The PageTrust algorithm: How to rank web pages when negative links are allowed?

Cristobald de Kerchove* Paul Van Dooren†

Abstract

The paper introduces a novel algorithm derived from the PageRank algorithm of Brin and Page. The PageRank algorithm interprets an hyperlink from page a to page b as being a positive vote from a to b. Starting from this interpretation, it attributes a rank to each page. However, it does not offer the possibility to take into account negative votes. The PageTrust algorithm includes negative links and converges to a trust value for each page. The PageTrust algorithm appears as a natural extension of PageRank and it preserves good properties of robustness against possible spammers. Moreover several parameters allow us to strengthen or weaken the role played by negative links.

1 Introduction

The importance of ranking methods that classify the nodes of a network by relevance is growing more and more these last years, especially in the context of search engines for the web. Many of them use eigenvector based techniques to extract information from the network [8]. For instance, Brin and Page's *PageRank* algorithm [16], the Kleinberg's *HITS* algorithm [7], the *SALSA* algorithm [10], and their variants [13, 15, 18] calculate dominant eigenvectors of matrices that represent the structure of that network. A key idea used to prove that these eigenvectors are positive is the Perron-Frobenius theorem [14] which uses that the matrix, from which the dominant eigenvector is calculated, is nonnegative, i.e., all its entries are nonnegative. That condition explains why it becomes nontrivial to apply eigenvector based techniques when negative links are permitted. These links would be represented by negative entries in the matrix and the nonnegativity assumption then gets lost. Though nontrivial, the consideration of negative links can be of interest to refine the measures of ranking methods. Moreover such links already exist in the web, but they are simply not taken into account by *Google*, see [12].

A first solution, that can be found for example in [5, 19], is to zero the entries corresponding to the negative links in the matrix representing the structure of the network. In that way, negative links do not give any trust to the nodes they point to. But then a negative link or an absence of link between two nodes amount to the same result: in both cases, the corresponding entry in the matrix is zero. The concern with that method is therefore that the rank of a node does not change after adding a negative inlink.

A second idea, proposed in [2], is to first ignore negative links and hence to satisfy the nonnegativity assumption. The obtained eigenvector gives the trust values for all nodes, and can be interpreted as some propagation of trust through the positive links of the network. Then in order to integrate negative links, one single step of propagation of distrust is applied. As a consequence, the distrust value due to a negative link given by node i is proportional to the trust value of i. It follows that highly trusted nodes have the possibility to highly decrease the trust value of other nodes. For example, it is enough for the web page *Yahoo!*, that has a very high PageRank, to negatively point to a node, say x, to degrade x to the end of the ranking list. This can encourage malicious nodes to negatively point to other competing nodes only to make them fall down in the ranking list.

Clearly other methods that take into account negative links exist, like for example in the sites *Ebay* [17], *Advogato* [11] and *Epinions* [20]. However this paper focuses on eigenvector based techniques where the final ranking for the nodes is given by some eigenvector.

An ideal ranking method should decrease trust values of nodes that receive more and more negative links, but it should also be robust to attackers that want to decrease trust values of competing nodes by the means of negative links. The rest of the paper shows how the PageTrust algorithm addresses these problems. Since the PageTrust algorithm is a natural extension of the PageRank algorithm, section 2 first introduces it. Then section 3 explains how to extend the PageRank algorithm to take into account negative links. The properties of the new algorithm are analyzed in section 4, and several examples are given to compare it with the PageRank algorithm.

*Dept Math. Engineering, UCL, Louvain-la-Neuve Belgium.
†Dept Math. Engineering, UCL, Louvain-la-Neuve Belgium.

2 Notations and the PageRank algorithm

This section serves as a basis for the rest of this paper. The principal notations are introduced and the PageRank algorithm is briefly described with its classical interpretation in terms of a random walk.

2.1 Notations Let $\mathcal{G}(\mathcal{N}, \mathcal{L}^+, \mathcal{L}^-)$ be a directed graph where $\mathcal{N}$, $\mathcal{L}^+$ and $\mathcal{L}^-$ are respectively its set of nodes, its set of positive links, and its set of negative links. The adjacency matrices for $\mathcal{G}_{pos}(\mathcal{N}, \mathcal{L}^+)$ and $\mathcal{G}_{neg}(\mathcal{N}, \mathcal{L}^-)$ are respectively noted A^+ and A^-. The outdegree of a node $i \in \mathcal{N}$ will be the number of positive outlinks of that node and is represented by d_i. From the graph $\mathcal{G}$, the vector π is calculated and its i^{th} entry π_i is the rank assigned to node i in $\mathcal{G}$.

