

Using Non-Linear Dynamical Systems for Web Searching and Ranking

Panayiotis Tsaparas^{*}
Dipartimento di Informatica e Systemistica
Universita di Roma, "La Sapienza"
tsap@dis.uniroma1.it

ABSTRACT

In the recent years there has been a surge of research activity in the area of information retrieval on the World Wide Web, using link analysis of the underlying hypertext graph topology. Most of the algorithms in the literature can be described as dynamical systems, that is, the repetitive application of a function on a set of weights. Algorithms that rely on eigenvector computations, such as HITS and PAGERANK, correspond to linear dynamical systems. In this work we consider two families of link analysis ranking algorithms that no longer enjoy the linearity property of the previous approaches. We study in depth an interesting special case of these two families. We prove that the corresponding non-linear dynamical system converges for any initialization, and we provide a rigorous characterization of the combinatorial properties of the stationary weights. The study of the weights provides a clear and insightful view of the mechanics of the algorithm. We also present extensive experimental results that demonstrate that our algorithm performs well in practice.

1. INTRODUCTION

Ranking is an integral component of any information retrieval system. In the case of Web search the role of ranking becomes even more important. Due to the size of the Web, and the impatient nature of Web users, it is imperative to have ranking functions that capture the user needs, and output the desired documents within the first few pages of results. To this end the Web offers a rich context of information which is expressed through the hyperlinks. Intuitively a link from page p to page q denotes an endorsement for the quality of page q . The seminal papers of Kleinberg [15], and Brin and Page [6] built upon this idea, and introduced the area of *Link Analysis Ranking*, where hyperlink structures are used to determine the relative *authority* of Web pages.

A Link Analysis Ranking (LAR) algorithm starts with a set of Web pages interconnected with hypertext links. It takes as input the underlying hyperlink graph, and returns as output an *authority*

^{*}A major part of this work was completed while the author was a graduate student at University of Toronto.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODS 2004 June 14-16, 2004, Paris, France.

Copyright 2004 ACM 1-58113-858-X/04/06...\$5.00.

weight for every node in the graph, that captures the authoritative-ness of the node. Most LAR algorithms can be defined as *discrete dynamical systems*. A dynamical system assigns an initial weight to each node, and then performs a weight-propagation scheme on the graph. The dynamical system iteratively updates the node weights, setting the new weights to be a function of the current weights, until the weights converge. The output of the algorithm is the stationary weights of the dynamical system. When the update function is linear, we have a *linear dynamical system*. Algorithms that depend on eigenvector computations [15, 6, 3, 16, 18, 17, 1] correspond to linear dynamical systems.

In this paper we work within the hubs and authorities framework introduced by Kleinberg [15]. We identify some potential weaknesses of the HITS algorithm, and we consider two families of algorithms that use alternative ways of computing hub and authority weights. A characteristic of the new families of algorithms is that they apply non-linear operators. The corresponding dynamical systems are non-linear. Non-linear systems are less popular, since the mathematical tools for analyzing their properties are not as well developed as in the case of linear systems. We study in detail the MAX algorithm, a special case of both families, and we prove that the corresponding dynamical system has good behavior, that is, it converges for any initialization. We also study the combinatorial properties of the stationary weights. The study provides valuable insight to the algorithm, and reveals a clear and structured mechanism for assigning weights. The MAX algorithm has been previously considered by Gibson, Kleinberg and Raghavan [11] for clustering categorical data. Our work resolves an open question raised in [11] regarding the rigorous analysis of this dynamical system.

We also present extensive experiments of our algorithm both for ranking and for finding related pages. The results indicate that our algorithm performs well, and in many cases outperforms other link analysis approaches.

The rest of this paper is structured as follows. Section 2 reviews some of the related work. In Section 3 we introduce the two (non-linear) families of algorithms, and we define the MAX algorithm. Section 4 presents the main concepts about dynamical systems. In Section 5 we study the convergence of the MAX algorithm, and the properties of the stationary configuration. Section 6 presents an experimental evaluation of various algorithms for ranking and finding related pages. Section 7 concludes the paper with some directions for future work.

2. BACKGROUND

A Link Analysis Ranking algorithm starts with a set of web pages, P , interconnected with hyperlinks. In this work we will assume that the set of pages is *query dependent*, that is, it depends

on a specific Web query. Kleinberg [15] describes a process for obtaining such a set of pages. Let n denote the size of the set P . The input to the link analysis algorithm is the $n \times n$ adjacency matrix W of the underlying *hyperlink 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 \mathbf{a} , where a_i , the i -th coordinate of the vector \mathbf{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 non-zero in-degree, and a *hub node* in the graph G to be a node with non-zero out degree. A node can be both a hub and an authority node. We use A to denote the set of authority nodes, and H to denote the set of hub nodes. We have that $P = A \cup H$.

Our work builds upon the hubs and authorities framework introduced by Kleinberg. 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 every page with 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 \mathbf{h} denote the n -dimensional vector of the hub weights, where h_i , the i -th coordinate of vector \mathbf{h} , is the hub weight of node i . We have that

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

Kleinberg 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 \mathbf{a} and \mathbf{h} become unit vectors in some norm. The algorithm iterates until the vectors converge. Kleinberg proves that after a sufficient number of iterations the vectors \mathbf{a} and \mathbf{h} converge to the principal eigenvectors of the matrices $W^T W$ and $W W^T$, respectively. This idea was later implemented as the HITS (Hyperlink Induced Topic Distillation) algorithm [12].

Independently, about the same time, Brin and Page introduced the PAGERANK algorithm [6], which later became an integral component of the Google¹ search engine. The PAGERANK algorithm assumes the random surfer model, where a user is following links on a graph, while at some points she performs a jump to a random page. The algorithm performs a random walk which proceeds at each step as follows. With probability ϵ it jumps to a page chosen uniformly at random, and with probability $1 - \epsilon$ it jumps uniformly at random to one of the pages linked from the current page. The authority weight of node i (the PageRank value of node i) is defined as the limiting fraction of time spent on page i by the random walk.

¹<http://www.google.com>

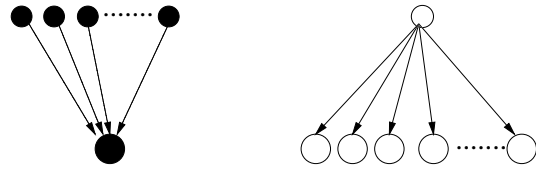


Figure 1: A bad example for the HITS algorithm

The HITS and PAGERANK algorithms were followed by a substantial number of variations and enhancements. Most of the subsequent work follows a similar algebraic approach, manipulating some matrix related to the web graph [5, 3, 16, 18, 2, 1, 17]. Recently, there were some interesting attempts in applying statistical, and machine learning tools for computing authority weights [5, 7, 14].

3. THE TWO FAMILIES OF 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 authority quality of some page p (captured by the authority weight of page p) depends on the hub quality of the pages that point to p (captured in the hub weight of the pages), and vice versa. Kleinberg proposes to associate the hub and authority weights through the addition operation. This definition has the following two implicit properties. It is *symmetric*, in the sense that both hub and authority weights are defined in the same way, and it is also *egalitarian*, in the sense that when computing the hub weight of 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).

However, these two properties may some times lead to non-intuitive results. Consider for example the graph in Figure 1. 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. If the number of white authorities is larger than the number of black hubs then 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 this non-intuitive result. Equality means that all authority weights of the nodes that are pointed to by a hub contribute equally to the hub weight of that node. As a result, quantity becomes quality. The hub weight of the white hub increases inordinately because it points to many weak authorities. This leads us to question the definition of the hub weight, and consequently the other implicit property 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 in a different manner.

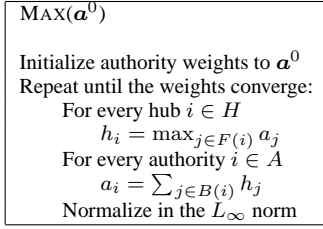


Figure 2: The MAX Algorithm

3.1 The Authority Threshold (AT(k)) family of algorithms

We would like to reduce the effect of the weak authorities on the computation of the hub weight, while at the same time we retain the positive effect of the strong authorities. A simple solution is to apply a threshold operator, that retains only the highest authority weights. Borodin et al. [5] proposed the *Authority-Threshold*, AT(k), algorithm which sets the hub weight of node i to be the sum of the k largest authority weights of the authorities pointed to by node i . This corresponds 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, 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.

