A survey on distributed approaches to graph based reputation measures

Konstantin Avrachenkov
INRIA Sophia Antipolis
2004 Route des Lucioles
06902 France
kavratch@sophia.inria.fr

Danil Nemirovsky
INRIA Sophia Antipolis
France, and
St.Petersburg State University
Russia
dnemirov@sophia.inria.fr

Kim Son Pham
St. Petersburg State University
Russia
sonsecure@yahoo.com.sg

ABSTRACT

Reputation systems are indispensable for the operation of Internet mediated services, electronic markets, document ranking systems, P2P networks and Ad Hoc networks. Here we survey available distributed approaches to the graph based reputation measures. Graph based reputation measures can be viewed as random walks on directed weighted graphs whose edges represent interactions among peers. We classify the distributed approaches to graph based reputation measures into three categories. The first category is based on asynchronous methods. The second category is based on the aggregation/decomposition methods. And the third category is based on the personalization methods which use local information.

1. INTRODUCTION

Trust and reputation are imperative for Internet mediated service provision, electronic markets, document ranking systems, P2P networks and Ad Hoc networks. It is necessary to distinguish clearly between the notions of trust and reputation. Following the works [16, 26], we can define Trust as the extent to which one party is willing to depend on something or somebody with a feeling of relative security, even though negative consequences are possible. And we can define Reputation as what is generally said or believed about a person's or thing's character or standing. Thus, Reputation is a more objective notion and Trust is a more subjective one. Reputation is typically acquired over a long time interval, whereas Trust is based on a personal reflection before taking a decision to interact with another person. In other words, the reputation about a person or a thing is given by the society and the trust is a decision taken by an individual member of the society to rely on the other party. The estimation of trust is quite application specific. Examples of trust metrics for instance can be found in [16, 19, 26] and in references therein. The computation of reputation measures is less application specific. Many reputation systems are based on graph centrality measures. A social/information network is typically represented by a directed/undirected/weighted graph. If the graph is directed, an edge from node A to node B signifies that node A recommends node B. Here we review distributed approaches to the graph based centrality/reputation measures. The distributed approaches are particularly needed for large systems such as WWW or P2P networks. Intuitively, graph centrality measures are based on the following two observations: (a) it is more likely that individuals will interact with friends of friends than with unknown parties. This is so-called transitivity of trust; and (b) individuals incline to trust more somebody who is trusted by some of their friends with high reputation. As case studies of the graph based reputation systems we shall take PageRank [29] and TrustRank [19]. The other graph based reputation measures appear to be modifications of the latter ones.

PageRank [29] is one of the principle criteria according to which Google search engine ranks Web pages. The basic idea of PageRank algorithm is to use the hyper-links as indication that one Web page recommends another Web page. Also, PageRank can be interpreted as the frequency that a random surfer visits a Web page. Thus, PageRank reflects the popularity and reputation of a Web page. The formal definition of PageRank is as follows: Denote by n the total number of pages on the Web and define the $n \times n$ hyper-link matrix P as follows. Suppose that page i has $k > 0$ outgoing links. Then $p_{ij} = 1/k$ if j is one of the outgoing links and $p_{ij} = 0$ otherwise. If a page does not have outgoing links, we call it a dangling page, and the probability is spread among all pages of the Web with some distribution v, namely, $p_{ij} = v_j$. In order to make the hyper-link graph connected, it is assumed that a random surfer goes with some probability to an arbitrary Web page with the distribution v. In the standard PageRank formulation, this distribution is chosen to be uniform. Thus, the PageRank is defined as a stationary distribution of a Markov chain whose state space is the set of all Web pages, and the transition matrix is

$$G = cP + (1 - c)ve, \tag{1}$$

where e is a vector whose all entries are equal to one, $v = \frac{1}{n}e^T$, and $c \in (0, 1)$ is the probability of following a link on the page and not jumping to a random page (it is chosen by Google to be 0.85). The constant c is often referred to as a damping factor. The Google matrix G is stochastic, aperiodic, and irreducible, so there exists a unique row vector π such that

$$\pi G = \pi, \quad \pi e = 1. \tag{2}$$

SMCTools '07, October 26, 2007, Nantes, France

The row vector π satisfying (2) is called a *PageRank vector*, or simply *PageRank*.

There is a drawback of PageRank as a reputation measure. All outgoing links from a node provide equal contributions. However, in many applications one node can have worse or better experience when interacting with another node and consequently, the relations between two nodes can have different level of trust. This factor was taken into account in [19, 35, 36]. Namely, now the entries of matrix P are not simply defined as uniform distributions over the outgoing links, but represent levels of trust the node has in respect to his peers. Thus, we could regard the *TrustRank* measure as a random walk on a weighted graph. We would like to mention that the other works that study the reputation systems for P2P networks are [30, 31, 33]. The authors of [35, 36] apply the graph based reputation measures to Semantic Web and the authors of [8] suggest to use the graph based reputation measures in mobile Ad Hoc networks.

Since PageRank and TrustRank are only different in the definition of the entries of matrix P, many algorithms designed for one reputation rank metric will work for the other reputation rank metric. Thus, in our survey, if an algorithm can be applied to either PageRank or TrustRank, we simply denote the outcome of the algorithm as a *Rank vector*. In particular, to find the value of the Rank vector, it is often convenient to transform the eigenproblem based definition to an equivalent form of the linear system [25, 21]:

$$\pi = \pi cP + (1-c)v. \tag{3}$$

The structure of the paper is organized as follows: In Section II we review the asynchronous approaches to the graph based reputation measures. Then, in Section III we review the aggregation/decomposition approaches. In fact, the aggregation/decomposition approaches can be regarded as some limiting cases of the asynchronous approaches. However, the class of aggregation/decomposition approaches is large and it deserves a special section. Finally, in Section IV we review the personalized approach to the graph based reputation measures. The personalized approach uses the information available locally. This is a natural approach to the reputation measures as the reputation discounts quickly in the chain of acquaintances.

