Delft University of Technology
Faculty Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

The second eigenvector of the Google matrix and its relation to link spamming.

Report for the
Delft Institute of Applied Mathematics
as part of

the degree of

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

ALEX SANGERS

Delft, the Netherlands
July 2012

BSc report APPLIED MATHEMATICS

"The second eigenvector of the Google matrix and its relation to link spamming."

ALEX SANGERS

Delft University of Technology

Thesis advisor

Dr.ir. M.B. van Gijzen

Other members of the graduation committee

Prof.dr.ir. C. Vuik

Dr. J.L.A. Dubbeldam

Dr. J.G. Spandaw

July, 2012

Delft

Contents

1 Introduction **6**

2 The Google Matrix **7**
2.1 Example . . . 9
2.2 Solving the eigenvalue problem . . . 9
2.3 The second eigenvalue . . . 11

3 Test problems **12**
3.1 Small test problem . . . 12
3.2 Large test problems . . . 13

4 Introduction to link spamming **14**
4.1 Introduction to energy . . . 15
4.2 Method 1 . . . 16
4.3 Method 2 . . . 17
4.4 Testing method 2 on test problems . . . 18

5 Detecting link spamming **18**
5.1 The second eigenvector . . . 19
5.1.1 The Cesàro sum . . . 20
5.1.2 Example . . . 21
5.1.3 Calculation of the eigenvector from the canonical form . . . 23
5.2 Structure of irreducible closed subsets . . . 24
5.3 Other structures . . . 25

6 Algorithms for computing the second eigenvector **25**
6.1 The block power method . . . 25
6.2 Direct solution . . . 27
6.3 Adjusting the power method . . . 28
6.3.1 Simple power method on matrix P . . . 28
6.3.2 Adaptation to Moler's power method . . . 28
6.3.3 Choosing a starting vector with sum equal to zero . . . 29
6.4 Tarjan's algorithm . . . 30
6.4.1 Tarjan's algorithm and detection . . . 31
6.5 Cesàro sum . . . 32
6.6 Numerical results . . . 32
6.7 Some observations . . . 35

7 Conclusion **35**

A MATLAB **37**
A.1 Spam detection algorithms . . . 37
A.1.1 `blockpower.m` . . . 37
A.1.2 `direct.m` . . . 38
A.1.3 `simpleP.m` . . . 38
A.1.4 `powerP.m` . . . 39
A.1.5 `startvector.m` . . . 40
A.1.6 `tarjan.m` . . . 40

A.1.7 `cesvec.m` 41
A.2 Other algorithms 41
A.2.1 `addpromo.m` 41
A.2.2 `Cesaro.m` 42
A.2.3 `detecttarjan.m` 42
A.2.4 `Ptwithoutdangling.m` 43

1 Introduction

As the web expands every year, so does the demand for efficient search engines. The last decade millions of websites have been launched with various subjects and of varying quality. Different search engines try to bring order to the web and help people to find for what they are looking for.
We will have a closer look at the algorithm proposed by Larry Page and Sergey Brin, better known as Google PageRank, or short: Google. For more background information about the founding of Google we refer to [3].

This report studies Google's search algorithm in mathematical context and in partical the problems of link spamming on the web. Link spamming (or short: spamming) is defined by deliberately abusing the structure of a search algorithm to gain an unfairly high ranking.

Google PageRank attempts to return the best ranking of websites when searching on the web. Intuitively, PageRank models a random web surfer. First, the web surfer starts at a random website. The web surfer will randomly follow one of the outgoing hyperlinks at the website with a chance p and switch to a totally random website with chance $1 - p$. This jump behaviour can be seen as the chance that the web surfer gets tired of following links and will choose another website. The web surfer does not mind visiting websites more than once and will never stop visiting websites. The PageRank is the distribution of the visiting frequency of each website. The PageRank of a website is the probability that the random web surfer chooses to view that website at a random time.
The description above is an intuitive explanation of a mathematical model known as a Markov chain. The PageRank is the first eigenvector of this transition matrix.

The question that we will try to answer is the following one:
How is the second eigenvector of the Google matrix related to link spamming?

To be more specific, we cite Haveliwala [4]: "The eigenvectors corresponding to the second eigenvalue are an artifact of certain structures in the web graph. (...) Analysis of the nonprincipal eigenvectors of $\mathbf{A}$ may lead to strategies for combating link spam." We will try to find the structure of the second eigenvectors and find a method to calculate them.

This report contains an explanation of the Google Matrix and different methods for solving Google's eigenvalue problem in Section 2. Furthermore, the second eigenvalue and its corresponding eigenvector are described in relation to link spamming in Sections 4 and 5. Different algorithms for finding the second eigenvectors are described in Section 6.

An important remark is that the terms 'websites', 'web pages' and 'nodes' as well as the terms 'hyperlinks' and 'web links' are used interchangeably.

One last remark about notation: in this report the i-th eigenvector is written as $\mathbf{x^{(i)}}$ and the j-th element of vector $\mathbf{x}$ is written as x_j. We will be noting a submatrix of matrix $\mathbf{A}$ with $\mathbf{A_{ij}}$ and an element with a_{ij}.

2 The Google Matrix

We introduce W, a set of the web pages, that are connected to each other with hyperlinks, i.e., incoming and outgoing web links. Another mathematical representation of W is a directed graph, with a (directed) connection if there is a (directed) hyperlink between the nodes of the graph.

Let n be the number of websites. Further, let $\mathbf{G}$ be the n-by-n connectivity matrix with $g_{ij} = 1$ if there is a hyperlink from page j to i and $g_{ij} = 0$ otherwise. One could say that $\mathbf{G}$ is the matrix representation of W. In general, when looking at the current web, $\mathbf{G}$ is a very large, sparse matrix, because W is huge and relatively few hyperlinks exist between the websites.
When building connectivity matrix $\mathbf{G}$ there is a choice to be made. Do we allow self-referencing nodes (so $g_{ii} = 1$)? This answer is different for every research and we chose to follow [1] and set this diagonal to zero per definition, i.e., self-referring hyperlinks are removed.

We introduce c_j as the column sums of $\mathbf{G}$, that is $c_j = \sum_i g_{ij}$. Note that c_j is the amount of outgoing hyperlinks of website j. We will also call this the out-degree of page j.

Surfing the web can be modelled as a Markov process, where one reaches one state from another state by following a hyperlink. Haveliwala [4] makes use of the row-stochastic matrix $\mathbf{P}$. Matrix $\mathbf{P}$ can be seen as a Markov matrix. We prefer to work with column-stochastic matrices for now, so let us formulate $\mathbf{P}$ (in terms of its elements):

$$p_{ij} = \begin{cases} g_{ij}/c_j & \text{if } c_j \neq 0, \\ 1/n & \text{if } c_j = 0. \end{cases} \tag{2.1}$$

Note that $\mathbf{P^T}$ is column-stochastic. We will call nodes without outgoing hyperlink *dangling nodes*. Thus, looking at (2.1), dangling nodes have a uniform chance to go to another node. From now on, we will call nodes without outgoing hyperlink dangling nodes. This is a very important observation, because this means that every dangling node is not a dead end, but actually a 'distributor'. See Figure 1 for a graphic representation of the change that a simple graph would make. For our own convenience, we will leave out the striped connections, as seen

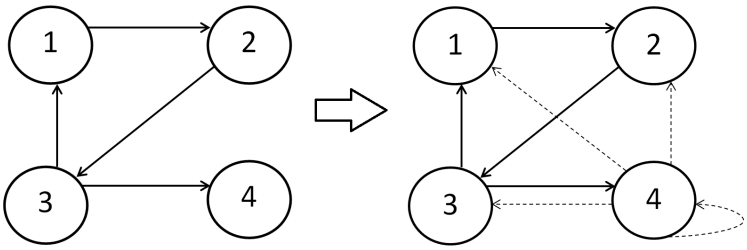


Figure 1: Changing the dangling nodes to 'distributors' when defining $\mathbf{P^T}$.

in Figure 1, in all following figures.

The column-stochastic matrix $\mathbf{P^T}$ has some shortcomings. Intuitively, a surfer will not always follow links, but sometimes 'jump' to another website by random choice. $\mathbf{P^T}$ does not model this jump behaviour and furthermore, $\mathbf{P^T}$ has some mathematically limitations. Let us sum up some of these disadvantages of $\mathbf{P^T}$:

1. The matrix $\mathbf{P^T}$ is reducible in general and therefore, its first eigenvectors are not necessarely unique. Thus, we have several 'PageRank vectors'.

2. As a consequence, the computation of any first eigenvector can be difficult (also in terms of convergence).

3. It is likely that most elements of all first eigenvectors are zero, which is undesirable.

Therefore, we shall discuss a new matrix which is column-stochastic (and irreducible). This matrix should remedy all shortcomings of $\mathbf{P^T}$. Let us say that we will follow an outlink with chance p and switch to a random page with chance $1-p$. Typically, p is chosen between 0.85 and 0.99 and often equal to 0.85 ([1],[3]). In this report we will use $p = 0.85$ for all test problems.

Let $\mathbf{A}$ be the n-by-n column-stochastic matrix with elements:

$$a_{ij} = \begin{cases} pg_{ij}/c_j + (1-p)/n & \text{if } c_j \neq 0. \\ 1/n & \text{if } c_j = 0. \end{cases} \tag{2.2}$$

Recognize that the following important equality holds:

$$\mathbf{A} = p\mathbf{P^T} + \frac{1-p}{n}\mathbf{E}, \tag{2.3}$$

with $\mathbf{E}$ the n-by-n matrix of all ones. Also, recognize that if page j is a dead end (dangling node) then each page has a chance $1/n$ $(= p/n + (1-p)/n)$ to be chosen. Thus, if column $\mathbf{a_j} = \mathbf{e}/n$, with $\mathbf{e}$ the n-vector of ones, then page j is a dangling node.

Furthermore, $\mathbf{A}$ is the transition probability matrix of the Markov chain. All elements are greater than zero and equal or smaller than one and the sum of its columns are equal to one. We will explain this more extensively later this section.

The PageRank is determined as the eigenvector of the dominant eigenvalue of the following system:

$$\mathbf{Ax^{(1)}} = \lambda_1 \mathbf{x^{(1)}}. \tag{2.4}$$

Intuitively, when recalling the random web surfer from Section 1, the eigenvector $\mathbf{x^{(1)}}$ is the distribution of the visiting frequency for each node. The more often the surfer passes node j, the higher its PageRank will be.

We say that $\mathbf{x^{(1)}}$ is the unique dominant eigenvector corresponding to the dominant eigenvalue $\lambda_1 = 1$. To show that $\lambda_1 = 1$ exists and is unique, we use the Perron-Frobenius theorem ([13]) for the Markov matrix $\mathbf{A}$.

Theorem 2.1. *(Perron-Frobenius) For any real square matrix* $\mathbf{A}$ *with positive entries, its unique Perron–Frobenius eigenvalue* λ_1 *satisfies the following inequalities:*

$$\min_j \sum_i A_{ij} \leq \lambda_1 \leq \max_j \sum_i A_{ij}. \tag{2.5}$$

The proof is given as an exercise in [13] (Exercise 8.2.7). As we know, all column sums of $\mathbf{A}$ are equal to one, so we can draw the conclusion that

$$\mathbf{x^{(1)}} = \mathbf{Ax^{(1)}} \tag{2.6}$$

has a unique solution within a scaling factor. If this scaling factor is chosen such that $\sum_i x_i^{(1)} = 1$ (or: $||\mathbf{x^{(1)}}||_1 = 1$), then $\mathbf{x^{(1)}}$ is the stationary stochastic vector of the Markov chain and also, $\mathbf{x^{(1)}}$ is the Google PageRank.

2.1 Example

To illustrate the theory, we present a simple example. To find the stationary vector $\mathbf{x^{(1)}}$ of the Markov chain, we introduce a small representation W of the web in Figure 2. We use $p = 0.85$, as we will do in all test problems. First, we construct connectivity matrix $\mathbf{G}$:

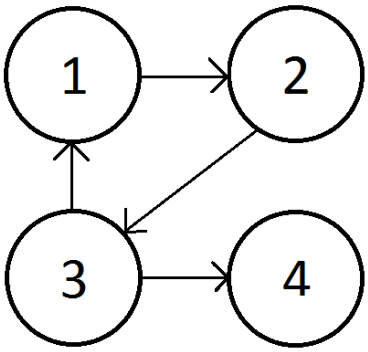


Figure 2: Directed graph with $n = 4$ and $W = \{1, 2, 3, 4\}$.

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{2.7}$$

and then calculate column-stochastic $\mathbf{P^T}$ with (2.1)

$$\mathbf{P^T} = \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{4} \\ 1 & 0 & 0 & \frac{1}{4} \\ 0 & 1 & 0 & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{1}{4} \end{pmatrix}. \tag{2.8}$$

We can now calculate $\mathbf{A} = p\mathbf{P^T} + \frac{1-p}{n}\mathbf{E}$ (or directly with (2.2)):

$$\mathbf{A} = \begin{pmatrix} \frac{3}{80} & \frac{3}{80} & \frac{37}{80} & \frac{1}{4} \\ \frac{71}{80} & \frac{3}{80} & \frac{3}{80} & \frac{1}{4} \\ \frac{3}{80} & \frac{71}{80} & \frac{3}{80} & \frac{1}{4} \\ \frac{3}{80} & \frac{3}{80} & \frac{37}{80} & \frac{1}{4} \end{pmatrix}. \tag{2.9}$$

Note that all columns of matrices $\mathbf{P^T}$ and $\mathbf{A}$ sum up to one and all elements of the last column are equal to 0.25 since this is a dangling node. Then, calculating the first eigenvector of $\mathbf{A}$ (in MATLAB for example) will return the following PageRank vector $\mathbf{x^{(1)}} = \begin{pmatrix} 0.214 \\ 0.265 \\ 0.318 \\ 0.214 \end{pmatrix}$, and its corresponding so-called ranking is $\begin{pmatrix} 3 \\ 2 \\ 1 \\ 3 \end{pmatrix}$. Note that node 1 and 4 have equal ranking.