REMARK 2.1. *For the sake of simplicity, we will assume in the sequel that all nodes in $\mathcal{N}$ have at least one positive child. Nonetheless the general case without that assumption only implies minor modifications.*

2.2 PageRank algorithm The PageRank algorithm iterates on variables x_i, $i \in \mathcal{N}$, that will eventually converge by the iterations in (2.1) to the ranks π_i. First, these variables are initialized to the same value: $x_i^{(0)} = 1/n$ for all $i \in \mathcal{N}$ where n is the number of nodes in $\mathcal{G}$. Then it iterates in the following way:

$$x_i^{(t+1)} = \alpha \sum_{j,(j,i)\in\mathcal{L}^+} x_j^{(t)}/d_j + (1-\alpha)\, z_i, \tag{2.1}$$

for all $i \in \mathcal{N}$. The parameter α belongs to $]0,1[$ and allows us to balance the rank obtained thanks to its parents and the one obtained by a constant source determined by the nonnegative vector z, the so-called personalization vector, satisfying $\sum_{i=1}^n z_i = 1$.

Hence the vector x converges to the PageRank vector π. This vector can be seen as the normalized eigenvector of the *Google* matrix G:

$$\lambda\pi = G\pi, \tag{2.2}$$

corresponding to the dominant eigenvalue λ, and for all $i, j \in \mathcal{N}$ we have $G_{ij} = \alpha A^+_{ji}/d_j + (1-\alpha)z_i$.

2.3 PageRank interpretation The final value π_i obtained by (2.1) can be interpreted as the probability of presence in node i of a random walker. At each step, that walker, say in node i, chooses, with a probability α, one of the children of i and, with a probability $1-\alpha$, any node $j \in \mathcal{N}$ (That last motion is commonly called *the zapping*). The choice between the children of node i is made with equal probability while the choice between all nodes during the zapping is distributed according to the personalization vector z.

2.4 PageRank properties The PageRank has interesting properties of robustness against attackers, see for example [1, 6], even though *spam farms* exist, i.e., groups of pages pointing to one single page. Simple techniques allow to detect such structure [3]. Moreover, the practical use of the PageRank algorithm in *Google* makes of it a reference among the search engine rankings [9]. However, as already pointed out in the introduction, it simply ignores the negative links in $\mathcal{L}^-$ or sometimes worse it uses them as positive links, see [12]. Next section explains how the PageTrust algorithm extends the PageRank algorithm to positive and negative links.

3 The PageTrust algorithm

The section is organized in five parts: first, we introduce the *distrust* matrix P which is the key element in the PageTrust algorithm. Then the iterative procedure on $P^{(t)}$ is described and interpreted. The section terminates with the two modifiable parameters of the PageTrust algorithm.

3.1 The distrust matrix Let us assume that several random walkers move in the graph $\mathcal{G}$ like in the PageRank algorithm, see § 2.3. The presence of negative links in the graph will imply that each random walker receives an opinion of every node. More precisely, each one trusts or distrusts a set of nodes in $\mathcal{N}$. The detailed dynamics of their opinions will be given next subsection. Let us first suppose, for the sake of clarity, that there is no opinion dynamics. Hence the proportion of walkers in node i who distrusts node k for any $i, k \in \mathcal{N}$ is given and remains constant. That proportion is denoted P_{ik}. This defines the $n \times n$ distrust matrix P. Therefore the diagonal of P gives the proportion of walkers that distrust the node they are in. Such walkers will not be considered as visitors but rather they will leave the graph. In that manner, $(1 - P_{ii})$ represents the proportion of remaining walkers in node $i \in \mathcal{N}$. This is thus characterized by the following iteration:

$$x_i^{(t+1)} = (1 - P_{ii}) \cdot \left[\alpha \sum_{j,(j,i)\in\mathcal{L}^+} x_j^{(t)}/d_j + (1-\alpha)z_i \right],$$