2. ASYNCHRONOUS APPROACH

The most standard way for the computation of the Rank vector is the method of power iteration. Namely, in the power iteration method one just need to iterate equation (3).

$$\pi^{(t+1)} = \pi^{(t)} cP + \frac{1-c}{n} e^T, \quad t = 0, 1, \dots , \tag{4}$$

with $\pi^{(0)} = \frac{1}{n} e^T$. Since the matrix cP is substochastic, the algorithm converges. Furthermore, its convergence rate is bounded by c [12]. The number of FLOPS required to achieve the accuracy ε is equal to $\frac{\log \varepsilon}{\log c} nnz(P)$, where $nnz(P)$ is the number of nonzero elements of the matrix P [32]. Even though an implementation of the power iteration method for sparse matrices can be very efficient, one still would like to distribute its computation. The reasons for this are two-fold. Firstly, the computation on parallel computers can significantly accelerate the basic algorithm. In particular, one can apply GRID technology [9]. Secondly, in some applications like P2P network a distributed approach to the computation of reputation measures is indispensable. Below we review deterministic and stochastic approaches to the asynchronous computation of the graph based reputation measures.

The asynchronous iterations for the solution of fixed point linear systems like (3) was proposed in [6]. The class of asynchronous iterative method of [6] can be described as follows:

$$x_j^{(t+1)} = \begin{cases} \sum_{i=1}^n cP_{ij} \pi_i^{(t-d(t,i,j))} + \frac{1-c}{n} & \text{if } j \in U(t), \\ x_j^{(t)} & \text{if } j \notin U(t), \end{cases} \tag{5}$$

where the function $U(t)$ gives a set of states to be updated at each step, and the function $d(t,i,j)$ gives the relative "age" of the entries used in the updates. Then, from [13, 14] we have the following result about the convergence of asynchronous methods.

THEOREM 1. *Let the functions $U(t)$ and $d(t,i,j)$ satisfy the following conditions:*

1. *Each vector entry, j, features in an infinite number of update sets;*
2. *For each pair of vector entries, i and j, we have that $(t - d(t,i,j)) \to \infty$ as $t \to \infty$ as well as $\forall t : d(t,i,j) \leq t$.*

Then, if the spectral radius of cP is strictly less than one, every sequence of iterates within the class given by (5) converges to the unique fixed point.

The authors of [20] have shown that the asynchronous iterates also converge in the eigenproblem formulation with the largest eigenvalue equal to one.

Monte Carlo method provides a framework for the construction of stochastic asynchronous methods [2, 5]. Let us for example describe one particular method from [2].

ALGORITHM 1. *Simulate N runs of the random walk initiated at a randomly chosen node. For any node j, evaluate π_j as the total number of visits to node j multiplied by $(1-c)/N$.*

We note that the random walks are generated independently, which provides a natural framework for distributed implementation. As was shown in [2], to find nodes with high reputation it is enough to simulate the random walk a number of times equal to the number of nodes. This is in turn equivalent to the complexity of just one iteration of the power iteration method.

3. AGGREGATION/DECOMPOSITION APPROACH

Aggregation/decomposition methods for computation of the Rank vector use the decomposition of the set of pages which we denote by $\mathcal{I}$. Let us assume that the set $\mathcal{I}$ is decomposed into $N \leqslant n$ non-intersecting sets $\mathcal{I}^{(i)}$, $i = 1, \dots, N$. Lets

$$\begin{aligned} \mathcal{I}^{(1)} &= \{1, \dots, n_1\}, \\ \mathcal{I}^{(2)} &= \{n_1 + 1, \dots, n_1 + n_2\}, \\ &\vdots \\ \mathcal{I}^{(N)} &= \left\{ \textstyle\sum_{i=1}^{N-1} n_i + 1, \dots, \sum_{i=1}^{N} n_i \right\}, \end{aligned} \tag{6}$$

where $\sum_{i=1}^{N} n_i = n$.

According to the decomposition of the set of pages the transition matrix can be also be partitioned as follows:

$$P = \begin{pmatrix} P_{11} & P_{12} & \dots & P_{1N} \\ P_{21} & P_{22} & \dots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \dots & P_{NN} \end{pmatrix},$$

where P_{ij} is a block with dimension $n_i \times n_j$. In the same manner the Google matrix G can be presented in blocks,

$$G = \begin{pmatrix} G_{11} & G_{12} & \dots & G_{1N} \\ G_{21} & G_{22} & \dots & G_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ G_{N1} & G_{N2} & \dots & G_{NN} \end{pmatrix}. \quad (7)$$

Respect to the partitioning of the Google matrix, the Rank vector is partitioned into components:

$$\pi = (\pi_1, \pi_2, \dots, \pi_N), \quad (8)$$

where π_i is a row vector with $dim(\pi_i) = n_i$. All aggregation methods use an aggregation matrix A. The matrix A is a matrix which each element correspond to a block of matrix G, i.e. $a_{ij} \Leftrightarrow G_{ij}$. Usually the elements of the matrix A are formed as $a_{ij} = \zeta_i G_{ij} e$, where ζ_i is a probability distribution vector. We call the vector ζ_i aggregation vector. Each aggregation method forms the aggregation matrix in its own way using different probability distributions as aggregation vectors and different partitioning. One can consider the aggregated matrix as a transition matrix of a Markov chain with state space formed by sets of pages.

The convergence rate of an aggregation method depends on the choice of the decomposition. The aggregation method converges faster than power iteration method if off-diagonal blocks P_{ij} are close to zero matrix. It means that the random walk performed by the transition matrix G most likely stays inside sets $\mathcal{I}^{(i)}$ and with small probability goes out.

In the following discussion aggregation methods are applied to the Google matrix (1) and the Rank vector (2), but some of them can be applied to a general (irreducible or primitive) stochastic matrix and its stationary probability distribution.

3.1 Block-diagonal case

Let us consider the case when all blocks excluding the diagonal ones are zeroes [1], i.e.

$$P = \begin{pmatrix} P_1 & 0 & \dots & 0 \\ 0 & P_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P_N \end{pmatrix}.$$

Since P is a stochastic matrix then all P_i are stochastic. For the i^{th} block define the Google matrix

$$G_i = cP_i + (1-c)(1/n_i)E,$$

and let vector π_i be the Rank vector of G_i,

$$\pi_i = \pi_i G_i.$$

Then the Rank vector π is expressed by

$$\pi = \left(\frac{n_1}{n}\pi_1, \frac{n_2}{n}\pi_2, \dots, \frac{n_N}{n}\pi_N\right).$$

The block-diagonal structure of the matrix P allows to produce computation of each component of the Rank vector in absolutely independent way from others.