2.2 Solving the eigenvalue problem

The most common way to solve a large system in (2.6) is the power method. The power method starts with a guess $\mathbf{u_0}$ and then we iteratively compute $\mathbf{u_{k+1}} = \mathbf{A}\mathbf{u_k}$. After each iteration we

scale $\mathbf{u_k}$ with $||\mathbf{u_k}||_1 = 1$ to make sure $\mathbf{u_k}$ sums up to 1 and thus is stochastic. For now this scaling is ignored because we are only interested in the direction of $\mathbf{u_k}$. We will iterate until $\mathbf{u_m}$ satisfies the convergence criterion for some large enough m and then the stationary vector of the Markov chain is given by $\mathbf{x^{(1)}} = \mathbf{u}_\infty \approx \mathbf{u_m}$.

Every iteration consists of a matrix-vector multiplication and after k iterations the algorithm returns $\mathbf{u_k} = \mathbf{A}^k\mathbf{u_0}$. We suppose that $\mathbf{A}$ is diagonalizable.

Theorem 2.2. *Suppose that* $\mathbf{A}$ *has eigenvalues* $\lambda_1, \ldots, \lambda_n$ *with a full set of associated eigenvectors* $\mathbf{x^{(1)}}, \ldots, \mathbf{x^{(n)}}$ *and that* $\mathbf{u_0}$ *can be expressed as a linear combination of eigenvectors. Then the following equation holds:*

$$\mathbf{A}^k\mathbf{u_0} = \mathbf{u_k} = c_1\lambda_1^k\mathbf{x^{(1)}} + \ldots + c_n\lambda_n^k\mathbf{x^{(n)}}. \tag{2.10}$$

Furthermore, we suppose that we have a unique largest eigenvalue. Thus $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. Then the component $c_1\lambda_1^k\mathbf{x^{(1)}}$ will gradually become dominant, as one can see when rewriting (2.10).

$$\frac{\mathbf{u_k}}{\lambda_1^k} = c_1\mathbf{x^{(1)}} + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{x^{(2)}} + \ldots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\mathbf{x^{(n)}}. \tag{2.11}$$

The convergence factor is determined by the second most dominant term, which is $c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{x^{(2)}}$ and the rate of convergence is equal to $|\lambda_2|/|\lambda_1|$.

To be more specific, we know that $\lambda_1 = 1$ and thus that $\lambda_2, \cdots, \lambda_n < 1$, so we can write (2.11) as

$$\begin{aligned}\mathbf{u_k} &= c_1\mathbf{x^{(1)}} + c_2\lambda_2^k\mathbf{x^{(2)}} + \ldots + c_n\lambda_n^k\mathbf{x^{(n)}} \\ &= c_1\mathbf{x^{(1)}}(\text{as } k \to \infty),\end{aligned} \tag{2.12}$$

with corresponding rate of convergence equal to $|\lambda_2|$.

We can compute the PageRank by writing the matrix $\mathbf{A}$ (see (2.2)) as

$$\mathbf{A} = p\mathbf{GD} + \mathbf{ez^T} \tag{2.13}$$

with

$$d_{jj} = \begin{cases} 1/c_j & \text{if } c_j \neq 0 \\ 0 & \text{if } c_j = 0, \end{cases} \tag{2.14}$$

$$z_j = \begin{cases} (1-p)/n & \text{if } c_j \neq 0 \\ 1/n & \text{if } c_j = 0, \end{cases} \tag{2.15}$$

and $\mathbf{e}$ the n-vector of all ones. The matrix $\mathbf{ez^T}$ accounts for the random jumps to websites.

In practice the matrix $\mathbf{A}$ will never be formed explicitly, as only forming such a matrix will take too much time and storage. Moler [1] described this variant of the power method without forming the Markov matrix $\mathbf{A}$ and preserving sparsity of $\mathbf{G}$, by directly repeating the statement $\mathbf{x} = p\mathbf{GDx} + \mathbf{ez^Tx}$. We refer to [1] or [11] for more information.

A variant of the power method is the inverse power method, which calculates $\mathbf{A} = p\mathbf{GD} + \frac{1-p}{n}\mathbf{ez^T}$ and then $\mathbf{x^{(1)}} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{e}$. Theoretically, $\mathbf{I} - \mathbf{A}$ is singular, but in practice (due to roundoff errors) this matrix is not exactly singular. Solving $\mathbf{x^{(1)}} = (\mathbf{I} - \mathbf{A})\backslash\mathbf{e}$ will blow up

$\mathbf{x^{(1)}}$, but in the right direction and scaling $\mathbf{x^{(1)}}$ back to a stochastic vector gives good results ([1]).

An alternative way to compute the PageRank is by rewriting (2.6) as a linear system

$$(\mathbf{I} - p\mathbf{GD})\mathbf{x^{(1)}} = \beta\mathbf{e} \tag{2.16}$$

with $\beta = \mathbf{z^T x^{(1)}}$. This is the direct solution of the inverse power method. Note that we do not know the value of scalar β, but we take $\beta = 1$ so the equation can be solved explicitely. Then $\mathbf{x^{(1)}}$ can be rescaled so that $\sum_i x_i^{(1)} = 1$. Though, this method is costly and only applicable on small test problems.

2.3 The second eigenvalue

Until now only the dominant eigenvalue has been taken into account. The power method solely reveals the dominant eigenvalue with its dominant eigenvector and only these results show us the PageRank in the basic algorithm.
Before we discuss the second eigenvalue, let us give some definitions.

Definition 2.3. *A set of states S is a closed subset of the Markov chain corresponding to $\mathbf{P^T}$ if and only if $i \in S$ and $j \notin S$ implies that $p_{ji} = 0$.*

Intuitively, Definition 2.3 tells us that a Markov chain is closed if it is not possible to get out of subset S as soon as you are in it. We call such a subset a rank sink, or short *sink*. It is important to remember that each dangling node is redefined as a 'distributor' in $\mathbf{P^T}$. This means that any subset containing a dangling node cannot be a sink and in particular, any dangling node cannot be a sink.
Now we can define irreducible closed subsets ([4]).

Definition 2.4. *A set of states S is an irreducible closed subset of the Markov chain corresponding to $\mathbf{P^T}$ if and only if S is a closed subset, and no proper subset of S is a closed subset.*

We assume that the web contains many irreducible closed subsets, so this theorem holds for our matrix $\mathbf{P^T}$ when big enough ([4]).

Having discussed the definitions above, let us have a look at the second eigenvalue. Haveliwala [4] has done research on the second eigenvalue, i.e., the second-largest eigenvalue λ_2. As mentioned before, the dominant eigenvalue is $\lambda_1 = 1$ and its corresponding eigenvector is the PageRank-vector $\mathbf{x^{(1)}}$. The results of Haveliwala make use of the matrix $\mathbf{P}$, recalling Equation 2.1, the elements of the column-stochastic matrix $\mathbf{P}$ are defined by:

$$p_{ij} = \begin{cases} g_{ij}/c_j & \text{if } c_j \neq 0. \\ 1/n & \text{if } c_j = 0. \end{cases} \tag{2.17}$$

We present the following theorem, found in [4]:

Theorem 2.5. *The second eigenvector $\mathbf{x^{(2)}}$ of $\mathbf{A}$ is orthogonal to $\mathbf{e}$: $\mathbf{e^T x^{(2)}} = 0$.*

Here, $\mathbf{e}$ is the vector of all ones. Theorem 2.5 can be found in [4] as Lemma 2 and with the corresponding proof.

As a consequence of Theorem 2.5, $\mathbf{ee^T x^{(2)}} = \mathbf{0}$ and thus, the second eigenvector of $\mathbf{A}$ is only dependent of $\mathbf{P^T}$, recalling that $\mathbf{A} = p\mathbf{P^T} + \frac{1-p}{n}\mathbf{E}$. This results in the following theorem that can be found in [4] and [7]:

Theorem 2.6. *If* $\mathbf{P^T}$ *has at least two irreducible closed subsets, then the second eigenvalue of* $\mathbf{A}$ *is* $\lambda_2 = p$*, with* $1 - p$ *the jump chance as introduced in Section 1.*

The prove is given in [4] and [7]. This result has some implications for the PageRank algorithm according to [4]. A few are listed below:

- The rate of convergence of the power method is equal to $\frac{|\lambda_2|}{|\lambda_1|} = p$.
- The greater the so called eigengap $|\lambda_1| - |\lambda_2|$, the more stable the stationary distribution of the Markov chain.
- The eigenvectors corresponding to λ_2 are an artifact of certain structures in the web. Link spamming (deliberately abusing the structure of Google to gain a higher ranking) could be detected through these eigenvectors.[1]

Note that $\lambda_1 = 1 > \lambda_2 = p \geq \ldots \geq |\lambda_n| \geq 0$. We mentioned before that often $p = 0.85$ is chosen. A higher p would give more accurate results (a fair PageRank). However, lowering p would provide faster convergence and a more stable distribution. Basically, there is a certain trade-off in the value of p and some balanced value has to be chosen. Emperically, this value $p = 0.85$ performs well, although the current value used by Google is unknown and research has been done to find a better-performing value ([8]).

We examined the influence of irreducible closed subsets in the web. It is stated that the second eigenvalue is equal to p if we have at least two irreducible closed subsets in W. To illustrate this, we introduce graph $\mathbf{G_1}$. Constructing the corresponding matrix $\mathbf{A}$ and calculating

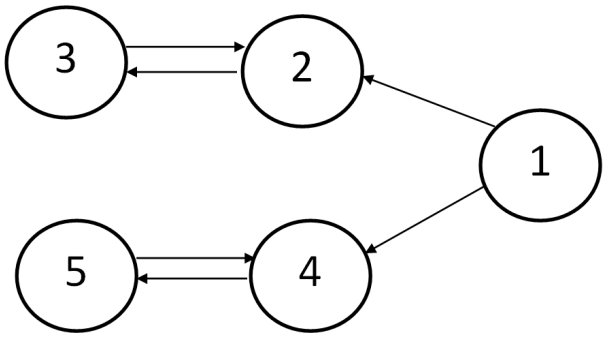


Figure 3: Simple graph $\mathbf{G_1}$ with two irreducible closed subsets.

its eigenvalues (using for example MATLAB) gives us $\lambda_2 = 0.85$, as we expected.

3 Test problems

3.1 Small test problem

We created our own small test problem. We introduce connectivity matrix $\mathbf{G_{test}}$. We will use Figure 4 in later sections to investigate link spamming. Note that W_{test} has one irreducible closed subset ($\{1, 2\}$) and two dangling nodes ($\{6\}$,$\{7\}$).

A variant of $\mathbf{G_{test}}$ is $\mathbf{G_{m2}}$, which will be introduced in Section 4.3.

[1] This, of course, is very interesting for our research and gives us a motivation to look at the second eigenvector.

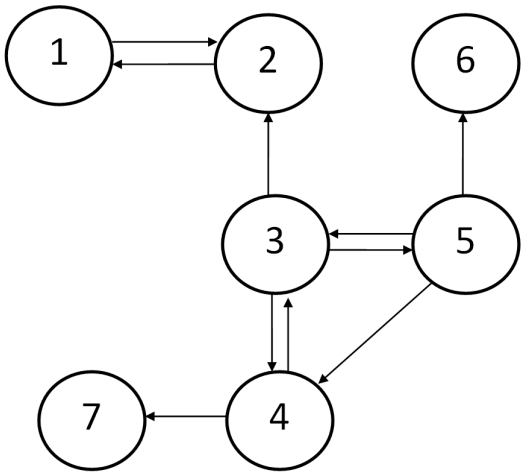


Figure 4: Directed graph W_{test} corresponding to the connectivity matrix $\mathbf{G_{test}}$.

3.2 Large test problems

Besides the small test problem we will use $\mathbf{G_{9914}}$ and $\mathbf{G_{2759}}$, publicly available at [16]. We should note that the source provides us (for both $n = 9914$ and $n = 2759$) with a matrix $\mathbf{A}$ and $\mathbf{P}$. Before we use these test problems, we should further investigate the structure of $\mathbf{A}$ and $\mathbf{P}$.

If we look at the upper left corner of both matrices $\mathbf{A}$ and $\mathbf{P}$, we note that the same elements of the matrices are filled. However, all elements of $\mathbf{A}$ are equal to one or zero, where $\mathbf{P}$ has elements between zero and one. Thus, $\mathbf{A}$ is the connectivity matrix and $\mathbf{P}$ is its stochastic matrix.

Now, a small test tells us that $\mathbf{P}$ is row-stochastic and thus, (because the same elements are filled), $\mathbf{A}$ is defined as $a_{ij} = 1$ if there is a hyperlink from page i to j and $a_{ij} = 0$ otherwise. We defined our connectivity matrix $\mathbf{G}$ with i and j switched. Furthermore, another small test tells us that $\mathbf{A}$ has self-referring hyperlinks (i.e., its diagonal contains ones). Therefore, we define $\mathbf{G_{9914}}$ (and similarly $\mathbf{G_{2759}}$) as follows:

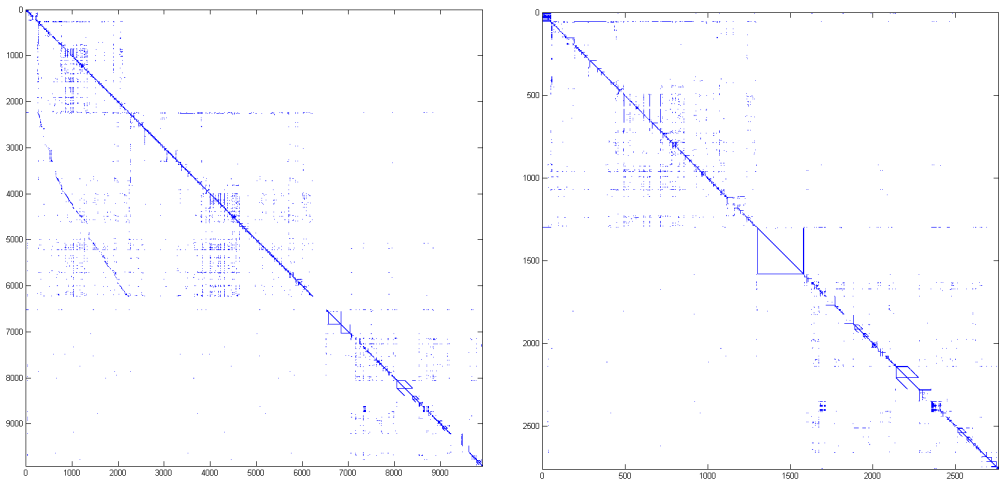
```
G9914 = A' - diag(diag(A));
```

See Figure 5 for the spy-plot of $\mathbf{G_{9914}}$ and $\mathbf{G_{2759}}$. Note that both $\mathbf{G_{9914}}$ and $\mathbf{G_{2759}}$ are very sparse in general and have some sort of dense 'diagonal', although its actual diagonal is defined zero per definition.