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

It is interesting to examine what happens at the extreme values of k . For $k = 1$, the threshold operator becomes the max operator. We will discuss this case in detail in Section 3.3. If d_{out} is the maximum out-degree in the graph G , then for $k \geq d_{out}$, the AT(k) algorithm is the HITS algorithm.

3.2 The NORM(p) Family of Algorithms

The Authority Threshold algorithm operates on the principle of *preferential treatment* of the authority weights. That is, higher authority weights should be more important in the computation of the hub weight. This principle is enforced by applying a threshold operator. A smoother approach is to *scale* the weights, so that lower authority weights contribute less to the hub weight. An obvious question is how to select the scaling factors. A natural solution is to use the weights themselves for the scaling factors.

This idea is implemented in the NORM(p) family of algorithms. In this case we set the hub weight of node i to be the p -norm of the vector of the authority weights of the nodes pointed to by node i . Recall that the p -norm of vector $\mathbf{x} = (x_1, \dots, x_n)$ is defined as $\|\mathbf{x}\|_p = (\sum_{i=1}^n x_i^p)^{1/p}$. The authority and hub weights are computed as follows:

$$a_i = \sum_{j \in B(i)} h_j \quad \text{and} \quad h_i = \left(\sum_{j \in F(i)} a_j^p \right)^{1/p}.$$

The value of p is passed as a parameter to the algorithm. We assume that $p \in [1, \infty]$. As p increases the value of the p -norm is dominated by the highest weights. For example, for $p = 2$, we essentially scale every weight with itself. An almost identical algorithm was

proposed by Gibson, Kleinberg and Raghavan [11] for clustering categorical data.

Again, it is interesting to examine the behavior of the algorithm in the extreme cases of the value p . For $p = 1$ the NORM(1) algorithm is the HITS algorithm. For $p = \infty$ the p -norm reduces to the max operator.

3.3 The MAX algorithm

The MAX algorithm is a special case of both the AT(k) algorithm for the threshold value $k = 1$, and the NORM(p) algorithm for the value $p = \infty$. 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.

Formally, we define the MAX algorithm as follows. The algorithm sets the hub weight of node i to be the maximum authority weight over all authority weights of the nodes pointed to by node i . The authority weights are computed as in the HITS algorithm. Therefore, the authority and hub weights as computed as follows.

$$a_i = \sum_{j \in B(i)} h_j \quad \text{and} \quad h_i = \max_{j \in F(i)} a_j.$$

The outline of the algorithm is shown in Figure 2. We set the normalization norm to be the *max* (or infinity) norm. This makes the analysis of MAX easier, but it does not affect the convergence, and the combinatorial properties of the stationary configuration.

4. DYNAMICAL SYSTEMS

A discrete dynamical system is defined [9] as a process that starts with an n -dimensional real vector, and repeatedly applies a function $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$. We define a *configuration* of the system as any intermediate value of the vector. The initial assignment of values is called the *initial configuration* of the dynamical system. Of particular interest are the *fixed configurations* (or fixed *points*) of the dynamical system. These are vectors \mathbf{x} , such that $g(\mathbf{x}) = \mathbf{x}$. We will also refer to these vectors as the *stationary configurations* of the dynamical system. An interesting question in dynamical systems is the limiting value of the dynamical system. That is, if $g^t(\mathbf{x})$ denotes the t -th iteration of the function g , then we are interested in understanding the limiting behavior of $g^t(\mathbf{x})$, as $t \rightarrow \infty$, for different initial values of \mathbf{x} . The critical question is whether the system reaches a fixed configuration (converges) as $t \rightarrow \infty$, and whether the stationary configuration depends on the initialization. For an introduction to dynamical systems, we refer the reader to the texts by Denavey [9], and Sandefur [19].

In the case of link analysis algorithms, the real vector is the authority weight vector, and the function g propagates the authority weight in the graph G . Let \mathbf{a}^t denote the authority weight vector after t iterations of the algorithm. The outline of a dynamical system for link analysis ranking is shown in Figure 3. Obviously, in order for the LAR algorithm to be well defined, we need some guarantees about the convergence of the dynamical system. Ideally, the dynamical system should always converge, and the stationary configuration should not depend on the initial configuration.

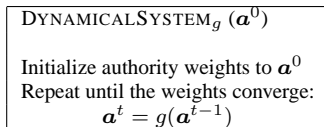


Figure 3: The outline of a dynamical system

Depending on the function g we distinguish between two types of dynamical systems: *linear* and *non-linear*. In linear dynamical systems, the function g is of the form $g(x) = Mx$, where M is an $n \times n$ matrix. The majority of the link analysis algorithms that have appeared so far in the literature [15, 6, 16, 18, 5, 1, 17] can be described as linear dynamical systems. For such systems, linear algebra offers the tools to analyze the limiting behavior of the system. The conditions for convergence and the combinatorial properties of the stationary weights are well understood. Things become more complicated when the function g is non-linear. Outside of the well understood world of linear algebra, we know very little about the behavior of dynamical systems. For non-linear systems, such as the AT(k) and NORM(p) families of algorithms we do not even know if they converge, which is a basic requirement for a well defined Link Analysis Ranking algorithm.

In the following we study in detail the MAX algorithm. This algorithm was previously explicitly considered by Gibson, Kleinberg and Raghavan [11], but it was not rigorously analyzed. We prove that the algorithm converges, and we characterize the combinatorial properties of the stationary configuration.

5. ANALYSIS OF THE MAX ALGORITHM

We will now introduce some of the terminology that we will use for the analysis of the MAX algorithm. We first define a notion of *time*. We define time t to be the moment immediately after the t -th iteration of the algorithm. We denote by \bar{a}_i^t the un-normalized weight of node i at time t . We assume that the weights are normalized in the L_∞ norm. This means that the normalization factor at step t is the maximum un-normalized authority weight, and the maximum normalized weight is 1. We use a_i^t to denote the normalized weight of node i at time t . When not specified, the weight of node i at time t refers to the normalized weight of node i at time t . We denote $a_i = \lim_{t \rightarrow \infty} a_i^t$, the limit of the weight of node i , as $t \rightarrow \infty$, assuming that the limit exists. When convenient we will use $\bar{a}^t(i)$, $a^t(i)$, and $a(i)$ for the quantities \bar{a}_i^t , a_i^t , and a_i respectively.

We also define the mapping $f^t : H \rightarrow A$, where the hub j is mapped to authority i , if at time t the authority i is the authority with the maximum weight among all the authorities pointed to by hub j . If there are many authorities in $F(j)$ that have the largest weight, we arbitrarily select one of the authorities (e.g., according to some predefined ordering). We use $f(j) = \lim_{t \rightarrow \infty} f^t(j)$ to denote the limit of the mapping function as $t \rightarrow \infty$, assuming again that the limit exists. For an authority i , the un-normalized weight of node i at time t is $\bar{a}_i^t = \sum_{j \in B(i)} a^{t-1}(f^{t-1}(j))$.

Recall that $G = (P, E)$ denotes the underlying hyperlink graph, and A denotes the set of authorities in the graph. Let $G_a = (A, E_a)$ denote the undirected graph, where there exists an undirected edge between two authorities if they share a hub. We will refer to G_a as the *authority graph*. Assume now that the graph G_a consists of k connected components C_1, C_2, \dots, C_k . Let \mathbf{a}^0 be the weight vector of the initial configuration. The weight assigned by configuration \mathbf{a}^0 to component C_i is the sum of weights of all authorities in C_i . We define a *fair* initial configuration as a configuration that assigns non-zero weight to all components in the graph G_a . We will assume that the initial configuration is always fair. If the component C_i is assigned zero weight by the vector \mathbf{a}^0 , then the weights of the nodes in C_i will immediately converge to zero. Thus, we can disregard the nodes in the component C_i and assume that the algorithm operates on a smaller graph \tilde{G} , initialized to a fair configuration $\tilde{\mathbf{a}}^0$.