for all $i \in \mathcal{N}$. Like in the PageRank algorithm, the vector x is initialized to $1/n$ for each entry. Moreover x is normalized to have its $1-$norm equals to 1 after each step. Eventually, the vector x will converge to the eigenvector π that gives the trust ranks. In matrix form, we have

$$x^{(t+1)} = \frac{D_P \cdot Gx^{(t)}}{\mathbf{1}^T D_P \cdot Gx^{(t)}}, \tag{3.3}$$

where $\mathbf{1}$ is the $n \times 1$ vector of 1's. The matrix D_P is the diagonal matrix containing the diagonal of $(I - P)$ with I, the identity matrix. This defines a scaling of the lines of the *Google* matrix G. Relatively speaking, the smaller elements of D_P will decrease the corresponding elements of $x^{(t+1)}$ and hence a large element P_{ii} (i.e., a stronger distrust in node i) will result in a smaller rank in component π_i. This can be visualized by inserting an intermediary node i_0 between the parents of node i and node i. Hence, a walker always visits node i_0 before node i. Then we also add an outlink from i_0 that represents a leak to leave the graph and the probability to use it is given by P_{ii}. Therefore a walker in node i_0 will reach node i with probability $(1 - P_{ii})$.

The next subsection describes how P can be adapted as well during the random walk.

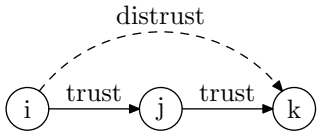

Figure 1: A three nodes graph with $(i,j),(j,k) \in \mathcal{L}^+$ and $(i,k) \in \mathcal{L}^-$.

3.2 The dynamics of distrust Let us introduce this part with the small example given in figure 1. Node i trusts node j, and hence i would advise a random walker to go and visit j. In the same way, node j would advise a walker to go and visit ode k. But i distrusts k, and we assume then that a walker leaving node i will keep in mind that information. In that manner, a walker leaving node i and reaching j should not visit k. We can compare this to the situation where you tell a secret to a friend of yours, but ask him not to repeat it to your enemies.

The example of Figure 1 shows us three principles for updating the matrix P that contains the proportions of walkers with some opinions about nodes (trust or distrust). Let us describe the motion of one single walker in the graph and suppose that the number of walkers is infinite.

1. A walker moves like a random walker in the PageRank algorithm and keeps his opinion (see **equation 3.4**).
2. A walker in node $i \in \mathcal{N}$ automatically adopts negative opinions of node i, that is he adds in his list of distrusted nodes the nodes negatively pointed by i (see **equation 3.5)**.
3. A walker who distrusts the node k leaves the graph if ever he visits k (see **equation 3.5**).

Formally, that leads to three operations to update the distrust matrix P. Since P is now time dependent, we have $P_{ik}^{(t)}$ representing the proportion of walkers in node $i \in \mathcal{N}$ who distrusts node $k \in \mathcal{N}$ at time t. We first consider one step of walkers with the conservation of their opinions:

$$\tilde{P}^{(t+1)} = T^{(t)} \cdot P^{(t)}, \tag{3.4}$$

where T is the *transition* matrix with $T_{ij}^{(t)}$ the ratio of walkers in node i who was in node j at time t,

$$T_{ij}^{(t)} = \frac{\alpha A_{ji}^{+} x_j^{(t)}/d_j + (1-\alpha) z_i x_j^{(t)}}{\alpha \sum_{k,(k,i)\in\mathcal{L}^+} x_k^{(t)}/d_k + (1-\alpha) z_i},$$

for all $i,j \in \mathcal{N}$. Then we apply two corrections that simply amount to putting some entries of P equal to 1 or 0 according to the adoption of the negative links of a node or the presence of walkers who distrust the node they reach:

$$P_{ij}^{(t+1)} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{L}^-, \\ 0 & \text{if } i = j, \\ \tilde{P}_{ij}^{(t+1)} & \text{otherwise,} \end{cases} \tag{3.5}$$

for all $i,j \in \mathcal{N}$. The complete iteration on the vector x and the matrix P is given in two steps: first x is updated from equation (3.3) that uses the diagonal of $\tilde{P}^{(t)}$ instead of P. Then P is updated from equations (3.4) and (3.5), with the initial values $P^{(0)} = \tilde{P}^{(0)} = A^-$.

