%	70%	20%	20%	0%	10%	10%	10%	**100%**	10%	20%
blues	60%	**80%**	60%	60%	70%	60%	70%	70%	50%	60%	60%
cheese	0%	20%	30%	30%	10%	0%	0%	10%	**50%**	0%	30%
classical guitar	**90%**	50%	70%	70%	50%	80%	50%	50%	**90%**	70%	70%
complexity	0%	50%	50%	50%	0%	**90%**	**90%**	0%	80%	0%	50%
computational complexity	90%	70%	90%	90%	90%	90%	90%	90%	90%	**100%**	90%
computational geometry	**100%**	40%	70%	70%	70%	**100%**	70%	70%	**100%**	80%	70%
death penalty	**100%**	70%	90%	90%	70%	**100%**	**100%**	**100%**	**100%**	**100%**	90%
genetic	**100%**	70%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	90%	**100%**	**100%**
geometry	**90%**	20%	**90%**	**90%**	**90%**	**90%**	**90%**	80%	**90%**	80%	**90%**
globalization	**100%**	70%	90%	90%	**100%**	**100%**	**100%**	**100%**	90%	**100%**	90%
gun control	0%	50%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	80%	**100%**
iraq war	40%	30%	30%	30%	10%	20%	20%	10%	**90%**	40%	30%
jaguar	0%	**30%**	0%	0%	0%	0%	0%	0%	10%	0%	0%
jordan	0%	30%	30%	30%	40%	**100%**	**100%**	**100%**	40%	30%	30%
moon landing	0%	30%	20%	20%	0%	0%	0%	0%	**100%**	0%	20%
movies	10%	20%	50%	40%	50%	**70%**	**70%**	**70%**	60%	60%	50%
national parks	0%	50%	10%	10%	**80%**	**80%**	**80%**	0%	70%	0%	10%
net censorship	0%	30%	80%	80%	60%	**90%**	**90%**	**90%**	80%	70%	80%
randomized algorithms	70%	**80%**	**80%**	**80%**	40%	50%	50%	50%	60%	**80%**	70%
recipes	0%	20%	70%	70%	30%	90%	90%	**100%**	80%	80%	70%
roswell	0%	20%	40%	40%	**70%**	**70%**	60%	0%	60%	10%	40%
search engines	80%	90%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	90%	80%	**100%**
shakespeare	**100%**	70%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
table tennis	90%	60%	**100%**	**100%**	90%	90%	90%	90%	90%	90%	**100%**
weather	80%	50%	80%	80%	60%	80%	80%	80%	**90%**	80%	80%
vintage cars	20%	10%	60%	60%	20%	60%	60%	20%	**70%**	40%	60%
avg	47%	48%	61%	62%	51%	67%	64%	54%	**78%**	56%	61%
max	**100%**	90%	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
min	0%	**10%**	0%	0%	0%	0%	0%	0%	**10%**	0%	0%
stdev	43%	23%	31%	31%	38%	36%	36%	42%	21%	37%	31%

The performance of the algorithm depends on how relevant this component is to the query. We discuss the TKC effect further in Section 5.3.

On the other hand, PAGERANK exhibits consistent, albeit poor, performance. It is the only algorithm that never achieves 100% relevance on any query, always producing at least one nonrelevant result. Furthermore, the PAGERANK algorithm is qualitatively different from the rest. It is the algorithm that most often promotes documents (relevant, or not) not considered by the remaining algorithms. The strong individuality of PAGERANK becomes obvious when examining the average distance of PageRank to the remaining algorithms (Tables IV, V, VI, VII in Appendix A), especially for the I and WI measures. This is something to be expected since the philosophy of the algorithm is different. Furthermore, the Base Set for each query is constructed in a way that is meant to take advantage of the mutual reinforcing relation between hubs and authorities, a property that does not affect the PageRank algorithm. We discuss this more in Section 5.3.

The Max algorithm emerges as the second best option after BFS, and it actually achieves the best high relevance score. The quality of the Max algorithm usually depends on the quality of the seed node and the nodes with the highest in-degree. We actually observed that, in most cases, the top 10 nodes returned by Max are a subset of the ten nodes with the highest in-degree, and the ten nodes that are most co-cited with the seed node. The ratios of Max indicate that the seed node is usually relevant to the query.

This observation agrees with the relatively high quality of the results of the InDegree algorithm. On average, 3 out of 10 of the most popular documents are highly relevant, and 6 out 10 are relevant. Because of its simplicity, one would expect the quality of the InDegree algorithm to set the bar for the remaining algorithms. Surprisingly, in many queries, the InDegree algorithm outperforms some of the more sophisticated algorithms. We should note though that, due to the simplicity of the algorithm, the InDegree algorithm is the one that is most affected by the choice of the search engine that is used for generating the Base Set of Web pages. Therefore, the performance of the InDegree algorithm reflects, in part, the quality of the Google search engine which uses, to some extent, link analysis techniques. It would be most interesting if one could generate a Base Set using a search engine that does not make use of any link analysis.

In our experiments, in most queries, the Salsa algorithm produces the same top 10 pages as the InDegree algorithm. The similarity of the two algorithms becomes obvious in the average values of all similarity measures in the tables of Appendix A, and especially in the I and WI measures. As can be seen in Table I, the graphs in our experiments contain a giant authority connected component which attracts most of the authority weight of the Salsa algorithm. As a result, the algorithm reduces to the InDegree algorithm.

For the other variants of Hits, the HubAvg is the one that performs the worst, only slightly better than Hits. HubAvg suffers from its own TKC effect that we describe in Section 5.3. For the AT-med and AT-avg, close examination of the results reveals that they usually have the same ratios as either the Hits or the Max algorithm. In most cases, the top 10 results of these algorithms are a subset of the union of the top 10 results of Hits and Max. Thus, it is not surprising that the average performance ratios of AT-med and AT-avg take values between the ratios of Max and Hits. We also note that all derivatives of Hits (including Max) exhibit similar erratic behavior to that of Hits. This is due to the various TKC phenomena that we describe in the next section.

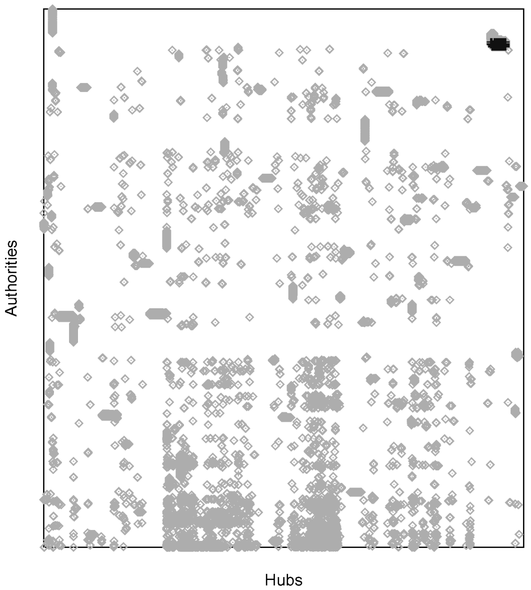
The study of the performance of the Bayesian algorithms provides the most intriguing observation of the experimental evaluation. The performance of the SBayesian algorithm is almost identical to that of the InDegree algorithm. This is corroborated by the average similarity measures in Appendix A. For all similarity measures, the two algorithms appear to be very close. It remains an interesting open problem to justify and, if possible, to formally prove the similarity of the two algorithms.

On the other hand, the BAYESIAN algorithm is less similar to the INDEGREE algorithm and more similar to the HITS algorithm and the derivatives of the HITS algorithm. As a result, the performance ratios of the BAYESIAN algorithm drop in comparison to the SBAYESIAN algorithm. Still it performs better than the HITS algorithm. There are queries such as "amusement parks" and "computational complexity" for which the algorithm achieves the best performance ratios. The difference in behavior between BAYESIAN and SBAYESIAN is obviously an effect of the "link tendency" parameters which "absorb" some of the weight of a link between two nodes, thus making the algorithm less dependent on the in-degree. It is interesting to note that the two algorithms are still close, especially when considering similarity measures that take into account the whole ranking (such as the d_1, or d_r distance). This is something that we expected since the underlying philosophy is similar.

5.3 Community Effects

A Link Analysis Ranking algorithm is defined as a function that maps a graph G to a real vector of weights. The function allocates the weight among the nodes depending on the graph theoretic properties of the nodes and the overall structure of the graph G. An intriguing question is to understand how the structure of the graph affects the ranking behavior of the algorithm. For example, it is well known that the HITS algorithm tends to favor the nodes that belong to the most dense component of the bipartite hubs and authorities graph. Lempel and Moran [2000] observed that the side-effect of this property of HITS is that, in a graph that contains multiple communities, the HITS algorithm will only focus on one of them in the top positions of the ranking, the one that contains the hubs and authorities that are most tightly interconnected. They termed this phenomenon the *Tightly Knit Community (TKC)* effect, and they compared the *focused* behavior of the HITS algorithm against the *mixing* behavior of the SALSA algorithm which tends to represent different communities in the top positions of the ranking. In this section, we study such *community effects* for all the algorithms that we consider. Our objective is to understand the kind of structures that the algorithms favor and the effects on the rankings they produce.