Finally, for some node $i \in A$, let d_i denote the in-degree of

authority i in the graph G , and let $d = \max\{d_i : i \in A\}$, denote the maximum in-degree of any authority in the graph G . Let $S \subseteq A$ denote the set of nodes with in-degree d . We call these nodes, the *seeds* of the algorithm. Seed nodes play an important role in the MAX algorithm. We define U to be the set of non-seed nodes. Thus, $A = S \cup U$.

5.1 Convergence of the MAX algorithm

In this section we prove that the algorithm converges for any initial configuration. First, we prove that the weights of the seed nodes always converge.

LEMMA 1. *The weight of every seed node $s \in S$ is a non-decreasing function of time.*

PROOF. Consider any seed node $s \in S$, and let $t \geq 0$ be some time in the execution of the algorithm. At iteration $t + 1$, for every hub node j , we have that $h_j = \max\{a_i^t : i \in F(j)\}$. Thus, for every hub $j \in B(s)$, $h_j \geq a_s^t$. Therefore, at time $t + 1$, the un-normalized weight of node s is $\bar{a}_s^{t+1} = \sum_{j \in B(s)} h_j \geq d a_s^t$. Let x be the authority node with maximum un-normalized weight at time $t + 1$. Since the weight of any authority at time t is at most 1, we have that for every hub $j \in B(x)$, $h_j \leq 1$. It follows that $\bar{a}_x^{t+1} = \sum_{j \in B(x)} h_j \leq d_x \leq d$. Therefore, after normalization

$$a_s^{t+1} \geq \frac{d}{\bar{a}_x^{t+1}} a_s^t \geq a_s^t$$

which concludes the proof. \square

COROLLARY 1. *The weight of every seed node $s \in S$ converges for any initial configuration.*

PROOF. For every seed node $s \in S$, the weight of s is a non-decreasing function that is upper-bounded, therefore it will converge. \square

Note that “non-decreasing” means that the weights either increase, or remain constant. We now prove that there always exists a time t_0 when the weight of some seed node takes the maximum value 1, which remains constant for all $t \geq t_0$.

LEMMA 2. *For every initial configuration there exists a point in time t_0 , such that for some seed node $s \in S$, $a_s^t = 1$, for all $t \geq t_0$.*

PROOF. Assume that for every $s \in S$, $a_s^t < 1$, for all $t \geq 0$. We will now prove that for every $s \in S$, $a_s^t \geq a_s^0 d^t / (d-1)^t$ by induction on t . For $t = 0$ it is trivially true. Now, assume that at time t , $a_s^t \geq a_s^0 d^t / (d-1)^t$. At time $t + 1$, we have (as in the proof of Lemma 1) that the un-normalized weight of s is $\bar{a}_s^{t+1} \geq d a_s^t$. Let x be the node with maximum un-normalized weight at time $t + 1$. Node x cannot be a seed node, since then we would have that $a_x^{t+1} = 1$, reaching a contradiction with our initial assumption. Since x is not a seed node, $\bar{a}_x^{t+1} \leq d_x \leq d - 1$. Normalizing the weight of s by \bar{a}_x^{t+1} we have that

$$a_s^{t+1} \geq \frac{d}{(d-1)} a_s^t \geq \frac{d^{t+1}}{(d-1)^{t+1}} a_s^0.$$

Therefore, the weight of every seed node is an increasing function of time. As $t \rightarrow \infty$, $a_s^t \rightarrow \infty$. Since, the weights are bounded, we reach a contradiction. Therefore, there must exist some point in time, t_0 , such that, for some node $s \in S$, $a_s^{t_0} = 1$. From Lemma 1 we know that the weight of the seed nodes is a non-decreasing function, thus, $a_s^t = 1$, for all $t \geq t_0$. Therefore, for this seed node, the weight increases until it becomes 1, and then it remains constant for the remaining iterations. \square

AUX(\mathbf{a}^0, \mathbf{u})

Run the MAX algorithm on \mathbf{a}^0

Let t_0 be the time that the seed nodes converge

$\mathbf{x}_S^0 = \mathbf{a}_S^{t_0} \quad \mathbf{x}_U^0 = \mathbf{u}$

Repeat until the weights converge:

For every hub $i \in H$

$h_i = \max_{j \in F(i)} x_j^t$

For every authority $i \in U$

$x_i^{t+1} = \sum_{j \in B(i)} h_j$

For every authority $s \in S$

$x_s^{t+1} = a_s^{t_0+t+1}$

Normalize in the L_∞ norm

Figure 4: The AUX dynamical system

For the following, given an *accuracy constant* δ , we say that the weight of some node i has converged at time t_i if $|a_i^{t+1} - a_i^t| \leq \delta$, for all $t \geq t_i$.² Corollary 1 and Lemma 2 guarantee that the seed nodes will converge, and at least one of the seeds will converge to weight 1. Let t_0 denote the first time that all seed nodes have converged, and let s be a seed node with weight 1. For $t \geq t_0$, the un-normalized weight of s is d . Furthermore, it is easy to see that for every other authority i , $\bar{a}_i^t \leq \bar{a}_s^t$. Therefore, for all $t \geq t_0$, the normalization factor $\|\bar{\mathbf{a}}^t\|_\infty$ is equal to d , the maximum in-degree of graph G , independent of the vector \mathbf{a}^t .

We are now ready to consider the convergence of the MAX algorithm. The proof proceeds roughly as follows. We first prove that as $t \rightarrow \infty$ the configuration \mathbf{a}^t of the MAX algorithm is independent of the weights of the non-seed nodes at time t_0 , and depends solely on the stationary weights of the seeds. Then, we set the weights of the non-seed nodes to zero at time t_0 and we prove that in this case the system converges. The fact that the configuration is independent of the non-seed weights implies that the system converges for any other configuration of the non-seed nodes, which in turn implies convergence of the MAX algorithm. However, “setting” the weights of the non-seed nodes to zero is not simple to do without disrupting the MAX algorithm. To this end, we need to introduce an auxiliary system AUX.

The system AUX is defined with two parameters. The first is the initial configuration \mathbf{a}^0 of the MAX algorithm. The second is a weight vector \mathbf{u} for the non-seed nodes in U . Given some configuration vector \mathbf{v} , we use \mathbf{v}_S to denote the projection of \mathbf{v} on the seed nodes S , and \mathbf{v}_U to denote the projection of \mathbf{v} on the non-seed nodes U . For the following we use \mathbf{a}^t to denote the weight vector of the MAX algorithm at time t , and \mathbf{x}^t to denote the weight vector of the system AUX at time t . The structure of AUX is given in Figure 4.

The system AUX runs the MAX algorithm with initial configuration \mathbf{a}^0 until time t_0 , when the weights of the seed nodes converge. It then initializes the weights of the seed nodes to $\mathbf{a}_S^{t_0}$, and the weights of the non-seed nodes to \mathbf{u} , and proceeds iteratively as follows. For the t -th iteration, it updates the weights of the non-seed nodes in the regular fashion, while it sets the weights of the seed nodes to the weight they would receive in the $(t_0 + t)$ -th iteration of the MAX algorithm. Essentially, the AUX system fixes the weights of the seed nodes to the stationary weights of the MAX algorithm when run on the initial configuration \mathbf{a}^0 , while it updates the weights of the non-seed nodes in the regular fashion. Note that if $\mathbf{u} = \mathbf{a}_U^{t_0}$, then for every node i , $x_i^t = a_i^{t_0+t}$, for all $t \geq 0$. That

²Any other method for testing convergence is applicable. Our analysis does not depend on the definition of convergence.

is, $\text{AUX}(\mathbf{a}^0, \mathbf{a}_U^{t_0})$ and $\text{MAX}(\mathbf{a}^0)$ are equivalent; the system AUX converges if and only if the system MAX converges. The AUX system serves the purpose of “disconnecting” the seed nodes from the non-seed nodes. This will become clear in the following.

