Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

A Distributed Randomized Approach for the PageRank Computation: Part 2

Hideaki Ishii
Department of Computational Intelligence & Systems Science
Tokyo Institute of Technology, Yokohama 226-8502, Japan
ishii@dis.titech.ac.jp

Roberto Tempo
IEIIT-CNR, Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
roberto.tempo@polito.it

***Abstract*— In the search engine of Google, the PageRank algorithm plays a crucial role in ranking the obtained results. The algorithm determines the importance of each web page based on the link structure of the web. In this two-part paper, we propose a distributed randomized approach for the PageRank computation, where the pages locally update their values by communicating with linked pages. This paper is the second part, and we develop two enhanced distributed schemes which deal with simultaneous updates and update termination of the computations, respectively.**

***Index Terms*— Distributed computation, Multi-agent consensus, PageRank algorithm, Randomization, Stochastic matrices**

I. Introduction

To efficiently search the web, it is critical to use search engines that provide lists of websites matching the users' needs. At Google, the search results take account of the rankings made by the so-called PageRank algorithm; see, e.g., [2], [3], [10]. This algorithm quantifies the importance and the quality of web pages by utilizing the link structure of the web. The basic idea is to count the number of pages that cite a given page under some normalization rules.

As the web today consists of billions of pages, the computation of the PageRank is a difficult and challenging task. Currently, this is performed at Google in a centralized fashion, and it is reported that this computation takes about a week. However, as the web continues to grow rapidly, it is clearly necessary to develop more efficient computational methods [1], [4], [9], [15].

In this two-part paper with [8], we propose a distributed randomized approach for the PageRank computation; for recent advances in probabilistic methods in systems and control, see [12]. The approach has three main features as follows: First, in principle, each page computes its own PageRank value locally by communicating with the pages that are connected by direct links. That is, each page exchanges its value with the pages that it links to and those linked to it. Second, the pages make the decision to initiate this communication at random times. This means that, in the implementation, there is neither a fixed order among the pages nor a centralized agent in the web that determines the pages to update their values. Third, the amount of computation required at each page is very mild.

This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology, Japan, under Grant-in-Aid for Scientific Research No. 17760344

As we have discussed in detail in [8], this approach has been motivated by the current research in multi-agent consensus problems; among the many recent works, it has an especially close relation to the stochastic approaches in, e.g., [5], [11], [13], [14]. At the conceptual level, it is natural to view the web as a network of agents that can make their own computation and communicate with neighboring agents. Also, at the technical level, the two problems share similarities related to the use of stochastic matrices.

In this second part, we aim at enhancing the flexibility in the approach especially from the implementation viewpoint. In particular, we focus on regulating the computation load at the pages and the communication rate among the pages in the distributed scheme. We propose two directions for extending the algorithm in [8] and establish convergence results.

The first scheme allows the random update times to be determined in a fully distributed way. Each page randomly updates its PageRank value, and hence simultaneous updates by multiple pages are possible. In contrast, in the previous scheme in [8], only one page is allowed to make an update; this may not be practical for the real web. The probability to update is given in terms of a new parameter. This scheme reduces to the original centralized algorithm, if all agents update all the time, and is consequently its generalization.

In the second scheme, the pages terminate the update in their PageRank values when they converge to a certain predetermined level. Once those values are communicated to the linked pages, for this page, no further computation/communication is needed. This method is based on [9].

The paper is organized as follows: An overview of the PageRank problem is given in Section II. In Section III, we provide the distributed algorithm where multiple pages can make simultaneous updates. In Section IV, the distributed scheme with update termination is described. We illustrate the results through a numerical example in Section V. The paper is concluded in Section VI.

Notation: For vectors and matrices, inequalities are used to denote entry-wise inequalities: For $X, Y \in \mathbb{R}^{n\times m}$, $X \leq Y$ implies $x_{ij} \leq y_{ij}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$; in particular, we say that the matrix X is nonnegative if $X \geq 0$ and positive if $X > 0$. A probability vector is a nonnegative vector $v \in \mathbb{R}^n$ such that $\sum_{i=1}^n v_i = 1$. Unless otherwise specified, by a stochastic matrix, we refer to a column-stochastic matrix, i.e., a nonnegative matrix $X \in \mathbb{R}^{n\times n}$ with the property that $\sum_{i=1}^n x_{ij} = 1$ for $j = 1, \ldots, n$.

Let $\mathbf{1} \in \mathbb{R}^n$ be the vector with all entries equal to 1 as $\mathbf{1} := [1 \ \cdots \ 1]^T$. Similarly, $S \in \mathbb{R}^{n\times n}$ is the matrix with all entries being 1. For $x \in \mathbb{R}^n$, we denote by $|x|$ the vector containing the absolute values of the corresponding entries of x. The norm $\|\cdot\|$ for vectors is the Euclidean norm. The spectral radius of the matrix $X \in \mathbb{R}^{n\times n}$ is denoted by $\rho(X)$.

II. The PageRank problem

We briefly introduce the PageRank problem based on, e.g., [2], [3], [10].