The HITS *algorithm.* It has been well documented that the HITS algorithm tends to favor the most "tightly interconnected component" of hubs and authorities in the graph G. This was first stated by Kleinberg [1998] in his original paper, and Drineas et al. [1999] provide some theoretical analysis to support this observation. The TKC effect is a byproduct of the properties of the Singular Value Decomposition. The hub and authority vectors, $\boldsymbol{h}$ and $\boldsymbol{a}$, (normalized in the Euclidean norm) are chosen so that the product $\boldsymbol{h}W\boldsymbol{a}$ is maximized. As a result, the nodes that belong to the most dense bipartite community of hubs and authorities receive the maximum weight. Another intuitive interpretation of the TKC effect comes from the analysis in Section 2.2.3, where we showed



(a) Matrix plot with the top-10 results of HITS

	HITS
1.	(1.000) Coffee Club URL: www.Batavia-rof.com
2.	(0.982) Hotel and Travel URL: www.bwdriftwood.com
3.	(0.935) Basement Writers URL: www.basement-writers.com
4.	(0.935) Before Today URL: www.beforetoday.com
5.	(0.935) Bennett Boxing URL: www.bennettboxing.com
6.	(0.935) Boeing Mail URL: www.boeingmail.com
7.	(0.935) Burdan USA URL: www.burdanusa.com
8.	(0.935) British Jokes URL: www.callusforfun.com
9.	(0.917) Religious Happenings URL: www.bellbrook-umc.com
10.	(0.917) Blade Liners URL: www.bladeliners.com

(b) Top-10 results of HITS

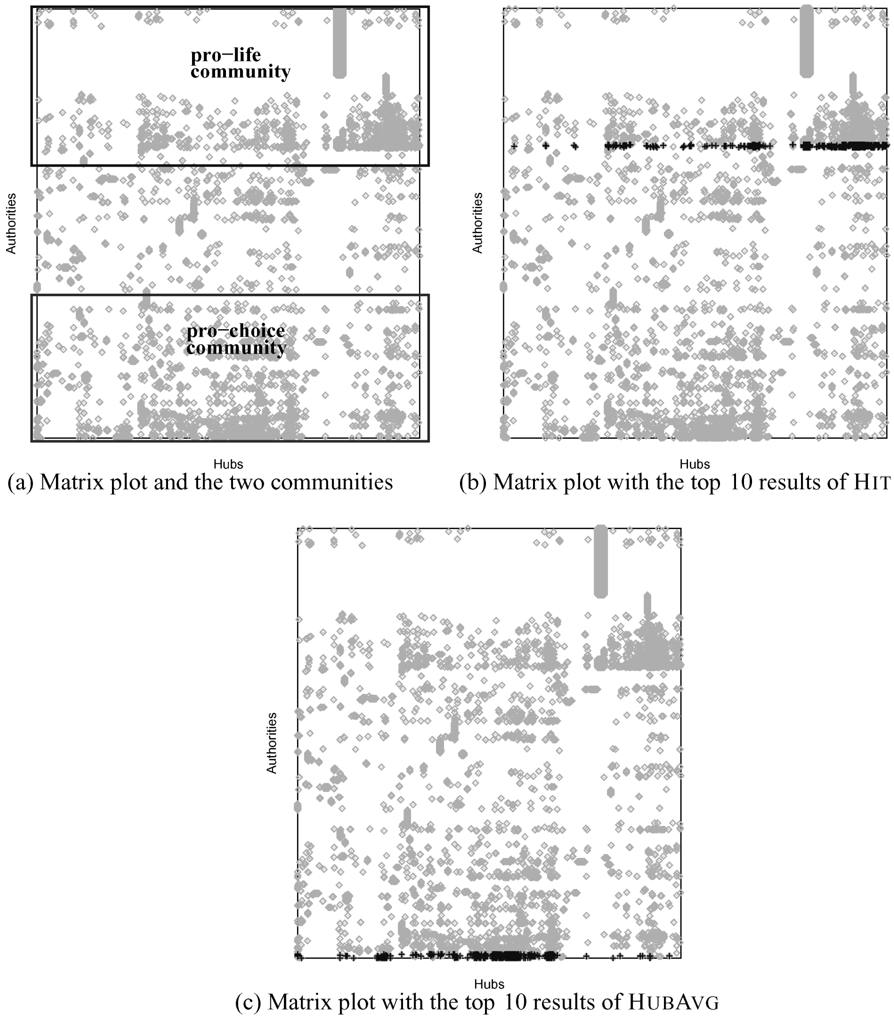
Fig. 13. The TKC effect for the HITS algorithm for the query "gun control".

that the weight on node i, after n iterations, is proportional to the number of $(BF)^n$ paths that leave node i.

The TKC effect is prominent in our experiments with the HITS algorithm, most of the time leading to a severe topic drift. Consider, for example, the query "gun control". Figure 13 shows the matrix plot of the graph and the top 10 results for HITS. In this query, HITS gets trapped in a tightly interconnected component of 69 nodes (63 hubs and 37 authorities) which is completely disconnected from the rest of the graph, and obviously unrelated to the query. Similar phenomena appear in many of the queries that we have tested (examples include, "vintage cars", "recipes", "movies", "complexity").

Consider now the query "abortion". Figure 14(a) shows the plot of the graph for this query. The graph contains two separated, but not completely disconnected, communities of Web pages: the pro-choice community, and the pro-life community. The pro-life community contains a set X of 37 hubs from a religious group[10] which form a complete bipartite graph with a set Y of 288 authority nodes. These appear as a vertical strip in the top-right part of the plot. This tightly interconnected set of nodes attracts the HITS algorithm to that community, and it ranks these 37 nodes as the best hubs. Among the 288 authorities, HITS ranks in the top 10, the authorities that are better interconnected with the remaining hubs in the community. The plot for the top 10 results for HITS is shown in Figure 14(b). The points in darker color correspond to the top 10 results of the HITS algorithm. The results can be found in Table VIII in Appendix A.

[10] http://www.abortion-and-bible.com/.



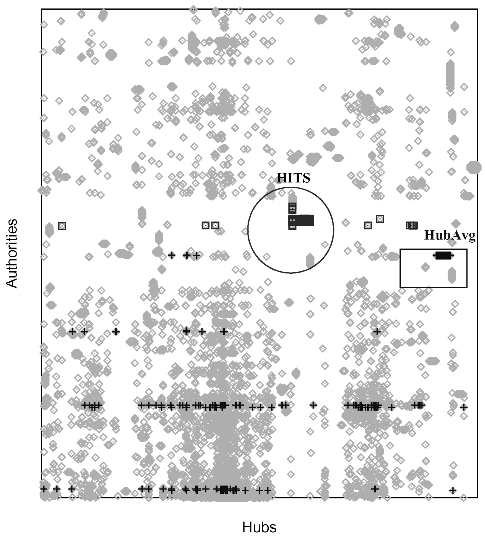
(a) Matrix plot and the two communities

(b) Matrix plot with the top 10 results of HITS

(c) Matrix plot with the top 10 results of HUBAVG

Fig. 14. The query “abortion”.

The HUBAVG *Algorithm.* In Section 3.1, we showed that the HUBAVG algorithm computes the Singular Value Decomposition of the normalized matrix W_e. Therefore, we expect the HUBAVG algorithm to exhibit similar TKC effects in the normalized authority space defined by the matrix W_e. This is indeed the case, and it accounts for the poor behavior of the HUBAVG algorithm. However, the normalization causes the HUBAVG algorithm to differ significantly from HITS. The HUBAVG algorithm favors tightly-knit communities, but it also poses the additional requirement of *exclusiveness*. That is, it requires that the hubs are *exclusive* to the community to which they belong. As a result, the HUBAVG algorithm tends to favor tightly-knit, isolated components in the graph that contain nodes of high in-degree. These correspond to long, thin horizontal strips in our plots. If such a strip exists, that is, if there exists a large set of hubs that all



(a) Matrix plot with the top 10 results of Hits and HubAvg

	Hits
1.	(1.000) HonoluluAdvertiser.com URL:www.hawaiisclassifieds.com
2.	(0.999) Gannett Company, Inc. URL:www.gannett.com
3.	(0.998) AP MoneyWire URL:apmoneywire.mm.ap.org
4.	(0.990) e.thePeople : Honolulu Adv URL:www.e-thepeople.com/affiliates
5.	(0.989) News From The Associated Press URL:customwire.ap.org/dynamic/fron
6.	(0.987) Honolulu Traffic Cameras, City URL:www.co.honolulu.hi.us/cameras/
7.	(0.987) News From The Associated Press URL:customwire.ap.org/dynamic/fron
8.	(0.987) News From The Associated Press URL:customwire.ap.org/dynamic/fron
9.	(0.987) News From The Associated Press URL:customwire.ap.org/dynamic/fron
10.	(0.987) News From The Associated Press URL:customwire.ap.org/dynamic/fron

(b) Top 10 results of Hits

	HubAvg
1.	(1.000) Le Web des iles www.chez.com/zanozile
2.	(0.991) Please stand by.. www.sofcom.com.au
3.	(0.968) Sign in - Yahoo! Groups groups.yahoo.com/group/mauriti
4.	(0.005) Microsoft bCentral - FastCounter fastcounter.bcentral.com/fc-jo
5.	(0.004) **Recipes are Cooking at Net** **URL:www.netcooks.com**
6.	(0.004) *Mauritian cuisine, cooking and* *URL:ile-maurice.tripod.com*
7.	(0.004) Mauritius Australia Connection www.cjp.net
8.	(0.003) Mauritius Australia Connection www.users.bigpond.com/clancy/t
9.	(0.003) SleepAngel.com - Are you snoring wcpsecure.com/app/aftrack.asp?
10.	(0.002) *Chef Jobs Foodservice Culinary* *URL:chef2chef.net*

(c) Top 10 results of HubAvg

Fig. 15. Hits and HubAvg for the query "recipes".

point to just a few authorities in the graph, then this community receives most of the weight of the HubAvg algorithm. Unfortunately, this occurs often in our experiments, resulting in topic drift for HubAvg.