Eventually x converges to the trust vector π that is the dominant eigenvector of the matrix $D_{\tilde{P}^\infty} \cdot G$ where the diagonal matrix $D_{\tilde{P}^\infty}$ is the diagonal part of $\tilde{P}^{(t)}$ after convergence. The iterations on x can still be interpreted in terms of a random walk.

3.3 PageTrust interpretation Let us imagine not one single random walker, but an infinite number of random walkers originally uniformly distributed in the graph. Each of these walkers moves according to the same rules than for the PageRank algorithm, see § 2.3. A walker in node j at time t chooses with equal probability a child of j or zaps to any node in $\mathcal{N}$, in both cases say it reaches node i . Many other walkers also reach node i with possibly different lists of distrusted nodes. The proportion of walkers that distrust node $k \in \mathcal{N}$ is given in equation (3.4) by $\tilde{P}_{ik}^{(t+1)}$. Then, we make the simplifying assumption that these walkers reach a consensus where finally the proportion of walkers that distrust the list of nodes $\mathcal{S} = \{i_1, i_2, \cdots, i_d\}$ is simply given by

$$\prod_{k\in\mathcal{S}} P_{ik}^{(t+1)} \cdot \prod_{k\in\mathcal{N}\setminus\mathcal{S}} (1 - P_{ik}^{(t+1)}).$$

In that manner, the opinions at the end of that consensus are uniformly distributed. A walker who distrusts node i then leaves the graph, hence the proportion of remaining walkers is $(1 - \tilde{P}_{ii}^{(t+1)})$, that factor is used in equation (3.3). Then the remaining walkers take into account the negative links of node i, equation (3.5), and choose a child of node i or zap, etc.

In that random walk, the entry $x_i^{(t)}$ represents the proportion of walkers in node i who trust i at time t.

3.4 Memory after zapping Let us introduce the binary parameter M that will represent whether the walkers keep their opinions after the zapping. So far, they conserved their lists of distrusted nodes after zapping, i.e., $M = 1$. As we will see in § 4.1, the loss of memory after zapping, i.e., $M = 0$, implies a decrease of the impact of negative links on the trust ranks. The addition of a parameter M only modifies the transition matrix T in equation 3.4. That dependence on M is given by

$$T_{ij}^{(t)} = \frac{\alpha A_{ji}^{+} x_j^{(t)}/d_j + M \cdot (1-\alpha) z_i x_j^{(t)}}{\alpha \sum_{k,(k,i)\in\mathcal{L}^+} x_k^{(t)}/d_k + (1-\alpha) z_i},$$

for all $i, j \in \mathcal{N}$. Let us remark that it is possible to have $M \in [0, 1]$. In that case the parameter M represents the probability to keep an opinion after zapping.

3.5 Degree of conviction As explained in § 3.3, once a random walker reaches node $i \in \mathcal{N}$, he mixes up his opinion with the other present walkers in order to reach a consensus. Then the walkers who distrust node i (their proportion is $\tilde{P}_{ii}^{(t)}$ at time t) leave node i. Therefore the proportion of remaining walkers after consensus is $(1 - \tilde{P}_{ii}^{(t)})$. Now let us suppose that the walkers who distrust node i have a degree of conviction β such that the proportion of remaining walkers after consensus becomes $(1-\tilde{P}_{ii}^{(t)})^{\beta}$. For $\beta = 0$, they have no influence and the PageTrust algorithm becomes exactly the same as the PageRank algorithm. Then, the higher β is, the more the negative links have an impact. As β becomes very large, one walker is enough to convince all present walkers in node i. This will modify the updates on the vector x in equation (3.3) as follows:

$$x^{(t+1)} = \frac{\left(D_{\tilde{P}^{(t)}}\right)^{\beta} \cdot G x^{(t)}}{\mathbf{1}^T \left(D_{\tilde{P}^{(t)}}\right)^{\beta} \cdot G x^{(t)}}, \tag{3.6}$$

where $\mathbf{1}$ is the $n \times 1$ vector of 1's and the matrix $D_{\tilde{P}^{(t)}}$ is the diagonal of $(I - \tilde{P}^{(t)})$.

The PageTrust algorithm with the parameters M and β is presented in Algorithm 4.1.