We will now prove that in the limit the configuration of AUX is independent of the initial configuration \mathbf{u} of the non-seed nodes. To assist the proof, we introduce the following conventions. We assume that at the initialization of the AUX system, each node i receives an amount of *mass* μ_i^0 of *color* i . The weight of this mass is x_i^0 , where a unit of mass corresponds to a unit of weight. That is, there is a one to one correspondence between mass and weight, except for the fact that mass has color. As mass is moved around in the graph, by measuring the amount of mass of color i at time t , we can quantify the contribution of the initial weight of authority i to the configuration \mathbf{x}^t at time t .

Consider the AUX system at time $t - 1$. Recall that the function f^{t-1} maps every hub j to the authority i which at time $t - 1$ has the maximum weight among all authorities in $F(j)$. We take the following view of the t -th iteration. Every authority i sends its mass to all hubs that map to i at time $t - 1$ (assuming that mass can be replicated). Consider a hub j , for which $f^{t-1}(j) = i$. The hub j receives the mass of the authority i , and sends it to all the authorities in $F(j)$, *except* the seed nodes in S . Every seed node $s \in S$ receives mass of color s , with weight $a_s^{t_0+t}$. Non-seed authority i receives mass from every hub in $B(i)$. The weight of i is the total weight of all the mass it receives. If node i receives μ units of mass of color k , we say that node i *contains* μ units of mass of color k . The amount of mass of color k contained in node i at time t is the contribution of the initial weight x_k^0 of node k to the weight x_i^t of node i , at time t . We use μ^t to denote the total mass of non-seed color in the system at time t , that is, the total mass of color k , for all $k \in U$.

We are now ready to prove the following lemma.

LEMMA 3. *In the AUX system, as $t \rightarrow \infty$, $\mu^t \rightarrow 0$.*

PROOF. First, we note that by definition of the AUX system, no seed node ever receives mass of non-seed color k , for any $k \in U$. We will prove that for all $t \geq 0$, every authority $i \in U$ contains at most $(d - 1)^t/d^t$ units of mass of non-seed color. For $t = 0$ the claim is trivially true. Assume that it is true at time t . At the iteration $t + 1$, the hub j receives the mass of the authority p , such that $f^t(j) = p$. By the inductive hypothesis, every authority contains at most $(d - 1)^t/d^t$ units of mass of non-seed color; therefore, after this first step of the iteration every hub j contains at most $(d - 1)^t/d^t$ units of mass of non-seed color.

Consider now some authority $i \in U$. Authority i receives the mass of $d_i \leq d - 1$ hubs. Since every hub contains at most $(d - 1)^t/d^t$ units of mass of non-seed color it follows that at the end of iteration $t + 1$ authority i contains at most $(d - 1)^{t+1}/d^t$ units of mass of non-seed color. At the normalization step, the mass at every authority is scaled by a factor $1/d$. Thus, at the end of iteration $t + 1$, authority i contains at most $(d - 1)^{t+1}/d^{t+1}$ units of mass of non-seed color.

Therefore, the total mass of non-seed color in the graph at time t is at most $\mu^t = |A|(d - 1)^t/d^t$, where $|A|$ is the number of authorities. Thus, as $t \rightarrow \infty$, $\mu^t \rightarrow 0$. \square

Corollary 2 follows immediately from Lemma 3.

COROLLARY 2. *The configuration $\lim_{t \rightarrow \infty} \mathbf{x}^t$ of the AUX system is independent of the initialization vector \mathbf{u} .*

Let $\mathbf{0}$ denote the vector of all zeros. We now prove the following lemma.

LEMMA 4. *The system $\text{AUX}(\mathbf{a}^0, \mathbf{0})$ converges for any configuration \mathbf{a}^0 .*

PROOF. We will prove that the weights of all authorities in the system are non-decreasing functions of time. Since the weights are upper bounded it follows that they will converge.

For every seed node $s \in S$, $x_s^t = a_s^{t+t_0}$, that is, the weight of the seed nodes in the AUX system at time t is the same with the weight of the seed nodes in the MAX system at time $t + t_0$. From Lemma 1 we know that for the MAX algorithm, the weights of all seed nodes are non-decreasing functions of time. Therefore, x_s^t is a non-decreasing function of time, for all $s \in S$.

We will now prove that for every authority $i \in U$, $x_i^t \geq x_i^{t-1}$ for all $t \geq 1$, using induction on time. For $t = 1$, $x_i^1 \geq 0$, so the claim is trivially true. Assume that it is true at time t . Consider now the difference $\bar{x}_i^{t+1} - \bar{x}_i^t$. We break up the hubs in $B(i)$ into two sets. The set V contains the hubs $j \in B(i)$ such that $f^t(j) = f^{t-1}(j)$; that is, the hubs whose mapping does not change at time t . The set W contains the hubs $j \in B(i)$ such that $f^t(j) \neq f^{t-1}(j)$, that is, the hubs whose mapping changes at time t .

We have that $\bar{x}_i^{t+1} - \bar{x}_i^t = S_1 + S_2$, where

$$\begin{aligned} S_1 &= \sum_{j \in V} (x^t(f^t(j)) - x^{t-1}(f^{t-1}(j))) \\ S_2 &= \sum_{j \in W} (x^t(f^t(j)) - x^{t-1}(f^{t-1}(j))) . \end{aligned}$$

For every $j \in V$, there exists $p \in A$ such that $f^t(j) = f^{t-1}(j) = p$. By the inductive hypothesis we have that $x_p^t - x_p^{t-1} \geq 0$. Therefore, $S_1 \geq 0$. For every $j \in W$, there exist $p, q \in A$ such that $f^t(j) = p$, and $f^{t-1}(j) = q$. Since at time t the mapping of the hub j switches from q to p , it follows that $x_p^t > x_q^t$, and $x_p^{t-1} \leq x_q^{t-1}$ (or $x_p^t \geq x_q^t$, and $x_p^{t-1} < x_q^{t-1}$ depending on the way that we break the ties). By the induction hypothesis we have that $x_q^t \geq x_q^{t-1}$. Therefore, $x_p^t - x_q^{t-1} \geq x_p^t - x_q^t \geq 0$. Thus, $S_2 \geq 0$, and $\bar{x}_i^{t+1} - \bar{x}_i^t \geq 0$. Since $x_i^{t+1} - x_i^t = (\bar{x}_i^{t+1} - \bar{x}_i^t)/d$, it follows that $x_i^{t+1} \geq x_i^t$. \square

The following theorem follows directly from Lemmas 3 and 4.

THEOREM 1. *The MAX algorithm converges for any initial configuration. The stationary configuration of MAX is determined by the stationary weights of the seed nodes.*

PROOF. For any initial configuration \mathbf{a}^0 the system $\text{AUX}(\mathbf{a}^0, \mathbf{0})$ converges. From Corollary 2 the limiting behavior of AUX is independent of the initial configuration of the non-seed nodes. Therefore, for any vector \mathbf{u} , the $\text{AUX}(\mathbf{a}^0, \mathbf{u})$ system will converge, and it will converge to the same vector as $\text{AUX}(\mathbf{a}^0, \mathbf{0})$. When $\mathbf{u} = \mathbf{a}_{U'}^{t_0}$, the system $\text{AUX}(\mathbf{a}^0, \mathbf{a}_{U'}^{t_0})$ is equivalent to the $\text{MAX}(\mathbf{a}^0)$ algorithm. Therefore, the MAX algorithm converges for any initial configuration. From Corollary 2 it follows that the stationary configuration depends only on the weights of the seed nodes at time t_0 . \square

We are particularly interested in the *uniform* initial configuration, when all nodes are initialized to the same weight. Since the configuration is a unit vector in the L_∞ norm all nodes are initialized to weight 1. In this case from Lemma 1, we know that the weight of the seed nodes will immediately converge to 1. In the case of the uniform initial configuration we also have a very clear characterization of the *rate of convergence* of the algorithm. In this case, the seed nodes converge immediately to weight 1. Given an accuracy constant δ , the MAX algorithm converges when the mass of the non-seed color becomes less than δ . Let d' denote the second-highest in-degree in the graph. As we saw in Lemma 3, after t

iterations, the mass of non-seed color is equal to $\mu^t \leq |A|(d'/d)^t$. We have that $|A|(d'/d)^t \leq \delta$, if $t \geq \frac{\log(|A|/\delta)}{\log(d/d')}$. Thus, the rate of convergence depends upon the size of the graph, and the ratio between the highest, and second-highest in-degree in the graph.

