Distributed Randomized Algorithms for PageRank Computation: Recent Advances

Hideaki Ishii and Atsushi Suzuki

Dedicated to Roberto Tempo

Abstract PageRank is a well-known centrality measure for the web used in search engines, representing the importance of each web page. Its computation is very large scale as the rankings for all pages in the entire web are to be calculated at once, and this has prompted various studies on the algorithmic aspects of this problem. In this chapter, we first present a short overview on the recent studies on distributed algorithms that have been developed in the systems control area. These algorithms share the features that (i) each page computes its own PageRank value by interacting with pages connected over hyperlinks and (ii) gossip-type randomization is employed in the update schemes. Then, we introduce a new class of distributed algorithms for PageRank, which is based on a simple but novel interpretation. It is demonstrated via analysis and numerical simulations that these algorithms have significant advantages in their convergence performances in comparison with other existing techniques. The chapter ends with a brief summary of the works on randomization-based distributed algorithms, heavily influenced by the collaboration with Roberto Tempo, to whom this writing is dedicated.

This work was supported in part by the JST CREST Grant No. JPMJCR15K3 and by JSPS under Grant-in-Aid for Scientific Research Grant No. 15H04020.

H. Ishii (✉) · A. Suzuki
Department of Computer Science, Tokyo Institute of Technology, Yokohama 226-8502, Japan
e-mail: ishii@c.titech.ac.jp

A. Suzuki
e-mail: suzuki@sc.dis.titech.ac.jp

T. Başar (ed.), *Uncertainty in Complex Networked Systems*,
Systems & Control: Foundations & Applications,
https://doi.org/10.1007/978-3-030-04630-9_12

1 Introduction

For search engines at Google, one of the many measures used for ranking the web pages in search results is the so-called PageRank. For each web page, the PageRank value provides a measure of its importance or popularity, which is based on the network structure of the web in terms of the hyperlinks. A page is considered more important and popular if it receives more hyperlinks from other pages and especially those that are important themselves. When such a structure in the hyperlinks is present around a page, it suggests how easily users surfing the web might arrive there, even by chance. The notion of PageRank was proposed by the co-founders of Google, Brin and Page, in [7]. It has received a great deal of interest especially in the context of complex networks as it is an effective measure of centrality. General references on this topic include the monograph [33] and the overview papers [23, 27].

The problem of computing PageRank itself has been a subject of extensive studies over the years. Despite the simple nature of the problem, because of the problem size involving billions of pages in the web, its efficient computation has remained as a difficult task. For centralized computation, the simple power method has been the realistic option for this reason. Alternative methods have been studied based on Monte Carlo simulations of the underlying Markov chain (e.g., [1]) and distributed algorithms (e.g., [48]).

Recently, in the systems control community, PageRank has gained much attention from the viewpoint of distributed computation. In particular, in [26], it was pointed out that the PageRank problem shared several similarities with the multi-agent consensus problem (e.g., [8, 40]) and randomized distributed algorithms were developed. The approach there is to view the web as a network of pages capable of communicating with neighbors connected via hyperlinks. Then, in a distributed manner, each web page can act as an agent which computes its own PageRank value iteratively by exchanging data with other pages. To cope with the network size, the pages randomly determine when to initiate updates, which is sometimes called gossiping [6]. The method is guaranteed to converge in the mean-square sense. However, it involves the time averaging of the state values, resulting in the convergence rate of order $1/k$ with respect to the updating time k.

The focus of this chapter is the research activities on the topic of PageRank that have taken place since. The chapter consists of roughly three parts. In the first, we provide a brief overview on the subject of distributed computation of PageRank, starting with the work of [26]. More recently, studies focusing on convergence speeds have appeared. In particular, it has been found that convergence with exponential rate is possible. Notably, in [57], the PageRank problem is formulated as a least-squares problem and then a distributed gradient-descent algorithm is applied. This work also points out the difficulty in assuming the global parameter of the total number of web pages to be known by all pages, leading to alternative algorithms that enable the PageRank calculation without the knowledge. The work [14] employs another technique of matching pursuit algorithms for solving linear equations and provides a randomized version. On the other hand, in [32], a modified gradient-descent algorithm is constructed so that the states of all pages remain to have the total equal to one throughout its execution. We also refer to [43], which studied stochastic gradient algorithms for PageRank. In this first part, we will introduce and discuss these algorithms and their different features such as their convergence speeds and required loads for communication and computation.

In the second part, we propose a novel approach towards the PageRank computation from a slightly different perspective [50]. By making use of the property that PageRank involves a stochastic matrix representing the network structure of the pages, we reformulate the problem in a certain way, expressed as an infinite matrix series. This formulation leads us to a completely different set of algorithms tailored to the problem. Specifically, we propose algorithms for both synchronous and asynchronous settings in the communication among the linked pages. Their convergence properties are fully analyzed in the development. For the asynchronous case, we employ randomization-based gossiping, but the probability to be selected for updates need not be uniform. We show that they have desirable characteristics including exponential convergence and relatively low requirements for the communication among agents. Through numerical examples, we carry out a detailed comparison of the algorithms discussed in the chapter.