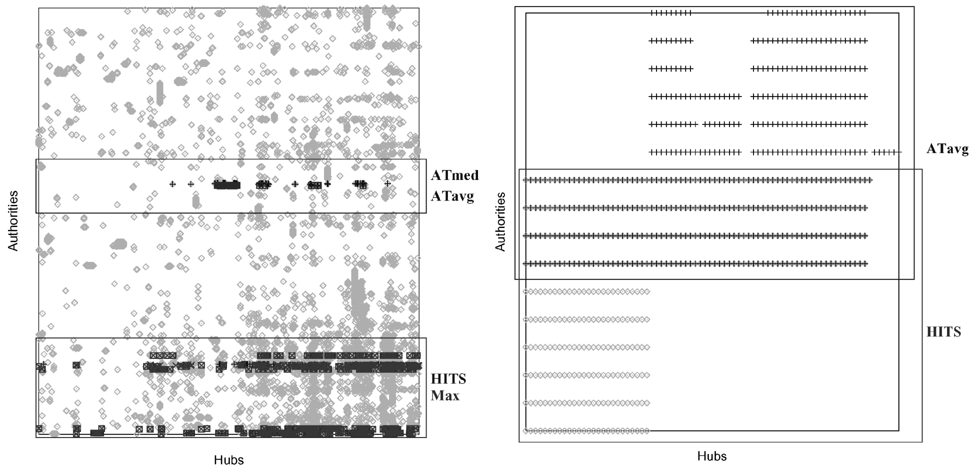
Figure 15 shows the plot of the graph for the query "recipes". The communities that attract the top 10 results of Hits and HubAvg are marked on the plot. The tables contain the top 10 tuples for each algorithm. The community on news and advertising that attracts Hits contains a set of hubs that point to nodes outside the community. This corresponds to the vertical strip above the marked rows for HITS. HUBAVG escapes this community, but it assigns almost all weight to a community of just three nodes that are interconnected by 45 hubs. Only two out of these 45 hubs point to nodes other than these three nodes. Note that, in order for such a structure to attract HUBAVG, the authorities (or at least some of the authorities) must have sufficiently large in-degree. In this case, the top three nodes for for HUBAVG correspond to the nodes with the 10th, 11th and 13th highest in-degree in the graph. This is a typical example of the behavior of the HUBAVG algorithm. Although the HUBAVG algorithm manages to escape the communities that pull HITS into topic drift, it still falls victim to its own TKC effect.

Consider the HUBAVG algorithm on the query "abortion". The authority nodes in the set Y are not all of equal strength. Some of them are well interconnected with other hubs in the pro-life community (the ones ranked high by HITS), but the large majority of them are pointed to only by the hubs in the set X. Recall that the HUBAVG algorithm requires that the hubs point only (or at least mainly) to good authorities. Most of the authorities in Y are poor. As a result, the hubs in X are penalized. The HUBAVG algorithm avoids the pro-life community and focuses on the pro-choice one. Note that this is the community that contains the node with the maximum in-degree. The top 10 results of HUBAVG are plotted in Figure 14(c).

The MAX *algorithm.* The influence of the various communities on the ranking of the MAX algorithm is primarily exerted through the seed node. The community that contains the seed node and the co-citation of the seed node with the remaining nodes in the community determine the focus of the MAX algorithm. For example, in the "movies" query, the seed node is the Internet Movie Database[11] (IMDB), and the algorithm converges to a set of movie databases and movie reviews sites. Similarly for the "net censorship" query, where the seed node is the Electronic Frontier Foundation[12] (EFF), the algorithm outputs highly relevant results. In both these cases, the MAX algorithm manages best to distill the relevant pages from the community to which it converges. On the other hand, in the case of the "affirmative action" query, the seed node is a copyright page from the University of Pennsylvania and, as a result, the MAX algorithm outputs a community of university home pages.

There are cases where the seed node may belong to more than one community. For example, for the query "basketball", the seed node is the NBA official Web page.[13] This page belongs to the basketball community, but it has the highest overlap with a community of nodes from MSN,[14] which causes the MAX algorithm to converge to this community. It may also be the case that there are multiple seed nodes in the graph. For example, for the "randomized algorithms" query, there are two seeds in the graph, one on algorithms, and one on computational geometry. As a result, the algorithm mixes pages of both communities.

[11] http://www.imdb.com.
[12] http://www.eff.org.
[13] http://www.nba.com.
[14] http://www.msn.com.



(a) Matrix plot for the “classical guitar” query.

(b) Matrix plot of (only) the top 10 results of AT-avg and Hits for “roswell query.

Fig. 16. The AT-med and AT-avg algorithms.

It is also interesting to observe the behavior of Max on the query “abortion”. The seed node in the graph is the “NARAL Pro-Choice” home page. Given that there is only light co-citation between the pro-choice and pro-life communities, one would expect that the algorithm would converge to pro-choice pages. However, the Max algorithm mixes pages from both communities. The third page in the ranking of Max is the “National Right To Life” (NRTL) home page, and there are two more in the fifth and seventh positions of the ranking. After examination of the data, we observed that the NRTL page has the second highest in-degree in the graph. Furthermore, its in-degree (189) is very close to that of the seed node (192), and, as we observed before, it belongs to a tightly interconnected community. In this case, the NRTL page acts as a *secondary* seed node for the algorithm, pulling pages from the pro-life community to the top 10. As a result, the algorithm mixes pages from both communities.

The AT*(k) algorithm.* For the AT-med and AT-avg algorithms, we observed that in most cases their rankings are the same as either that of Hits or that of Max. The cases that deviate from these two reveal some interesting properties of the algorithms. Consider the query “classical guitar”. The plot of the graph and the top 10 results of the algorithms are shown in Figure 16(a). The top 10 results for all algorithms appear in Table X in Appendix A. For this graph $k = 3$ for AT-med, and $k = 5$ for AT-avg. In this query, the AT-med and AT-avg algorithms allocate most of the weight to just four nodes, which are ranked lower by Hits and Max. We discovered that these four nodes belong to a complete bipartite graph pointed to by approximately 120 hubs. The Hits algorithm favors a community of nodes drawn together by a set of strong hubs. As we decrease k, the effect of the strong hubs decreases and other structures emerge as the most tightly connected ones. Surprisingly, the Max algorithm, which corresponds to the other extreme value of k, produces a set of top 10 results that are more similar to that of Hits than to that of AT-med and AT-avg.

We observed the similar phenomena for the queries “cheese” and “amusement parks”.

In order to better understand the mechanics of the threshold algorithms, we consider the “roswell” query. In this case, the AT-AVG produces the same top 4 authorities as HITS, but the lower part of the ranking is completely different. In this graph, $k = 4$ for the AT-AVG algorithm. Figure 16(b) shows the plot for the top 10 authorities of both algorithms. The middle part corresponds to the common authorities, the bottom rows are the authorities of HITS, and the upper rows are the authorities of AT-AVG. Note that the authorities of HITS are more tightly interconnected. However, given that $k = 4$, all hubs receive the same weight, since they all point to the middle authorities which are the strongest. These four authorities play a role similar to that of the seed in the MAX algorithm. Therefore, the authorities of AT-AVG are ranked higher because they have higher co-citation with the “seed” nodes despite the fact that they are not as well interconnected. It seems that as we decrease the value of k, the effect of the in-degree increases.

The INDEGREE *and* SALSA *algorithms.* The effect of different communities to the INDEGREE algorithm is straightforward. Communities that contain nodes with high in-degree will be promoted, while communities that do not contain “popular” nodes are not represented regardless of how tightly interconnected they are. As a result, INDEGREE usually mixes the results of various communities in the top 10 results. One characteristic example is the “genetic” query where the INDEGREE algorithm is the only one to introduce a page from the Genetic Algorithms community. The simplistic approach of the INDEGREE algorithm appears to work relatively well in practice. However, it has no defense mechanism against *spurious* authorities, that is, nodes with high in-degree that are not related to the query, as in the case of the “amusement parks” query. Another such example is the “basketball” query, where the algorithm is drawn to the set of spurious authorities from the MSN community.

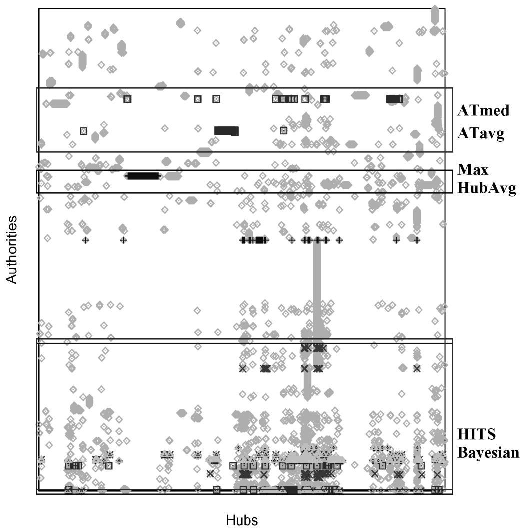
As we have already discussed, in most cases, the SALSA algorithm outputs the same top 10 results as INDEGREE. The “amusement parks” query is one of the rare cases where the rankings of the two algorithms differ. For this query, the node with the highest in-degree is a completely isolated node. SALSA is designed to handle such situations (i.e., nodes that are very popular, but belong to weak communities), so the high in-degree isolated node does not appear in the top 10 of the SALSA algorithm. The top 10 results for both SALSA and INDEGREE algorithms are shown in Figure 17. The premise of SALSA is interesting. It sets the weight of a node as a combination of the popularity of the community it belongs to as well as its popularity within the community. However, a community, in this case, is defined as an authority connected component (ACC) of the graph. This is a very strong definition since, if a node shares even just one hub with the ACC, it immediately becomes part of the community. It would be interesting to experiment with SALSA on graphs that contain multiple ACCs of comparable size, and observe how the algorithm mixes between the different communities. In our experiments, all graphs contain a giant ACC and many small ACCs that do not contribute to the ranking. Thus, the SALSA algorithm reduces to INDEGREE with the additional benefit of avoiding the occasional isolated high in-degree

	SALSA
1.	(1.000) **Empty title field** **URL:www.sixflags.com**
2.	(0.912) **Busch Gardens Adventure Parks** **URL:www.buschgardens.com**
3.	(0.879) **IAAPA** **URL:www.iaapa.org**
4.	(0.868) Free Legal Advice in 100+ Law freeadvice.com
5.	(0.868) AttorneyPages Helps You Find attorneypages.com
6.	(0.868) Do It Yourself Home Improvement doityourself.com
7.	(0.824) ExpertPages.com - Books, Tapes expert-pages.com/books.htm
8.	(0.758) Accidents Happen - Why are Lawyers law.freeadvice.com/resources/c
9.	(0.736) *Exhibits Collection – Amusement* *URL:www.learner.org/exhibits/parkp*
10.	(0.736) **Cedar Point Amusement Park** **URL:www.cedarpoint.com**

(a) Top 10 results of the SALSA algorithm

	INDEGREE
1.	(1.000) HONcode: Principles www.hon.ch/HONcode/Conduct.htm
2.	(0.645) **Empty title field** **URL:www.sixflags.com**
3.	(0.589) **Busch Gardens Adventure Parks** **URL:www.buschgardens.com**
4.	(0.567) **IAAPA** **URL:www.iaapa.org**
5.	(0.560) Free Legal Advice in 100+ Law freeadvice.com
6.	(0.560) AttorneyPages Helps You Find attorneypages.com
7.	(0.560) Do It Yourself Home Improvement doityourself.com
8.	(0.532) ExpertPages.com - Books, Tapes expert-pages.com/books.htm
9.	(0.489) Empty title field imgserv.adbutler.com/go2/;ID=1
10.	(0.489) Empty title field imgserv.adbutler.com/go2/;ID=1

(b) Top 10 results of the INDEGREE algorithm



(c) Matrix plot for the “amusement parks” query

Fig. 17. The query “amusement parks”.

node for the queries “amusement parks”, “automobile industries”, “moon landing”, “movies”, and “national parks”.