5.2 The stationary configuration

In this section we give a characterization of the way the MAX algorithm assigns the weights to the authorities. We first introduce the auxiliary graph G_A . Assume that the algorithm has converged, and let a_i denote the stationary weight of node i . 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 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 .

An example of the graphs G , G_a , and G_A is shown in Figure 5. Every edge $\{i, j\}$ in the graph G_a is tagged with the number of hubs $B(i) \cap B(j)$ that point to both i and j nodes. The numbers next to the nodes of graph G_A are the stationary weights, and the weights on the edges are the $c(i, j)$ values. Note that there is no edge between nodes x and y in the graph G_A . Although they share a hub, this hub is mapped to node s .

The following proposition gives a recursive formula for weight a_i , given the weights of the nodes in $N(i)$.

PROPOSITION 1. *The weight of node i satisfies the equation*

$$a_i = \sum_{j \in N(i)} c(j, i)a_j/d + b_i a_i/d . \quad (2)$$

PROOF. Recall that for every node i , $a_i = \sum_{j \in B(i)} a(f(j))/d$. From the hubs in $B(i)$, b_i of them are mapped to node i , or to some node with weight equal to a_i . These hubs recycle the weight of node i , and they contribute weight $b_i a_i/d$ to the weight of node i . The remaining hubs bring in the weight of some other authority. For every $j \in N(i)$, there are $c(j, i)$ hubs in $B(i)$ that are mapped to node j . These hubs propagate the weight a_j of node j to node i . Thus, they collectively contribute weight $c(j, i)a_j/d$ to the weight of node i . Therefore, we obtain equation 2. \square

By definition, the graph G_A is a DAG. Therefore, there must exist some nodes, such that no node in G_A points to them. We define a *source node* in the graph G_A to be a node x , such that

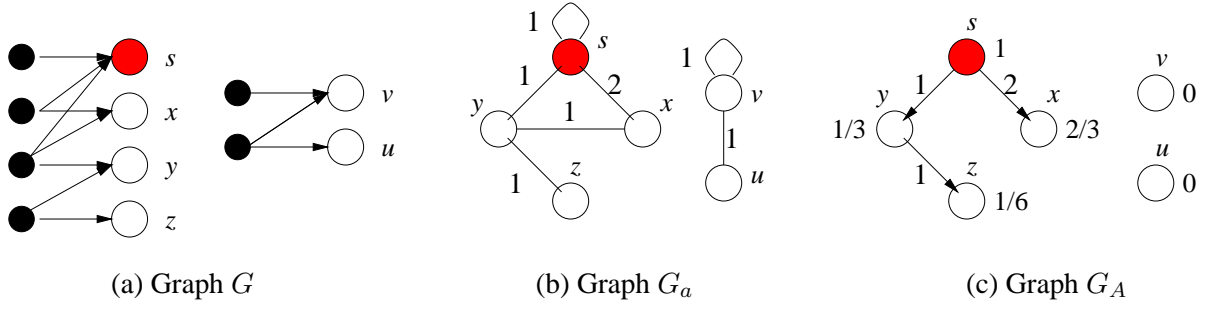


Figure 5: Graphs G , G_a , and G_A .

$N(x) = \emptyset$ (i.e., there is no node in G_A that points to x), and $a_x > 0$. Lemma 2 guarantees that at least one such node exists. In the example of Figure 5(c), there is only one source node, the seed node s . Nodes v and u have no incoming edges, but they are not source nodes, since they have zero weight. We now prove that the set of source nodes is identical to the set of the seed nodes.

LEMMA 5. *A node is a source node of the graph G_A if and only if it is a seed node in the graph G .*

PROOF. Let x be a source node of the graph G_A . Since $N(x) = \emptyset$, it follows that all d_x hubs that point to x are mapped to x , or to some node with weight equal to a_x . Therefore $b_x = d_x$. We have that $a_x = a_x d_x / d$. Since $a_x > 0$, it follows that $d_x = d$.

Let s be a seed node. Assume that s is not a source node in the graph G_A . Then, either $a_s = 0$, or $N(s) \neq \emptyset$. We have assumed that the initial configuration is a fair configuration, that is, the initial configuration assigns to every component of the graph G_a non-zero weight. If C_s is the component in the graph G_a that contains node s , then at least one node in C_s was initialized to non-zero weight. Therefore, there exists some point in time t_s such that the $a_s^{t_s} > 0$. From Lemma 1 we know that the weight of every seed node is a non-decreasing function of time, therefore, $a_s^t \geq a_s^{t_s}$ for all $t \geq t_s$. Therefore, $a_s > 0$.

Assume that $N(s) \neq \emptyset$. We have that

$$a_s = \sum_{i \in N(s)} c(i, s) a_i / d + b_s a_s / d.$$

For every $i \in N(s)$ we have that $a_i > a_s$. Therefore, it follows that

$$a_s = \sum_{i \in N(s)} c(i, s) a_i / d + b_s a_s / d > c_s a_s / d + b_s a_s / d = a_s d / d = a_s,$$

thus reaching a contradiction. \square

We now turn our attention to the non-seed nodes of the graph. For the following, we say that node i is *connected* to a seed node in the graph G_a if there exists a path in the graph G_a from a seed node to node i . We say that node i is *reachable* from a seed node in the graph G_A if there exists a directed path in the graph G_A from a seed node to the node i . We will often say that a node is reachable to indicate that it is reachable from a seed node in the graph G_A . In the example of Figure 5, nodes x, y, z are connected to, and reachable from the seed node s , while nodes u and v are neither connected to, nor reachable from seed node s .

LEMMA 6. *A node i is reachable from a seed node in the graph G_A if and only if $a_i > 0$.*

PROOF. We will prove that every reachable node has positive weight using induction on the length of the shortest path from a seed node to node i in the graph G_A . Let $radius_A(s, i)$ be the length of the shortest path from seed node s to node i in the graph G_A . Let $radius_A(i) = \min_{s \in S} radius_A(s, i)$ be the shortest path from any seed node to node i in the graph G_A . For every node i with $radius_A(i) = 0$, that is, the seed nodes themselves, the lemma is trivially true. Assume that it is true for every node i with $radius_A(i) \leq \ell$. Every node j with $radius_A(j) = \ell + 1$ must be connected to a node i with $radius_A(i) = \ell$. From Proposition 1 we have that $a_j \geq c(i, j) a_i / d > 0$.

Assume now that node $i \in U$ is not reachable from a seed node. If $N(i) = \emptyset$, then it must be that $a_i = 0$. Otherwise, node i is a source node. From Lemma 5 this is not possible, since node i is not a seed node. Assume now that $N(i) \neq \emptyset$, that is, there exists some node in the graph G_A that points to node i . Then starting from node i we can follow edges backwards in the graph G_A to other non-reachable nodes. Since the graph G_A contains no cycles, we will eventually find a node j that is not reachable, and has no incoming edges. Since j is not a seed node, we have that $a_j = 0$, and $a_j > a_i \geq 0$, thus reaching a contradiction. Therefore, there cannot be any node pointing to node i , and $a_i = 0$. \square

LEMMA 7. *A node is reachable from a seed node in the graph G_A if and only if it is connected to a seed node in the graph G_a .*

PROOF. Obviously, by definition of the graphs G_a and G_A , if a node is not connected to a seed node, then it is not reachable from a seed node. We will now prove that every node i , that is connected to a seed node in the graph G_a , it is also reachable from a seed node in the graph G_A , using induction on the length of the shortest path from a seed node to node i in the graph G_a . Let $radius_a(s, i)$ be the length of the shortest path from seed node s to i in the graph G_a . Let $radius_a(i) = \min_{s \in S} radius_a(s, i)$ be the shortest path from any seed node to node i in the graph G_a . For every node i such that $radius_a(i) = 0$, that is, the seed nodes themselves, the lemma is trivially true. Assume that it is true for every node i with $radius_a(i) = \ell$. Now consider some node j with $radius_a(j) = \ell + 1$. Since node j is connected to a seed node in the graph G_a , there exists a node i with $radius_a(i) = \ell$ such that the edge (i, j) belongs to graph G_a . This implies that there exists at least one hub h that points to both i and j . Let $f(h) = k$ be the mapping of this hub. Node k is not necessarily node i or j , and it is not necessarily the case that $radius_a(k) \leq \ell$. However, we know that hub h points to both i and k , and that $a_k \geq a_i$. By the inductive hypothesis, node i is reachable, so $a_i > 0$. Thus, $a_k > 0$.