ALGORITHM 4.1. (PAGETRUST ALGORITHM.)

input : A graph $\mathcal{G}(\mathcal{N}, \mathcal{L}^+, \mathcal{L}^-)$, $\alpha \in]0, 1[$, $z > 0$, $M \in \{0, 1\}$ and $\beta \geq 0$.
output: A ranking vector π and the final distrust matrix P.

Initialization of $x^{(0)}$, $P^{(0)}$, $\tilde{P}^{(0)}$ *and* $t = 0$;
Build the Google matrix G;
while $\max(x^{(t+1)} - x^{(t)}) > \epsilon$ **do**
  for $i = 1$ **to** $|\mathcal{N}|$ **do**
    $x_i^{(t+1)} \leftarrow (1 - \tilde{P}_{ii}^{(t)})^{\beta} \cdot \sum_{k\in\mathcal{N}} G_{ik}\, x_k^{(t)}$;
    Build the transition matrix $T^{(t)}$;
    for $j = 1$ **to** $|\mathcal{N}|$ **do**
      $\tilde{P}_{ij}^{(t+1)} \leftarrow \sum_{k\in\mathcal{N}} T_{ik}^{(t)} P_{kj}^{(t)}$;
      if $(i, j) \in \mathcal{L}^-$ **then**
        $P_{ij}^{(t+1)} \leftarrow 1$;
      else if $i = j$ **then**
        $P_{ij}^{(t+1)} \leftarrow 0$;
      else
        $P_{ij}^{(t+1)} \leftarrow \tilde{P}_{ij}^{(t+1)}$;
      end
    end
  end
  $x^{(t+1)} \leftarrow x^{(t+1)} / \mathbf{1}^T x^{(t+1)}$;
  $t \leftarrow t + 1$;
end

4 Properties and examples

This section analyzes several properties of the PageTrust algorithm and illustrates them by different examples where the vector z is always taken as uniform.

4.1 Zapping, memory and degree of conviction The impact of zapping has already been analyzed for the PageRank algorithm [4, 9], and is similar for the PageTrust algorithm. The difference resides in the memory M that tells us whether a random walker keeps or looses his memory after zapping. Clearly, if he looses it, negative links will be less penalizing. If in addition the zapping is high, i.e., if α is close to 0, then the effect of negative links will still decrease. Therefore the binary M used with the zapping α allows us to control the strength of negative linkage. Table 1 illustrates this discussion for the graph in Figure 2 with $\beta = 1$ and compares the PageTrust with the PageRank of the nodes.

The degree of conviction is another parameter that allows us to control the PageTrust vector. That parameter ranges from 0 to ∞, from the recovering of the

	$\alpha = .9$			$\alpha = .5$		
	PR	$M = 0$	$M = 1$	PR	$M = 0$	$M = 1$
1	.18	.24	.31	.18	.19	.30
2	.18	.19	.20	.18	.19	.21
3	.27	.36	.49	.26	.28	.45
4	**.18**	**.10**	**0**	**.18**	**.17**	**0**
5	.18	.11	.01	.19	.18	.04

Table 1: The effect of the memory after zapping in PageTrust for the graph given in figure 2. Two columns compare the results with PageRank.

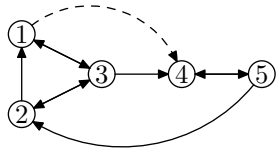

Figure 2: Example of graph with one distrust from node 1 to node 4.

PageRank algorithm to the absolute contagion where one random walker is enough to convince all the other ones. Obviously, the higher β, the more penalizing the negative links. Table 2 illustrates that fact with $\alpha = .9$ and $M = 0$.

	$\beta = 0$	$\beta = 1$	$\beta = 2$	$\beta = \infty$
	PR			
1	.18	.24	.30	.31
2	.18	.19	.20	.20
3	.27	.36	.47	.49
4	**.18**	**.10**	**.01**	**0**
5	.18	.11	.02	.01

Table 2: Depending on parameters in PageTrust, node 4 and indirectly node 5 are more and more penalized by the link of distrust $(1, 4) \in \mathcal{L}^-$. The first column amounts to PageRank.

4.2 Robustness to malicious behaviors Ideally, as explained in the introduction, the PageTrust must be robust to attacks. Two possible attacks are to use negative links to increase its own PageTrust or to decrease the PageTrust of competing nodes. Linkage strategy with positive links have already been investigated in [1, 6].