3.2 Full aggregation method (FAM)

The method is based on the theory of stochastic complement and the coupling theorem [24]. Here we introduce it for the completeness.

DEFINITION 1 (**Stochastic complement**). *For a given index i, let G_i denote the principal block submatrix of G obtained by deleting the i^{th} row and i^{th} column of blocks from G, and let G_{i*} and G_{*i} designate*

$$G_{i*} = (G_{i1} G_{i2} \cdots G_{i,i-1} G_{i,i+1} \cdots G_{iN})$$

and

$$G_{*i} = \begin{pmatrix} G_{1i} \\ \vdots \\ G_{i-1,i} \\ G_{i+1,i} \\ \vdots \\ G_{Ni} \end{pmatrix}.$$

That is, G_{i*} is the i^{th} row of blocks with G_{ii} removed, and G_{*i} is the i^{th} column of blocks with G_{ii} removed. The stochastic complement of G_{ii} in G is defined to be the matrix

$$S_i = G_{ii} + G_{i*}(I - G_i)^{-1} G_{*i}.$$

THEOREM 2 (**The coupling theorem [24, Theorem 4.1]**). *The Rank vector of the Google matrix G partitioned as (7) is given by*

$$\pi = (\nu_1 \sigma_1, \nu_2 \sigma_2, \dots, \nu_N \sigma_N),$$

where σ_i are the unique stationary distribution vector for the stochastic complement

$$S_i = G_{ii} + G_{i*}(I - G_i)^{-1} G_{*i}$$

and where

$$\nu = (\nu_1, \nu_2, \dots, \nu_N)$$

is the unique stationary distribution vector for the aggregation matrix A whose entries are defined by

$$a_{ij} = \sigma_i G_{ij} e.$$

In respect to the scope of the survey, Theorem 2 is given in application to the Google matrix. For the most general formulation of the theorem an interested reader is referred to [24]. Theorem 2 implies that the Rank vector can be found by the exact aggregation but it forces to compute the stochastic complements of diagonal blocks and their stationary distributions. One can avoid it by using approximate iterative aggregation method.

ALGORITHM 2. *Determine an approximation $\pi^{(k)}$ to the Rank vector π of a Google matrix G in k iterations.*

1. *Select a vector $\pi^{(0)} = \left(\pi_1^{(0)}, \pi_2^{(0)}, \dots, \pi_N^{(0)}\right)$ with $\pi^{(0)} e =$ 1.*

2. *Do* $k = 0, 1, 2 \ldots$

   (a) *Normalize* $\sigma_i^{(k)} = [\pi_i^{(k)}]$, $i = 1, \ldots, N$.

   (b) *Form aggregated matrix* $A^{(k)}$

$$a_{ij} = \sigma_i^{(k)} G_{ij} e.$$

   (c) *Determine the stationary distribution* $\nu^{(k)}$ *of* $A^{(k)}$

$$\nu^{(k)} = \nu^{(k)} A^{(k)}.$$

   (d) *Determine disaggregated vector*

$$\widetilde{\pi}^{(k)} = \left(\nu_1^{(k)} \sigma_1^{(k)}, \nu_2^{(k)} \sigma_2^{(k)}, \ldots, \nu_N^{(k)} \sigma_N^{(k)} \right).$$

   (e) *Do* l *steps of power iteration method*

$$\pi^{(k+1)} = \widetilde{\pi}^{(k)} G^l.$$

The Rank vector is the fix point of Algorithm 2. Indeed, if $\pi^{(k)} = \pi$, then $A^{(k)} = A$ and $\nu^{(k)} = \nu$. Therefore, $\widetilde{\pi}^{(k)} = \pi$, and $\pi^{(k+1)} = \pi$.

For the local convergence of the Algorithm 2 is required to fulfill one of the condition:

1. $G \gg 0$ and $G \geqslant \delta Q$, where $Q = e\pi$,
2. $G \geqslant eb$, where b is a row vector, $be = \delta$.

Since the Google matrix satisfies the both, Algorithm 2 converges locally [23, Theorem 1]. Algorithm 2 converges globally if l is large enough [23].

Let us provide an estimation of the rate of convergence of the method [27].

1. $G \geqslant \delta_1 Q$. Lets find δ_1. Lets denote by g_{min} the minimum entry of the matrix G. If $p_{ij} = 0$ then $g_{ij} = g_{min}$. Hence,

$$g_{min} = \frac{1-c}{n}. \tag{9}$$

   The maximum of one of the element of the Rank vector π is achieved when all the other elements achieving minimum, because of $\pi > 0$ and $\pi e = 1$. The minimum entry of the Rank vector for a page is realized if there is no other page referring to it. The minimum entry of the Rank vector is $\frac{1-c}{n}$. Therefore the maximum of one of the element of the Rank vector is equal to

$$\pi_{max} = 1 - \frac{1-c}{n}(n-1) = \frac{1+c(n-1)}{n}. \tag{10}$$

   Hence, if we find δ_1 from the constraint

$$g_{min} \geqslant \delta_1 \pi_{max}, \tag{11}$$

   it ensure us that $G \geqslant \delta_1 Q$. From the equalities (9,10,11) we get

$$\delta_1 \leqslant \frac{1-c}{1+c(n-1)}.$$

2. $G \geqslant eb$, where $be = \delta_2$. Lets determine δ_2. From the equalities (1) we obtain, that $G \geqslant \frac{1-c}{n} E$. The equality can be rewritten $G \geqslant e\left(\frac{1-c}{n} e^T\right)$. Therefore, as the vector b one can take $\left(\frac{1-c}{n} e^T\right)$. Hence,

$$\delta_2 = 1 - c.$$

The difference vector of the method at the k^{th} iteration is given by

$$\pi^{(k+1)} - \pi = (\pi^{(k)} - \pi) J(\pi^{(k)}).$$

The definition and expressions for the matrix $J(v)$ can be found in [23].