Consider now the nodes j and k . If $a_j \geq a_k$, then $a_j > 0$, therefore, node j is reachable. Otherwise, for the pair (k, j) we have that: [a] there exists an edge (k, j) in the graph G_a (since the nodes j and k share the hub h); [b] $B(k, j) \cap H(k) \neq \emptyset$ (since the

hub h is mapped to k); [c] $a_k > a_j$. Therefore, there must exist a directed edge (k, j) in the graph G_A . Since $a_k > 0$, Lemma 6 guarantees that node k is reachable from a seed node in G_A . Thus, node j is also reachable. \square

For some node i , and some seed node s , 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. We now summarize the results of this section in the following theorem.

THEOREM 2. *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 , $1 \leq i \leq k$, 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 every node x in C_i is reachable from a seed node in C_i , and $a_x > 0$. Given the weights of the seed nodes, we can recursively compute the weight of a reachable (in the graph G_A) 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 $j \in N(i)$, $dist(j) < \ell$.

PROOF. Let C_i denote the i -th component of the graph G_a . Obviously, if a node is not connected to a seed node in graph G_a , it cannot be reachable from a seed node in the graph G_A . Therefore, if component C_i does not contain a seed node, then, from Lemma 6, for every x in C_i , $a_x = 0$. Assume now that the component C_i contains a seed node. Lemma 7 guarantees that every node x in C_i becomes reachable from a seed node in the graph G_A . The weight of node $x \in C_i$ can be computed recursively using Proposition 1. We have that

$$a_x = \sum_{j \in N(x)} c(j, x) a_j / d + b_x a_x / d.$$

Therefore,

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

If node x is at distance ℓ , then all nodes $j \in N(x)$ have $dist(j) < \ell$. Therefore, starting from the seed nodes, we can iteratively compute the weights of all nodes at increasing distances. \square

Theorem 2 is in agreement with our findings in the Section 5.1, where we observed that the stationary configuration depends solely on the stationary weights of the seed nodes. Note that Theorem 2 does not provide a constructive way of assigning weights to the nodes, since the graph G_A depends on the stationary configuration. However, it provides a useful insight in the mechanics of the algorithm, and in the way the weight is propagated from the seed nodes to the remaining authorities. All weight emanates from the seed nodes, and it floods the rest of the nodes, propagated in the graph G_A . As the distance from the seed nodes increases, the weight decreases exponentially by a scaling factor d . However, well connected nodes, and nodes with high in-degree in the graph G , reinforce their own weight. For node i , there are b_i hubs that recycle the weight of node i . Thus, high in-degree can increase the weight of a node, even if it is far from a seed node.

The uniform initial configuration case: In the case of the uniform initial configuration we know that the seed nodes will converge immediately to weight 1. Therefore, the MAX algorithm will rank the seed nodes first. The rest of the nodes receive less weight than the seed nodes.

The arbitrary initial configuration case: If we knew the stationary weights of the seed nodes then we would be able to compute the weights of the rest of the nodes recursively, using the formula in Theorem 2. However, the weights of the seeds depend on the initial configuration. Lemma 2 guarantees that at least one seed will receive weight 1. In the case that the graph contains a single seed node (a case we encounter often in our experiments) the MAX algorithm converges to the same configuration as in the uniform case. One would hope that all seeds converge to weight 1, for all initial configurations, in which case the stationary configuration would be unique. However, this is not the case. One can construct simple examples of graphs that consist of multiple disconnected components, where, depending on the weight assigned to each component, the algorithm converges to different configurations. A natural question is whether we can prove a similar result if we consider an *authority connected* graph, that is, a graph G , such that the authority graph G_a is connected. We now present a counter example, where we show that for an authority connected graph G , there exists an initial configuration such that one of the seed nodes converges to a weight less than 1. Furthermore, there exist non-seed nodes that have weight greater than the weight of that seed node.

PROPOSITION 2. *The MAX algorithm does not always converge to the same weight vector for all initial configurations, even when restricted on authority connected graphs.*

PROOF. Consider the graph G in Figure 6(a). The large red and white nodes are the authorities, while the small black nodes are the hubs. The shaded (red) nodes are the seed nodes of the graph G . There are four seeds in the graph, each with in-degree 3. Figure 6(b) shows the corresponding graph G_a . The initial configuration assigns weight 1 to all seed nodes, except for the central seed, which receives zero weight. The non-seed nodes are also initialized to zero weight. The initial weights for each node are shown next to vertices of the graph in Figure 6(a).

When the algorithm converges, we obtain graph G_A shown in Figure 6(c). The numbers next to the nodes in the graph are the stationary weights of the nodes. The weights on the edges are equal to the $c(i, j)$ values. Obviously, the algorithm does not converge to the same stationary configuration as when initialized to the uniform configuration, since the central seed node receives weight less than 1. Also, in this example there exist non-seed nodes that receive weight greater than the weight of the central seed node. \square

6. EXPERIMENTAL EVALUATION

In this section we study the performance of the MAX algorithm for Web search queries, and for finding related pages.

6.1 Web Search queries

We experiment on thirty four different queries. The data sets were constructed in the fashion described by Kleinberg [15]. We start with a Root Set of 200 pages that are returned by a search engine. In our experiments we use the Google search engine. Then, for each page in the Root Set, we include all the pages that are pointed to by this page, and the first 50 pages (in the order returned by the Google search engine) that point to this page. Given this Base Set of web pages, we construct the underlying hyperlink graph. We remove *navigational links*, that is, links between the same domain.

We compare the MAX algorithm against the HITS, PAGERANK, AT(k), and NORM(p) algorithms, as well as the INDEGREE heuristic where all nodes are ranked according to their in-degree in the hyperlink graph G . For the NORM(p) family, we set $p = 2$ and we denote this algorithm as NORM. For the AT(k) family of algorithms,

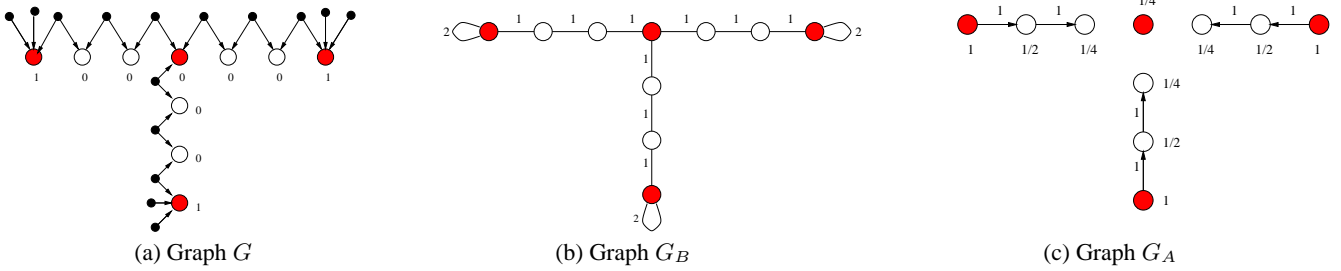


Figure 6: An example with a non-uniform initial configuration for authority connected graphs

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. For the PAGERANK algorithm, we set the jump probability to $\epsilon = 0.2$, a value that has been previously considered by Brin and Page [6]. A detailed comparison of MAX with other LAR algorithms can be found at [20]. Results are presented at the Web pages <http://www.cs.toronto.edu/~tsap/experiments/thesis>, and <http://www.cs.toronto.edu/~tsap/experiments/journal>.

For HITS and its variants (NORM, AT(k), MAX) we initialize all weights to 1, that is, to the uniform configuration. This is a natural initialization that assumes no a-priori knowledge, thus, assigning equal initial weight to all nodes. We note that in our experiments for almost all queries the authority graph G_a contains a giant component, which contains a single seed node. Thus, with respect to the HITS and MAX algorithms any *fair* initial configuration produces the same output as the uniform initialization.