The “amusement parks” query is particularly interesting since it reveals the behavior of the algorithms in extreme settings. The top 10 results for all algorithms appear in Table IX in Appendix A. The plot of the graph for this query is shown in Figure 17. Both MAX and HUBAVG allocate all their weight to the isolated node with the maximum in-degree, and zero to the rest of the nodes. The HITS algorithm escapes this component easily, and converges to the most relevant community. The AT-MED, AT-AVG algorithms converge again to a small tight bipartite graph different from that of HITS.

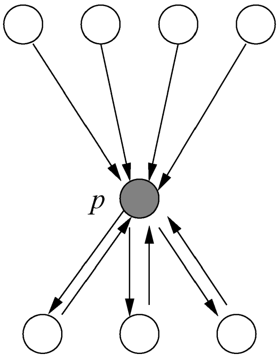


Fig. 18. Topic drift for PAGERANK.

The BFS *algorithm.* The BFS algorithm exhibits the best performance among the LAR algorithms. This can be attributed to the fact that the BFS algorithm is not as sensitive to tightly-knit communities. The weight of a node in the BFS algorithm depends on the number of neighbors that are reachable from that node, and not on the number of paths that lead to them. Highly interconnected components (i.e., components with high reachability) influence the ranking, but *tightly* interconnected components (components with large number of *paths* between nodes) do not have a significant effect on BFS. As a result, the BFS algorithm avoids strong TKC effects and strong topic drift. The experimental results suggest that reachability is a better measure of quality than connectivity.

Furthermore, the BFS algorithm is not strongly affected by the existence of spurious authorities since it considers the popularity of a node in a neighborhood of larger radius. For example, in the "basketball" query, the BFS algorithm does not output any page from the MSN[15] community which contains many nodes within the top 10 results of the INDEGREE algorithm.

The PAGERANK *algorithm.* The existence of dense communities of hubs and authorities does not have a significant effect on the PAGERANK algorithm since it does not rely on mutual reinforcement for computing authority weights. However, we observed that there are certain structures to which the PAGERANK algorithm is sensitive. For example, in the query "amusement parks", the PAGERANK algorithm assigns a large weight to the isolated node with the maximum indegree. We observed that, in general, PAGERANK tends to favor isolated nodes of high in-degree. In this case, the hubs that point to the isolated node transfer all their weight directly to that node since they do not point anywhere else, thus increasing its weight. Furthermore, the PAGERANK algorithm favors structures like the one shown in Figure 18. There exists a node p that is pointed to exclusively by a set of hubs (not necessarily many of them), and it creates a two link cycle with one or more nodes. In this case, the node p reinforces itself since the random walk will iterate within this cycle until it performs a random jump. This explains the fact that in certain cases the performance

[15] http://www.msn.com.

of the algorithm improves when we increase the jump probability. Note that such structures are very common in the Web where p may be the entry point to some Web site and all pages within this site point back home. They appeared very often in our experiments (even with pages that are not in the same site), and they account for most of the topic drift of the PAGERANK algorithm. The effect of such structures has also been investigated by Bianchini et al. [2002].

Overall, the PAGERANK algorithm appears to be mixing between different communities. This should probably be attributed to the random jumps that the algorithm performs. The random jumps are probably also responsible for the fact that the algorithm performs better than the rest of the algorithms on very sparse graphs (like in the "jaguar" and "randomized algorithms" queries).

The BAYESIAN *and* SBAYESIAN *algorithms.* As indicated by all similarity measures, the SBAYESIAN algorithm exhibits behavior very similar to that of the INDEGREE algorithm. Thus, the effect of communities on SBAYESIAN is very similar to that of the INDEGREE algorithm. It favors the communities that contain popular nodes, regardless of density or the size of the communities. We note that SBAYESIAN agrees with the INDEGREE, even in the special case of the "amusement parks" query where the node with the maximum in-degree consists of an isolated node.

On the other hand, the BAYESIAN algorithm is far less dependent upon the in-degree of the nodes. In the "amusement parks" query, the algorithm disregards completely the isolated node with the highest in-degree, and focuses on the same community as the HITS algorithm, producing the best set of top 10 results. One possible explanation is that the effect of many nodes pointing to a single node is absorbed by the parameters e_i, and thus the weight of the pointed node does not increase inordinately. That is, the extra parameters e_i can be "used" to explain high in-degrees, so in-degree does not overly affect the resulting authority weights. This also implies that the BAYESIAN algorithm favors communities that contain hubs with relatively high out-degree. For example, in the case of the "abortion" query, the algorithm converges to the pro-life community which contains a set of strong hubs. This could also explain the high dissimilarity between the BAYESIAN and the HUBAVG algorithms since HUBAVG tends to promote isolated components with high in-degree nodes.

In general, the BAYESIAN algorithm exhibits high similarity with the HITS algorithm and its variants (especially AT-AVG). In all queries, the top 10 results of BAYESIAN contain at least one node of the principal community favored by HITS. This suggests that the BAYESIAN algorithm is affected by densely interconnected components. However, there are many cases, such as the queries "gun control", "movies", "net censorship", "randomized algorithms", where the BAYESIAN algorithm deviates from the HITS algorithm. In most such cases, HITS converges to small isolated dense communities. These are represented in the top 10 results of BAYESIAN, and the remaining positions are filled with pages promoted by SBAYESIAN or MAX. It is very common that the top 10 results of BAYESIAN are some combination of results of HITS and SBAYESIAN or MAX. It is an interesting puzzle to understand the behavior of the BAYESIAN algorithm. Furthermore, it would be interesting to study how changing the distribution parameters for the e_i random variables affects the properties of the algorithm.

5.4 Concluding Remarks for the Experimental Study

We performed an experimental analysis of Link Analysis Ranking. Using user feedback, we evaluated the ability of each algorithm to rank highly documents relevant to the queries. Also, in order to better understand the performance of the algorithms, we studied how the existence of various communities in the graphs affect the behavior of the algorithms. The experimental analysis revealed several limitations and weaknesses of Link Analysis Ranking. Overall, LAR algorithms were not effective in retrieving the most relevant pages in the dataset within the top few positions of the ranking. The main reason for the shortcomings of the LAR algorithms appears to be the various community effects that we described in Section 5.3. The structures that are promoted by the various algorithms are usually not relevant to the query at hand. We observed that the algorithms that exhibit a "mixing" behavior, that is, they allocate the weight across different communities in the graph, tend to perform better than more "focused" algorithms that tend to allocate all weight to the nodes of a single community. In our experiments, the graphs were often fragmented. As a result, focused algorithms often produced a lopsided weighting scheme where almost all weight was allocated to just a few nodes. This suggests that in the future we should consider relaxations of the existing algorithms so that they employ more moderate approaches.

Alternatively, we could consider improving the input graph so that it does not include structures that cause topic drift. Our experimental study indicates that the properties of the graph that are given as input to the LAR algorithm are critical to the quality of the output of the LAR algorithm. Little research [Gibson et al. 1998; Bharat and Henzinger 1998] has been devoted to the problem of improving the algorithm that generates the input graph. During the expansion of the Root Set, many nonrelevant pages are introduced into the Base Set. Content analysis could be applied to filter some of the noise introduced in the graph and steer the algorithm towards the relevant community of nodes. This could be achieved either by pruning some of the nonrelevant pages, or by downweighting their effect by adding weights to the links. Other approaches include grouping similar nodes together to avoid extreme TKC phenomena, as described in the work of Roberts and Rosenthal [2003]. The problem of understanding the input graph is of fundamental importance for the study of Link Analysis Ranking.

Furthermore, the analysis of Section 5.3 indicates that, in order to evaluate an LAR algorithm, we need to understand the interplay between the graph and the algorithm, as well as the connection between graph structures and topical relevance. An LAR algorithm is a mapping from a graph to a set of weights. Thus, we need to understand how the structural properties of the graph affect the ranking of the algorithm. Then, we need to study how the relevance of the Web pages relates to these structural properties. For example, assume that we observe that the graphs are likely to contain cycles. Then, we need to understand which algorithms are affected by the existence of cycles in the graph, and how likely it is for the nodes that belong to the cycles to be relevant to the query. Alternatively, if we know that an algorithm favors cycles, then we need to estimate how often cycles appear in the graphs, and, again, how likely it is to be relevant to the query. Performing such analysis will enable us to predict the combinations of graphs and algorithms that we expect to perform well and work towards improving the algorithms, or the construction of the input graph. Ideally, we would like to be able to characterize the structures that an LAR algorithm favors within the theoretical framework we introduced in Section 4. Then, we would be able to argue formally about the performance of the LAR algorithms on specific families of graphs.

In our datasets, we observed that tightly-knit communities, cycles, and isolated components are usually off topic. Such structures appear often, and the algorithms that promote them (HITS, HUBAVG, PAGERANK) perform poorly. On the other hand, reachability and high in-degree are good indications of relevance, and the algorithms that rely on them (BFS, INDEGREE) perform better. Furthermore, the fact that the INDEGREE algorithm performs relatively well allows us to predict that the MAX algorithm will perform well since it is strongly influenced by the quality of the node with the highest in-degree. In a similar fashion, the knowledge that the graphs contain a giant component allows us to predict that the properties of the SALSA algorithm are immaterial, and the algorithm reduces to the INDEGREE algorithm.