From the above estimation and [23] we can conclude that the spectral radius of the matrix $J(\pi)$

1. is less than $1 - \delta_1 = \frac{cn}{1+c(n-1)} < 1$,
2. is less than $\sqrt{1-\delta_2} = \sqrt{c} < 1$.

For the big enough n the second estimation becomes better than the first one. The second estimation ensure that the convergence rate of the method is no less than $\sqrt{c}$. Although, the estimation doesn't ensure us that the method converges faster than the power iteration method, but for the partial aggregation method which is discussed in the next subsection and is actually the particular case of the full aggregation method it was shown that there exists such partitioning of the Google matrix which provides faster convergence than power iteration method.

3.3 Partial aggregation method (PAM)

The partial aggregation method is considered in detail in [10]. Here we discuss the application of the method to the Google matrix and the Rank vector. The method is applied to the 2×2 case, i.e. $N = 2$, and the matrix G is partitioned as follows

$$G = \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix}.$$

The matrix $I - G$ is singular, and the matrix $I - G_{11}$ is nonsingular [3]. Hence we can factor $I - G = LDU$ [24, proof of Theorem 2.3], where

$$\begin{aligned} L &= \begin{pmatrix} I & 0 \\ -G_{21}(I - G_{11})^{-1} & I \end{pmatrix}, \\ D &= \begin{pmatrix} I - G_{11} & 0 \\ 0 & I - S_2 \end{pmatrix}, \\ U &= \begin{pmatrix} I & -(I - G_{11})^{-1} G_{12} \\ 0 & I \end{pmatrix}, \end{aligned}$$

where S_2 is a stochastic complement of the block G_{22}.

Since the matrix U is nonsingular we have $\pi(I - G) = 0$ if and only if $\pi LD = 0$. Hence

$$\pi_2 S_2 = \pi_2 \qquad \pi_1 = \pi_2 G_{21} (I - G_{11})^{-1}, \tag{12}$$

which means that π_2 is a stationary distribution for the matrix S_2. The expression (12) represent a particular case of the Theorem 2 for the 2×2 decomposition of the state [24, Colorary 4.1]. S_2 has unique stationary distribution

$$\sigma_2 S_2 = \sigma_2, \qquad \sigma_2 e = 1.$$

And we can find π_2 as $\pi_2 = \rho \sigma_2$, where the factor ρ is responsible for the normalization $\pi e = 1$.

The component π_1 and the factor ρ can be expressed as components of the stationary distribution of the aggregated matrix

$$A_1 = \begin{pmatrix} G_{11} & G_{12} e \\ \sigma_2 G_{21} & \sigma_2 G_{22} e \end{pmatrix}.$$

From (12), $\pi_2 = \rho\sigma_2$ ï£¡$\sigma_2 e = 1$ we get

$$(\pi_1, \rho)(I - A_1) = 0, \qquad (\pi_1, \rho)e = 1.$$

Since A_1 is stochastic and irreducible [24, Teorem 4.1], it has a unique stationary distribution α,

$$\alpha A_2 = \alpha, \qquad \alpha e = 1.$$

From the uniqueness we get $\alpha = (\pi_1, \rho)$.

The above analysis imply that the Rank vector can be found by the partial exact aggregation but it forces to compute the stochastic complement of G_{22} block of the Google matrix G and its stationary distribution. One can avoid it by using approximate iterative partial aggregation method.

ALGORITHM 3. *Determine an approximation $\pi^{(k)}$ to the Rank vector π of a Google matrix G in k iterations.*

1. *Select a vector $\pi^{(0)} = \left(\pi_1^{(0)}, \pi_2^{(0)}\right)$ with $\pi^{(0)}e = 1$.*
2. *Do $k = 0, 1, 2 \ldots$*
   (a) *Normalize $\sigma_2^{(k)} = [\pi_2^{(k)}]$.*
   (b) *Form aggregated matrix $A_1^{(k)}$*
   $$A_1^{(k)} = \begin{pmatrix} G_{11} & G_{12}e \\ \sigma_2^{(k)}G_{21} & \sigma_2^{(k)}G_{22}e \end{pmatrix}.$$
   (c) *Determine the stationary distribution $\alpha^{(k)}$ of $A_1^{(k)}$*
   $$\alpha^{(k)} = \alpha^{(k)}A_1^{(k)}.$$
   (d) *Partition $\alpha^{(k)}$*
   $$\alpha^{(k)} = (\omega_1^{(k)}, \rho^{(k)}).$$
   (e) *Determine disaggregated vector*
   $$\widetilde{\pi}^{(k)} = \left(\omega_1^{(k)}, \rho^{(k)}\sigma_2^{(k)}\right).$$
   (f) *Do l steps of power iteration method*
   $$\pi^{(k+1)} = \widetilde{\pi}^{(k)}G^l.$$

Lets consider $l = 1$. Algorithm 3 is the power iteration methods with matrix $\widetilde{G}$ [10, Proposition 5.1, Theorem 5.2], where

$$\widetilde{G} = \begin{pmatrix} 0 & 0 \\ G_{21}(I - G_{11})^{-1} & S_2 \end{pmatrix}$$

Therefore, the rate of convergence of Algorithm 3 is equal to $|\lambda_2(S_2)|$ [10, Theorem 5.2]. If power iteration methods converges for matrix G then Algorithm 3 converges, too [10, Proposition 7.1]. If we consider a general stochastic matrix instead of the Google matrix Algorithm 3 can converge slower power iteration method [10, Example 6.3], but for the Google matrix exists such decomposition which ensure that the method converges faster than power iteration method.

3.4 BlockRank Algorithm (BA)

The method exploit the site structure of the Web. According to the experiments made by Kamvar [17] the majority of the links are links between pages inside sites. Hence, the method decompose the set of pages $\mathcal{I}$ into the site, i.e. $\mathcal{I}^{(i)}$ is the pages of site i. The Google matrix is partitioned according to the decomposition of $\mathcal{I}$.