The measure that we will use for the evaluation of the quality rankings is *precision over the top-10*. We define the *high relevance ratio* as the fraction of documents in the top 10 positions of the ranking that are highly relevant to the query. We also define the *relevance ratio* as the fraction of documents within the top-10 positions that are relevant, or highly relevant to the query. Similar quality measures are used in the TREC conferences for evaluating Web search algorithms.³

In order to determine the relevance of the results, we performed a user study. The study was performed on-line. In the starting page of the study, the set of queries was displayed and the users had the option to select any query they felt they were capable of evaluating. Then, they were presented with a set of results that belong to union of the top-10 results of the different algorithms. The pages were presented in a random order, and the users did not know which algorithm introduced each result. They were asked to rate the results as “Highly relevant”, “Relevant”, “Non-Relevant”, or “Don’t Know”, if they could not assess the relevance of the result.

The evaluations of each user define a relevance and a high relevance ratio for each algorithm. We take the average of these ratios over all users and over all queries. The average ratios are shown in Table 1. The MAX algorithm emerges as the clear best among the algorithms we consider. The second best options are the INDEGREE and AT-MED algorithms, followed by the AT-AVG algorithm. At the low end, the PAGERANK, NORM and HITS algorithms ex-

³For TREC relevance and high relevance is usually predefined by a set of experts.

algorithm	avg HR ratio	avg R ratio
HITS	22%	45%
PAGERANK	24%	46%
INDEGREE	35%	58%
MAX	38%	64%
AT-MED	34%	60%
AT-AVG	28%	51%
NORM	24%	44%

Table 1: Web Search: Average Performance Ratios

hibit the worst performance.

The performance of the MAX algorithm is strongly affected by the quality of the seed node, as well as the quality of the nodes with the highest in-degree, and the nodes that are most heavily co-cited with the seed node. 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 performance ratios of the INDEGREE algorithm. The INDEGREE algorithm achieves the second best high relevance ratio, and the third best relevance ratio and it outperforms more sophisticated algorithms like HITS and PAGERANK. This is rather surprising given the simplicity of the INDEGREE algorithm. We should note though that exactly due to its simplicity, the INDEGREE algorithm is the one that is most affected by the choice of the search engine that it 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.

For the AT-MED, AT-AVG and NORM algorithms, 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 the average performance ratios of AT-MED, AT-AVG and NORM take values between the ratios of MAX and HITS.

The poor performance of the HITS and the PAGERANK algorithms can be attributed to type of *communities* that the algorithms tend to promote in the top positions of the rankings. For the HITS algorithm it is well known [15, 16, 5, 10] that it tends to promote the most *Tightly Knit Community* of hubs and authorities in the graph. The PAGERANK algorithm tends to favor isolated nodes with high in-degree that form two link cycles with one or more

nodes [4]. In both cases, we observed that the communities that HITS and PAGERANK tend to favor are usually not relevant to the query, which accounts for the topic drift of the two algorithms.

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, Table 4 (Appendix A) shows the top-10 results for the query “movies”. The seed node is the Internet Movie Database⁴ (IMDB), and the algorithm converges to a set of movie databases and movie reviews sites. In this case, the MAX algorithm manages to distill the relevant pages from the community to which it converges. On the other hand, in the case of the “affirmative action” query (Table 5, Appendix A) 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.

It is also interesting to observe the behavior of MAX on the query “abortion”. Table 6 (Appendix A) shows the top-10 results of the algorithm. The seed node in the graph is the “NARAL Pro-Choice” home page. We observed that there is only light co-citation between the pro-choice and pro-life communities, so 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 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.

Similar mixing behavior for the MAX algorithm is observed for the cases where the graph contains two seeds (“randomized algorithms” query – Table 7, Appendix A), or that the seed node belongs to two different communities (“basketball” query). More results can be found at the Web pages that contain the results for all queries.

6.2 Related pages queries

The property of the MAX algorithm to diffuse the weight from the seed node to the remainder of the graph has the effect that the pages that are ranked highly are usually “related” to the seed node. Therefore, if we could set the seed node to some selected page, then we could use the MAX algorithm to find pages *related* to that page. Finding pages related to a query Web page is a standard feature of most modern search engines. This is an active research area with a growing literature [15, 8, 13]. The current techniques use content analysis, link analysis, or a combination of both. We propose the use of the MAX algorithm as a tool for discovering Web pages, related to a query Web page.

The idea of using link analysis algorithms for finding related pages was first suggested by Kleinberg [15], and it was later extended by Dean and Henzinger [8]. In this section, we use the terminology of Dean and Henzinger [8]. First, we note that we need a different algorithm for constructing the hyperlink graph that will be given as input to the LAR algorithm. Given a query page q , Dean and Henzinger propose to construct a “vicinity graph” around q as follows. Let B denote a step that follows a link backwards, and let F denote a step that follows a link forward. Starting from the query

algorithm	avg HR ratio	avg R ratio
HITS	41%	68%
PAGERANK	25%	44%
INDEGREE	40%	62%
MAX	49%	76%
COCITATION	49%	76%
GOOGLE	52%	72%

Table 2: Related Pages: Average Performance Ratios

algorithm	avg HR ratio	avg R ratio	avg grade
HITS	23%	56%	2.2
PAGERANK	34%	68%	2.2
INDEGREE	33%	66%	3.0
MAX	36%	66%	2.9
COCITATION	31%	59%	2.7
GOOGLE	18%	41%	1.4

Table 3: Homepages: Average Performance Ratios

page q , collect a set of pages that can be reached by following B , F , BF , and FB paths. The vicinity graph is the underlying hyperlink graph of this set of pages. The authors then propose to run the HITS algorithm, or other heuristics for discovering related pages.

We propose the MAX algorithm as a novel alternative for discovering related pages. In order for the algorithm to work, the query page q must be the seed of the algorithm. The rest of the nodes will then be ranked according to their relation to q , where relation is defined naturally by the MAX algorithm. However, it may not always be the case that the page q is the seed of the vicinity graph. In these cases, we engineer the graph, so as to make sure that the page q has the highest in-degree. We go through the nodes of the graph and find the node with the highest in-degree d . We then add enough extra “dummy” nodes in the graph, that point only to node q , so that the in-degree of q becomes greater than d . Thus the page q becomes the seed node for the Base Set. The MAX algorithm will assign maximum weight 1 to page q . Following the discussion in Section 5.2, the weight will be diffused from the seed node to the remaining nodes of the graph, through the hubs. The amount of weight that reaches node i will be used as a measure of its relatedness to the seed node.

Other than MAX, we also experiment with the HITS, INDEGREE, and PAGERANK algorithms. We also consider the COCITATION heuristic (also considered by Dean and Henzinger [8]) which is defined as follows. Given the query page q , for each page p in the vicinity graph compute the number of hubs that point to both q and p . Then, rank the pages according to the number of hubs that they have in common with the query page. Furthermore, we also performed a comparison with the actual Google search engine.

We experimented with 20 different queries, and we collected user feedback in a similar fashion as for the Web Search queries. Table 2 reports the average high relevance and relevance ratios. The GOOGLE search engine achieves the best high relevance ratio, while the MAX and COCITATION algorithms tie in the second position. The results indicate that co-citation plays a significant role when handling queries for related pages. The MAX and COCITATION algorithms outperform the GOOGLE search engine when considering the relevance ratio. This was rather surprising, given the fact that Google is a complete search engine that uses a combination of link analysis, text analysis, and (possibly) user statistics. It was also interesting to observe that the HITS algorithm performs significantly better in this context, and that the simple INDEGREE heuristic remains competitive.