Consider a network of n web pages indexed by integers from 1 to n. The network is represented by the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V} := \{1, 2, \ldots, n\}$ is the set of vertices corresponding to the web page indices while $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges representing the links among the pages. The vertex i is connected to the vertex j by an edge, i.e., $(i, j) \in \mathcal{E}$, if page i has an outgoing link to page j, or in other words, page j has an incoming link from page i.

The objective of the PageRank algorithm is to provide some measure of importance to each web page. The PageRank value, or simply the value, of page $i \in \mathcal{V}$ is a real number $x^* \in [0, 1]$. The values are ordered: $x_i^* > x_j^*$ implies that page i is more important than page j.

The pages are ranked according to the rule that a page having links from important pages is also important. This is done in such a way that the value of one page equals the sum of the contributions from all pages that have links to it. In particular, we define the value of page i by

$$x_i^* = \sum_{j\in\mathcal{L}_i} \frac{x_j^*}{n_j},$$

where $\mathcal{L}_i := \{j : \ (j, i) \in \mathcal{E}\}$, i.e., this is the set of page indices that are linked to page i, and n_j is the number of outgoing links of page j. It is customary to normalize the total of all values as $\sum_{i=1}^n x_i^* = 1$.

Let the values be in the vector form as $x^* \in [0, 1]^n$. Then, the PageRank problem can be restated as

$$x^* = Ax^*, \quad x^* \in [0, 1]^n, \quad \sum_{i=1}^n x_i^* = 1, \tag{1}$$

where the link matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ is given by

$$a_{ij} := \begin{cases} \frac{1}{n_j} & \text{if } j \in \mathcal{L}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The value vector x^* is a nonnegative unit eigenvector corresponding to the eigenvalue 1 of A. In general, however, for this eigenvector to exist and then to be unique, it is sufficient that (i) the so-called dangling nodes, which are pages having no links to others, do not exist, and (ii) the web as a graph is strongly connected[1].

To simplify the issue regarding dangling nodes, we redefine the graph by bringing in artificial links. As a result, the link matrix A becomes a stochastic matrix, having at least one eigenvalue equal to 1. For more on this, see [8].

[1]A directed graph is said to be strongly connected if for any two vertices $i, j \in \mathcal{V}$, there is a sequence of edges which connects i to j.

The web is known not to be strongly connected in general. To avoid this problem, a modified version of the values has been introduced in [2] as follows: Let $m \in (0, 1)$, and let the modified link matrix $M \in \mathbb{R}^{n\times n}$ be defined by

$$M := (1 - m)A + \frac{m}{n}S. \tag{3}$$

This matrix is clearly positive and is further stochastic being a convex combination of two stochastic matrices A and S/n. By the Perron-Frobenius Theorem [6], there exists a unique positive eigenvector for the eigenvalue 1. Hence, we redefine the value vector x^* by using M in place of A in (1) as

$$x^* = Mx^*, \quad x^* \in [0, 1]^n, \quad \sum_{i=1}^n x_i^* = 1. \tag{4}$$

Because of the large dimension of the link matrix M, the computation of the value vector x^* relies on the power method. That is, x^* is computed through the recursion

$$x(k + 1) = Mx(k), \tag{5}$$

where $x(k) \in \mathbb{R}^n$ and the initial condition $x(0) \in \mathbb{R}^n$ is a probability vector.

The following lemma shows that, using this method, we can asymptotically find the value vector (e.g., [6]).

Lemma 2.1: For any initial condition $x(0)$, in the update scheme (5) using the modified link matrix M, it holds that $x(k) \to x^*$ as $k \to \infty$.

We note that the rate of convergence relies on the value of m. In [2], a typical value is set as $m = 0.15$;

III. A distributed scheme with simultaneous updates

In this section, we propose a distributed algorithm to compute the value vector x^*. This is a generalization of the simpler scheme in [8].

Consider the web with n pages from the previous section. It is assumed that each page computes its own value locally by communicating with the pages linked to it. At any time instant, some pages initiate the updates of their values. For each page, this is performed by (i) sending its value to the pages that it is linked to and (ii) requesting the pages that link to it for their values. All pages involved in this process renew their values based on the latest available information.

These updates can take place in a fully distributed and randomized manner. The decision to make an update is a random variable. In particular, this is determined under a given probability $\alpha \in (0, 1]$ at each time k, and hence, the decision can be made locally at each page. In practice, this scheme can be realized without requiring a centralized decision maker or any fixed order among the pages for updates. Note however that the probability α is a global parameter in that all pages have the same α.