ALGORITHM 4. *Determine an approximation $\pi^{(k)}$ to the Rank vector π of the Google matrix G in k iterations.*

1. *Determine local Rank vector for each diagonal block P_i*
   (a) *Normalize P_i, i.e. $\left(\bar{P}_i\right)_{jk} = \frac{(P_i)_{jk}}{(P_i)_j \mathbf{1}}$.*
   (b) *Form G_i, $G_i = c\bar{P}_i + (1-c)(1/n)E$.*
   (c) *Approximately determine $\bar{\pi}_i$*
      i. *Select a vector $\pi_i^{(0)}$.*
      ii. *Do $k = 1, 2 \ldots$*
      $$\pi_i^{(k)} = \pi_i^{(k-1)}G_i.$$
2. *Determine BlockRank*
   (a) *Form aggregated matrix A*
   $$a_{ij} = \bar{\pi}_i P_{ij} e.$$
   (b) *Form B, $B = cA + (1-c)(1/n)E$.*
   (c) *Approximately determine β*
      i. *Select a vector $\beta^{(0)}$.*
      ii. *Do $k = 1, 2 \ldots$*
      $$\beta^{(k)} = \beta^{(k-1)}B.$$
3. *Determine global Rank vector*
   (a) *Form the vector $\pi^{(0)} = (\beta_1\bar{\pi}_1, \beta_2\bar{\pi}_2, \ldots, \beta_N\bar{\pi}_N)$.*
   (b) *Do $k = 1, 2 \ldots$*
   $$\pi^{(k)} = \pi^{(k-1)}G.$$

For the method it was empirically shown that it speeds up power iteration method by factor 2 and higher [17].

3.5 Fast Two-Stage Algorithm (FTSA)

The methods exploit the structure of the Web and precisely dangling pages [22]. The main idea of the method is to lump dangling pages into one state and find the Rank vector of the new aggregated matrix at the first stage and to aggregate non-dangling pages into one state at the second stage. Therefore, the set of pages is decomposed into two sets S_{ND} and S_D, where $S_{ND} \cup S_D = \mathcal{I}$, and S_{ND} contains all non-dangling pages and S_D contains all dangling pages. Hence, the matrix G is represented in the following way:

$$G = \begin{pmatrix} G_{11} & G_{12} \\ e_{n_1}v_{n_1} & e_{n_2}v_{n_2} \end{pmatrix},$$

where $e = \left(e_{n_1}^T, e_{n_2}^T\right)^T$ and $v = (v_{n_1}, v_{n_2})$.

ALGORITHM 5. *Determine an approximation $\pi^{(k)}$ to the Rank vector π of the Google matrix G in k iterations.*

1. *The first stage: lump dangling pages*
   (a) *Form the lumped matrix $G^{(1)}$*
   $$G^{(1)} = \begin{pmatrix} G_{11} & G_{12}e_{n_2} \\ v_{n_1} & v_{n_2}e_{n_2} \end{pmatrix}.$$
   (b) *Approximately determine π_1*
      i. *Select a vector $\pi_1^{(0)}$.*

ii. Do $k = 1, 2 \ldots$

$$\pi_1^{(k)} = \pi_1^{(k-1)} G^{(1)}.$$

(c) Determine aggregation weights of the second stage

$$\eta = \frac{\pi_1}{\sum_{i=1}^{n_1} (\pi_1)_i}.$$

2. The second stage: aggregate non-dangling pages

(a) Form aggregated matrix $G^{(2)}$

$$G^{(2)} = \begin{pmatrix} \eta G_{11} e_{n_1} & \eta G_{12} \\ v_{n_1} e_{n_1} & e_{n_2} v_{n_2} \end{pmatrix}.$$

(b) Approximately determine π_2

i. Select a vector $\pi_2^{(0)}$.

ii. Do $k = 1, 2 \ldots$

$$\pi_2^{(k)} = \pi_2^{(k-1)} G^{(2)}.$$

3. Form the result Rank vector

$$\pi = (\pi_1, \pi_2).$$

The first stage requiring less computation work, roughly $O(n_1)$ as opposed to $O(n)$, converges at least as fast as power iteration method. The second stage usually converges to the exact solution after three iterations. If the second stage doesn't converge after three iterations the exact limit vector can be found using Aitken Extrapolation [18]:

$$(\pi_2)_i = \frac{\left(\left(\pi_2^{(2)}\right)_i - \left(\pi_2^{(1)}\right)_i\right)^2}{\left(\pi_2^{(3)}\right)_i - 2\left(\pi_2^{(2)}\right)_i + \left(\pi_2^{(1)}\right)_i}.$$

Because of the second stage converge after three iterations and, possibly, applying Aitken Extrapolation, the method as a whole converges at least as fast as power iteration method.

3.6 Distributed PageRank Computation (DPC)

The method is designed naturally for distributed computation of the Rank vector [34]. The set of pages is decomposed by sites, i.e. $\mathcal{I}^{(i)}$ is the pages of site i. The main idea of the method is to allow each site to compute the Rank vector for local pages and after that construct the entire Rank vector vector. Further we refer to a site as a node which makes computations. Let us consider the method theoretically.

Lets $S(\pi)$ denote a $N \times n$ disaggregation matrix as

$$S(\pi) = \begin{pmatrix} S(\pi)_1 & 0 & \ldots & 0 \\ 0 & S(\pi)_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & S(\pi)_N \end{pmatrix},$$

where $S(\pi)_i = [\pi_i]$ is a row vector denoting the censored stationary distribution of pages in site i. Also let us denote by G_i the ith block row of the matrix G partitioned as (7), i.e.

$$G_i = (G_{i1}, G_{i2}, \ldots, G_{iN}).$$

ALGORITHM 6. *Determine an approximation $\pi^{(k)}$ to the Rank vector π of a Google matrix G in k iterations.*

1. Each node i

(a) Construct an $n_i \times n_i$ local transition matrix $\bar{G}_{ii}$, i.e. normalize G_{ii}

$$\left(\bar{G}_{ii}\right)_{jk} = \frac{(G_{ii})_{jk}}{(G_{ii})_j \mathbf{1}}.$$

(b) Determine the stationary distribution $\pi_i^{(0)}$ for G_i.

2. Do k = 0, 1, 2, ...

3. Main node

(a) Normalize $\sigma_i^{(k)} = [\pi_i^{(k)}]$.