Finally, although it appears that the LAR algorithms cannot always capture the true quality of the documents, it was interesting to observe that the LAR algorithms were very successful in discovering the “greater picture” behind the topic of the query. For example, for the query “computational geometry”, the algorithms returned pages on math. For the query “armstrong”, we got many pages on jazz music, even if they did not contain any reference to Louis Armstrong. For the query “globalization”, the algorithms returned independent media sites, anti-Bush sites, and workers’ movements sites. It is questionable whether the users would like such a wide perspective of the query (and our user study proved that they usually do not); however, it is important to have a tool that can provide the “Web context” of a query. The algorithms for finding related pages to a query page build upon exactly this property of Link Analysis. Preliminary experiments indicate that LAR algorithms are successful in handling such queries. Furthermore, as a result of this generalization property of Link Analysis, LAR algorithms proved successful in finding highly related pages that do not contain the actual query words. This was the case in the “search engines” query, the “automobile industries” query, and the “globalization” query where the algorithms discover and rank highly pages like the World Trade Organization site. Despite its limitations, Link Analysis is a useful tool for mining and understanding the Web.

6. CONCLUSIONS

In this article, we studied the problem of Link Analysis Ranking. We approached the problem from three different perspectives. First, we extended the existing techniques to produce new LAR algorithms. Working within the

the hubs and authorities paradigm defined by Kleinberg [1999], we proposed new ways of computing hub and authority weights, namely the HubAvg and BFS algorithms, the AT(k) family of algorithms, and two algorithms based on a statistical Bayesian approach.

Second, we developed a formal framework for analyzing LAR algorithms that allows us to define specific properties of LAR algorithms such as monotonicity, distance, similarity, stability, and locality. Within this framework, we were able to provide an axiomatic characterization of the InDegree algorithm. We proved that any algorithm that is monotone, label-independent, and local produces the same ranking as the InDegree algorithm. This result indicates that our framework and the properties we defined are both meaningful and useful.

Last, we experimented with the algorithms that we proposed, as well as some of the existing ones. We performed an extensive experimental study on multiple queries, using user feedback, where we studied the quality of the algorithms, as well as the effect of graph structures on the behavior of the ranking algorithms. We observed that some of the theoretically predicted properties (for example, the TKC effect for the Hits algorithm) were indeed prominent in our experiments. We were surprised to discover that some of the "simpler" algorithms, such as InDegree and BFS, appear to perform better than more sophisticated algorithms, such as PageRank and Hits.

Our work opens a number of interesting questions for future work. First, it would be interesting to consider other variations of the hubs and authorities definitions that combine ideas from the Hits and PageRank algorithm or make use of other machine learning techniques. Furthermore, we consider our theoretical framework as a first step towards the formal analysis of LAR algorithms. There are plenty of research questions that emerge within this framework. The stability and similarity of the Bayesian algorithms remains undetermined. Also, although most of the stability and similarity results are negative, it is possible that, if we restrict ourselves to smaller classes of graphs, we can obtain positive results. We are currently examining the conditions for the stability of the Hits algorithm as well as the similarity between Hits and InDegree on a specific family of random graphs.

Another possible research direction for obtaining some positive results for rank similarity between LAR algorithms is to consider weaker notions of similarity. Rather than classifying two algorithms as rank similar, or rank dissimilar, we could instead try to characterize the degree of (dis)similarity of the two algorithms. This *rank similarity coefficient* would be defined as the maximum fraction of pairs of nodes that are ranked in a different order by the two algorithms. The maximum value for the coefficient is 1, in which case the two algorithms produce two completely reversed rankings. A similar *rank stability coefficient* can be defined for the study of rank stability.

APPENDIX

A. DISTANCE MEASURES AND SAMPLE QUERY RESULTS

In this appendix, we present the average values for all the similarity measures that we consider. We also present tables with the results of three sample queries.

Table IV. Average $I(10)$ Measure

	Hits	PageRank	InDegree	Salsa	HubAvg	Max	AT-med	AT-avg	BFS	Bayesian	SBayesian
Hits	10.0	1.1	4.1	4.1	3.4	4.3	3.9	5.2	2.8	6.6	4.0
PageRank	1.1	10.0	3.2	3.1	2.2	2.1	2.1	1.8	2.2	1.8	3.2
InDegree	4.1	3.2	10.0	9.8	5.1	6.3	6.2	6.0	5.6	6.3	9.5
Salsa	4.1	3.1	9.8	10.0	5.1	6.3	6.2	5.9	5.6	6.3	9.4
HubAvg	3.4	2.2	5.1	5.1	10.0	5.5	6.2	6.6	3.3	4.4	5.1
Max	4.3	2.1	6.3	6.3	5.5	10.0	8.7	6.7	5.5	6.0	6.2
AT-med	3.9	2.1	6.2	6.2	6.2	8.7	10.0	7.5	5.0	5.6	6.1
AT-avg	5.2	1.8	6.0	5.9	6.6	6.7	7.5	10.0	4.1	6.7	5.9
BFS	2.8	2.2	5.6	5.6	3.3	5.5	5.0	4.1	10.0	4.5	5.5
Bayesian	6.6	1.8	6.3	6.3	4.4	6.0	5.6	6.7	4.5	10.0	6.2
SBayesian	4.0	3.2	9.5	9.4	5.1	6.2	6.1	5.9	5.5	6.2	10.0

Table V. Average $WI(10)$ Measure

	Hits	PageRank	InDegree	Salsa	HubAvg	Max	AT-med	AT-avg	BFS	Bayesian	SBayesian
Hits	10.0	1.0	3.7	3.6	3.3	4.2	3.7	5.2	2.9	6.3	3.6
PageRank	1.0	10.0	2.8	2.7	2.1	2.1	2.0	1.5	1.6	1.4	2.8
InDegree	3.7	2.8	10.0	9.7	5.0	7.2	6.8	5.9	5.6	5.9	9.6
Salsa	3.6	2.7	9.7	10.0	5.0	7.1	6.9	5.8	5.6	5.9	9.4
HubAvg	3.3	2.1	5.0	5.0	10.0	5.1	6.0	6.3	2.7	4.2	4.9
Max	4.2	2.1	7.2	7.1	5.1	10.0	8.8	6.6	5.1	6.3	7.1
AT-med	3.7	2.0	6.8	6.9	6.0	8.8	10.0	7.4	4.7	5.7	6.8
AT-avg	5.2	1.5	5.9	5.8	6.3	6.6	7.4	10.0	4.0	6.9	5.7
BFS	2.9	1.6	5.6	5.6	2.7	5.1	4.7	4.0	10.0	4.5	5.6
Bayesian	6.3	1.4	5.9	5.9	4.2	6.3	5.7	6.9	4.5	10.0	5.8
SBayesian	3.6	2.8	9.6	9.4	4.9	7.1	6.8	5.7	5.6	5.8	10.0

Table VI. Average d_r Distance

	Hits	PageRank	InDegree	Salsa	HubAvg	Max	AT-med	AT-avg	BFS	Bayesian	SBayesian
Hits	0.00	0.53	0.42	0.45	0.31	0.24	0.20	0.16	0.24	0.35	0.41
PageRank	0.53	0.00	0.32	0.30	0.47	0.45	0.44	0.46	0.46	0.38	0.27
InDegree	0.42	0.32	0.00	0.08	0.40	0.36	0.36	0.37	0.39	0.33	0.21
Salsa	0.45	0.30	0.08	0.00	0.44	0.40	0.40	0.41	0.42	0.34	0.23
HubAvg	0.31	0.47	0.40	0.44	0.00	0.28	0.25	0.22	0.36	0.41	0.42
Max	0.24	0.45	0.36	0.40	0.28	0.00	0.07	0.13	0.19	0.30	0.33
AT-med	0.20	0.44	0.36	0.40	0.25	0.07	0.00	0.08	0.16	0.28	0.32
AT-avg	0.16	0.46	0.37	0.41	0.22	0.13	0.08	0.00	0.19	0.30	0.35
BFS	0.24	0.46	0.39	0.42	0.36	0.19	0.16	0.19	0.00	0.27	0.33
Bayesian	0.35	0.38	0.33	0.34	0.41	0.30	0.28	0.30	0.27	0.00	0.24
SBayesian	0.41	0.27	0.21	0.23	0.42	0.33	0.32	0.35	0.33	0.24	0.00

Table VII. Average d_1 Distance

	Hits	PageRank	InDegree	Salsa	HubAvg	Max	AT-med	AT-avg	BFS	Bayesian	SBayesian
Hits	0.00	1.64	1.22	1.25	1.32	1.11	1.12	0.84	1.46	1.33	1.33
PageRank	1.64	0.00	0.94	0.93	1.64	1.38	1.38	1.49	1.13	0.63	0.65
InDegree	1.22	0.94	0.00	0.11	1.41	0.86	0.87	1.01	0.96	0.84	0.38
Salsa	1.25	0.93	0.11	0.00	1.42	0.89	0.89	1.03	0.96	0.86	0.41
HubAvg	1.32	1.64	1.41	1.42	0.00	1.12	1.01	0.91	1.67	1.61	1.48
Max	1.11	1.38	0.86	0.89	1.12	0.00	0.21	0.57	1.19	1.16	1.01
AT-med	1.12	1.38	0.87	0.89	1.01	0.21	0.00	0.42	1.20	1.17	1.01
AT-avg	0.84	1.49	1.01	1.03	0.91	0.57	0.42	0.00	1.35	1.24	1.14
BFS	1.46	1.13	0.96	0.96	1.67	1.19	1.20	1.35	0.00	1.06	0.99
Bayesian	1.33	0.63	0.84	0.86	1.61	1.16	1.17	1.24	1.06	0.00	0.56
SBayesian	1.33	0.65	0.38	0.41	1.48	1.01	1.01	1.14	0.99	0.56	0.00

Table VIII. Query "Abortion"