We also performed a different experiment, where we used as

⁴<http://www.imdb.com>

query pages the home pages of ten researchers and professors. We then asked them to evaluate the results, and also to (blindly) rate the six algorithms with an (integer) grade between 0 (unacceptable) and 4 (excellent). Table 3 presents the average relevance and high relevance ratios, as well as, the average grade for each algorithm. A number of surprising facts emerge from this table. First, the GOOGLE search engine emerges as the worst algorithm with respect to both ratios and grades. The PAGERANK algorithm, which had the worst performance in Table 2, exhibits the best relevance ratio. The best high relevance ratio is achieved by the MAX algorithm. However, the best average grade is given to the INDEGREE algorithm, followed closely by the MAX and COCITATION algorithms. We note that for these queries we usually had to add to the graph a large number of dummy nodes (on average 42) to make the query page the seed node of the graph. The special nature of these queries may explain the discrepancy between the previous results.

7. CONCLUSIONS

In this paper, we considered two families of non-linear dynamical systems and we studied in detail the MAX algorithm, a special case of both families. We proved that the algorithm converges for any initial configuration, and we provided a combinatorial description of the stationary weights. We also performed extensive experiments that indicate that the algorithm performs well in practice.

Our work suggests as a possible future research direction the study of other non-linear systems. For the families $AT(k)$ and $NORM(p)$, for the two extreme values of p and k , we have a very good understanding of how the algorithms behave. It is an intriguing question to understand what happens for the intermediate values of p and k . Do the algorithms converge, and are there any other values of p and k for which they meet?

8. ACKNOWLEDGEMENTS

I would like to thank Allan Borodin for his continuous guidance throughout this work; Jon Kleinberg for his advice, support, and enthusiastic encouragement; Jeff Rosenthal, Ken Sevcik, Renee Miller, Ken Jackson, and Sam Roweis for useful comments. Many thanks to the numerous people who contributed to the on-line evaluation survey.

9. REFERENCES

- [1] D. Achlioptas, A. Fiat, A. Karlin, and F. McSherry. Web search through hub synthesis. In *Proceedings of the 42nd Foundation of Computer Science (FOCS 2001)*, Las Vegas, Nevada, 2001.
- [2] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proceedings of the 33rd Symposium on Theory of Computing (STOC 2001)*, Hersonissos, Crete, Greece, 2001.
- [3] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Research and Development in Information Retrieval*, pages 104–111, 1998.
- [4] M. Bianchini, M. Gori, and F. Scarselli. Pagerank: A circuitial analysis. In *Proceedings of the Eleventh International World Wide Web (WWW) Conference*, 2002.
- [5] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the World Wide Web. In *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, 2001.
- [6] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998.
- [7] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *Proceedings of the 17th International Conference on Machine Learning*, pages 167–174, Stanford University, 2000.
- [8] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. In *Proceedings of the Eighth International World-Wide Web Conference (WWW9)*, 1999.
- [9] R. L. Devaney. *An Introduction to Chaotic Dynamical Systems*. W. Benjamin, New York, 1986.
- [10] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1999.
- [11] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *Proceedings of the 24th Intl. Conference on Very Large Databases (VLDB)*, 1998.
- [12] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proceedings of 9th ACM Conference on Hypertext and Hypermedia*, 1998.
- [13] T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Similarity search on the Web: Evaluation and scalability considerations. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, 2002.
- [14] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence, UAI'99*, Stockholm, 1999.
- [15] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.
- [16] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proceedings of the 9th International World Wide Web Conference*, May 2000.
- [17] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *Proceedings of the 24th International Conference on Research and Development in Information Retrieval (SIGIR 2001)*, New York, 2001.
- [18] D. Rafiei and A. Mendelzon. What is this page known for? Computing web page reputations. In *Proceedings of the 9th International World Wide Web Conference*, Amsterdam, Netherlands, 2000.
- [19] J. T. Sandefur. *Discrete dynamical systems*. Oxford: Clarendon Press, 1990.
- [20] P. Tsaparas. *Link Analysis Ranking*. PhD thesis, University of Toronto, 2004.

APPENDIX

A. SAMPLE QUERY RESULTS

In this appendix we present the top-10 results of the MAX algorithm for some sample queries. In the tables below, each result is marked as “Highly Relevant”, “Relevant”, or “Not-Relevant”. Relevant results are the pages for which the “Relevant” and “Highly Relevant” votes were more than the “Non-Relevant” ones. “Highly Relevant” are the Relevant pages that have more “Highly Relevant” than “Relevant” votes. Highly Relevant documents are marked with boldface, Relevant documents with italics, while Non-Relevant documents are in regular font.

MAX	
1.	(1.000) The Internet Movie Database (IMDb). URL:www.imdb.com
2.	(0.296) Signs on DVD www.signs.movies.com
3.	(0.253) Google www.google.com
4.	(0.219) Hollywood.com - Your entertainment URL:www.hollywood.com
5.	(0.211) Empty title field URL:www.film.com
6.	(0.174) Get Wild - GetWild - getwild.com www.getwild.com
7.	(0.161) All Movie Guide URL:www.allmovie.com
8.	(0.159) Movie Review Query Engine URL:www.mrqe.com
9.	(0.159) ROTTEN TOMATOES: Movie Reviews URL:www.rottentomatoes.com
10.	(0.142) Greatest Films URL:www.filmsite.org

Table 4: Top-10 results of the MAX algorithm for the query “movies”

MAX	
1.	(1.000) Copyright Information www.psu.edu/copyright.html
2.	(0.447) <i>PSU Affirmative Action</i> <i>URL:www.psu.edu/dept/aaoffice</i>
3.	(0.314) Welcome to Penn State’s Home on the www.psu.edu
4.	(0.010) University of Illinois www.uiuc.edu
5.	(0.009) Purdue University-West Lafayette, I www.purdue.edu
6.	(0.008) UC Berkeley home page www.berkeley.edu
7.	(0.008) University of Michigan www.umich.edu
8.	(0.008) The University of Arizona www.arizona.edu
9.	(0.008) The University of Iowa Homepage www.uiowa.edu
10.	(0.008) Penn: University of Pennsylvania www.upenn.edu

Table 5: Top-10 results of the MAX algorithm for the query “affirmative action”

MAX	
1.	(1.000) <i>prochoiceamerica.org : NARAL Pro-Ch</i> <i>URL:www.naral.org</i>
2.	(0.946) <i>Planned Parenthood Federation of Am</i> <i>URL:www.plannedparenthood.org</i>
3.	(0.918) <i>National Right to Life</i> <i>URL:www.nrlc.org</i>
4.	(0.819) NAF - The Voice of Abortion Provide URL:www.prochoice.org
5.	(0.676) <i>Priests for Life Index</i> <i>URL:www.priestsforlife.org</i>
6.	(0.624) Pregnancy Centers Online URL:www.pregnancycenters.org
7.	(0.602) <i>ProLifeInfo.org</i> <i>URL:www.prolifeinfo.org</i>
8.	(0.557) Abortion Clinics OnLine URL:www.gynpages.com
9.	(0.551) After Abortion: Information on the URL:www.afterabortion.org
10.	(0.533) <i>FEMINIST MAJORITY FOUNDATION ONLINE</i> <i>URL:www.feminist.org</i>

Table 6: Top-10 results of the MAX algorithm for the query “abortion”

MAX	
1.	(1.000) <i>Algorithms Courses on the WWW</i> <i>URL:www.cs.pitt.edu/kirk/algorithm</i>
2.	(1.000) <i>Computational Geometry, Algorithms</i> <i>URL:www.cs.uu.nl/geobook</i>
3.	(0.270) Directory of Computational Geometry www.geom.umn.edu/software/cgli
4.	(0.258) LEDA moved to Algorithmic Solutions www.mpi-sb.mpg.de/LEDA/leda.ht
5.	(0.257) <i>ANALYSIS of ALGORITHMS HOME PAGE</i> <i>URL:pauillac.inria.fr/algo/AofA</i>
6.	(0.237) IEEE Computer Society computer.org
7.	(0.205) <i>Center for Discrete Mathematics and</i> <i>URL:dimacs.rutgers.edu</i>
8.	(0.183) <i>MFCS’98 home page</i> <i>URL:www.fi.muni.cz/mfcs98</i>
9.	(0.182) Computer Science Papers NEC Researc citeseer.nj.nec.com/cs
10.	(0.178) Welcome to Springer, springer-verla www.springer.de

Table 7: Top-10 results of the MAX algorithm for the query “randomized algorithms”