Furthermore, note that $\mathbf{G_{9914}}$ has quite some dangling nodes between 6000 and 7000, visible through the empty columns. Also, $\mathbf{G_{2759}}$ has a dense block in upper left corner.

Note that $\mathbf{P}$ from [16] is not defined as in this report, because the self-referring hyperlinks should have been removed from the corresponding $\mathbf{A}$ (or in this report: $\mathbf{G}$). Therefore, only $\mathbf{G_{2759}}$ and $\mathbf{G_{9914}}$ are used and other corresponding matrices are calculated by using for example `Ptwithoutdangling.m` as described in Appendix A.2.4.

A last note on test problems $\mathbf{G_{9914}}$ and $\mathbf{G_{2759}}$: with the MATLAB algorithm `tarjan.m` (see Appendix A.1.6) we tested to see whether there are irreducible closed subsets. We found one such subset in $\mathbf{G_{2759}}$, that is $\{1578, 1579, 1580, 1581\}$ and 175 subsets in $\mathbf{G_{9914}}$. However, these results can be unreliable. Adding an extra irreducible closed subset to $\mathbf{G_{9914}}$ (which should



(a) $\mathbf{G_{9914}}$. (b) $\mathbf{G_{2759}}$.

Figure 5: Two spy-plots using `spy(.)`

make a total of 176) gave only one irreducible closed subset.
Thus, unfortunately, we cannot be sure of the structure of these two test problems.

4 Introduction to link spamming

There are companies (such as SearchKing), also called link farms, which solely aim to increase the PageRank of its customers. Those customers pay to achieve a higher PageRank in order to reach more customers. One way to achieve a higher PageRank is to use other websites which refer with hyperlinks to one (group of) website(s). This is called link spamming.

It is Google's goal to filter out as much spam as possible, since the quality of a search algorithm mainly depends on a fair search result. This battle between link farms and Google will last forever and this is one of the reasons why Google made its current PageRank closed for public. Most recent literature limited itself to the basic (or slightly extended) PageRank algorithm. The only thing we know is that Google still *does* use the PageRank algorithm, but we do not exactly know how it influences the PageRank score or how it adapted to recent developments. Google needs to be one step ahead.

As mentioned before, the most significant way to achieve higher PageRank is to increase the number of important inlinks. Furthermore, Bianchini ([5],[6]) found more tools to boost PageRank. Let us first introduce some terminology:
The *target website* or *target page* is the website whose PageRank we want to increase.
A *target group* is a group of websites which we include to get a higher PageRank for our target website and allow to get a higher PageRank itself.

We cite from [5] (pages 2-3):

- The same content divided into many small pages yields a higher score than if it is concentrated into a single large page.
- Sinks should be avoided or carefully limited.

- External hyperlinks[2] must be limited and must belong to pages with many internal hyperlinks and/or with small PageRank.
- Pages that point to sinks should have a small score and/or many internal hyperlinks.

Before going in-depth on link spamming techniques, let us give some information about the concept of energy.

4.1 Introduction to energy

To give an alternative point of view when looking at the PageRank, we follow Bianchini [6] and introduce *energy*. Energy is a measure for the PageRank: the more energy a website has, the higher its PageRank. This point of view allows us to provide additional motivations and explanation. The energy of a set W_I of websites can be calculated in the following way:

$$E_I = |I| + E_I^{in} - E_I^{out} - E_I^{dn} \tag{4.1}$$

where
$|I|$: amount of pages in W_I,
E_I^{in}: energy from outside W_I going inside W_I,
E_I^{out}: energy from inside W_I going outside W_I,
E_I^{dn}: energy going to dangling nodes inside W_I.

When finding an efficient method for increasing the PageRank of a node, we neglect having influence on E_I^{in} (i.e. we assume we cannot find significant websites that will refer to our website). However, we do have influence on $|I|$, E_I^{out} and E_I^{dn}.

A typical structure to collect as much energy as possible is to have many websites, which all link to one website, whose PageRank increases. We introduce the following definition:

Definition 4.1. *Within the n-by-n connectivity matrix* $\mathbf{G}$*, node* j *is called a* ***promotion node*** *for node* k $(\neq j)$ *if for all* $i = 1, \ldots, n$ *the following equation holds:*

$$g_{ij} = g_{ji} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{else.} \end{cases}$$

Deriving from Definition 4.1, we conclude that a node is a promotion node for node k if it has one incoming and one outgoing hyperlink, only to and from node k. Recalling Figure 4, we recognize that node 1 is a promotion node for node 2 and this is the only promotion node within W_{test}.

See Figure 6 (similar to figure 7 in [6]). Intuitively, random surfers spend most time in node 1 having a large number of internal paths. In practice, node 1 could also have some external links or links to dangling nodes, but this structure ensures that the chance is small that the random surfer selects one of these links. Therefore, little energy is lost (and only) through node 1. Moreover, lots of energy is stored in this structure containing many websites, see (4.1).

To illustrate different methods how one can increase the PageRank of a website, we recall $\mathbf{G_{test}}$, graphically recalled in Figure 7. Our target website is node 4 (i.e. our goal is to increase

[2]External hyperlinks can be descripted as hyperlinks that refer outside the target group. Internal hyperlinks are hyperlinks referring within the target group.

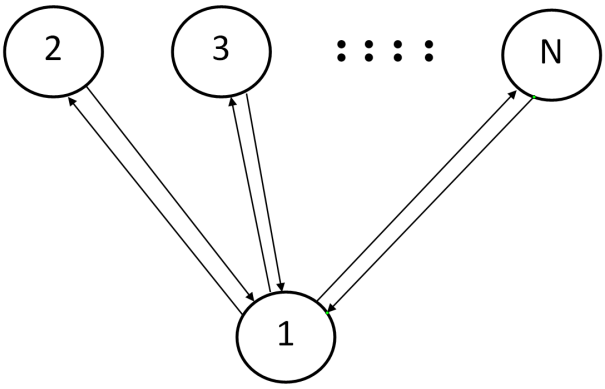


Figure 6: Collecting energy to node 1.

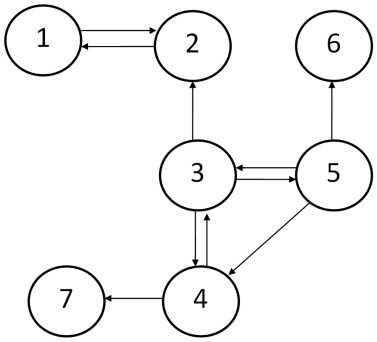


Figure 7: Test problem $\mathbf{G_{test}}$.

the PageRank of node 4). The current PageRank of $\mathbf{G_{test}}$ is given by

$$\mathbf{x}^{(1)} = \begin{pmatrix} 0.318 \\ 0.332 \\ 0.087 \\ 0.078 \\ 0.061 \\ 0.054 \\ 0.070 \end{pmatrix}.$$

From now on, we will focus on the PageRank for node 4. In this 'focused' PageRank we include the score for node 4 and the two highest nodes (excluding node 4), so in this case $\mathbf{x}^{(1)}_{\{4,2,1\}} = \begin{pmatrix} 0.078 & 0.332 & 0.318 \end{pmatrix}^T$.

We will now introduce two link spamming methods to increase the PageRank for node 4 of the test problem in Figure 7. For now, the target group is solely our target website 4.

4.2 Method 1

An effective way to increase your PageRank is to add promotion nodes. Speaking in terms of energy, as we add more promotion nodes, we have a higher amount of energy. It will increase the factor $|I|$, see (4.1). Also, removing dangling nodes is beneficial, for it reduces E_I^{dn}. Remember that a dangling node actually is a 'distributor', recalling Figure 1. In practice, we would (if possible) replace the dangling node with a promotion node, but with the same content.

Now, to increase the PageRank of node 4, we removed the dangling node and included three promotion nodes, see Figure 8. The (focused) PageRank for node 4 is

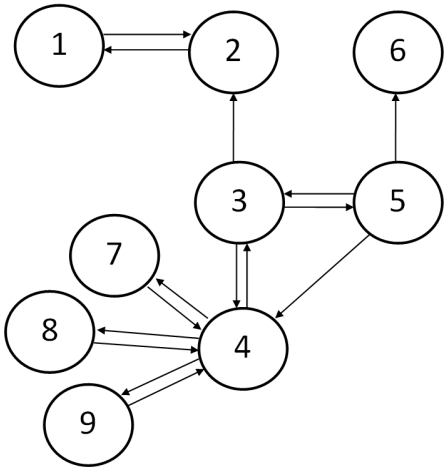


Figure 8: Method 1: Changes to be made to improve the PageRank for node 4.

$\mathbf{x}^{(\mathbf{1})}_{\{\mathbf{4},\mathbf{2},\mathbf{1}\}} = \begin{pmatrix} 0.228 & 0.213 & 0.201 \end{pmatrix}^T$. Note that the target group is set to $\{4, 7, 8, 9\}$ and that only node 4 managed to reach the top 3.

4.3 Method 2

Summarizing all conclusions of Section 4.1, the best way to increase your website is to create an irreducible closed subset. To do this, first remove all dangling nodes *and* external hyperlinks and then add sufficient promotion nodes. We refer to Figure 9, where we removed the dangling node and external hyperlink to node 3 and included one promotion node. The (focused) PageRank

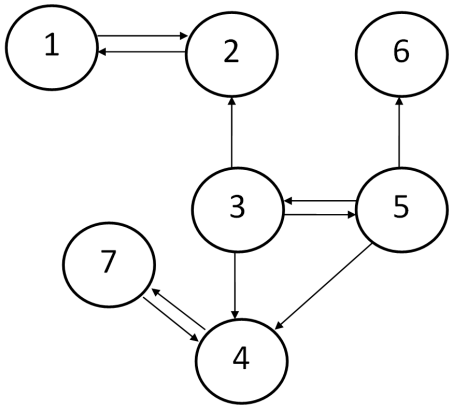


Figure 9: Method 2: Changes to be made to optimally improve the PageRank for node 4.

for node 4 is $\mathbf{x}^{(\mathbf{1})}_{\{\mathbf{4},\mathbf{7},\mathbf{2}\}} = \begin{pmatrix} 0.246 & 0.235 & 0.209 \end{pmatrix}^T$. Note that the target group is set to $\{4, 7\}$ and that the target group is top-ranked. We will call the corresponding connectivity matrix of Figure 9 from now on $\mathbf{G_{m2}}$. This matrix will become important, as it is a small test problem with two irreducible closed subsets.

The difference between method 1 and method 2 is that all outgoing hyperlinks are removed with method 2. Intuitively, when using method 2 we see that all outgoing energy is minimized

to zero. Formally, we recognize that only method 2 creates an irreducible closed subset. Note that removing the dangling nodes is crucial for creating an irreducible closed subset, because we defined the dangling nodes in $\mathbf{P^T}$ as so-called 'distributors' (see Figure 1).
Note that method 2 is far more efficient, because only one extra promotion node is required to sufficiently increase the PageRank of node 4 and furthermore, the target group gained more PageRank.

We will focus on detecting method 2, or in general the detection of: 'link spamming using irreducible closed subsets'.

4.4 Testing method 2 on test problems

We have already tested method 2 on $\mathbf{G_{test}}$ with target node 1 and we obtained $\mathbf{G_{m2}}$.

Let us try to promote the first website in $\mathbf{G_{2759}}$. We shall use the focused PageRank including the target website 1 and the two highest PageRanks (excluding the target website). The current score for $\mathbf{G_{2759}}$ is $\mathbf{x}^{(\mathbf{1})}_{\{\mathbf{1,54,786}\}} = 10^{-1}\begin{pmatrix} 0.018 & 0.195 & 0.132 \end{pmatrix}^T$.

We will use method 2 to increase the PageRank of node 1 with the MATLAB algorithm `addpromo.m` (see appendix A.2.1). Therefore, we introduce 80 promotion nodes for node 1. The PageRank score is now given by $\mathbf{x}^{(\mathbf{1})}_{\{\mathbf{1,54,786}\}} = 10^{-1}\begin{pmatrix} 0.191 & 0.189 & 0.127 \end{pmatrix}^T$. Note that we need to add 2.9% of the initial amount of nodes to obtain the highest PageRank for node 1. We will call $\mathbf{G_{2759}}$ *with* the 80 promotion nodes from now on $\mathbf{G_{2759+80}}$.