Hits	PageRank	InDegree
1. (1.000) *Priests for Life Index URL:www.priestsforlife.org*	1. (1.000) *WCLA Feedback URL: www.janeylee.com/wcla*	1. (1.000) *prochoiceamerica.org : NARAL URL:www.naral.org*
2. (0.997) *National Right to Life URL:www.nrlc.org*	2. (0.911) *Planned Parenthood Action Ne URL:www.ppaction.org/ppaction/prof*	2. (0.984) *National Right to Life URL:www.nrlc.org*
3. (0.994) **After Abortion: Information URL:www.afterabortion.org**	3. (0.837) *Welcome to the Westchester C URL:www.wcla.org*	3. (0.969) *Planned Parenthood Federatio URL:www.plannedparenthood.org*
4. (0.994) *ProLifeInfo.org URL:www.prolifeinfo.org*	4. (0.714) *Planned Parenthood Federatio URL:www.plannedparenthood.org*	4. (0.865) **NAF—The Voice of Abortion URL:www.prochoice.org**
5. (0.990) **Pregnancy Centers Online URL:www.pregnancycenters.org**	5. (0.633) GeneTree.com Page Not Found www.qksrv.net/click-1248625-91	5. (0.823) *Priests for Life Index URL:www.priestsforlife.org*
6. (0.989) *Human Life International URL:www.hli.org*	6. (0.630) Bible.com Prayer Room www.bibleprayerroom.com	6. (0.807) **Pregnancy Centers Online URL:www.pregnancycenters.org**
7. (0.987) *Abortion—Breast Cancer Lin URL:www.abortioncancer.com*	7. (0.609) *United States Department of URL:www.dhhs.gov*	7. (0.740) *ProLifeInfo.org URL:www.prolifeinfo.org*
8. (0.985) **Abortion facts and informati URL:www.abortionfacts.com**	8. (0.538) **Pregnancy Centers Online URL:www.pregnancycenters.org**	8. (0.734) **After Abortion: Information URL:www.afterabortion.org**
9. (0.981) *Campaign Life Coalition Brit URL:www.clcbc.org*	9. (0.517) Bible.com Online World bible.com	9. (0.672) **Abortion Clinics OnLineURL:www.gynpages.com**
10. (0.975) Empty title field www.heritagehouse76.com	10. (0.516) *National Organization for Wo URL:www.now.org*	10. (0.625) *Abortion—Breast Cancer Lin URL:www.abortioncancer.com*

HubAvg	Max	AT-med
1. (1.000) *prochoiceamerica.org : NARAL URL:www.naral.org*	1. (1.000) *prochoiceamerica.org : NARAL URL:www.naral.org*	1. (1.000) *prochoiceamerica.org : NARAL URL:www.naral.org*
2. (0.935) *Planned Parenthood Federatio URL:www.plannedparenthood.org*	2. (0.946) *Planned Parenthood Federatio URL:www.plannedparenthood.org*	2. (0.933) *Planned Parenthood Federatio URL:www.plannedparenthood.org*
3. (0.921) **NAF—The Voice of Abortion URL:www.prochoice.org**	3. (0.918) *National Right to Life URL:www.nrlc.org*	3. (0.837) **NAF—The Voice of Abortion URL:www.prochoice.org**
4. (0.625) **Abortion Clinics OnLine URL:www.gynpages.com**	4. (0.819) **NAF—The Voice of Abortion URL:www.prochoice.org**	4. (0.717) *National Right to Life URL:www.nrlc.org*
5. (0.516) *FEMINIST MAJORITY FOUNDATION URL:www.feminist.org*	5. (0.676) *Priests for Life Index URL:www.priestsforlife.org*	5. (0.552) *FEMINIST MAJORITY FOUNDATION URL:www.feminist.org*
6. (0.484) *The Alan Guttmacher Institut URL:www.guttmacher.org*	6. (0.624) **Pregnancy Centers OnlineURL:www.pregnancycenters.org**	6. (0.545) **Abortion Clinics OnLineURL:www.gynpages.com**
7. (0.439) **center for reproductive righ URL:www.crlp.org**	7. (0.602) *ProLifeInfo.org URL:www.prolifeinfo.org*	7. (0.538) *The Alan Guttmacher Institut URL:www.guttmacher.org*
8. (0.416) *The Religious Coalition for URL:www.rcrc.org*	8. (0.557) **Abortion Clinics OnLine URL:www.gynpages.com**	8. (0.523) **center for reproductive righ URL:www.crlp.org**
9. (0.415) *National Organization for Wo URL:www.now.org*	9. (0.551) **After Abortion: Information URL:www.afterabortion.org**	9. (0.518) *Priests for Life Index URL:www.priestsforlife.org*
10. (0.408) *Medical Students for Choice: URL:www.ms4c.org*	10. (0.533) *FEMINIST MAJORITY FOUNDATION URL:www.feminist.org*	10. (0.478) *The Religious Coalition for URL:www.rcrc.org*

Table VIII. Continue

AT-avg	BFS	Bayesian
1. (1.000) *National Right to Life URL:www.nrlc.org*	1. (1.000) *National Right to Life URL:www.nrlc.org*	1. (7.04) *National Right to Life URL:www.nrlc.org*
2. (0.905) *Priests for Life Index URL:www.priestsforlife.org*	2. (0.930) *Priests for Life Index URL:www.priestsforlife.org*	2. (6.79) *Priests for Life Index URL:www.priestsforlife.org*
3. (0.844) *ProLifeInfo.org URL:www.prolifeinfo.org*	3. (0.928) **After Abortion: Information URL:www. afterabortion.org**	3. (6.31) *ProLifeInfo.org URL:www.prolifeinfo.org*
4. (0.785) **Pregnancy Centers Online URL:www. pregnancycenters.org**	4. (0.905) *Pro-life news and informatio URL:www.all.org*	4. (6.23) **Pregnancy Centers Online URL:www. pregnancycenters.org**
5. (0.778) **After Abortion: Information URL:www. afterabortion.org**	5. (0.893) *ProLifeInfo.org URL:www.prolifeinfo.org*	5. (6.09) **After Abortion: Information URL:www. afterabortion.org**
6. (0.777) *prochoiceamerica.org : NARAL URL:www.naral.org*	6. (0.869) **Pregnancy Centers Online URL:www. pregnancycenters.org**	6. (5.67) *Human Life International URL:www.hli.org*
7. (0.741) *Human Life International URL:www.hli.org*	7. (0.860) *Human Life International URL:www.hli.org*	7. (5.40) *Abortion—Breast Cancer Lin URL:www.abortioncancer.com*
8. (0.704) *Planned Parenthood Federatio URL:www. plannedparenthood.org*	8. (0.852) **Abortion facts and informati URL:www. abortionfacts.com**	8. (5.35) **Abortion facts and informati URL:www. abortionfacts.com**
9. (0.683) **Abortion facts and informati URL:www. abortionfacts.com**	9. (0.847) *prochoiceamerica.org : NARAL URL:www.naral.org*	9. (5.08) *Campaign Life Coalition Brit URL:www.clcbc.org*
10. (0.677) *Abortion—Breast Cancer Lin URL:www. abortioncancer.com*	10. (0.839) *Planned Parenthood Federatio URL:www. plannedparenthood.org*	10. (4.36) Empty title field www.heritagehouse76.com

Table IX. Query “Amusement Parks”

Hits	PageRank	InDegree
1. (1.000) *Welcome to RENOLDI rides am URL:www.renoldi.com*	1. (1.000) HONcode: Principles www.hon.ch/HONcode/Conduct.htm	1. (1.000) HONcode: Principles www.hon.ch/HONcode/Conduct.htm
2. (0.933) **IAAPA URL:www. iaapa.org**	2. (0.495) abc,Inflatable,tmoonwalk, moon www.adventure- bounce.com	2. (0.645) **Empty title field URL:www.sixflags.com**
3. (0.834) **Knott’s URL:www. knotts.com**	3. (0.335) Local Business Listings Beac business.beachcomberii.com	3. (0.589) **Busch Gardens Adventure Park URL:www. buschgardens.com**
4. (0.799) *Traditional Amusement parks URL:www.tradition.cjb.net*	4. (0.332) *NAARSO National Association URL:www.naarso.com*	4. (0.567) **IAAPA URL:www. iaapa.org**
5. (0.788) **HUSS URL:www. hussrides.com**	5. (0.332) AttorneyPages Helps You Find attorneypages.com	5. (0.560) Free Legal Advice in 100+ La freeadvice.com
6. (0.779) *Empty title field URL:www.aimsintl.org*	6. (0.332) Do It Yourself Home Improvem doityourself.com	6. (0.560) AttorneyPages Helps You Find attorneypages.com
7. (0.772) **Screamscape URL:www.screamscape.com**	7. (0.321) *Theme Parks Classifieds URL:adlistings.themeparks.about.co*	7. (0.560) Do It Yourself Home Improvem doityourself.com
8. (0.770) **REVERCHON : HOME PAGE URL:www.reverchon. com**	8. (0.317) FreeFind Site Search search.freefind.com/find.html?	8. (0.532) ExpertPages.com—Books, Tap expert-pages.com/books.htm
9. (0.769) *Empty title field URL:www.zierer.com*	9. (0.315) Free Legal Advice in 100+ La freeadvice.com	9. (0.489) Empty title field imgserv.adbutler.com/go2/;ID=1
10. (0.767) *DE URL:www.drewexpo.com*	10. (0.303) **IAAPA URL:www.iaapa.org**	10. (0.489) Empty title field imgserv.adbutler.com/go2/;ID=1