(b) Construct aggregated matrix $A^{(k)}$

$$a_{ij} = \sigma_i^{(k)} G_{ij} e.$$

(c) Determine the stationary distribution $\nu^{(k)}$ for $A^{(k)}$.

4. Each node i

(a) Construct an $(n_i + 1) \times (n_i + 1)$ extended local transition matrix

$$B_i^{(k)} = \begin{pmatrix} G_{ii} & e^T (I - G_{ii}) \\ \frac{\left(\nu^{(k)} S(\pi^{(k)}) G_i - \nu_i^{(k)} \pi_i^{(k)} G_{ii}\right)}{1 - \nu_i^{(k)}} & \alpha^k \end{pmatrix},$$

where the scalar α^k ensures the row sum of $B_i^{(k)}$ is one.

(b) Determine the stationary distribution b_i^k for $B_i^{(k)}$.

(c) Partition b_i^k

$$b_i^k = (\omega_i^{(k)}, \beta_i^{(k)}),$$

where $\beta_i^{(k)}$ is a scalar.

(d) Form local vector $\tilde{\pi}_i^{(k)}$

$$\tilde{\pi}_i^{(k)} = \frac{1 - \nu_i^{(k)}}{\beta_i^{(k)}} \omega_i^{(k)}.$$

5. Main node

(a) Construct vector $\tilde{\pi}^{(k)}$

$$\tilde{\pi}^{(k)} = \left(\tilde{\pi}_1^{(k)}, \tilde{\pi}_2^{(k)}, \ldots, \tilde{\pi}_N^{(k)}\right).$$

(b) Normalize $\pi^{(k)} = [\tilde{\pi}^{(k)}]$.

The method is proved to be theoretically equivalent to iterative aggregation-disaggregation method based on Block Jordan decomposition [34]. The main advantage of the method is that it provides a distributed way to calculate the Rank vector, at the same time the communication overhead is not high due to only scalars and vector are being sent between nodes and never matrices are. The entire communication overhead is of the magnitude $O(nnz((\bar{L} + \bar{U})S(\pi))) + O(n)$, where $\bar{L}$ and $\bar{U}$ are low-triangular and upper-triangular matrices of the matrix P, correspondingly.

3.7 Discussion of the Aggregation/Decomposition methods

The considered Aggregation/Decomposition methods can be classified into two groups. Whiles FAM and PAM are general methods which can be applied to any decomposition of the set of pages, BA and FTSA propose to use a particular decomposition. The rate of convergence of FAM and PAM depends on the chosen decomposition essentially and the problems of "optimal" decomposition of the set of pages and convergence acceleration are actual. BA and FTSA made a step in the direction of solving both the problems. Whilst BA experimentally speeds up power iteration method, FTSA converging as power iteration method reduces the dimension of matrices and vectors participating in iterations. DPC being distributed and parallel method which converges like aggregation-disaggregation method and has low communication overhead provides another way to calculate the Rank vector.

4. PERSONALIZED APPROACH

The presented above algorithms compute the global reputation measure. Namely, to calculate the Rank vector we need an input (may be indirect) from all the nodes. However, the reputation discounts quickly in the chain of acquaintances. This provides a motivation to consider "localized" or "personalized" versions of the graph based reputation measures. Furthermore, one often needs to encompass different notions of importance or reputation for different users and queries. Thus, the original algorithm should be modified to take into account personalized view for the reputation of the nodes.

As we mentioned in the introduction, in general a random walk follows the outgoing links with probability c, and makes a random jump with probability $(1-c)$ according to the probability distribution given in v. Depending on the "type" of users, vector v will not be uniform, but biased to some set of nodes, which are considered to be important for these "types" of users. For this reason, the vector v is referred as *personalization* vector. Let $\pi(v)$ denote the personalized Rank vector, corresponding to the personalization vector v. It can be computed by solving the equation $\pi = \pi G$

$$\begin{aligned} \pi &= \pi cP + (1-c)v, \\ \pi - \pi cP &= (1-c)v, \\ \pi(I-cP) &= (1-c)v. \end{aligned}$$

Since c is different from one, the matrix $(I-cP)$ is invertible and we have

$$\pi(v) = v(1-c)(I-cP)^{-1}. \quad (13)$$

Let $Q = (1-c)(I-cP)^{-1}$. By letting $v = e_i^T$ [1] we see that $\pi(e_i) = Q^i -$ the i^{th} row of Q. Thus, rows of Q comprise a complete basis for personalized Rank vectors (PRVs). Any PRV can be expressed as a convex linear combination of these basic vectors. This statement is based on the following theorem:

THEOREM 3. *Given two arbitrary π_1, π_2 PRVs and v_1, v_2 are their corresponding personalization vectors. Then, for any constant $\alpha_1, \alpha_2 > 0$ such that $\alpha_1 + \alpha_2 = 1$*

$$\alpha_1\pi_1 + \alpha_2\pi_2 = c(\alpha_1\pi_1 + \alpha_2\pi_2)P + (1-c)(\alpha_1 v_1 + \alpha_2 v2) \quad (14)$$

For any personalization vector v, the corresponding PRV is given by : vQ. Unfortunately, approach to use the complete basis for the personalized Rank vector is infeasible in practice. Computing the dense matrix Q off line is impractical due to its huge size. However, rather than using the full basis, we can use a reduce basis with $k < n$ vectors. In this case, we can not express all PRVs but only those corresponding to convex combinations of the vectors in reduced basis set

$$\pi(\omega) = w\hat{Q}. \quad (15)$$

4.1 Scaled Personalization

In [15] the authors have presented a method that enables the computation of PRVs which scales well with the increasing number of users. The authors of [15] have developed their method in the context of information retrieval. Then, the authors of [7] have adopted the method of [15] to the reputation management in P2P networks. In this survey, we also present a broader reputation measure based interpretation of the algorithm of [15]. We would like to mention that the division of the users in two groups: pre-trusted peers and regular users provide yet another application of the results of [7, 15] in the context of Bionets [4].