Now, we will do the same for the fourth website[3] in $\mathbf{G_{9914}}$. The current focused PageRank is $\mathbf{x}^{(\mathbf{1})}_{\{\mathbf{4,2264,8059}\}} = 10^{-2}\begin{pmatrix} 0.055 & 0.799 & 0.594 \end{pmatrix}^T$. Adding 80 promotion nodes give $\mathbf{x}^{(\mathbf{1})}_{\{\mathbf{4,2264,8059}\}} = 10^{-1}\begin{pmatrix} 0.079 & 0.078 & 0.059 \end{pmatrix}^T$. This time we needed only 0.8% promotion nodes. We will call $\mathbf{G_{9914}}$ *with* the 80 promotion nodes from now on $\mathbf{G_{9914+80}}$.

5 Detecting link spamming

First, we list measures or typical structures that can be used to limit the influence of spam. One should see this short list as a motivation and expectation for later sections.

1. The second eigenvectors have a certain structure and spam could be detectable through this structure ([4], page 6).

2. A stochastic personalization vector $\mathbf{v}$ could be helpful to control spamming ([8], page 16). This vector $\mathbf{v}$ is a variant of the n-vector $\mathbf{e}$ of all ones. Instead, $\mathbf{v}$ would describe a certain class of surfers. It is important to note that [8] uses row-stochastic vectors and matrices. Introducing $\mathbf{v}$ would replace the jump part $\frac{1-p}{n}\mathbf{e}\mathbf{e^T}$ with $(1-p)\mathbf{v}\mathbf{e^T}$, with $\mathbf{v}$ a stochastic vector.

As mentioned in Section 1 we have indications that the second eigenvectors are an artifact for detecting link spamming ([4]). Generally, we will consider the second eigenvectors and mostly

[3]Trying node 1 gave bad results when using `pagerankpow.m` of Moler [1]. The initial PageRank of node 1 is (relatively) very low and a lot of promotion nodes are needed. Maybe the large irreducible closed subset causes a failure.

ignore the personalization vector for this report. Furthermore, we will limit ourselves to link spamming method 2 in Section 4.3, which is an efficient technique and creates an irreducible closed subset.

5.1 The second eigenvector

We will look at the second eigenvector of $\mathbf{A}$ to retrieve some information about $\mathbf{G}$. If $\mathbf{G}$ has nodes within an irreducible closed subset, than these nodes will absorb a lot of energy. Recalling Figure 1, it is good to remember that a node is not in an irreducible closed subset as soon as there is a hyperlink to a dangling node.
First, we give a result of [4]:

Lemma 5.1. *The second eigenvector* $\mathbf{x^{(2)}}$ *of* $\mathbf{A}$ *must be an eigenvector* $\mathbf{y^{(i)}}$ *of* $\mathbf{P^T}$*, and the corresponding eigenvalue is* $\gamma_i = \lambda_i/p$*.*

The proof is given in [4] (Lemma 4).
We assume that the second eigenvalue of $\mathbf{A}$ is $\lambda_2 = p$ and thus $\mathbf{x^{(2)}}$ is an eigenvector of $\mathbf{P^T}$ corresponding to the eigenvalue $\gamma_1 = 1$ of $\mathbf{P^T}$. We know that $\gamma_1 = 1$ exists, because of the following lemma from [12] on page 126:

Lemma 5.2. *The multiplicity of the eigenvalue* 1 *for* $\mathbf{P^T}$ *is equal to the number of irreducible closed subsets of* $\mathbf{P^T}$*.*

The lemma's above provide us the information that looking at the second right eigenvector of the column-stochastic matrix $\mathbf{A}$ is equivalent to looking at the first left eigenvector of the row-stochastic matrix $\mathbf{P}$. Therefore, we shall take a closer look at $\mathbf{P}$, its so-called canonical form for reducible matrices and its first left eigenvector corresponding to $\gamma = 1$.

Let l be the number of irreducible closed subsets of $\mathbf{P}$. We know that the row-stochastic matrix $\mathbf{P}$ is reducible in general and we can rewrite $\mathbf{P}$ in canonical form ([13]) by renumbering the nodes. In general, each reducible matrix can be written in the following form:

$$\mathbf{P} \sim \begin{pmatrix} \mathbf{T_{11}} & \mathbf{T_{12}} \\ \mathbf{0} & \mathbf{T_{22}} \end{pmatrix} = \left(\begin{array}{cccc|cccc} \mathbf{P_{11}} & \mathbf{P_{12}} & \cdots & \mathbf{P_{1r}} & \mathbf{P_{1,r+1}} & \mathbf{P_{1,r+2}} & \cdots & \mathbf{P_{1m}} \\ \mathbf{0} & \mathbf{P_{22}} & \cdots & \mathbf{P_{2r}} & \mathbf{P_{2,r+1}} & \mathbf{P_{2,r+2}} & \cdots & \mathbf{P_{2m}} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P_{rr}} & \mathbf{P_{r,r+1}} & \mathbf{P_{r,r+2}} & \cdots & \mathbf{P_{rm}} \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P_{r+1,r+1}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{P_{r+2,r+2}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P_{mm}} \end{array}\right), \quad (5.1)$$

where $l = m - r$ and each $\mathbf{P_{11}}, \ldots, \mathbf{P_{rr}}$ is either irreducible or $[\mathbf{0}]_{\mathbf{1\times 1}}$, and $\mathbf{P_{r+1,r+1}}, \ldots, \mathbf{P_{mm}}$ are irreducible. First, note that each $\mathbf{P_{ij}}$ is a submatrix of the n-by-n matrix $\mathbf{P}$. Let us call the dimension of the block $\mathbf{T_{11}}$ $\tilde{r}$-by-$\tilde{r}$ and thus, the dimension of the block $\mathbf{T_{22}}$ is $(n-\tilde{r})$-by-$(n-\tilde{r})$.

The subset of states $\mathbf{P_{11}}, \ldots, \mathbf{P_{rr}}$ are called transient and once left, a transient class cannot be re-entered. The subset of states $\mathbf{P_{r+1,r+1}}, \ldots, \mathbf{P_{mm}}$ are called ergodic (irreducible closed subsets) and once entered, an ergodic class cannot be left. The subset $\mathbf{T_{12}}$ describes the probability of transitioning from some transient state to some ergodic state. Note that within an ergodic class the state vector could oscillate, but the chain will be trapped in this ergodic class

forever([13], pages 695-700).

We are interested in the first left eigenvector(s) of $\mathbf{P}$ and using its canonical form provides us more insight in the structure of $\mathbf{P}$. There are several ways to determine these eigenvectors and the different approaches are a matter of taste. We shall discuss two of them.
In Section 5.1.1 we will discuss the Cesàro sum and its application to $\mathbf{P}$. We shall also provide an example in the section thereafter. Next, we will discuss the calculation of the eigenvector of $\mathbf{P}$ in canonical form using linear algebra in Section 5.1.3.

5.1.1 The Cesàro sum

In this section we will discuss the Cesàro sum, which can intuitively be explained as the average distribution matrix. Because we do not necessarily have a unique first eigenvector for $\mathbf{P}$ as explained later in this report, we will search for the matrix $\mathbf{C}$ that send any random vector to some eigenvector of $\mathbf{P}$. The matrix $\mathbf{C}$ is the Cesàro sum, defined as

$$\mathbf{C} = \lim_{k\to\infty} \frac{\mathbf{I} + \mathbf{P} + \cdots + \mathbf{P}^{k-1}}{k}. \tag{5.2}$$

Let $\tilde{\mathbf{y}}^{\mathbf{T}}_{\mathbf{r+j}}$ be the left eigenvector for $\mathbf{P}_{\mathbf{r+j},\mathbf{r+j}}$ $(1 \leq j \leq l)$ corresponding to $\gamma = 1$. All elements of $\tilde{\mathbf{y}}^{\mathbf{T}}_{\mathbf{r+j}}$ within block $\mathbf{P}_{\mathbf{r+j},\mathbf{r+j}}$ are strictly positive and all other elements are 'padded' with zeros to appropriate size.

Furthermore, we know that the uniform vector $\mathbf{e}_{\mathbf{r+j}}$ is the right eigenvector for each $\mathbf{P}_{\mathbf{r+j},\mathbf{r+j}}$ $(1 \leq j \leq l)$, because $\mathbf{P}$ is row-stochastic and 1 is an eigenvalue of each irreducible $\mathbf{P}_{\mathbf{r+j},\mathbf{r+j}}$.

We will use the Cesàro sum instead of $\lim_{k\to\infty} \mathbf{P}^k$ to determine the eigenvector $\mathbf{y}^{\mathbf{T}}$, because we are not sure whether the ergodic class is periodic. In other words, the distribution in an irreducible closed subset can alterate and furthermore, an eigenvector of $\mathbf{P}$ corresponding to $\gamma_i = 1$ is a linear combination of the eigenvectors of $\mathbf{P}_{\mathbf{r+j},\mathbf{r+j}}$ padded with zeros to appropriate size. The Cesàro sum will ensure us convergence. For example, $\mathbf{P_{jj}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ could be an irreducible closed subset, but its distribution would alterate between its two states. If we use the power method $\mathbf{y^T_{k+1}} = \mathbf{y^T_k P_{jj}}$ for any random stochastic starting vector $\mathbf{u}^{\mathbf{T}}$, then every even (respectively odd) iteration will be the same, but no convergence would occur. However, the long-run fraction of time that the chain spends in each state will have an average distribution: $\mathbf{y}^{\mathbf{T}} = \mathbf{u}^{\mathbf{T}}\mathbf{C} = \mathbf{u}^{\mathbf{T}} \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix} = \begin{pmatrix} 0.5 & 0.5 \end{pmatrix}$.
Note that the Cesàro sum is a unique matrix with an average distribution, but that in general the randomly chosen vector $\mathbf{u}^{\mathbf{T}}$ does still have influence on the result (although is does not have influence in this example).

The limit of the Cesàro sum exists for all stochastic matrices ([13], page 698) and, again citing [13], the long-run fraction of time that the chain spends in state S_j is y_j, which is the j^{th} component of the Cesàro limit or, equivalently, the j^{th} component of the left Perron vector for $\mathbf{P}$. We will now calculate the limit of the Cesàro sum for matrix $\mathbf{P}$:

$$\lim_{k\to\infty} \frac{\mathbf{I} + \mathbf{P} + \cdots + \mathbf{P}^{k-1}}{k} = \begin{pmatrix} \mathbf{0} & (\mathbf{I} - \mathbf{T_{11}})^{-1}\mathbf{T_{12}}\mathbf{E} \\ \mathbf{0} & \mathbf{E} \end{pmatrix} = \mathbf{C}, \tag{5.3}$$

where

$$\mathbf{E} = \begin{pmatrix} \mathbf{e_{r+1}}\tilde{\mathbf{y}}^{\mathbf{T}}_{\mathbf{r+1}} & & \\ & \ddots & \\ & & \mathbf{e_m}\tilde{\mathbf{y}}^{\mathbf{T}}_{\mathbf{m}} \end{pmatrix}. \tag{5.4}$$

Here, $\mathbf{C}$ is the projector onto Null$(\mathbf{I}-\mathbf{P})$ along Range$(\mathbf{I}-\mathbf{P})$ ([13], page 698).

Now, for the Cesàro sum we know that for any initial vector $\mathbf{u^T}$ the following equation holds

$$\begin{aligned} \overline{\mathbf{y^T}} &= \lim_{k\to\infty} \mathbf{u^T}\frac{\mathbf{I}+\mathbf{P}+\cdots+\mathbf{P}^{k-1}}{k} = \mathbf{u^T C} = \begin{pmatrix} \mathbf{u_1^T}, & \mathbf{u_2^T} \end{pmatrix} \begin{pmatrix} \mathbf{0} & (\mathbf{I}-\mathbf{T_{11}})^{-1}\mathbf{T_{12}E} \\ \mathbf{0} & \mathbf{E} \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{0}, & \mathbf{u_1^T}(\mathbf{I}-\mathbf{T_{11}})^{-1}\mathbf{T_{12}E} + \mathbf{u_2^T E} \end{pmatrix}. \end{aligned} \tag{5.5}$$

Here, the components of $\overline{\mathbf{y^T}}$ provide the expected proportion of time the chain spent in each state. Note that the left eigenvectors $\tilde{\mathbf{y}}^{\mathbf{T}}_{\mathbf{r+j}}$ of each $\mathbf{P_{r+j,r+j}}$ are strictly positive[4]. Thus, each block $\mathbf{e_{r+j}}\tilde{\mathbf{y}}^{\mathbf{T}}_{\mathbf{r+j}}$ is non-zero in each column. Moreover, all columns have at least one non-zero element, because all elements of $\mathbf{T_{22}}$ are part of an irreducible closed subset (if there was a zero column, then that state would not be in $\mathbf{T_{22}}$).[5]

Now assume that all elements of $\mathbf{u^T}$ have a component along the eigenvector $\tilde{\mathbf{y}}^{\mathbf{T}}_{\mathbf{r+j}}$ (corresponding to $\gamma = 1$) of each block $\mathbf{P_{r+1,r+1}}, \ldots, \mathbf{P_{nn}}$, i.e., $\mathbf{u^T}$ is not in the left null space of $\mathbf{E}$, then we have: [6]

$$\mathbf{y^T} = \begin{pmatrix} \mathbf{0_{[1\times\tilde{r}]}}, & y_{\tilde{r}+1}, \cdots, y_n \end{pmatrix}, \tag{5.6}$$

with $y_{\tilde{r}+1}, \cdots, y_n \neq 0$. Remember that $\mathbf{y^T}$ is not unique and everything is in canonical form.

5.1.2 Example

In this example we will illustrate how a second eigenvector of $\mathbf{A}$ can be determined in the analytical way described above, using $\mathbf{G_{m2}}$ in Figure 9. First, we will renumber the nodes to get the canonical form as in (5.1). For a graphical representation of the renumbering, we refer to Figure 10.