Table IX. Continue

HubAvg	Max	AT-med
1. (1.000) HONcode: Principles www.hon.ch/HONcode/Conduct.htm	1. (1.000) HONcode: Principles www.hon.ch/HONcode/Conduct.htm	1. (1.000) AttorneyPages Helps You Find attorneypages.com
2. (0.000) AttorneyPages Helps You Find attorneypages.com	2. (0.000) **Empty title field URL:www.sixflags.com**	2. (1.000) Do It Yourself Home Improvem doityourself.com
3. (0.000) Do It Yourself Home Improvem doityourself.com	3. (0.000) **Busch Gardens Adventure Park URL:www. buschgardens.com**	3. (0.987) Free Legal Advice in 100+ La freeadvice.com
4. (0.000) Free Legal Advice in 100+ La freeadvice.com	4. (0.000) **Cedar Point Amusement Park, URL:www. cedarpoint.com**	4. (0.949) ExpertPages.com—Books, Tap expert-pages.com/books.htm
5. (0.000) ExpertPages.com—Books, Tap expert-pages.com/books.htm	5. (0.000) **IAAPA URL:www. iaapa.org**	5. (0.866) Accidents Happen-Why are L law.freeadvice.com/resources/c
6. (0.000) Accidents Happen-Why are L law.freeadvice.com/resources/c	6. (0.000) **Knott's URL:www. knotts.com**	6. (0.032) Expert Witness Directory| −F expertpages.com
7. (0.000) Empty title field imgserv. adbutler.com/go2/;ID=1	7. (0.000) **Universal Studios URL:www.usf.com**	7. (0.017) Get Your Discount Card Today www.usaphonetime.com
8. (0.000) Empty title field imgserv. adbutler.com/go2/;ID=1	8. (0.000) *Welcome to RENOLDI rides am URL:www.renoldi.com*	8. (0.016) MapQuest: Home www.mapquest.com
9. (0.000) Site Meter—Counter and Sta s10.sitemeter.com/stats.asp?si	9. (0.000) **Kennywood : America's Finest URL:www. kennywood.com**	9. (0.014) Adventure travel outdoor r www.outsidemag.com
10. (0.000) Empty title field imgserv.adbutler.com/go2/;ID=1	10. (0.000) *Exhibits Collection – Amuse URL:www.learner.org/ exhibits/parkp*	10. (0.013) **Disneyland Resort—The offi URL:www. disneyland.com**

AT-avg	BFS	Bayesian
1. (1.000) AttorneyPages Helps You Find attorneypages.com	1. (1.000) **Knott's URL:www. knotts.com**	1. (1.88) **Empty title field URL:www.sixflags.com**
2. (1.000) Do It Yourself Home Improvem doityourself.com	2. (0.910) *Welcome to Gillette Shows URL:www.gilletteshows.biz*	2. (1.71) *Welcome to RENOLDI rides am URL:www.renoldi.com*
3. (0.995) Free Legal Advice in 100+ La freeadvice.com	3. (0.885) *Welcome to RENOLDI rides am URL:www.renoldi.com*	3. (1.62) **Busch Gardens Adventure Park URL:www. buschgardens.com**
4. (0.973) ExpertPages.com—Books, Tap expert-pages.com/books.htm	4. (0.884) **IAAPA URL:www. iaapa.org**	4. (1.60) **Knott's URL:www. knotts.com**
AT-avg	BFS	Bayesian
5. (0.900) Accidents Happen—Why are L law.freeadvice.com/resources/c	5. (0.881) *e-musementparkstore. com—Am URL:www.e-musementparkstore.com*	5. (1.59) **Cedar Point Amusement Park, URL:www.cedarpoint. com**
6. (0.016) Expert Witness Directory| −F expertpages.com	6. (0.879) *Great Adventure Source URL:greatadventure.8m.com*	6. (1.52) **Kennywood : America's Finest URL:www.kennywood. com**
7. (0.012) Get Your Discount Card Today www.usaphonetime.com	7. (0.868) Web Page Under Construction www.carousel.org	7. (1.52) **IAAPA URL:www.iaapa. org**
8. (0.008) The Expert Pages—About Adv expertpages.com/about.htm	8. (0.854) amutech amutech.homestead.com	8. (1.46) **Universal Studios URL:www.usf.com**
9. (0.008) Terms amp; Conditions at Ex expertpages.com/conditions.htm	9. (0.850) **Joyrides—Amusement Park an URL:www.joyrides.com**	9. (1.44) **Beachboardwalk.com: Californ URL:www. beachboardwalk.com**
10. (0.008) Expert Pages Privacy Policy expertpages.com/privacy.htm	10. (0.849) **Pharaohs Lost Kingdom Advent URL:www. pharaohslostkingdom.com**	10. (1.35) **LEGO.com Plug-in Download URL:www. legolandca.com**

Table X. Query “Classical Guitar”

HITS	PAGERANK	INDEGREE
1. (1.000) **earlyromanticguitar.com URL:www.earlyromanticguitar.com**	1. (1.000) Take Note Publishing Limited www.takenote.co.uk	1. (1.000) Guitar Alive-GuitarAlive—www.guitaralive.com
2. (0.927) **Empty title field URL: classicalguitar.freehosting.ne**	2. (0.724) *Guitar music, guitar books a URL:www.booksforguitar.com*	2. (0.895) **earlyromanticguitar.com URL:www.earlyromanticguitar.com**
3. (0.889) *Adirondack Spruce.com URL:adirondackspruce.com*	3. (0.588) **Registry of Guitar Tutors URL:www.registryofguitartutors.com**	3. (0.889) **Empty title field URL:www.guitarfoundation.org**
4. (0.766) *The Classical Guitar Homepag URL:www.ak-c.demon.nl*	4. (0.528) Guitar Alive-GuitarAlive—www.guitaralive.com	4. (0.882) *Hitsquad.com-Musicians Web URL:www.hitsquad.com*
5. (0.732) Guitar Alive-GuitarAlive—www.guitaralive.com	5. (0.416) *Hitsquad.com-Musicians Web URL:www.hitsquad.com*	5. (0.850) Hitsquad Privacy Policy www.hitsquad.com/privacy.shtml
6. (0.681) **Empty title field URL:www.guitarfoundation.org**	6. (0.413) Hitsquad Privacy Policy www.hitsquad.com/privacy.shtml	6. (0.850) Advertising on Hitsquad Musi www.hitsquad.com/advertising.s
7. (0.676) *GUITAR REVIEW URL:www.guitarreview.com*	7. (0.413) Advertising on Hitsquad Musi www.hitsquad.com/advertising.s	7. (0.784) *Adirondack Spruce.com URL:adirondackspruce.com*
8. (0.644) *Avi Afriat—Classical guita URL:afriat.tripod.com*	8. (0.387) **Empty title field URL: www.guitarfoundation.org**	8. (0.778) **Empty title field URL: classicalguitar.freehosting.ne**
9. (0.605) **The Classical Guitar Home Pa URL:www.guitarist.com/cg/cg.htm**	9. (0.343) **Guitar Foundation of America URL:64.78.54.231**	9. (0.765) *Empty title field URL: www.vicnet.net.au/simieasyjamn*
10. (0.586) *Empty title field URL:www.duolenz.com*	10. (0.322) Vivendi Universal www.vivendiuniversal.com	10. (0.739) *The Classical Guitar Homepag URL:www.ak-c.demon.nl*

HUBAVG	MAX	AT-MED
1. (1.000) *Hitsquad.com-Musicians Web URL:www.hitsquad.com*	1. (1.000) Guitar Alive-GuitarAlive—www.guitaralive.com	1. (1.000) *Hitsquad.com-Musicians Web URL:www.hitsquad.com*
2. (0.995) Hitsquad Privacy Policy www.hitsquad.com/privacy.shtml	2. (0.619) **earlyromanticguitar.com URL:www.earlyromanticguitar.com**	2. (0.983) Hitsquad Privacy Policy www.hitsquad.com/privacy.shtml
3. (0.995) Advertising on Hitsquad Musi www.hitsquad.com/advertising.s	3. (0.506) **Empty title field URL:www.guitarfoundation.org**	3. (0.983) Advertising on Hitsquad Musi www.hitsquad.com/advertising.s
4. (0.856) *Empty title field URL: www.vicnet.net.au/simieasyjamn*	4. (0.451) *Adirondack Spruce.com URL:adirondackspruce.com*	4. (0.880) *Empty title field URL: www.vicnet.net.au/simieasyjamn*
5. (0.135) *AMG All Music Guide URL:www.allmusic.com*	5. (0.441) **Empty title field URL: classicalguitar.freehosting.ne**	5. (0.205) *AMG All Music Guide URL:www.allmusic.com*
6. (0.132) Free Music Download, MP3 Mus ubl.com	6. (0.378) *GUITAR REVIEW URL:www.guitarreview.com*	6. (0.193) Free Music Download, MP3 Mus ubl.com
7. (0.130) *2000 Guitars Database URL:dargo.vicnet.net.au/guitar/lis*	7. (0.377) *The Classical Guitar Homepag URL:www.ak-c.demon.nl*	7. (0.169) *2000 Guitars Database URL:dargo.vicnet.net.au/guitar/lis*
8. (0.115) Guitar Alive-GuitarAlive—www.guitaralive.com	8. (0.371) **The Classical Guitar Home Pa URL:www.guitarist.com/cg/cg.htm**	8. (0.132) Guitar Alive-GuitarAlive—www.guitaralive.com
9. (0.096) CDNOW www.cdnow.com/from=sr-767167	9. (0.336) *Hitsquad.com-Musicians Web URL:www.hitsquad.com*	9. (0.129) CDNOW www.cdnow.com/from=sr-767167
10. (0.056) *OLGA—The On-Line Guitar Ar URL:www.olga.net*	10. (0.312) Hitsquad Privacy Policy www.hitsquad.com/privacy.shtml	10. (0.082) *OLGA—The On-Line Guitar Ar URL:www.olga.net*

AT-AVG	BFS	BAYESIAN
1. (1.000) *Hitsquad.com-Musicians Web URL:www.hitsquad.com*	1. (1.000) **Empty title field URL: classicalguitar.freehosting.ne**	1. (3.66) **earlyromanticguitar.com URL:www.earlyromanticguitar.com**
2. (0.986) Hitsquad Privacy Policy www.hitsquad.com/privacy.shtml	2. (0.991) **earlyromanticguitar.com URL: www.earlyromanticguitar.com**	2. (3.54) **Empty title field URL: classicalguitar.freehosting.ne**