The central notion of the scaled personalization algorithm is the set of pre-trusted peers (or hub peers). A regular pear can choose some of pre-trusted peers. This does not mean that the hub peers selected by a user more trustworthy than the other hub peers. This simply means that a user might prefer certain hub peers because they supply a specific service of a very good quality. Next let us describe several stages of the scaled personalization algorithm.

Specification.

Let us consider a set of the personalization vectors u_h where $u_h = e_h$ is biased to a specific hub node $h \in H$. We denote by H the set of hub nodes. The personalized Rank vector corresponding to u_h is called a basis hub vector π_h. If the basis vector for each hub node $h \in H$ were computed and stored, then, from Theorem 3 any PRV corresponding to a preference set $P \subseteq H$ can be computed. The preference set P corresponds to the set of hub nodes chosen by a user as preferred pre-trusted peers.

Each hub vector can be computed naively by power method. However, this task is very expensive in time and resources. The algorithm of [15] enables a more scalable computation by constructing hub vectors from shared components.

Decomposition of Basis Vectors.

To compute a large number of basis hub vectors efficiently, one can decomposed them into partial vectors and hubs skeleton, components from which hub vectors can be constructed quickly.

Let define the *inverse P-distance* $r'_p(q)$ from p to q as

$$r'_p(q) = \sum_{t:p \rightsquigarrow q} P[t](1-c)c^{l(t)}, \quad (16)$$

where the summation is taken over all tour t, starting from p and finishing at q, possibly touching p and q more than one time, $l(t)$ is the length of the tour, and $P[t]$ is interpreted as the probability of taking the tour t.

[1] e_i has 1 in i^{th} place, and 0 elsewhere.

Consider tour $t =< w_1, \ldots, w_k >$, then $P[t] = \prod_{i=1}^{k-1} \frac{1}{Outdeg(w_i)}$, or 1 if $l(t) = 0$. If there is no any tours from p to q, the summation is taken to be equal to 0. It is proven that $\pi_p(q) = r_p(q)$ [15].

Similarly, *restricted inverse P-distance is defined:*

Let $H \subseteq V$ be some nonempty set of nodes. For $p, q \in V$, $r_p^H(q)$: a restriction of $r_p(q)$ that considers only tours from p to q that pass through H.

$$r_p^H(q) = \sum_{t:p \rightsquigarrow H \rightsquigarrow q} P[t](1-c)c^{l(t)} \quad (17)$$

Intuitively, $r_p^H(q)$ is the influence of p on q through H. Obviously, if all paths from $p \rightsquigarrow q$ come through H, then $r_p^H(q) = r_p(q)$. For *carefully* chosen H, $r_p(q) - r_p^H(q) = 0$ for many pages p, q. The strategy is to take advantage of this property by breaking r_p into components $(r_p - r_p^H)$ and r_p^H.

$$\pi_p = r_p = (r_p - r_p^H) + r_p^H. \quad (18)$$

$(r_p - r_p^H)$ is called the partial vector. Computing and storing partial vectors is cheaper, since they can be represented as a list of their nonzero entries. Moreover, the size of each partial vector will decrease as H increases in size, making this approach particularly scalable. It can be proven that any r_p^H vector can be expressed in terms of the partial vectors $(r_h - r_h^H)$ for $h \in H$ (see the Hub Theorem in [15])

THEOREM 4. *For any $p \in V, H \subseteq V$,*

$$r_p^H = \frac{1}{1-c} \sum_{h \in H} (r_p(h) - (1-c)x_p(h))(r_h - r_h^H - (1-c)x_h), \quad (19)$$

where $x_h = e_h$. The quantity $(r_h - r_h^H)$ appears on the right side of (19) is the partial vector. Suppose we have computed $r_p(H) = \{(h, r_p(h)) | h \in H\}$ for a hub node p. Substitute into equation (18):

$$r_p = (r_p - r_p^H) +$$

$$\frac{1}{1-c} \sum_{h \in H} (r_p(h) - (1-c)x_p(h))[(r_h - r_h^H) - (1-c)x_h]. \quad (20)$$

The equation is central to the construction of hub vectors from partial vectors. The set $S = \{r_p(H) | p \in H\}$ forms the hubs skeleton, giving the interrelationships among partial vectors. Computing $(r_p - r_p^H), p \in H$ naively by power method is inefficient due to the huge number of hub nodes. Three scalable algorithms for computing these partial values, using dynamic programming were presented. All of them are based on the decomposition theorem

THEOREM 5. *For any $p \in V$*

$$r_p = \frac{c}{|O(p)|} \sum_{i-1}^{|O(p)|} r_{O_i(p)} + (1-c)x_p \quad (21)$$

where $O_i(p)$ is the i^{th} neighbor of node p.

The above theorem gives the interpretation for PRV. The p's view of r_p is the average of the views of its out-neighbors, but with extra importance given to p itself.

Construction of PRV's.

Let $u = \alpha_1 p_1 + \ldots + a_z p_z$ be a preference vector, where $p_i \in H$. Let

$$r_u(h) = \sum_{i=1}^{z} \alpha_i (r_{p_i}(h) - c.x_{p_i}(h)). \quad (22)$$

Then, the PRV π can be computed as follows:

$$\pi = \sum_{i=1}^{z} \alpha_i (r_{p_i} - r_{p_i}^H) + \frac{1}{1-c} \sum_{h \in H} r_u(h)[(r_h - r_h^H) - (1-c)x_h]. \quad (23)$$

The choice of H.

The choice of hub nodes can have a strong effect to the overall performance. Particularly, the size of partial vectors is smaller when pages in H have high Rank vector values, since nodes with high Rank vector values are closer in term of P-reverse distance to other pages. In the context of P2P networks, it's reasonable for the members in the pre-trusted peers to have high Pagerank.

4.2 Relation to the Aggregation/Decomposition approach

Let us relate the Personalized Rank vector approach to the Aggregation/Decomposition approach. The BlockRank algorithm proposed by Kamvar et al. [17] computes $n \times k$ matrix corresponding to k blocks. Each block corresponds to a host. Instead of being able to choose a distribution over pages to which the user jumps, he may chooses host. So, we can encode the personalization vector in the k-dimensional space. With the restriction of this model, the local Rank vector $\vec{l_j}$ will not change for different personalizations. Only the block rank $\vec{b}$ depends on the personalizations. Therefore, we only need to recompute the BlockRank $\vec{b}$ for each block-personalization vector v_k. The BlockRank algorithm is able to exploit the graph's block structure to compute efficiently many of the block-oriented basis vectors.