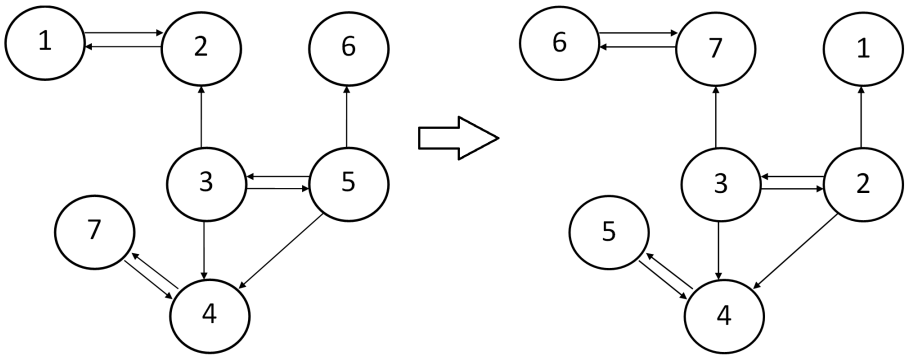


Figure 10: Renumbering the nodes of Figure 9 to canonical form.

[4] All elements which are in the block $\mathbf{P_{r+j,r+j}}$ are strictly positive, other elements are zero

[5] Note that $\mathbf{E}$ can be written as $\mathbf{I}$ (for another purpose), so certainly every column has a non-zero element ([13], page 699).

[6] Choosing $\mathbf{u^T}$ randomly should satisfy this assumption, because the chance that a random vector will not satisfy is practically zero.

Thus, rewriting $\mathbf{P}$ to $\mathbf{P_{canon}}$:

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\sim \left(\begin{array}{ccc|cc|cc} \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right) = \mathbf{P_{canon}}. \tag{5.7}$$

Let us take a closer look at $\mathbf{P_{canon}}$ in (5.7). First, we recognize the block on the lower left side of all zeros. Also, it is clear that we have two irreducible closed subsets ($\mathbf{P_{22}}$ and $\mathbf{P_{33}}$), which can be reached by $\mathbf{T_{12}}$. However, $\mathbf{T_{12}}$ includes all other nodes that are not in $\mathbf{T_{22}}$ and thus, $\mathbf{P_{11}}$ is the only block in the upper left side of $\mathbf{P_{canon}}$ (i.e., there are no nodes that do not refer to one of the irreducible closed subsets). Note that $\mathbf{P_{11}}$ is irreducible, but not closed and therefore represents the transient class. $\mathbf{P_{22}}$ and $\mathbf{P_{33}}$ represent the ergodic classes.

Let us calculate the Cesàro sum for $\mathbf{P_{canon}}$ in (5.7). Using the MATLAB function described in Appendix A.2.2, we find for $k = 50$ and $k = 500$ the following Cesàro sums:

$$\mathbf{C_{50}} = \begin{pmatrix} 0.026 & 0.006 & 0.006 & 0.263 & 0.262 & 0.219 & 0.220 \\ 0.010 & 0.025 & 0.010 & 0.343 & 0.224 & 0.140 & 0.140 \\ 0.003 & 0.008 & 0.023 & 0.278 & 0.270 & 0.205 & 0.213 \\ 0 & 0 & 0 & 0.500 & 0.500 & 0 & 0 \\ 0 & 0 & 0 & 0.500 & 0.500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.500 & 0.500 \\ 0 & 0 & 0 & 0 & 0 & 0.500 & 0.500 \end{pmatrix},$$

$$\mathbf{C_{500}} = \begin{pmatrix} 0.003 & 0.001 & 0.001 & 0.272 & 0.272 & 0.227 & 0.227 \\ 0.001 & 0.003 & 0.001 & 0.351 & 0.350 & 0.147 & 0.147 \\ 0.000 & 0.001 & 0.002 & 0.283 & 0.283 & 0.215 & 0.216 \\ 0 & 0 & 0 & 0.500 & 0.500 & 0 & 0 \\ 0 & 0 & 0 & 0.500 & 0.500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.500 & 0.500 \\ 0 & 0 & 0 & 0 & 0 & 0.500 & 0.500 \end{pmatrix}. \tag{5.8}$$

We find (for increasing k) that $\mathbf{C} \to \begin{pmatrix} \mathbf{0} & \mathbf{Z} \\ \mathbf{0} & \mathbf{E} \end{pmatrix}$. Note that the convergence is relatively slow, since we take an average distribution each iteration. Now, let us choose the stochastic vector $\mathbf{u^T}$ randomly:

```
u = rand(1,7);
u = u/sum(u);
```

For our example, we find the 'averaged eigenvectors' $\overline{\mathbf{y_{50}^T}}$ and $\overline{\mathbf{y_{500}^T}}$:

$$\overline{\mathbf{y_{50}^T}} = \mathbf{u^T} \cdot \mathbf{C_{50}} = \begin{pmatrix} 0.008 \\ 0.007 \\ 0.004 \\ 0.354 \\ 0.351 \\ 0.138 \\ 0.138 \end{pmatrix}^T, \overline{\mathbf{y_{500}^T}} = \mathbf{u^T} \cdot \mathbf{C_{500}} = \begin{pmatrix} 0.001 \\ 0.001 \\ 0.000 \\ 0.358 \\ 0.358 \\ 0.141 \\ 0.141 \end{pmatrix}^T . \tag{5.9}$$

As k increases we recognize for $\overline{\mathbf{y^T}}$ that all nodes in $\mathbf{P_{11}}$ (the transient class) go to zero and all nodes in $\mathbf{P_{22}}$ and $\mathbf{P_{33}}$ (the ergodic classes) are unequal to zero.

5.1.3 Calculation of the eigenvector from the canonical form

In this section we will discuss the calculation of the first eigenvector of $\mathbf{P}$ in canonical form. Let us recall $\mathbf{P}$ in canonical form:

$$\mathbf{P} \sim \begin{pmatrix} \mathbf{T_{11}} & \mathbf{T_{12}} \\ \mathbf{0} & \mathbf{T_{22}} \end{pmatrix} = \left(\begin{array}{cccc|cccc} \mathbf{P_{11}} & \mathbf{P_{12}} & \cdots & \mathbf{P_{1r}} & \mathbf{P_{1,r+1}} & \mathbf{P_{1,r+2}} & \cdots & \mathbf{P_{1m}} \\ \mathbf{0} & \mathbf{P_{22}} & \cdots & \mathbf{P_{2r}} & \mathbf{P_{2,r+1}} & \mathbf{P_{2,r+2}} & \cdots & \mathbf{P_{2m}} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P_{rr}} & \mathbf{P_{r,r+1}} & \mathbf{P_{r,r+2}} & \cdots & \mathbf{P_{rm}} \\ \hline \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P_{r+1,r+1}} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{P_{r+2,r+2}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P_{mm}} \end{array}\right) . \tag{5.10}$$

We are looking for the eigenvectors corresponding to eigenvalues $\gamma_i = 1$. These eigenvalues (or: this eigenvalue) exist, because we were able to write $\mathbf{P}$ in canonical form with $\mathbf{T_{22}}$ non-empty. Remember that each block $\mathbf{P_{r+j,r+j}}$ $(1 \leq j \leq l)$ in $\mathbf{T_{22}}$ has eigenvalue 1 (see Section 5.1.1).

Now, let us directly calculate the eigenvector(s) corresponding to eigenvalue $\gamma_i = 1$ for $\mathbf{P}$ in canonical form:

$$\begin{aligned} \mathbf{y^T P} &= \gamma_i \mathbf{y^T} \\ \begin{pmatrix} \mathbf{y_1^T}, & \mathbf{y_2^T} \end{pmatrix} \begin{pmatrix} \mathbf{T_{11}} & \mathbf{T_{12}} \\ \mathbf{0} & \mathbf{T_{22}} \end{pmatrix} &= \begin{pmatrix} \mathbf{y_1^T}, & \mathbf{y_2^T} \end{pmatrix} \\ \Rightarrow \begin{cases} \mathbf{y_1^T T_{11}} & = \mathbf{y_1^T} \\ \mathbf{y_1^T T_{12} + y_2^T T_{22}} & = \mathbf{y_2^T} \end{cases} \end{aligned} \tag{5.11}$$

We know that $(\mathbf{T_{11}} - \mathbf{I})$ is not singular, since $|\gamma_i| < 1$ for $\mathbf{T_{11}}$ (refer to [13], page 698). Thus, Equation (5.11) implies $\mathbf{y_1^T} = \mathbf{0}$. It follows that $\mathbf{y_2^T T_{22}} = \mathbf{y_2^T}$.

We get $\mathbf{y_2^T}(\mathbf{T_{22}} - \mathbf{I}) = \mathbf{0}$, where $(\mathbf{T_{22}} - \mathbf{I})$ is singular and thus, $\mathbf{y_2^T}$ is an eigenvector of $\mathbf{T_{22}}$ corresponding to $\gamma_i = 1$. Moreover, this eigenvector is a linear combination of the eigenvectors of $\mathbf{P_{r+j,r+j}}$ $(1 \leq j \leq l)$ corresponding to $\gamma_i = 1$, where each eigenvector of of $\mathbf{P_{r+j,r+j}}$ is padded with zeros to get the appropriate size.

Furthermore, no element of $\mathbf{y_2^T}$ is equal to zero, because $\mathbf{y_2^T}$ is an eigenvector of each block $\mathbf{P_{r+j,r+j}}$ $(1 \leq j \leq l)$ and all blocks $\mathbf{P_{r+j,r+j}}$ are row-stochastic (and thus no zero-row exists in

$\mathbf{T_{22}}$).

Putting everything together, we calculated the (non-unique) left eigenvector $\mathbf{y^T}$ of $\mathbf{P}$ in canonical form:

$$\mathbf{y^T} = \begin{pmatrix} \mathbf{y_1^T} & \mathbf{y_2^T} \end{pmatrix} = \begin{pmatrix} \mathbf{0_{[1\times\tilde{r}]}} & y_{\tilde{r}+1}, \cdots, y_n \end{pmatrix}, \tag{5.12}$$

with $y_{\tilde{r}+1}, \cdots, y_n \neq 0$.

5.2 Structure of irreducible closed subsets

We will now try to detect link spamming by recalling $\mathbf{G_{m2}}$ in Figure 9. We explicitly calculate matrix $\mathbf{A}$ as in Equation (2.2) and find the (unscaled) second eigenvector $\mathbf{x^{(2)}}$ corresponding to $\lambda_2 = 0.85$:

$$\mathbf{x^{(2)}} = \begin{pmatrix} 0.5000 \\ 0.5000 \\ 0.0000 \\ -0.5000 \\ 0.0000 \\ 0.0000 \\ -0.5000 \end{pmatrix}. \tag{5.13}$$

This eigenvector $\mathbf{x^{(2)}}$ is not unique, but a linear combination of the eigenvectors corresponding to the irreducible closed subsets. Equation (5.13) shows us a certain hypothesis. Before we express that hypothesis, we consider $\mathbf{G_{2759+80}}$. Again calculating the eigenvalues gives us $\lambda_2 = 0.85$ with corresponding eigenvector $\mathbf{x^{(2)}} = \begin{pmatrix} 0.926 \\ 0 \\ \vdots \\ 0 \\ -0.179 \\ \vdots \\ -0.179 \\ 0 \\ \vdots \\ 0 \\ 0.012 \\ \vdots \\ 0.012 \end{pmatrix}$.

We found for both test problems that $x_j^{(2)} = 0$ if j is not in an irreducible closed subset and $x_j^{(2)} \neq 0$ if j is in an irreducible closed subset.

We will now introduce the following theorem:

Theorem 5.3. *Each eigenvalue $\lambda_i = p$ of $\mathbf{A}$ has a corresponding right eigenvector $\mathbf{x^{(i)}} = (x_1, \cdots, x_n)$, which has the following properties:*

$$\begin{cases} x_j \neq 0 & \text{if node } j \in \text{ irreducible closed subset,} \\ x_j = 0 & \text{if node } j \notin \text{ irreducible closed subset.} \end{cases}$$

Proof. Write $\mathbf{P}$ in canonical form (5.1). Note that renumbering the columns and rows of $\mathbf{P}$ has no influence on the eigenvalues and the eigenvectors are permuted in analogy to the renumbering.

Calculate the Cesàro sum (refer to (5.3)) or directly calculate the eigenvector corresponding to $\gamma_i = 1$ (refer to (5.11)).
The non-unique left eigenvector $\mathbf{y^T}$ (in canonical form) corresponding to $\gamma_1 = 1$ is equal to $\mathbf{y^T} = (\mathbf{0_{[1\times r]}}, y_{\tilde{r}+1}, \cdots, y_n)$ with $y_{\tilde{r}+1}, \cdots, y_n \neq 0$ (refer to Equation (5.6)). Putting $\mathbf{P}$ back in original form will renumber the elements of $\mathbf{y^T}$ likewise. The non-zero elements $y_{\tilde{r}+1}, \cdots, y_n$ still correspond to the irreducible closed subsets and the zero elements do not.

We will now use Lemma 5.1 reversed. This is allowed as can be easily seen by reversing the proof of Lemma 4 in [4]. Thus, any left eigenvector $\mathbf{y^T}$ of $\mathbf{P}$ corresponding to $\gamma_1 = 1$ is a right eigenvector $\mathbf{x^{(i)}}$ of $\mathbf{A}$ corresponding to $\lambda_i = p$. Thus, $\mathbf{x^{(i)}} = (x_1, \cdots, x_n)$ has the following properties:
$\begin{cases} x_j \neq 0 & \text{if node } j \in \text{ irreducible closed subset,} \\ x_j = 0 & \text{if node } j \notin \text{ irreducible closed subset.} \end{cases}$ □

5.3 Other structures

There are ways to camouflage promotion structures. For our specific detection method, it is sufficient to add any outgoing link (i.e. creating a promotion set which is not an irreducible closed subset). We recognize such a structure as link spamming method 1 in Section 4.2, which is not detectable in the same way as method 2. It would also be sufficient to add a dangling node, as this would act as a distributor.