Table X. Continue

3. (0.986) Advertising on Hitsquad Musi www.hitsquad.com/ advertising.s	3. (0.974) *Adirondack Spruce.com URL:adirondackspruce.com*	3. (3.53) *Adirondack Spruce.com URL:adirondackspruce.com*
4. (0.906) *Empty title field URL: www.vicnet.net.au/simieasyjamn*	4. (0.962) **Empty title field URL:www.guitarfoundation.org**	4. (3.42) *Hitsquad.com—Musicians Web URL:www.hitsquad.com*
5. (0.210) *AMG All Music Guide URL:www.allmusic.com*	5. (0.945) Guitar Alive-GuitarAlive—www.guitaralive.com	5. (3.38) Guitar Alive-GuitarAlive—www.guitaralive.com
6. (0.199) Free Music Download, MP3 Mus ubl.com	6. (0.933) *The Classical Guitar Homepag URL:www.ak-c.demon.nl*	6. (3.33) **Empty title field URL:www.guitarfoundation.org**
7. (0.179) *2000 Guitars Database URL:dargo.vicnet.net.au/guitar/lis*	7. (0.917) **The Classical Guitar Home Pa URL:www.guitarist.com/cg/cg.htm**	7. (3.33) Hitsquad Privacy Policy www.hitsquad.com/privacy.shtml
8. (0.135) CDNOW www.cdnow.com/from=sr-767167	8. (0.898) *Avi Afriat—Classical guita URL:afriat.tripod.com*	8. (3.33) Advertising on Hitsquad Musi www.hitsquad.com/ advertising.s
9. (0.122) Guitar Alive-GuitarAlive—www.guitaralive.com	9. (0.889) *GUITAR REVIEW URL:www.guitarreview.com*	9. (3.27) *The Classical Guitar Homepag URL:www.ak-c.demon.nl*
10. (0.080) *OLGA—The On-Line Guitar Ar URL:www.olga.net*	10. (0.881) *Empty title field URL:www.duolenz.com*	10. (3.25) *GUITAR REVIEW URL:www.guitarreview.com*

ACKNOWLEDGMENTS

We would like to thank Alberto Mendelzon, Renee Miller, Ken Sevcik, Ken Jackson, Jon Kleinberg, Sam Roweis, Ronald Fagin, Ronny Lempel, Shlomo Moran, and the anonymous reviewers for valuable comments and corrections. We would also like to thank all the people who participated in the online survey for determining the relevance of the search results.

REFERENCES

Achlioptas, D., Fiat, A., Karlin, A., and McSherry, F. 2001. Web search through hub synthesis. In *Proceedings of the 42nd Foundation of Computer Science (FOCS 2001)*. Las Vegas, NY.

Andritsos, P., Tsaparas, P., Miller, R., and Sevcik, K. 2003. LIMBO: Scalable clustering of categorical data. Submitted for publication.

Azar, Y., Fiat, A., Karlin, A., McSherry, F., and Saia, J. 2001. Spectral analysis of data. In *Proceedings of the 33rd Symposium on Theory of Computing (STOC 2001)*. Hersonissos, Crete, Greece.

Bernardo, J. and Smith, A. 1994. *Bayesian Theory*. John Wiley & Sons, Chichester, England.

Bharat, K. and Henzinger, M. R. 1998. Improved algorithms for topic distillation in a hyperlinked environment. In *Research and Development in Information Retrieval*. 104–111.

Bharat, K. and Mihaila, G. A. 2001. When experts agree: Using non-affiliated experts to rank popular topics. In *Proceedings of the 10th International World Wide Web Conference*.

Bianchini, M., Gori, M., and Scarselli, F. 2002. PageRank: A circuital analysis. In *Proceedings of the 11th International Word Wide Web Conference (WWW 2002)*. Poster Session. Hawai.

Borodin, A., Roberts, G. O., Rosenthal, J. S., and Tsaparas, P. 2001. Finding authorities and hubs from link structures on the World Wide Web. In *Proceedings of the 10th International World Wide Web Conference*. Hong Kong.

Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International World Wide Web Conference*. Brisbane, Australia.

Broder, A. 2002. Web searching technology overview. In *Advanced School and Workshop on Models and Algorithms for the World Wide Web*. Udine, Italy.

Chakrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Raghavan, P., and Rajagopalan, S. 1998. Automatic resource compilation by analysing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*.

Chien, S., Dwork, C., Kumar, R., Simon, D., and Sivakumar, D. 2002. Towards exploiting link evolution. In *Workshop on Algorithms for the Web*. Vancouver, Canada.

Cohn, D. and Chang, H. 2000. Learning to probabilistically identify authoritative documents. In *Proceedings of the 17th International Conference on Machine Learning*. Stanford University, 167–174.

Davison, B. 2000. Recognizing nepotistic links on the web. In *AAAI-2000 Workshop on Artificial Intelligence for Web Search*. AAAI Press, Austin, TX.

Dempster, A., Laird, N., and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc., Series B*, *39*, 1–38.

Diaconis, P. and Graham, R. 1977. Spearman's footrule as a measure of disarray. *J. Roy. Statist. Soc. 39*, 2, 262–268.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. 1999. Clustering in large graphs and matrices. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. 2001. Rank aggregation methods for the Web. In *Proceedings of the 10th International World Wide Web Conference*. Hong Kong.

Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., and Vee, E. 2004. Comparing and aggregating rankings with ties. In *Symposium on Principles of Database Systems (PODS)*.

Fagin, R., Kumar, R., and Sivakumar, D. 2003. Comparing top k lists. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Gibson, D., Kleinberg, J., and Raghavan, P. 1998. Inferring Web communities from link topology. In *Proceedings of 9th ACM Conference on Hypertext and Hypermedia*.

Gilks, W., Richardson, S., and Spiegelhalter, D. 1996. *Markov Chain Monte Carlo in practice*. Chapman and Hall, London.

Haveliwala, T. H. 2002. Topic sensitive Page Rank. In *Proceedings of the 11th International Word Wide Web Conference (WWW 2002)*. Hawai.

Hofmann, T. 1999. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence, UAI'99*. Stockholm, Sweden.

Hofmann, T. 2000. Learning probabilistic models of the Web. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval (ACM SIGIR'00)*.

Jansen, B. J., Spink, A., Bateman, J., and Saracevic, T. 1998. Real Life Information Retrieval: A Study of User Queries on the Web. *ACM SIGIR Forum 32*, 5–17.

Jen, G. and Widom, J. 2003. Scaling personalized Web search. In *Proceedings of the 12th International World Wide Wed Conference(WWW2003)*. Budapest, Hungary.

Katz, L. 1953. A new status index derived from sociometric analysis. *Psychometrika 18*, 39–43.

Kendall, M. G. 1970. *Rank Correlation Methods*. Griffin, London, UK.

Kleinberg, J. 1998. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. 668–677.

Kleinberg, J. 1999. Authoritative sources in a hyperlinked environment. *J. ACM 46*.

Lee, H. C. and Borodin, A. 2003. Perturbation of the hyperlinked environment. In *Proceedings of the 9th International Computing and Combinatorics Conference*.

Lempel, R. and Moran, S. 2000. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proceedings of the 9th International World Wide Web Conference*.

Lempel, R. and Moran, S. 2001. Rank stability and rank similarity of Web link-based ranking algorithms. Tech. Rep. CS-2001-22. Technion—Israel Institute of Technology.

Lempel, R. and Moran, S. 2003. Rank stability and rank similarity of Web link-based ranking algorithms. In *2nd Workshop on Algorithms and Models for the Web-Graph (WAW2003)*. Budapest, Hungary.

Lin, J. 1991. Divergence measures based on the Shannon entropy. *Mach. Learn. 37*, 1, 145–151.

Marchiori, M. 1997. The quest for correct information on Web: Hyper search engines. In *Proceedings of the 6th International World Wide Web Conference*.

Mendelzon, A. and Rafiei, D. 2000. What do the neighbours think? Computing Web page reputations. *IEEE Data Eng. Bull. 23*, 3, 9–16.

Ng, A. Y., Zheng, A. X., and Jordan, M. I. 2001a. Link analysis, eigenvectors, and stability. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Seattle, Washington.

Ng, A. Y., Zheng, A. X., and Jordan, M. I. 2001b. Stable algorithms for link analysis. In *Proceedings of the 24th International Conference on Research and Development in Information Retrieval (SIGIR 2001)*. New York, NY.

PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. 1998. The PageRank citation ranking: Bringing order to the web. Tech. rep. Stanford Digital Library Technologies Project.

RAFIEI, D. AND MENDELZON, A. 2000. What is this page known for? Computing Web page reputations. In *Proceedings of the 9th International World Wide Web Conference*. Amsterdam, Netherlands.

RICHARDSON, M. AND DOMINGOS, P. 2002. The intelligent surfer: Probabilistic combination of link and content information in PageRank. In *Advances in Neural Information Processing Systems (NIPS) 14*.

ROBERTS, G. AND ROSENTHAL, J. 1998. Markov chain Monte Carlo: Some practical implications of theoretical results (with discussion). *Canadian J. Statist. 26*, 5–31.

ROBERTS, G. O. AND ROSENTHAL, J. S. 2003. Downweighting tightly knit communities in World Wide Web rankings. *Adv. Appl. Statist. 3*, 199–216.

SILVERSTEIN, C., HENZINGER, M., MARAIS, H., AND MORICZ, M. 1998. Analysis of a very large AltaVista query log. Tech. Rep. 1998-014. Digital SRC.

SLONIM, N. AND TISHBY, N. 1999. Agglomerative Information Bottleneck. In *Advances in Neural Information Processing Systems (NIPS)*. Breckenridge, CO.

SMITH, A. AND ROBERTS, G. 1993. Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods (with discussion). *J. Roy. Statist. Soc., Series B*, 55, 3–24.

TIERNEY, L. 1994. Markov chains for exploring posterior distributions (with discussion). *Ann. Statist. 22*, 1701–1762.

TISHBY, N., PEREIRA, F. C., AND BIALEK, W. 1999. The Information Bottleneck method. In *37th Annual Allerton Conference on Communication, Control and Computing*. Urban-Champaign, IL.

TOMLIN, J. A. 2003. A new paradigm for ranking pages on the World Wide Web. In *Proceedings of the 12th International World Wide Wed Conference (WWW2003)*. Budapest, Hungary.

TSAPARAS, P. 2004a. Link analysis ranking. Ph.D. thesis, University of Toronto.

TSAPARAS, P. 2004b. Using non-linear dynamical systems for Web searching and ranking. In *Symposium on Principles of Database Systems (PODS)*. Paris, France.

Received November 2001; revised March 2004; accepted April 2004