Formally, the proposed distributed update scheme is described as follows. Let $\eta_i(k) \in \{0, 1\}$, $i = 1, \ldots, n$, $k \in \mathbb{Z}_+$, be Bernoulli processes given by

$$\eta_i(k) = \begin{cases} 1 & \text{if page } i \text{ initiates an update at time } k, \\ 0 & \text{otherwise} \end{cases}$$

where their probability distributions are specified as

$$\alpha := \mathrm{Prob}\{\eta_i(k) = 1\}. \tag{6}$$

As in the previous section, we first describe the approach in the A-matrix domain to simplify the discussion. The distributed update scheme is given by

$$x(k+1) = A_{\eta_1(k),\ldots,\eta_n(k)}x(k), \tag{7}$$

where $x(k) \in \mathbb{R}^n$ is the state whose initial condition $x(0)$ is a probability vector, and $A_{\eta_1(k),\ldots,\eta_n(k)}$ are called the distributed link matrices.

Let $y_i(k)$ be the time average of the states $x_i(0), \ldots, x_i(k)$ given by

$$y_i(k) := \frac{1}{k+1}\sum_{\ell=0}^{k} x_i(\ell), \quad k \in \mathbb{Z}_+,\ i \in \mathcal{V}. \tag{8}$$

We say that, for the distributed update scheme, the PageRank value x^* is obtained through the time average y if, for each initial condition $x(0)$, $y(k)$ converges to x^* in the mean-square sense as

$$E\left[\left\|y(k) - x^*\right\|^2\right] \to 0, \quad k \to \infty.$$

The problem of distributed PageRank computation is formulated as follows: Find the distributed link matrices such that, for the corresponding distributed update scheme, the PageRank value x^* is obtained through the time average.

This problem is a generalization of that in [8], where the class of distributed link matrices is limited; there, only one of the pages is updated at a time. To emphasize the difference, the current approach is called the distributed scheme with simultaneous updates. The analysis of this scheme is more involved as we shall see.

A. Distributed link matrices

Let the distributed link matrices $A_{\eta_1,\ldots,\eta_n}$ be given by

$$\left(A_{\eta_1,\ldots,\eta_n}\right)_{ij} := \begin{cases} a_{ij} & \text{if } \eta_i = 1 \text{ or } \eta_j = 1, \\ 1 - \sum_{h:\ \eta_h = 1} a_{hj} & \text{if } \eta_i = 0 \text{ and } i = j, \\ 0 & \text{if } \eta_i = \eta_j = 0 \text{ and } i \neq j \end{cases} \tag{9}$$

for $\eta_r \in \{0,1\}$, $r \in \{1,\ldots,n\}$, and $i,j \in \{1,\ldots,n\}$. Clearly, there are 2^n matrices. They have the property that (i) if $\eta_i = 1$, then the ith column and the ith row are the same as those in the original link matrix A, (ii) if $\eta_i = 0$, then the ith diagonal entry is chosen such that the entries of the ith column add up to 1, and (iii) all other entries are 0. Hence, these are constructed as stochastic matrices. We note that these link matrices coincide with those in equation (11) of [8] when $\eta_i = 1$ for one i and $\eta_j = 0$ for all $j \neq i$.

B. The average link matrix

We now analyze the average dynamics of the distributed update scheme in (7). We define the average link matrix by

$$\overline{A} := E\left[A_{\eta_1(k),\ldots,\eta_n(k)}\right], \tag{10}$$

where $E[\,\cdot\,]$ is the expectation with respect to $\eta_i(k)$, $i \in \mathcal{V}$. This matrix $\overline{A}$ is clearly nonnegative and stochastic.

The following proposition shows that the average link matrix $\overline{A}$ has a clear relation to the original link matrix A. In particular, it implies the possibility that the two matrices share the value vector x^* as their eigenvectors.

Proposition 3.1: For the average link matrix $\overline{A}$ given in (10), we have the following:

(i) $\overline{A} = \left[1 - (1-\alpha)^2\right]A + (1-\alpha)^2 I$.

(ii) There exists a vector $z_0 \in \mathbb{R}^n_+$ which is an eigenvector corresponding to the eigenvalue 1 for both A and $\overline{A}$.

C. A modified distributed update scheme

To guarantee that the distributed scheme yields the PageRank value, we now introduce a modified version.

Consider the distributed update scheme in the form of

$$x(k+1) = M_{\eta_1(k),\ldots,\eta_n(k)}x(k), \tag{11}$$

where $x(k) \in \mathbb{R}^n$, the initial condition $x(0)$ is a probability vector, and $\eta_i(k) \in \{0,1\}$, $i \in \mathcal{V}$, are specified by (6).

Here, the objective is to find the modified link matrices $M_{\eta_1,\ldots,\eta_n}$ such that their average and the link matrix M from (3) share an eigenvector corresponding to the eigenvalue 1. Since such an eigenvector is unique for M, it is necessarily equal to the value vector x^*.

For the definition of the distributed link matrices, we use an additional parameter $\hat{m}$ given by

$$\hat{m} := \frac{[1-(1-\alpha)^2]m}{1 - m(1-\alpha)^2}. \tag{12}$$

Now, for $\eta_1, \ldots, \eta_n \in \{0,1\}$, let

$$M_{\eta_1,\ldots,\eta_n} := (1-\hat{m})A_{\eta_1,\ldots,\eta_n} + \frac{\hat{m}}{n}S. \tag{13}$$

Then, let their average value be $\overline{M} := E[M_{\eta_1(k),\ldots,\eta_n(k)}]$. Here, the distributed link matrices are positive stochastic matrices, which implies that the average matrix $\overline{M}$ enjoys the same property.