Acknowledgments

This work was supported by the European project BioNets. The authors would also like to thank Sara Alouf, Roberto Cascella and the anonymous reviewer for the helpful suggestions.

5. REFERENCES

[1] K. Avrachenkov and N. Litvak. Decomposition of the Google PageRank and Optimal Linking Strategy. *Inria Sophia Antipolis, University of Twente*, 2004.

[2] K. Avrachenkov, N. Litvak, D, Nemirovsky and N. Osipova, "Monte Carlo methods in PageRank computation: When one iteration is sufficient", *SIAM Journal on Numerical Analysis*, v.45, no.2, pp.890-904, 2007.

[3] A. Berman and R.J. Plemmons. Nonnegative Matrices in the Mathematical Sciences. *SIAM Classics In Applied Mathematics, SIAM, Philadelphia*, 1994.

[4] Biologically inspired Network and Services (BIONETS): http://www.bionets.org/

[5] L.A. Breyer, "Markovian Page Ranking Distributions: Some Theory and Simulations", Technical report,

2002; available online at http://www.lbreyer.com/preprints.html.

[6] D. Chazan and W.L. Miranker, "Chaotic relaxation, *Linear Algebra and its Applications*, v.2, pp.199-222, 1969.

[7] P.A. Chirita, W. Nejdl, M. Schlosser, and O. Scurtu, "Personalized reputation management in P2P networks", in Proceedings of the Trust, Security, and Reputation Workshop, 2004.

[8] L. Eschenauer, V. Gligor and J. Baras, "On trust establishment in mobile Ad Hoc networks", in *Proceedings of Security Protocols Workshop*, pp.47-66, 2002.

[9] I. Foster and C. Kesselman, (eds.) *The GRID: Blueprint for a new computing infrastructure*, Elsevier, San Francisco, 2004.

[10] C.F. Ipsen and S. Kirklad. Convergence analysis of an improved PageRank algorithm. *NCSU CRSC Technical Report*, 2004.

[11] T.H. Haveliwala, "Topic-Sensitive PageRank", in *Proceedings of the 11th International World Wide Web Conference*, 2002.

[12] T.H. Haveliwala and S.D. Kamvar, "The Second Eigenvalue of the Google Matrix", Stanford University Technical Report, March 2003.

[13] , D. de Jager, *PageRank: Three distributed algorithms*, Master thesis, Imperial College (University of London), 2004.

[14] D. de Jager and J.T. Bradley, "Asynchronous iterative solution for state-based performance metrics", in *Proceedings of ACM SIGMETRICS 2007.*

[15] G. Jeh and J. Widom, "Scaling Personalized Web Search", in *Proceedings of the 12th International World Wide Web Conference*, 2003.

[16] A. Josang, R. Ismail and C. Boyd, "A survey of trust and reputation systems for online service provision", *Decision Support Systems*, v.43, no.2, pp.618-644, 2007.

[17] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. *Stanford University Technical Report*, 2003.

[18] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, and G.H. Golub, "Extrapolation methods for accelerating PageRank computations", in *Proceedings of the 12th International World Wide Web Conference*, 2003.

[19] S.D. Kamvar, M.T. Schlosser and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks", in *Proceedings of the 12th International World Wide Web Conference*, 2003.

[20] G. Kollias, E. Gallopoulos and D. Szyld, "Asynchronous iterative computations with Web information retrieval structures: The PageRank case", in *Proceedings of the International Conference ParCo 2005. Parallel Computing: Current & Future Issues of High-End Computing*, pp.309-316.

[21] A.N. Langville and C.D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2005.

[22] Ch.P.Ch. Lee, G.H. Golub, and S.A. Zenios, "A fast two-stage algorithms for computing PageRank", SCCM Report, 2002.

[23] I. Marek and I. Pultarova. Two notes on local and global convergence analysis of iterative aggregation-disaggregation method. *available on the Web*, 2005.

[24] C.D. Meyer and R.J. Plemmons. Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems. *SIAM Rev.*, 1989.

[25] C.B. Moler. *Numerical Computing with MATLAB.* SIAM, 2004.

[26] L. Mui, M. Mohtashemi and A. Halberstadt, "A computational model of trust and reputation", in *Proccedings of the 35th Hawaii International Conference on System Sciences*, 2002.

[27] D.A. Nemirovsky. Analysis of iterative method for PageRank computation based on decomposition of the Web graph. *Master thesis, St.Petersburg State University, in Russian*, 2005.

[28] The Open Directory Project *www.dmoz.org*

[29] L. Page, S. Brin, R. Motwani and T. Winograd, "The PageRank citation ranking: Bringing order to the Web", Technical Report, Stanford Digital Library Technologies Project, 1998.

[30] K. Sankaralingam, S. Sethumadhavan and J.C. Browne, "Distributed pagerank for P2P systems", in *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, 2003.

[31] S.M. Shi, J. Yu, G.W. Yang and D.X. Wang, "Distributed page ranking in structured P2P networks", in *Proceeding of International Conference on Parallel Processing*, 2003.

[32] W.J. Stewart. Introduction to the numerical solutions of Markov chains. *Princeton University Press, Princeton*, 1994.

[33] A. Yamamoto, D. Asahara, T. Itao, S. Tanaka and T. Suda, "Distributed pagerank: a distributed reputation model for open peer-to-peer network", in *Proceedings of the International Symposium on Applications and the Internet Workshops*, 2004.

[34] Y. Zhu and Sh. Ye and X. Li, "Distributed PageRank computation based on iterative aggregation-disaggregation methods", in *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005.

[35] C.-N. Ziegler and G. Lausen, " Spreading activation models for trust propagation", in *Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2004.

[36] C.-N. Ziegler and G. Lausen, "Propagation models for trust and distrust in social networks", *Information Systems Frontiers*, v.7, no.4/5, pp.337-358, 2005.