Since PageTrust of node i can be interpreted as the proportion of walkers in i once the steady state reached, one strategy could be to try to lure back walkers to itself via negative links. A similar idea was used in [1] with positive links. They explain that optimal linkage strategy is obtained for a node when it points to one of its parents in order to make the random walker return to itself. With negative links, such a strategy seems less obvious. For instance, a natural idea consists in pointing negatively to nodes that represent a leak for node i, that is nodes that send the walkers far away from node i. But that strategy, illustrated in Figure 3, does not help since a walker that distrusts the leaking nodes, node 3 in the figure, will not choose between the remaining outlinks (link $(3, 2)$ in the figure) but it will rather leave the graph as explained in § 3.3. Therefore this way is not interesting. For example in Figure 3, the PageTrust that is lost by node 3 is mainly earned by node 5 and 6. Even though node i does not necessarily

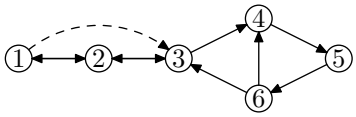

Figure 3: Node 3 is a leak for node 1 since it can send walkers to the nodes $4, 5, 6$ what makes the walkers move away from node 1.

increase its PageTrust, it is able to decrease the one of nodes it distrusts. And this is interesting when they are compared in the same ranking list. A reasonable way to avoid such a behavior is to consider negative links in both directions. In other words, if node i declares itself against node k, we have to consider that automatically node k will declare itself against node i. Taking back the example in Figure 3, the mutual distrust between node 1 and 3 is not interesting for node 1 nor for node 3. Table 3 shows the results when a mutual distrust exist between node 1 and 3 in Figure 3 with $M = 0$ and $\alpha = .9$.

	$\mathcal{L}^- = \emptyset$	$\mathcal{L}^- = \{(1, 3), (3, 1)\}$	
	PR	$\beta = 1$	$\beta = 2$
1	**.09**	**.04**	**.01**
2	.18	.13	.11
3	**.18**	**.17**	**.18**
4	.18	.21	.22
5	.18	.22	.23
6	.18	.23	.25

Table 3: PageTrust with different parameters shows that distrusting a leaking node is not synonym of increasing its trust rank. The first column amounts to PageRank.

4.3 Local Trust Metric Thus far, the PageTrust algorithm allows us to obtain a global trust metric given by the vector π. By global trust metric, we mean a measure of trust that does not depend on the point of view of any user. This is of interest when users have no a priori about the graph $\mathcal{G}$, like for instance in the case of the web graph where no web pages are trusted a priori. However, this can be done via the personalization vector z that distributes a kind of initial trusts over the nodes.

In contrast, a local trust metric depends on a certain view point of some user, say represented by node i. A natural idea, proposed in [19] with the PageRank algorithm, is to put z_i to 1 and the rest of the entries of z to 0. In that way, it is as if the random walkers start in node i. Thereafter, there is a probability $(1-\alpha)$ that they come back to node i. In that manner, node i stays the reference for the walkers, and for example no nodes distrusted by node i will be visited. Therefore parameter α allows us to favor or not *trust proximity*: close to 1 the walker often comes back to the source node i, and close to 0 he will more probably go further through some chains of friends. Table 4 illustrates two different opinions depending on the source node for the example in Figure 2 where we add a mutual distrust between node 1 and node 4.

	$\alpha = .5,\ \beta = 1,\ M = 0$	
	*node*2	*node*5
1	**.23**	**.13**
2	.30	.20
3	.34	.20
4	**.06**	**.16**
5	.06	.31

Table 4: PageTrust with selected parameters gives the view points of node 2 and 5. This is for the graph in figure 2 with in addition a distrust link from node 4 to node 1.

4.4 Convergence and complexity The iteration in equation (3.3) with the fixed distrust matrix P is guaranteed to converge. The same can be shown when we iterate on $\tilde{P}^{(t)}$ by equations (3.5) and (3.4) with a fixed vector x. However, the Algorithm 4.1 iterates both on the vector x and the matrix $\tilde{P}$ that mutually influence each other, and the proof of its convergence has still to be established. Nevertheless we observed experimentally that the algorithm converges linearly. If we increase the memory M, the zapping α, or the degree of conviction β, then the rate of convergence decreases. The complexity of the algorithm is in $\mathcal{O}(\bar{k}nn^-)$ per iteration step where $\bar{k}$, n^- and n are respectively the mean degree, the number of nodes that have received negative links and the total number of nodes.