The next lemma is the key to establish the desired relation between the distributed link matrices and their average.

Lemma 3.2: The scalar $\hat{m}$ in (12) and the link matrices $M_{\eta_1,\ldots,\eta_n}$ in (13) have the following properties:

(i) $\hat{m} \in (0,1)$ and $\hat{m} \leq m$.

(ii) $\overline{M} = \frac{\hat{m}}{m}M + \left(1 - \frac{\hat{m}}{m}\right)I$.

(iii) The value x^* in (4) is the unique eigenvector of the average matrix $\overline{M}$ corresponding to the eigenvalue 1.

We can show by (iii) in the lemma that, in an average sense, the distributed update scheme asymptotically obtains the correct values. More precisely, we have $E[x(k)] = \overline{M}^k x(0) \to x^*$ as $k \to \infty$.

We are now ready to state the main result of this section.

Theorem 3.3: Consider the distributed scheme with simultaneous updates in (11). For any update probability $\alpha \in (0,1]$, the PageRank value x^* is obtained through the time average y in (8) as $E\left[\left\|y(k)-x^*\right\|^2\right] \to 0$ as $k \to \infty$.

Several remarks are in order. This distributed update scheme is a generalization of the original scheme in Section II. This can be observed by taking the update probability α to be 1. In particular, when all pages initiate their updates, the distributed link matrix is $M_{1,\ldots,1}$; this coincides with the original M because of $\hat{m}=m$ and $A_{1,\ldots,1}=A$ in (9).

On the other hand, when $\alpha < 1$, the scheme is fully decentralized. It is parameterized by α, which determines the frequency in the updates, communication load among the pages, and the rate of convergence in the mean, as we have seen above. In practice, the recursion in (11) must be implemented at each page based on the equivalent expression

$$x(k+1) = (1-\hat{m})A_{\eta_1(k),\ldots,\eta_n(k)}x(k) + \frac{\hat{m}}{n}\mathbf{1}, \qquad (14)$$

where we used the fact that $Sx(k) = \mathbf{1}$ for all $k \in \mathbb{Z}_+$. Clearly, communication is required only over the links corresponding to the nonzero entries in the link matrices. Each page then performs weighted additions of its own value, the values that it has just received, and the extra term $\hat{m}/n$. Hence, the amount of computation is fairly small.

IV. Update termination in PageRank computation

In this section, we further develop the distributed algorithm for calculating the PageRank. We relax the objective and aim at obtaining approximate values of the PageRank. The key feature here is to allow the pages to terminate their updates at the point when the values have converged to a certain level. The benefit is that such values need to be transmitted only once to the linked pages; hence, the computation and communication load can be reduced.

A. Convergence properties in a centralized setting

The idea of update termination has been introduced by [9] in a centralized computational setting. Here, we outline the basics of this idea and provide an analysis on convergence.

Consider the original centralized update scheme in (5). Suppose that at time $k_0 \geq 0$, some pages have a good estimate of their values in terms of relative errors. This is determined by the new parameter $\delta \in (0,1)$. For fixed k_0, let $\mathcal{C}$ be the set of indices of such pages defined by

$$\mathcal{C} := \{i \in \mathcal{V} : \ |x_i(k_0) - x_i(k_0-1)| \leq \delta x_i(k_0)\}.$$

The cardinality of $\mathcal{C}$ is denoted by n_1. Also, let $\mathcal{N}$ be the indices not in $\mathcal{C}$ as $\mathcal{N} := \mathcal{V} \setminus \mathcal{C}$.

Now, for the state vector $x(k)$, we introduce a coordinate transformation such that

$$x(k) = \begin{bmatrix} x_{\mathcal{C}}(k) \\ x_{\mathcal{N}}(k) \end{bmatrix},$$

where $x_{\mathcal{C}}(k) \in \mathbb{R}^{n_1}$ contains the values of the pages in $\mathcal{C}$ and $x_{\mathcal{N}}(k) \in \mathbb{R}^{n-n_1}$ contains those of the pages in $\mathcal{N}$. Similarly, the link matrix M is partitioned as

$$M := \begin{bmatrix} M_{\mathcal{C}\mathcal{C}} & M_{\mathcal{C}\mathcal{N}} \\ M_{\mathcal{N}\mathcal{C}} & M_{\mathcal{N}\mathcal{N}} \end{bmatrix}.$$

Thus, in the new coordinate system, the scheme (5) becomes

$$\begin{bmatrix} x_{\mathcal{C}}(k+1) \\ x_{\mathcal{N}}(k+1) \end{bmatrix} = \begin{bmatrix} M_{\mathcal{C}\mathcal{C}} & M_{\mathcal{C}\mathcal{N}} \\ M_{\mathcal{N}\mathcal{C}} & M_{\mathcal{N}\mathcal{N}} \end{bmatrix} \begin{bmatrix} x_{\mathcal{C}}(k) \\ x_{\mathcal{N}}(k) \end{bmatrix}.$$