Moreover, let us introduce a promoting set W_P, which solely aim to increase the PageRank of another set websites W_I. This promoting set has no qualitative content, but all nodes refer to the set W_I (which wants to get a higher PageRank). Further, W_P is an irregular (various PageRanks and a variety of interconnections) and interconnected set with the property that all nodes in W_P refer to all nodes in set W_I. This set W_I will also have interconnected nodes, outgoing links and dangling nodes and will behave just like any other set websites. On the other hand, W_I get buffed by its promoting set W_P and receive an unfair higher PageRank.

As one might notice, it is hard to detect link spamming in this form. However, this form of link spamming requires an enormous amount of effort (and probably money) and thus is less attractive for businesses.

6 Algorithms for computing the second eigenvector

In the previous section we gave insight in the structure of the second eigenvector of $\mathbf{A}$. This section we will discuss some algorithms about how to compute this eigenvector. The algorithms can be found in Appendix A.1, written in MATLAB.

6.1 The block power method

The block power method differs from the 'normal' power method as it uses several vectors, instead of a vector $\mathbf{u_k}$. We start with r orthonormal vectors, multiplying them all with $\mathbf{A}$ and orthogonalize them again. This way we will find r different eigenvalues and their associated eigenvectors at the same time.

First, we define a random n-by-r matrix $\mathbf{Z_0}$ and then we iterate as follows:

for $k = 0, 1, 2, \ldots$
$\mathbf{Q_k} = \text{orth}(\mathbf{Z_k})$;
$\mathbf{Z_{k+1}} = \mathbf{AQ_k}$;
end

Now, determine the eigenvalues and associated eigenvectors for the system

$[\mathbf{Y}, \mathbf{D}] = \text{eig}(\mathbf{Q_k^T AQ_k})$;
$\mathbf{X} = \mathbf{QY}$;

and scale $\mathbf{x^{(1)}}$ such that $||\mathbf{x^{(1)}}||_1 = 1$. See Appendix A.1.1 for more information.

We would expect that the convergence ratio for the first eigenvector is now given by $|\lambda_{r+1}|/|\lambda_1|$. Knowing that $|\lambda_1| \geq \ldots \geq |\lambda_n|$, we would conclude that the block power method converges at least as fast as the basic power method. However, one should take into account that the second eigenvector is not necessarily unique and above all, several eigenvalues of $\mathbf{A}$ could be near p. Furthermore, when r increases, we can expect increasing calculation time for each iteration.

We tested this method for $\mathbf{G_{m2}}$ using `[x,y,iter] = blockpower(Gmethod2,4)` and we found:

$$\mathbf{x} = \begin{pmatrix} 0.203 & 0.500 & -0.316 & -0.682 \\ 0.209 & 0.500 & 0.316 & 0.682 \\ 0.036 & -0.000 & -0.000 & 0.000 \\ 0.246 & -0.500 & 0.632 & 0.186 \\ 0.036 & -0.000 & -0.000 & -0.000 \\ 0.036 & -0.000 & -0.000 & 0.000 \\ 0.235 & -0.500 & -0.632 & -0.186 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 1.000 \\ 0.850 \\ -0.850 \\ -0.850 \end{pmatrix}, \quad iter = 22. \tag{6.1}$$

Note that the second eigenvalue is 0.85. The second eigenvector shows us clearly that node 1 and 2 as well as node 4 and 7 form two irreducible closed subsets. This second eigenvector is not unique, of course.
We checked the first eigenvector and it is correct.

Furthermore, we set the maximum amount of iterations to 3000 and we stop iterating if the eigenvalues converge within a small tolerance or if the maximum amount of iterations is reached.

A limitition of this method, as already mentioned, is that the second eigenvector is not unique in general. Furthermore, another limition of the algorithm is that the performance of the algorithm is likely to decrease when trying larger matrices. This is due to input parameter r (the amount of columns of the block matrix $\mathbf{Q}$). The following test problem should be enlightening.

Running the algorithm for $\mathbf{G_{2759+80}}$ returns:
`[x,y,iter] = blockpower(G2759_80,4);`

$$\mathbf{y} = \begin{pmatrix} -0.849 \\ 1.000 \\ 0.849 \\ 0.850 \end{pmatrix}, \quad iter = 1346. \tag{6.2}$$

with the eigenvector corresponding to 0.850 equal to $\mathbf{x_{1,1578,1580}} = \begin{pmatrix} -0.926 & 0.179 & 0.179 \end{pmatrix}^T$. Here we used $r = 4$, but some other tests gave that $r = 3$ or $r = 5$ work as well. Anyway, in the

successful attempt above we see that the link spamming on node 1 is detected.

In general, the results above are not unique as a result of the input parameter r. As a consequence, the block power method as implemented in `blockpower.m` gives various results for different r. For example, for test problem $\mathbf{G_{9914+80}}$ it was hard to find a working r. Eventually, we found that $r = 25$ gave an eigenvalue 0.832, close to p. The corresponding eigenvector had a relative high value in the link spamming node 4, but it was not the largest element in the vector. We think this failure is due to the large amount of irreducible closed subsets in $\mathbf{G_{9914+80}}$ and therefore, r should be increased significantly. More research should be done in investigating these failings. For more insight in the algorithm, we refer to Appendix A.1.1.

6.2 Direct solution

Suppose we have a set W of websites with at least two irreducible closed subsets, so we know $\lambda_2 = p$. We define $\mathbf{A}$ the usual way and rewrite $\mathbf{A} = p\mathbf{GD} + \mathbf{ez^T}$ as in (2.13).
We are interested in $\mathbf{x^{(2)}}$ (the eigenvector(s) corresponding to λ_2) and we know that for this vector the following equations holds:

$$\begin{aligned} \lambda\mathbf{x} &= \mathbf{Ax} \\ p\mathbf{x} &= (p\mathbf{GD} + \mathbf{ez^T})\mathbf{x} \\ p\mathbf{x} &= p\mathbf{GDx} + \mathbf{ez^T}\mathbf{x} \end{aligned} \tag{6.3}$$

$$\begin{aligned} &\Rightarrow (p\mathbf{I} - p\mathbf{GD})\mathbf{x} = \beta\mathbf{e} \\ &\Rightarrow (\mathbf{I} - \mathbf{GD})\mathbf{x} = \beta'\mathbf{e} \end{aligned} \tag{6.4}$$

with $\beta' = \frac{\beta}{p}$. Similar as when solving (2.16), $\beta' = \frac{z^T x}{p}$ is set to 1 temporary.

In general, the matrix $(\mathbf{I} - \mathbf{GD})$ (also: $(\mathbf{I} - \mathbf{P^T})$) is singular, because we assume that $\mathbf{GD}$ has irreducible closed subsets, so that $\gamma_1 = 1$. Therefore, the system $(\mathbf{I} - \mathbf{GD})$ has no unique solution. However, in practice the following calculation will blow up $\mathbf{x}$ in the right direction:

$$\mathbf{x} = (0.99999 * \mathbf{I} - \mathbf{P^T})\backslash\mathbf{e} \tag{6.5}$$

This way, all elements in an irreducible closed subset will blow up compared to the one that are not in an irreducible closed subset.
For example, when testing the MATLAB function `direct.m` (A.1.2) on our test problem $\mathbf{G_{m2}}$, then we get: $\mathbf{x} = \begin{pmatrix} 0.227 \\ 0.227 \\ -0.000 \\ 0.273 \\ -0.000 \\ -0.000 \\ 0.273 \end{pmatrix}$. Note that `direct.m` also scales the returned vector to a stochastic vector and thus, this vector does not sum up to zero. This algorithm does not return a second eigenvector of $\mathbf{A}$, but blows up the link spamming elements.
When testing this algorithm on $\mathbf{G_{2759+80}}$ and $\mathbf{G_{9914+80}}$ we do get good results, but the calculation time increases significantly for larger matrices. The limitation of this algorithm is obvious: direct calculation is not possible for really large matrices.

6.3 Adjusting the power method

In this section we will discuss three variants of the power method to determine the second eigenvector of $\mathbf{A}$.

6.3.1 Simple power method on matrix P

Instead of looking at the second eigenvector of $\mathbf{A}$, another option is to look directly at the first eigenvector of $\mathbf{P^T}$ using the power method. Thus, we find the irreducible closed subsets of $\mathbf{P^T}$ directly. The most obvious algorithm is just iterating $\mathbf{u_{k+1}} = \mathbf{P^T u_k}$ with $\mathbf{u_0}$ a random stochastic vector.

A disadvantage of this method is that we cannot assure a proper convergence. Note that we can assume that (an arbitrarily) $\mathbf{P^T}$ has more than one irreducible closed subsets and thus, that $\gamma = 1$ has multiplicity greater than one. The situation we discussed in Section 2.2, where $\lambda_1 = 1$ is unique, is not applicable. We now have a set of eigenvalues which will dominate and no unique eigenvector exists (i.e., in general the eigenvector will not converge to a unique vector, but will oscillate to a linear combination of eigenvectors of $\mathbf{P^T}$ corresponding to $\gamma = 1$).

Nevertheless, we will follow the same approach as in Section 2.2 to show this method's limitation.
Suppose that $\mathbf{P^T}$ has eigenvalues $\gamma_1, \ldots, \gamma_n$ with a full set of associated eigenvectors $\mathbf{y^{(1)}}, \ldots, \mathbf{y^{(n)}}$ and that $\mathbf{u_0}$ can be expressed as a linear combination of eigenvectors. Then the following equation holds, due to Theorem 2.2:
$\mathbf{P^T}^k \mathbf{u_0} = \mathbf{u_k} = c_1\gamma_1\mathbf{y^{(1)}} + \ldots + c_n\gamma_n^k\mathbf{y^{(n)}}$.

Furthermore, suppose that $1 = \gamma_1 = \ldots = \gamma_i > \gamma_{i+1} \geq \ldots \geq \gamma_n$ (i.e., $\mathbf{P^T}$ has i irreducible closed subsets). Then the components $c_1\gamma_1^k\mathbf{y^{(1)}}, \ldots, c_i\gamma_i^k\mathbf{y^{(i)}}$ will gradually become dominant (likewise as in (2.11)):

$$\begin{aligned}\mathbf{u_k} = \frac{\mathbf{u_k}}{\gamma_1^k} &= c_1\mathbf{y^{(1)}} + \ldots + c_i\mathbf{y^{(i)}} + c_{i+1}\left(\frac{\gamma_{i+1}}{\gamma_1}\right)^k\mathbf{y^{(i+1)}} + \ldots + c_n\left(\frac{\gamma_n}{\gamma_1}\right)^k\mathbf{y^{(n)}} \\ &= c_1\mathbf{y^{(1)}} + \ldots + c_i\mathbf{y^{(i)}} + c_{i+1}\gamma_{i+1}^k\mathbf{y^{(i+1)}} + \ldots + c_n\gamma_n^k\mathbf{y^{(n)}}.\end{aligned} \tag{6.6}$$

The convergence factor is determined by the second most dominant term, which is $c_{i+1}\gamma_{i+1}^k\mathbf{y^{(i+1)}}$ and the rate of convergence is equal to $|\gamma_{i+1}|$. Here, we use that $\gamma_1, \ldots, \gamma_i = 1$. However, we do not know the value of the largest eigenvalue unequal to one and thus, the rate of convergence is unknown.

See Section 6.6 for results and Appendix A.1.3 for `simpleP.m`.

6.3.2 Adaptation to Moler's power method

We slightly adjusted the algorithm `pagerankpow.m` of [1] by ignoring the 'jumping' part. We refer to A.1.4.

Let us illustrate what this algorithm shows by recalling $\mathbf{G_{test}}$ (see also Figure 4). After 17 iterations in `powerP.m` we find $\mathbf{x} = \begin{pmatrix} 0.480 & 0.498 & 0.006 & 0.005 & 0.004 & 0.003 & 0.005 \end{pmatrix}^T$. Note that only node 1 and 2 have relative high value and the remaining (non-spamming) website have value almost equal to zero.

If we test `powerP.m` on $\mathbf{G_{2759+80}}$ we can only detect the link spamming on node 1 (and not the corresponding promotion nodes).

Testing on $\mathbf{G_{9914+80}}$ gives us the following result:
$\mathbf{x_{4,8059,8057}} = \begin{pmatrix} 0.024 & 0.017 & 0.015 \end{pmatrix}^T$ with 600 iterations. Thus, the detection should be possible, which can be recognized in the returned vector.

It is important that this algorithm also tries to recognize link spamming nodes by determining which terms are non-zero (in an irreducible closed subset) and which are zero (not in an irreducible closed subset). This is the only implemented algorithm with this functionality and it is hard to find a suitable stopping criterium.
The test problem $\mathbf{G_{9914+80}}$ reached the maximum tolerated amount of iteration and the returned vector $\mathbf{x}$ shows us some link spamming nodes, but the vector **spam** (refer to Appendix A.1.4) returned no spam.

6.3.3 Choosing a starting vector with sum equal to zero

Let us perform a standard power method with $\mathbf{u_{k+1}} = \mathbf{A}\mathbf{u_k}$ and let us choose $\mathbf{u_0}$ as follows:

$$\begin{aligned} \mathbf{u_0} &= -1 + 2 \cdot \text{rand}(n, 1); \\ \mathbf{u_0} &= \mathbf{u_0} - \text{sum}(\mathbf{u_0})/n; \end{aligned} \tag{6.7}$$

Hence, we choose the elements of $\mathbf{u_0}$ uniformly in $[-1, 1]$, but with the property that $||\mathbf{u_0}||_1 = 0$.