5 Conclusion

In this paper we described how the PageTrust algorithm naturally extends the PageRank algorithm to the inclusion of negative links in a graph. We believe that this method has interesting properties and can find applications in the context of Web mining. The PageTrust can be used as global metric, but also as local metric. The latter metric is more convenient for sites with exchanges of opinions like for example in Ebay, Epinions, peer to peer systems, etc. But further investigations are needed to validate this. Future research will focus on validating our method with real data sets and comparing results with existing methods. In addition we still need to analyze the sensitivity of its parameters, its rate of convergence, its memory space and its implementation.

The key idea of this paper is the addition of a sort of memory in the random walk on a graph. The same principle can be applied for other algorithms where an interpretation of random walk exists. Therefore we think that other eigenvector based methods can benefit from that principle and can be extended to take into account negative links in a graph.

Acknowledgements

This paper presents research supported by a grant "Actions de recherche concertées – Large Graphs and Networks" of the "Communauté Française de Belgique" and by the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office.

References

[1] K. Avrachenkov and N. Litvak, *Decomposition of the Google PageRank and Optimal Linking Strategy*, Tech. report, INRIA, (2004), http://www.inria.fr/rrrt/rr-5101.html.

[2] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, *Propagation of Trust and Distrust*, Proc. of the 13th Int. Conf. on WWW, (2004), pp. 403–412

[3] Z. Gyongyi and H. Garcia-Molina, *Link Spam Alliances*, Proc. of the 31st Int. Conf. on Very large data bases, VLDB Endowment, (2005), pp. 517-528.

[4] I. C. F. Ipsen and R. S. Wills, *Mathematical Properties and Analysis of Googles PageRank*, Bol. Soc. Esp. Mat. Apl., 34 (2006), pp. 191-196.

[5] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, *The EigenTrust Algorithm for Reputation Management in P2P Networks*, Proc. of the 12th Int. Conf. on WWW, (2003), pp. 640–651.

[6] C. de Kerchove, L. Ninove, and P. Van Dooren, *Maximizing PageRank via Outlinks*, tentatively accepted to LAA.
[7] J. M. Kleinberg, *Authoritative Sources in a Hyperlinked Environment*, J. ACM 46, 5 (1999), pp. 604–632.
[8] A. N. Langville and C. D. Meyer, *A survey of eigenvector methods of web information retrieval*, SIAM Review, 47(1), (2005), pp. 135-161.
[9] ____ , *Google's PageRank and Beyond: the Science of Search Engine Rankings*, Princeton University Press, Princeton, NJ, (2006).
[10] R. Lempel and S. Moran, *The Stochastic Approach for Link-Structure Analysis and the TKC Effect*, Proc. of the 11th Int. Conf. on WWW, (2000), pp. 387–401.
[11] R. Levien, *Advogato Trust Metric*, PhD Dissertation, UC Berkeley, USA, (2003).
[12] P. Massa and C. Hayes, *Page-reRank: Using Trusted Links to re-Rank Authority*, in Proceedings of Web Intelligence Conference, France, Sept. 2005.
[13] A. O. Mendelzon and D. Rafiei, *An Autonomous Page Ranking Method for Metasearch Engines*, Proc. of the 11th Int. Conf. on WWW, (2002).
[14] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, (2000).
[15] A. Y. Ng, A. X. Zheng, and M. I. Jordan, *Stable Algorithms for Link Analysis*, Proc. of the 24th annual Int. ACM SIGIR Conf., (2001), pp. 258–266.
[16] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bring Order to the Web*, Technical report, Computer Science Departement, Stanford University, (1998).
[17] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood, *The Value of Reputation on eBay: A Controlled Experiment Revue*, Experimental Economics, Springer Netherlands, Vol 9/2, (2006), pp. 79-101.
[18] M. Richardson and P. Domingos, *The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank*, Advances in Neural Information Processing Systems 14, (2002), pp. 1441–1448.
[19] M. Richardson, R. Agrawal, and P. Domingos, *Trust Management for the Semantic Web*, Proc. of the 2nd Int. Semantic Web Conf., LNCS., vol 2870, (2003), pp. 351–368.
[20] http://www.epinions.com/