Now, since the state $x_{\mathcal{C}}(k)$ has converged at time k_0 in an approximate sense, this state will be fixed. The proposed algorithm updates the state $x(k)$ through the recursion

$$\begin{bmatrix} x_{\mathcal{C}}(k+1) \\ x_{\mathcal{N}}(k+1) \end{bmatrix} = \begin{bmatrix} I & 0 \\ M_{\mathcal{N}\mathcal{C}} & M_{\mathcal{N}\mathcal{N}} \end{bmatrix} \begin{bmatrix} x_{\mathcal{C}}(k) \\ x_{\mathcal{N}}(k) \end{bmatrix}, \ \ k \geq k_0. \quad (15)$$

We note that the matrix appearing on the right-hand side is nonnegative, but is no longer stochastic; the sums of the entries of the first n_1 columns are larger than 1 while those of the other columns are smaller than 1. Hence, though $x(k) \geq 0$ still holds, the state $x(k)$ may not be a probability vector.

It is clear that, after the coordinate change, the value vector x^* is an equilibrium of the system (15). Then, we introduce the notation

$$x^* = \begin{bmatrix} x^*_{\mathcal{C}} \\ x^*_{\mathcal{N}} \end{bmatrix}. \qquad (16)$$

It is also straightforward to show that, for each $x_{\mathcal{C}} \in \mathbb{R}^{n_1}$, the vector given by

$$\begin{bmatrix} x_{\mathcal{C}} \\ (I - M_{\mathcal{N}\mathcal{N}})^{-1} M_{\mathcal{N}\mathcal{C}} x_{\mathcal{C}} \end{bmatrix}$$

is an equilibrium of (15). We note that the inverse of $I - M_{\mathcal{N}\mathcal{N}}$ always exists because all of its nondiagonal entries are nonpositive, that is, $I - M_{\mathcal{N}\mathcal{N}}$ is a so-called M-matrix [7]. Let $\widetilde{x}(k_0)$ be the equilibrium defined by $x_{\mathcal{C}}(k_0)$ as

$$\widetilde{x}(k_0) = \begin{bmatrix} \widetilde{x}_{\mathcal{C}}(k_0) \\ \widetilde{x}_{\mathcal{N}}(k_0) \end{bmatrix} := \begin{bmatrix} x_{\mathcal{C}}(k_0) \\ (I - M_{\mathcal{N}\mathcal{N}})^{-1} M_{\mathcal{N}\mathcal{C}} x_{\mathcal{C}}(k_0) \end{bmatrix}. \qquad (17)$$

After the pages in $\mathcal{C}$ have terminated their updates, the dynamics of the scheme can be characterized as follows.

Lemma 4.1: Consider the update scheme (15) with the two equilibria x^* and $\widetilde{x}(k_0)$, respectively, given by (16) and (17). Then, the following statements hold.

(i) The state $x(k)$ converges to $\widetilde{x}(k_0)$ and, in particular, $x_{\mathcal{N}}(k) \to \widetilde{x}_{\mathcal{N}}(k_0)$ as $k \to \infty$.

(ii) If $|\widetilde{x}_{\mathcal{C}}(k_0) - x^*_{\mathcal{C}}| \leq \delta x^*_{\mathcal{C}}$, then $|\widetilde{x}_{\mathcal{N}}(k_0) - x^*_{\mathcal{N}}| \leq \delta x^*_{\mathcal{N}}$.

The lemma shows that if the values in $x_{\mathcal{C}}(k_0) = \widetilde{x}_{\mathcal{C}}(k_0)$ are close to the true values $x^*_{\mathcal{C}}$, then via the proposed update scheme (15), we can still obtain an approximate value $\widetilde{x}_{\mathcal{N}}(k_0)$ for the rest of the states; the level of approximation is the same, represented by the parameter δ.

B. A distributed approach

We extend the idea of update termination to the distributed update scheme of Section III. First, to provide a convergence result as in the previous subsection, we consider a simple case. Then, we provide the details of the proposed algorithm.

Consider the distributed scheme in (14) for computing the values $x(k)$ together with their time average $y(k)$. Suppose

that at a given time k_0, some of the time averages $y_i(k_0)$ have, in an approximate sense, converged. This is measured by finding those that have varied only within sufficiently small ranges for a certain number of time steps. Here, we introduce two parameters: Let $\delta \in (0,1)$ be the relative error level, and let N_s be the number of steps. Using the history of its own time average y_i, each page i then determines at time k_0 whether the following condition holds:

$$|y_i(k_0) - y_i(k_0 - l)| \leq \delta y_i(k_0), \quad l = 1, 2, \ldots, N_s. \quad (18)$$

If so, then (i) the page i will terminate its update and fix its estimate at $y_i(k_0)$, and then (ii) this value $y_i(k_0)$ is transmitted to the pages connected to i by direct links where these values will be used for further updates.