Let us explain why we would choose such a starting vector. Recall that $\mathbf{A} = p\mathbf{P^T} + \frac{1-p}{n}\mathbf{E}$ and $\mathbf{u_0}$ sums up to zero, thus we find the next iteration by computing:

$$\begin{aligned} \mathbf{u_1} &= \mathbf{A}\mathbf{u_0} \\ &= p\mathbf{P^T}\mathbf{u_0} + \frac{1-p}{n}\mathbf{E}\mathbf{u_0} \\ &= p\mathbf{P^T}\mathbf{u_0}. \end{aligned} \tag{6.8}$$

Knowing that $\mathbf{u_0}$ sums up to zero, it should be easy to recognize that $\mathbf{u_0}$ and any matrix $\mathbf{E}$ with uniform rows are orthogonal and thus that $\mathbf{E}\mathbf{u_0} = \mathbf{0}$, similar as in Theorem 2.5.
Furthermore, $\mathbf{u_1}$ in Equation (6.8) also sums up to zero, because $\mathbf{P^T}$ is column-stochastic ($||\mathbf{u_1}||_1 = 1 \cdot (u_0)_1 + \cdots + 1 \cdot (u_0)_n = 0$).
In general, we find that each $\mathbf{u_k}$ sums up to zero and the uniform matrix $\mathbf{E}$ is of no influence on the vector $\mathbf{u_{k+1}}$. Therefore, we chose the starting vector $\mathbf{u_0}$ so that the model *with* jumping chance changed to a model *without* jumping chance.
Thus, in the end $\mathbf{u_k}$ will give a second eigenvector $\mathbf{x^{(2)}}$ of $\mathbf{A}$.

When testing `startvector.m` (see Appendix A.1.5) on $\mathbf{G_{2759+80}}$ and $\mathbf{G_{9914+80}}$ we can detect the link spamming and got the following two vectors:

$$\mathbf{x}^{(\mathbf{2})}_{\{\mathbf{1,2422,1117}\}} = 10^{-5} \begin{pmatrix} 0.389 \\ 0.035 \\ 0.021 \end{pmatrix}, \qquad \mathbf{x}^{(\mathbf{2})}_{\{\mathbf{4,2264,8059}\}} = 10^{-16} \begin{pmatrix} -0.105 \\ -0.103 \\ -0.078 \end{pmatrix}. \tag{6.9}$$

Note that also elements that are not in an irreducible closed subset can get value, and thus caution is required when determining the spamming nodes (determining which elements are non-zero and in an irreducible closed subset).

6.4 Tarjan's algorithm

A visual supportive algorithm is described in Appendix A.1.6. This algorithm uses the function `graphconncomp(.)`, which makes use of Tarjan's algorithm. The algorithm finds the strongly connected components of the graph, i.e., it finds the groups of nodes that are mutually reachable through following the hyperlinks. It returns a graph as well as a vector showing the groups of strongly connected nodes.

First, the algorithm adds outgoing links from dangling nodes, just as in the row-stochastic matrix $\mathbf{P}$. Each dangling node refers to all other nodes (except to itself). Self-referring nodes give a warning in MATLAB and are therefore removed. This has no consequences for ordering the nodes, because a self-referring node is already reached before it makes itself reachable (i.e., the strongly connected components do not change when deleting the self-referring hyperlinks).

Let us illustrate the algorithm with an example using $\mathbf{G_{m2}}$. Figure 11 is the graphic representation returned by the algorithm, comparable to Figure 9. The corresponding vector returned by the algorithm is $\mathbf{c} = \begin{pmatrix} 1 & 1 & 3 & 2 & 3 & 3 & 2 \end{pmatrix}$.

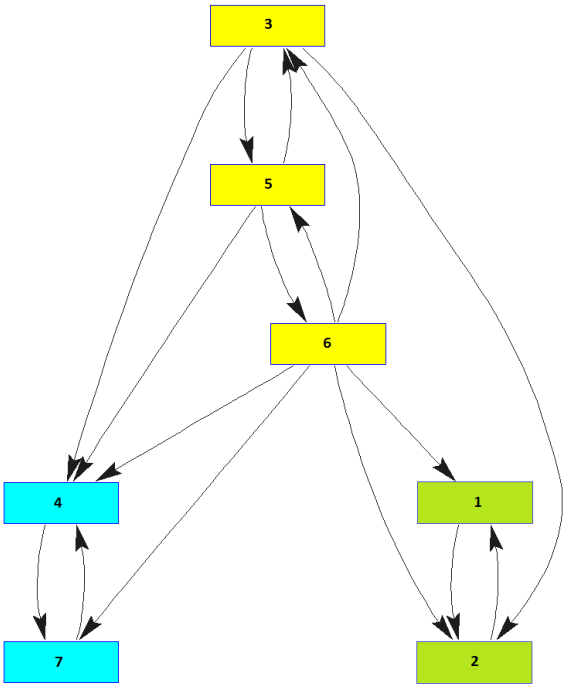


Figure 11: Using `tarjan.m` on $\mathbf{G}$ corresponding to Figure 9.

Note that the dangling node 6 in the middle of Figure 11 is replaced by a node which refers to all nodes (except itself). Furthermore, we recognize three groups which are mutually reachable. The two groups below ({4,7} in light blue and {1,2} in green) represent link spamming websites, the group above ({3,5,6} in yellow) represent the remaining web.

The MATLAB function `graphconncomp(.)` does not always work proper for large matrices. Applying the function to $\mathbf{G_{9914}}$ gave some contradicting results. Still though, in $\mathbf{G_{2759+80}}$ and $\mathbf{G_{9914+80}}$ the link spamming nodes clearly appeared in a seperate group.
For more information about Tarjan's algorithm, we refer to [15].

6.4.1 Tarjan's algorithm and detection

Tarjan's algorithm can have some interesting applications when we investigate the structure of link spamming nodes. Therefore, we combined a detection method with Tarjan's algorithm. We refer to Appendix A.2.3.

This algorithm uses `powerP.m`, which first computes the second eigenvector and thereafter, it determines the spamming nodes. Hence, `powerP.m` determines the second eigenvector of $\mathbf{A}$ and which elements of the second eigenvector are non-zero (in an irreducible closed subset) and which elements are 'zero-enough' (not in an irreducible closed subset). Thus, `detecttarjan.m` uses `powerP.m` to find the spamming nodes and then try to find nodes that belong to the same irreducible closed subset using Tarjan's algorithm.

Let us give us the following example ($\mathbf{G_{m2}}$):

```
Y = detecttarjan(Gm2);
```

The algoritm returns $\mathbf{Y} = \begin{pmatrix} 1 & 2 \\ 4 & 7 \end{pmatrix}$. Thus, node 1 and 2 correspond to an irreducible closed subset as well as node 4 and 7. If desirable, we can make a graphic representation by using the command `[Y,h] = detecttarjan(G)`, which returns matrix $\mathbf{Y}$ and a graphic representation as can be seen in Figure 12. Note that the nodes have been renumbered in the figure.

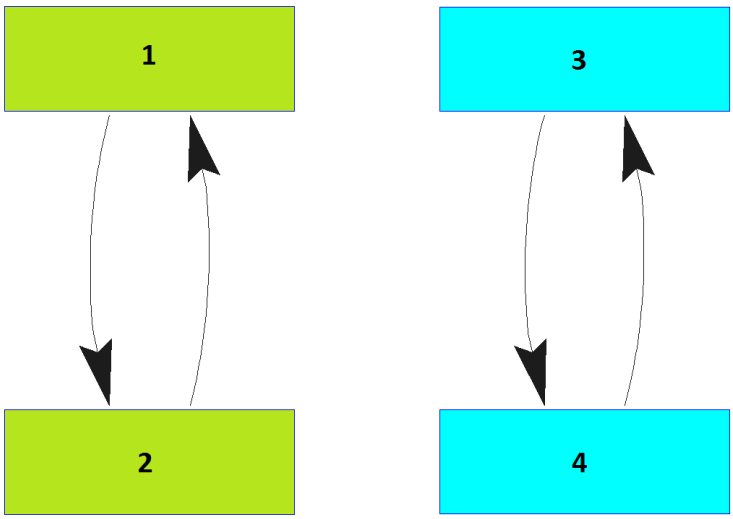


Figure 12: Graphic representation of combining detection with Tarjan's algorithm for $\mathbf{G_{m2}}$.

Testing this algorithm on $\mathbf{G_{2759+80}}$ returns

$$\mathbf{Y} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1578 & 1579 & 1580 & 1581 \\ 1670 & 0 & 0 & 0 \\ 2422 & 0 & 0 & 0 \\ 2423 & 0 & 0 & 0 \end{pmatrix}. \tag{6.10}$$

Here, link spamming node 1 is detected, but as well some other nodes (which may or may not be link spamming). Note that the 80 promotion nodes of node 1 remained undetected, i.e., these nodes are not found as spamming nodes. That is why we find node 1 as a single link spamming node using this algorithm. A nice detail is that nodes 1578 till 1581 are recognized as a group of link spamming nodes (althought we do not know for sure whether this is the case).

A last important remark is that this algorithm uses the information obtained by another detection algorithm, such as `powerP.m` discussed in Section 6.3.2 and it is not capable of finding link spamming nodes itself. As a consequence, the algorithm `detecttarjan.m` is only useful if the supportive detection algorithm was successful.

6.5 Cesàro sum

Another way to find a link spamming vector is to determine the Cesàro sum. This is, to be precise, no second eigenvector, because its 1-norm is equal to 1. Thus, we refer to this vector with $\mathbf{c}$. We refer to Section 2.3 and especially to (5.5) for the mathematical background.

An important note is that we know that the Cesàro sum converges very slowly. Recall that the Cesàro sum is given by:

$$\mathbf{C} = \lim_{k\to\infty} \frac{\mathbf{I} + \mathbf{P} + \cdots + \mathbf{P}^{k-1}}{k}. \tag{6.11}$$

Comparing with the power method, which computes $\mathbf{u_k} = \mathbf{P^k u_0}$, we recognize that the Cesàro sum also computes all previous terms and then calculates the average.

Thus many iterations are required, but on the other hand, we find for all test problems good results. For example, when applying `cesvec.m` (refer to Appendix A.1.7) to $\mathbf{G_{2759+80}}$ using 10000 iterations returns:
$\mathbf{c}_{\{\mathbf{1,1578,1579}\}} = \begin{pmatrix} 0.451 & 0.007 & 0.007 \end{pmatrix}^T$.

We should note that the promotion nodes get a value ($c_{2760-2839} = 0.006$) and thus a total value of $0.451 + 80 \cdot 0.006 = 2 \cdot 0.451 = 0.902$ out of 1.000 is catched in these link spamming nodes. Although we cannot detect link spamming for the promotion nodes within 10000 iterations, the detection of link spamming node 1 is excellent.

Now applying `cesvec.m` to $\mathbf{G_{9914+80}}$ using 10000 iteration returns:
$\mathbf{c}_{\{\mathbf{4,8059,8057}\}} = \begin{pmatrix} 0.026 & 0.016 & 0.014 \end{pmatrix}^T$.

We clearly recognize link spamming in node 4, as that node absorbs more energy with an increasing amount of iterations.

6.6 Numerical results

This section contains the numerical results of the discussed algorithms in previous sections. These results were obtained with a Windows 7 64-bit computer with an Intel Core Quad CPU Q8400 2.67GHz processor.

The elapsed time was determined by the function `tic, toc` in MATLAB and was put in the beginning respectively at the end of each algorithm.

We refer to table 1 on page 34 for an overview of all discussed algorithms. The column "matrix-vector per iteration" is short for the amount of matrix-vector multiplications per iteration. The time refers to the elapsed time in seconds. An algorithm is defined as successful if the known link spamming nodes have significantly the largest absolute value in the returned (second eigen)vector. That means that for $\mathbf{G_{m2}}$ that the nodes 1,2,4 and 7 are detected, for $\mathbf{G_{2759+80}}$ node 1 and for $\mathbf{G_{9914+80}}$ node 4. This does not exclude some results with other large absolute values and thus, additional found link spamming nodes. Yet, for all results these additional found nodes are limited.

In table 1, the amount of iterations is an input parameter for the following algorithms: `simpleP.m`, `startvector.m` and `cesvec.m`. These amount of iterations are only an indication for the required amount of iterations.

As we mentioned in Section 6.1, the success of `blockpower.m` depends on the right parameter r. The results for this algorithm in table 1 are the first successful r. This means that this algorithm sometimes needed some trial and error. For $\mathbf{G_{9914+80}}$ we see that the block power algorithm fails, which could be due to (probably) 176 irreducible closed subsets and we only tried $r \leq 30$.

Algorithm	Matrix-vector per iteration	$\mathbf{G_{m2}}$			$\mathbf{G_{2759+80}}$			$\mathbf{G_{9914+80}}$		
		Success	Iterations	Time	Success	Iterations	Time	Success	Iterations	Time
`blockpower.m`	r	yes	23	$3.1 \cdot 10^{-3}$	yes	1346	1.1	no	3000	140
`direct.m`	N/A	yes	N/A	$1.5 \cdot 10^{-3}$	yes	N/A	0.21	yes	N/A	83
`simpleP.m`	1	yes	10	$2.0 \cdot 10^{-3}$	yes	50	0.13	yes	500	0.52
`powerP.m`	0	yes	22	$1.6 \cdot 10^{-2}$	yes	256	2.8	yes[7]	600	34
`startvector.m`	1	yes	10	$3.0 \cdot 10^{-3}$	yes	150	0.22	yes	1250	3.5
`tarjan.m`	N/A	yes	N/A	$2.2 \cdot 10^{-4}$	yes	N/A	$9.5 \cdot 10^{-3}$	yes	N/A	153
`cesvec.m`	1	yes	500	$1.2 \cdot 10^{-2}$	yes	10000	1.0	yes	10000	3.2

Table 1: Numerical results for different algorithms on three test problems.

[7] `powerP.m` did return a proper vector $\mathbf{x}$, but an empty vector **spam** (see Appendix A.1.4).

6.7 Some observations

Page and Brin introduced the jump chance $1 - p$ for a few reasons. One reason was to make sure that most rankings do not converge to zero. Another reason was to ensure convergence to a unique dominant eigenvector. Moreover, adding a jump chance gave a convergence rate for the power method equal to one minus this jump chance.