The question of interest is whether the pages that continue with updates will reach a good estimate of their true values. In what follows, we show that the answer is positive and the approximation level achievable in the estimate will be as good as that for the pages that have terminated their updates.

Let $\mathcal{C}(k_0)$ be the set of page indices that have reached good estimates at time k_0 as

$$\mathcal{C}(k_0) := \big\{ i \in \mathcal{V} : \ |y_i(k_0) - y_i(k_0 - l)| \leq \delta y_i(k_0), \\ l = 1, 2, \ldots, N_s \big\}.$$

The cardinality of this set is denoted by n_1. Also, let $\mathcal{N}$ be the set of indices not in $\mathcal{C}(k_0)$ as $\mathcal{N}(k_0) := \mathcal{V} \setminus \mathcal{C}(k_0)$.

Following the notation in the previous subsection, for the state $x(k)$, we introduce a coordinate change based on these sets and then partition it as $x(k) = \big[x_{\mathcal{C}}(k)^T \ x_{\mathcal{N}}(k)^T\big]^T$. Next, the distributed link matrices A_η in (9) and the average link matrix $\overline{A}$ in (10) are partitioned accordingly as

$$A_\eta = \begin{bmatrix} A_{\eta,\mathcal{C}\mathcal{C}} & A_{\eta,\mathcal{C}\mathcal{N}} \\ A_{\eta,\mathcal{N}\mathcal{C}} & A_{\eta,\mathcal{N}\mathcal{N}} \end{bmatrix}, \quad \overline{A} = \begin{bmatrix} \overline{A}_{\mathcal{C}\mathcal{C}} & \overline{A}_{\mathcal{C}\mathcal{N}} \\ \overline{A}_{\mathcal{N}\mathcal{C}} & \overline{A}_{\mathcal{N}\mathcal{N}} \end{bmatrix}, \quad (19)$$

where we write A_η for $A_{\eta_1,\ldots,\eta_n}$, $\eta_i \in \{0,1\}$, $i \in \mathcal{V}$.

Now, since the time average $y_{\mathcal{C}}(k)$ has converged sufficiently by time k_0, the proposed approach employs the value $y_{\mathcal{C}}(k_0)$ as $x_{\mathcal{C}}(k)$ for all $k \geq k_0$. Hence, the value at time k_0 is reset as $x(k_0) = \big[y_{\mathcal{C}}(k_0)^T \ x_{\mathcal{N}}(k_0)^T\big]^T$. The updates are carried out through the distributed algorithm given by

$$x(k+1) = \widetilde{A}_{\eta(k)} x(k) + \frac{\hat{m}}{n} \tilde{s}, \quad k \geq k_0, \quad (20)$$

where

$$\widetilde{A}_{\eta(k)} = \begin{bmatrix} I & 0 \\ \widetilde{A}_{\eta(k),\mathcal{N}\mathcal{C}} & \widetilde{A}_{\eta(k),\mathcal{N}\mathcal{N}} \end{bmatrix}, \quad \tilde{s} := \begin{bmatrix} 0 \\ s \end{bmatrix}, \quad (21)$$

with $\widetilde{A}_{\eta(k),\mathcal{N}\mathcal{C}} := (1 - \hat{m}) A_{\eta(k),\mathcal{N}\mathcal{C}}$, $\widetilde{A}_{\eta(k),\mathcal{N}\mathcal{N}} := (1 - \hat{m})$ $A_{\eta(k),\mathcal{N}\mathcal{N}}$, and $\mathbf{1} \in \mathbb{R}^{n-n_1}$. We remark that the link matrices $\widetilde{A}_\eta$ are not stochastic.

The time average $y(k)$ is also modified by fixing the entries for $i \in \mathcal{C}(k_0)$ as

$$y(k) = \begin{bmatrix} y_{\mathcal{C}}(k_0) \\ y_{\mathcal{N}}(k) \end{bmatrix}, \quad k \geq k_0,$$

where $y_{\mathcal{N}}(k)$ is determined through the original formula (8).

For the approximate update scheme (20), its average state $\overline{x}(k) := E[x(k)]$ follows the recursion

$$\overline{x}(k+1) = \widehat{A}\, \overline{x}(k) + \frac{\hat{m}}{n} \tilde{s}, \quad k \geq k_0, \quad (22)$$

where the average link matrix $\widehat{A} := E[\widetilde{A}_{\eta(k)}]$ is given by

$$\widehat{A} = \begin{bmatrix} I & 0 \\ \widehat{A}_{\mathcal{N}\mathcal{C}} & \widehat{A}_{\mathcal{N}\mathcal{N}} \end{bmatrix} \quad (23)$$

with $\widehat{A}_{\mathcal{N}\mathcal{C}} := (1 - \hat{m}) \overline{A}_{\mathcal{N}\mathcal{C}}$ and $\widehat{A}_{\mathcal{N}\mathcal{N}} := (1 - \hat{m}) \overline{A}_{\mathcal{N}\mathcal{N}}$. This matrix is not stochastic and in fact has the eigenvalue 1 with multiplicity greater than or equal to 1.

Regarding the average link matrix, the following result will be useful in the subsequent development.

Lemma 4.2: The submatrix $\widehat{A}_{\mathcal{N}\mathcal{N}}$ of the average link matrix $\widehat{A}$ as given in (23) satisfies the following:

(i) $\rho(\widehat{A}_{\mathcal{N}\mathcal{N}}) \in [0, 1 - \hat{m}]$ and $\widehat{A}_{\mathcal{N}\mathcal{N}}$ is a stable matrix.
(ii) $(I - \widehat{A}_{\mathcal{N}\mathcal{N}})^{-1} \geq 0$.

As in the centralized case discussed in the previous subsection, we focus on two equilibria of this average system: The true x^* and the approximate values $\widetilde{x}(k_0)$ given by

$$\widetilde{x}(k_0) = \begin{bmatrix} \widetilde{x}_{\mathcal{C}}(k_0) \\ \widetilde{x}_{\mathcal{N}}(k_0) \end{bmatrix} := \begin{bmatrix} y_{\mathcal{C}}(k_0) \\ \big(I - \widehat{A}_{\mathcal{N}\mathcal{N}}\big)^{-1} \big(\widehat{A}_{\mathcal{N}\mathcal{C}} y_{\mathcal{C}}(k_0) + \frac{\hat{m}}{n} s \big) \end{bmatrix}. \quad (24)$$

This vector always exists due to the lemma above.

The average system in (22) has the following properties.

Lemma 4.3: Consider the distributed update scheme (20) with the two equilibria x^* and $\widetilde{x}(k_0)$, respectively, given by (16) and (24). Then, the following statements hold.

(i) The average state $\overline{x}(k)$ converges to $\widetilde{x}(k_0)$ and, in particular, $\overline{x}_{\mathcal{N}}(k) \to \widetilde{x}_{\mathcal{N}}(k_0)$ as $k \to \infty$.
(ii) If $|\widetilde{x}_{\mathcal{C}}(k_0) - x^*_{\mathcal{C}}| \leq \delta x^*_{\mathcal{C}}$, then $|\widetilde{x}_{\mathcal{N}}(k_0) - x^*_{\mathcal{N}}| \leq \delta x^*_{\mathcal{N}}$.

The following is the main convergence result.

Theorem 4.4: Consider the distributed scheme in (20), where n_1 pages have terminated their updates at time k_0. The time average $y_{\mathcal{N}}(k)$ converges to the equilibrium $\widetilde{x}_{\mathcal{N}}(k_0)$ in the mean square sense as

$$E\left[\big\| y_{\mathcal{N}}(k) - \widetilde{x}_{\mathcal{N}}(k_0) \big\|^2 \right] \to 0, \quad k \to \infty.$$

C. Distributed algorithm with update termination

We now present the distributed algorithm with update termination based on the results outlined in this section.

Algorithm 4.5: For $i \in \mathcal{V}$, page i executes the following.

0) Initialize the parameters n, α, $x_i(0)$, N_s, and δ. Set $k = 0$, $\mathcal{C}(0) := \emptyset$, and $n_1 = 0$.
1) At time k, generate $\eta_i(k) \in \{0,1\}$ under the probability α. If $\eta_i(k) = 1$, then send the value $x_i(k)$ to pages $j \notin \mathcal{C}(k)$ that it is linked to. Also request pages $j \notin \mathcal{C}(k)$ that have links to it for their values.

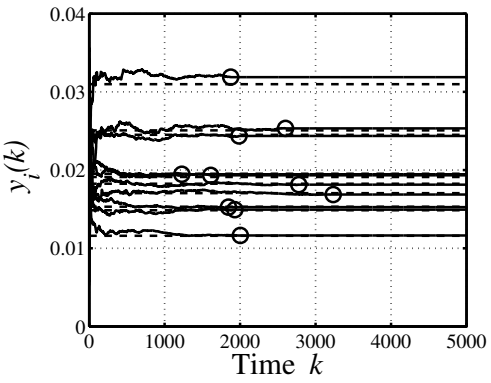

Fig. 1. Sample paths of $y_i(k)$ (solid lines) with the time updates stopped (marked by ◯ for each page) and the PageRank values x_i^* (dashed lines) for $i = 1, 2, \ldots, 10$.

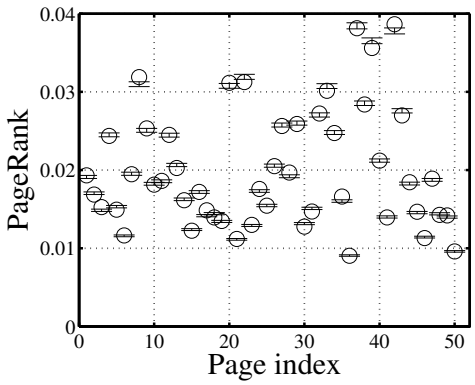

Fig. 2. The range of approximate values $((1-\delta)x_i^*, (1+\delta)x_i^*)$ (marked by two lines) and $y_i(k)$, $i = 1, \ldots, 5$, at $k = 5000$ (marked as ◯)

2) Update the value $x_i(k)$ and its time average by

$$x_i(k+1) = \sum_{j=1}^{n} \big(\widetilde{A}_{\eta(k)}\big)_{ij} x_j(k) + \frac{\hat{m}}{n},$$

$$y_i(k) = \frac{1}{k+1} \sum_{\ell=0}^{k} x_i(\ell),$$

where $\widetilde{A}_{\eta(k)}$ is constructed by (21) using $\mathcal{C}(k)$.