The last two reasons for Page and Brin also hold for our detection problem. Whatever algorithm we take, in general we will have to deal with a non-unique second eigenvector of $\mathbf{A}$. Furthermore, the 'convergence' is not very good. For example, if we want to force convergence, we could work with a Cesàro sum, but such an algorithm will have a slow convergence rate, because of the averaging aspect (see (5.8)). On the other hand, this is a convincing algorithm in terms of absolute convergence and certainty about link spamming nodes. Other algorithms are dependent on the ratio of eigenvalues, but a lot of the eigenvalues of $\mathbf{A}$ can be close to p, which would cause a poor convergence (to zero) of elements that are not in an irreducible closed subset.

Another limitation is the performance of the algorithms when no irreducible closed subset exists. It can be difficult to recognize the difference between a lot of spamming nodes and no spamming nodes at all, i.e., determining which terms are zero and which terms are non-zero. We assume we have at least one irreducible closed subset, but this will not always be the case. However, we expect that this will not be a problem for a 'real' (large) web sample ([4]).

7 Conclusion

In this report we examined the second eigenvector of the Google matrix and its relation to link spamming. Furthermore, we discussed different methods to perform link spamming and we proposed different algorithms for detecting link spamming using knowledge about the second eigenvector of the Google matrix.

We found that creating an irreducible closed subset with as much promotion nodes as possible is the most efficient way of link spamming. This structure absorbs a lot of energy and receives a high PageRank. We refer to Section 4.1 for more information about promotion nodes and to Section 4.3 for more information about the most efficient link spamming method.

Furthermore, we discovered that irreducible closed subsets can be found with the second eigenvector of the Google matrix. A second eigenvector of $\mathbf{A}$ is a first eigenvector of $\mathbf{P^T}$. The elements of this eigenvector have (energy) value in the irreducible closed subset and no value in other nodes, because all energy ultimately gets absorbed into the ergodic states. We refer to Section 5.2 for more information.

Thereafter, we proposed several algorithms to find the second eigenvector of the Google matrix. Some of these algorithms are slow when applied to large matrices, other algorithms are more effective. Either way, we found some limitations.
First, the second eigenvector of $\mathbf{A}$ is not unique in general and thus, convergence is not guaranteed. This means that the states in the irreducible closed subsets can alternate.
Furthermore, the rate at which other components (not within an irreducible closed subset) go to zero is unknown. We know that the rate of convergence for the power method is equal to the ratio of the two largest unequal eigenvalues. For Google matrix $\mathbf{A}$ this is $|\lambda_2|/|\lambda_1| = p$ ([4]),

but the ratio of the two largest unequal eigenvalues of $\mathbf{P^T}$ can be very close to 1. Therefore, the convergence factor of many algorithms which look for the second eigenvector of $\mathbf{A}$ can be poor, especially when the matrix is huge. For more information about the rate of convergence, we refer to Section 2.2 and 6.3.1.

We conclude that the second eigenvector of the Google matrix can be used to detect the most efficient way of link spamming (speaking in terms of energy [6]), namely creating an irreducible closed subset. The elements of second eigenvector that are unequal to zero are using this method of link spamming. All other elements are equal to zero.

However, detection of other link spamming techniques using the second eigenvector is limited. This vector only detects nodes in an irreducible closed subset, which is one, but certainly not the only method to perform link spamming. Moreover, in practice, finding the second eigenvector can be hard due to poor convergence and non-uniqueness. Another point of interest is that some websites could be within an irreducible closed subset without performing link spamming on purpose.

In Section 5 we mentioned the stochastic personalization vector $\mathbf{v}$. Let us consider the consequences of using the personalization vector instead of the uniform vector. Recall Theorem 2.5, which tells us that $\mathbf{e^T x^{(2)}} = 0$. Using the personalization vector gives:

$$\mathbf{A} = p\mathbf{P^T} + (1-p)\mathbf{v e^T}. \tag{7.1}$$

As a consequence, the following equality holds:

$$\mathbf{v e^T x^{(2)}} = \mathbf{v} \cdot 0 = \mathbf{0}. \tag{7.2}$$

Thus still, the second eigenvector of $\mathbf{A}$ is only dependent of $\mathbf{P^T}$. Therefore, we expect that the resulting Theorem 5.3 does still hold when using the personalization vector.
Future research should be done to test this hypothesis.

References

[1] Cleve Moler, *Experiments with MATLAB*, Chapter 7: Google PageRank, MathWorks, Inc., 2011.

[2] L. Page, S. Brin and R. Motwani, *The PageRank Citation Ranking: Bringing Order to the Web*, Technical report, Stanford University, 1998.

[3] Rebecca S. Wills, Google's PageRank. The Math Behind the Search Engine, *Springer Science + Business Media* **28** (2006), no. 4.

[4] Taher H. Haveliwala and Sepandar D. Kamvar, *The Second Eigenvalue of the Google Matrix*, Technical report, Stanford University, 2003.

[5] Monica Bianchini, Marco Gori and Franco Scarselli, *PageRank, A Circuital Analysis*, In Proceedings of the Eleventh International WWW Conference, 2002.

[6] Monica Bianchini, Marco Gori and Franco Scarselli, *Inside PageRank*, ACM transactions on Internet technology **5** (2005), no. 1.

[7] Lars Eldén, *A Note on the Eigenvalues of the Google Matrix*, Report, Linköping University, 2003.

[8] Amy N. Langville and Carl D. Meyer, Deeper Inside PageRank, *Internet Mathematics* **1** (2004), no. 3, 335-380.

[9] Gilbert Strang, *Linear Algebra and its Applications*, 3rd ed., Brooks Cole, 1988.

[10] Jim Lambers, *The Eigenvalue Problem: Power Iterations*, Lecture notes 14, University of Southern Mississippi, 2010.

[11] Maysum Panju, Iterative Methods for Computing Eigenvalues and Eigenvectors, *The Waterloo Mathematics Review* **1** (2011), 9-18.

[12] D. L. Isaacson and R. W. Madsen. *Markov Chains: Theory and Applications*, chapter IV, pages 126 – 127. John Wiley and Sons, Inc., New York, 1976.

[13] Carl D. Meyer, *Matrix Analysis and Applied Linear Algebra*, Chapter 7 and 8, Society for Industrial and Applied Mathematics, 2000.

[14] Darald J. Hartfiel, *Markov Set-Chains*, Lecture notes in Mathematics, Springer-Verlag Berlin Heidelberg, 1991.

[15] *http://www.mathworks.nl/help/toolbox/bioinfo/ref/graphconncomp.html*, The MathWorks, Inc., R2012a Documentation, 2012.

[16] David F. Gleich, *http://www.cs.purdue.edu/homes/dgleich/nmcomp/matlab/*, wb-cs.stanford.mat, Last modified: 15-Nov-2011.

A MATLAB

A.1 Spam detection algorithms

A.1.1 `blockpower.m`

```
function [x,y,iter] = blockpower(G,r)

%Google's PageRank with block power method
% G: connectivity matrix
% r:  the size of the block, creates a n by r block
% returns x:  the eigenvectors corresponding to the r largest eigenvalues

p = 0.85;
maxiter = 3000;
[n,n] = size(G);
delta = (1-p)/n;
e = ones(n,1);

% Constructing Pt without dangling
[Pt,k] = Ptwithoutdangling(G);

% Choose Q(n,r) such that Q^T * Q = I_r
Z = rand(n,r);
iter = 0;
y2 = rand(r,1);
y = rand(r,1);

%QR decomposition
while max(abs(sort(y2) - sort(y))) > 10^(-6) && iter < maxiter
      Q = orth(Z);
```

```
      Z = p*Pt*Q + delta*e*(e'*Q) + p/n*e*(k'*Q);
      if size(Q,2) ~= r
         warning('Found at least one eigenvalue equal to zero.  Try smaller r.');
         return;
      end
      y = eig(Q'*Z);
      iter = iter + 1;
end

[s,D] = eig(Q'*Z);

%Determining and scaling x
x = Q*s;
% k = find(abs(y)>0.95);
% x(:,k) = x(:,k)/sum(x(:,k));


%Checking iteration amount
if iter == maxiter
   warning('Maximum iterations (%d) reached:  solution probably inaccurate!',maxiter);
end

end
```

A.1.2 direct.m

```
function x = direct(G)

%Blowing up the spamming nodes

[n,n] = size(G);
I = speye(n);
[Pt,k] = Ptwithoutdangling(G);
e = ones(n,1);
% Making P∧T
Pt = Pt + sparse(1/n*e*k');

% Direct solution
x = (0.99999*I-Pt)\ e;
x = x/sum(x);
end
```

A.1.3 simpleP.m

```
function x = simpleP(G,iter)

[n,n] = size(G);
[Pt,k] = Ptwithoutdangling(G);
```

```
e = ones(n,1);
x = rand(n,1);
cnt = 0;

while cnt<iter
      x = Pt*x + 1/n*e*(k'*x);
      cnt = cnt+1;
end

x = x/sum(x);
end
```

A.1.4 powerP.m

```
function [x,iter,spam] = powerP(G)

%Efficient algorithm for blowing up spam
[n,n] = size(G);
n2 = n^(-2.5);


for j = 1:n
    L{j} = find(G(:,j));
    c(j) = length(L{j});
end

x = ones(n,1)/n;
z = zeros(n,1);
y1 = 0;
y2 = 0;
iter = 0;
maxiter = max(round(6*sqrt(n)),250);
stop =0;
spam = 0;

while iter < maxiter && stop ==0
      z = x;
      x = zeros(n,1);
      for j = 1:n
          if c(j) == 0
             x = x + z(j)/n;
          else
             x(L{j}) = x(L{j}) + z(j)/c(j);
          end
      end
      iter = iter+1;
      %stopping criterium
      y1 = x.^6;
      y1 = y1/sum(y1);
      y2 = z.^6;
```

```
      y2 = y2/sum(y2);
      t1 = length(find(y1 > n2));
      t2 = length(find(y2 > n2);
      if t1 - t2 == 0
         stoparray(iter) = 1;
      end
      if iter>20 && sum(stoparray(iter-20:iter)) == 21
         stop = 1;
         spam = find(y1>n2);
      end
end

if iter == maxiter
   warning('Max iter reached');
end

end
```

A.1.5 startvector.m

```
function x = startvector(G,iter)

[Pt,k] = Ptwithoutdangling(G);
[n,n] = size(G);

x = -1 + 2*rand(n,1);
x = x -sum(x)/n;
p = 0.85;
delta = (1-p)/n;
cnt = 0;
e = ones(n,1);

while cnt < iter
      x = p*Pt*x + delta*e*(e'*x) + p/n*e*(k'*x);
      cnt = cnt+1;
end

end
```

A.1.6 tarjan.m

```
function c = tarjan(G)

%input:  connectivity matrix G with g(i,j)=1 if j -> i.

%deleting selfreferring nodes (to avoid warning)
G = G - diag(diag(G));

%dangling nodes artificially refer to all nodes
c = sum(G,1);
```

```
for j = 1:size(G,1)
    if c(j) == 0
       G(:,j) = 1;
    end
end


%making G sparse with g(i,j)=1 if i -> j
G = sparse(G');

[s,c] = graphconncomp(G);
% h = view(biograph(G));

%%coloring h
%colors = jet(s);
%for i = 1:numel(h.nodes)
%    h.Nodes(i).Color = colors(c(i),:);
%end

end
```

A.1.7 cesvec.m

```
function ces = cesvec(G,k)

%input:G
%output:  column cesaro sum as vector

[n,n] = size(G);
[Pt,k2] = Ptwithoutdangling(G);
y = rand(n,1);
y = y/sum(y);
x = y;
e = ones(n,1);

for i = 1:k-1
    x = Pt*x + 1/n*e*(k2'*x) + y;
end

ces = x/k;
end
```

A.2 Other algorithms

A.2.1 addpromo.m

```
function G = addpromo(G,g,p)

%G: square connectivity matrix
%g:  natural number within size of G
```

```
%p:  natural number >0

%adds p nodes to G with promoting websites for node 'g'
%also:  clears all outgoing links from node 'g', except to the added promotions.

[n,n] = size(G);

%clear all outgoing links from g.
G(:,g) = 0;

%add promoting nodes to 'g'
G(:,n+1:n+p) = 0;
G(g,n+1:n+p) = 1;
%add 'g' to promoting nodes
G(n+1:n+p,:)  = 0;
G(n+1:n+p,g) = 1;

end
```

A.2.2 Cesaro.m

```
function ces = Cesaro(P,k)

%input:  row-stochastic P, k iterations
%output:  row-stochastic Cesaro matrix

I = eye(size(P,1));
Psum = I;

for i = 1:k-1
    Psum = Psum*P + I;
end

ces = Psum/k;

end
```

A.2.3 detecttarjan.m

```
function [Y,h] = detecttarjan(G)

%Find spam vector
[∼,∼,spm] = powerP(G);
if spm'*spm == 0
   Y = 0;
   warning('Detection failed');
   return
end
```

```
Gnew = G(spm,spm);
%dangling nodes should not be included and therefore artifical hyperlinks
%should not be added

Gnew = sparse(Gnew);
[s,c] = graphconncomp(Gnew);


%Y = zeros(length(c));
for i = 1:s
    k = (c == i);
    Y(i,1:sum(k)) = spm(k)';
end

if any(Y(:,2) == 0)
   warning('Single-node spamming found:  possible error in detect.m');
end

if nargout > 1
   h = view(biograph(Gnew));
end

end
```

A.2.4 Ptwithoutdangling.m

```
function P = Ptwithoutdangling(G)

%returns the (dense-defined) row-stochastic matrix P
%possibility of returning column-stochastic matrix A

[n,n] = size(G);
c = sum(G,1);
k = zeros(n,1);
Pt = sparse(n,n);

for j = 1:n
    L{j} = find(G(:,j));
    c(j) = length(L{j});
end

for j=1:n
    if c(j) ~= 0
       Pt(L{j},j) = 1/c(j);
    else
       k(j) = 1;
    end
end

end
```