3) Check whether $y_i(k)$ has sufficiently converged based on (18). If so, then (i) add i to the set $\mathcal{C}(k)$, (ii) send $y_i(k)$ to the pages having direct links to page i, and (iii) fix $x_i(\ell) = y_i(\ell) = y_i(k)$ for $\ell \geq k$.
4) If $\mathcal{C}(k) = \mathcal{V}$, then terminate. Otherwise, set $\mathcal{C}(k+1) = \mathcal{C}(k)$ and $k = k+1$, and then go to Step 1. ▽

We remark that, in this scheme, the choice of the parameters N_s and δ affects the accuracy in the values when the pages terminate their updates as well as the time when the pages decide to do so. Taking N_s larger and/or δ smaller will improve the value estimates, but will require longer time before the updates terminate; this in turn will keep the communication and computation load higher.

V. Numerical example

We present an example using the web with 50 pages from [8]. The links among the pages were randomly generated and, for each page, there were between 2 and 13 links. The parameter m of the link matrices M in (3) and M_η in (13) was taken as $m = 0.15$. Simulations were carried out using Algorithm 4.5. The parameters were taken as $\alpha = 0.1$, $N_s = 800$, and $\delta = 0.01$. We chose them so that the characteristics of this scheme are visible in the plots.

We computed a sample path of the state $x(k)$. The initial state $x(0)$ was taken as a probability vector that was randomly chosen. The time averages $y_i(k)$ of the states for pages $i = 1, \ldots, 10$ are shown in Fig. 1. We observe that they become close to the true values indicated by the dashed lines. The times when the corresponding pages terminated their updates are marked by ◯. All 50 pages stopped by time $k = 4349$ and on average by about 2160.

In Fig. 2, the final values of $y_i(k)$ at $k = 5000$ are plotted together with the acceptable range of error, that is, $[(1-\delta)x_i^*, (1+\delta)x_i^*]$ by two lines connected in the middle. Overall, the errors are fairly small. As we have mentioned, the time average vector $y(k)$ is no longer normalized. However, we had $\|y(5000)\|_1 = 0.999$, which is close to 1.

VI. Conclusion

We have extended the distributed randomized approach for the PageRank computation in [8] to enhance flexibility. The first scheme incorporates simultaneous updates by multiple pages while the second one is equipped with a criterion to terminate updates to obtain approximate values. Future research will study the effects of communication delays.

References

[1] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte Carlo methods in PageRank computation: When one iteration is sufficient. *SIAM J. Numer. Anal.*, 45:890–904, 2007.
[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Comp. Networks & ISDN Systems*, 30:107–117, 1998.
[3] K. Bryan and T. Leise. The $25,000,000,000 eigenvector: The linear algebra behind Google. *SIAM Rev.*, 48:569–581, 2006.
[4] D. V. de Jager and J. T. Bradley. Asynchronous iterative solution for state-based performance metrics. In *Proc. ACM SIGMETRICS*, pages 373–374, 2007.
[5] Y. Hatano and M. Mesbahi. Agreement over random networks. *IEEE Trans. Autom. Control*, 50:1867–72, 2005.
[6] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge Univ. Press, 1985.
[7] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge Univ. Press, 1991.
[8] H. Ishii and R. Tempo. A distributed randomized approach for the PageRank computation: Part 1. In *Proc. 47th IEEE Conf. on Decision and Control*, 2008.
[9] S. Kamvar, T. Haveliwala, and G. Golub. Adaptive methods for the computation of PageRank. *Linear Algebra Appl.*, 386:51–65, 2004.
[10] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton Univ. Press, 2006.
[11] A. Tahbaz-Salehi and A. Jadbabaie. A necessary and sufficient condition for consensus over random networks. *IEEE Trans. Autom. Control*, 53:791–795, 2008.
[12] R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer, London, 2005.
[13] R. Tempo and H. Ishii. Monte Carlo and Las Vegas randomized algorithms for systems and control: An introduction. *European J. Control*, 13:189–203, 2007.
[14] C. W. Wu. Synchronization and convergence of linear dynamics in random directed networks. *IEEE Trans. Autom. Control*, 51:1207–1210, 2006.
[15] Y. Zhu, S. Ye, and X. Li. Distributed PageRank computation based on iterative aggregation-disaggregation methods. In *Proc. 14th ACM Conf. Info. and Knowledge Management*, pages 578–585, 2005.