The novel aspects of the proposed algorithms can be summarized as follows. First, the reformulation idea is simple and its advantage may not be immediately clear. This is partly because additional states are introduced for the pages, which increase the computational burden. In fact, in the synchronous case, the convergence is not necessarily faster than the power method. Second, in the proposed randomized algorithms, the states are guaranteed to reach the true PageRank values from below in a monotonic fashion. Hence, despite the randomization, the responses of the states are smooth, which may explain the efficiency of the approach. Third, in the randomization, the pages to initiate updates can be chosen under arbitrary distributions. It should be noted that no change is necessary in the algorithm due to the chosen distribution. This leaves a certain degree of freedom in enhancing the convergence speed as discussed in the numerical example section. Furthermore, the pages communicate over only their outgoing hyperlinks and do not require the knowledge of the incoming ones as in some methods in the literature.

As the last part, which is the shortest, we discuss the different roles that randomization may play in networked systems problems and, in particular, multi-agent consensus problems. In addition to gossiping in communication, probabilistic techniques can be useful in enhancing distributed decision-making as well as cybersecurity levels for systems in hazardous environments where malicious attackers may take advantage to disrupt the execution of algorithms and control.

Finally, we should note that, in the area of systems control, studies on PageRank have grown in a spectrum of interesting directions; see [27] for more discussions. For distributed algorithms, the approach of [26] has been extended, for example, to incorporate aggregation of pages to realize more efficient computation in [29]. Stronger convergence properties with probability one are established with the help of stochastic approximation results in [58]. Moreover, in [12, 28, 36], different probability distributions are employed for the randomized updates in the pages, making them capable to function, e.g., even if the channels for the communication among pages are unreliable. Other works conducted studied on the problem of finding the ranges of PageRank values when a subset of the hyperlinks is uncertain in the sense whether they are actually present [25], optimization of PageRank for pages of interest by changing the link structure [13, 19], and a game theoretic analysis for enhancing PageRank through aggregation of pages [38].

This chapter is organized as follows: We first give an overview on the PageRank problem in Sect. 2. In Sect. 3, we introduce the recent works on distributed computation approaches for the PageRank. In Sect. 4, an alternative formulation for the problem is presented, which is then used for deriving two novel distributed algorithms based on randomized gossiping. Illustrative numerical examples are provided in Sect. 5. A more general discussion on the topic of randomization-based techniques in the context of multi-agent systems is provided in Sect. 6. The chapter is finally concluded in Sect. 7.

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 all i, j; 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 $\sum_{i=1}^n v_i = 1$. A matrix $X \in \mathbb{R}^{n\times n}$ is said to be (column) stochastic if it is nonnegative and each column sum equals 1, i.e., $\sum_{i=1}^n x_{ij} = 1$ for each j. Let $\mathbf{1}_n \in \mathbb{R}^n$ be the vector whose entries are all 1 as $\mathbf{1}_n := [1 \cdots 1]^T$. For a vector x, we use $\|x\|$ to denote its the Euclidean norm. For a discrete set $\mathscr{D}$, its cardinality is given by $|\mathscr{D}|$.

2 The PageRank Problem

In this section, we introduce the basics of PageRank and its interpretations commonly employed for its computation [7, 27, 33].

The underlying idea for PageRank is to regard the entire web as a directed graph consisting of web pages with hyperlinks. By solely using the network structure there, PageRank provides a powerful measure of centrality, indicating how important or popular each web page is.

Let n be the total number of pages in the web; we assume $n \geq 2$ to avoid the trivial case. The web graph is given by $\mathscr{G} := (\mathscr{V}, \mathscr{E})$, where $\mathscr{V} := \{1, 2, \ldots, n\}$ is the set of vertices representing the web pages, and $\mathscr{E}$ is the set of hyperlinks connecting the pages. Here, $(i, j) \in \mathscr{E}$ holds if and only if page i has a hyperlink to page j. In such a case, for page i, page j becomes its out-neighbor, whereas page i is the in-neighbor of page j.

The hyperlinks are not always mutual, so this graph is generally a directed graph. When a node does not have any outgoing link, it is referred to as a dangling node. Here, to simplify the discussion, we assume that all pages have at least one outgoing hyperlink. This is commonly done by slightly modifying the structure of the web, specifically by adding hyperlinks from such dangling nodes, which correspond to the use of back buttons; see, e.g., [33] for more details.

Next, we define the hyperlink matrix $A = \left(a_{ij}\right) \in \mathbb{R}^{n\times n}$ of this graph by

$$a_{ij} := \begin{cases} \frac{1}{n_j} & \text{if } \ i \in \mathscr{L}_j, \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

where $\mathscr{L}_i := \{j : (i, j) \in \mathscr{E}\}$ is the set of outgoing neighbors of page i and n_i is its cardinality, i.e., $n_i := |\mathscr{L}_i|$. By the assumption that all pages have one or more hyperlinks, this matrix A is (column) stochastic, that is, it is a nonnegative matrix where the sum of entries in each column is equal to 1.

For the web consisting of n pages, the PageRank vector $x^* \in [0, 1]^n$ is defined as

$$x^* = (1-m)Ax^* + \frac{m}{n}\mathbf{1}_n, \ \ \mathbf{1}_n^T x^* = 1, \tag{2}$$

where the parameter is chosen as $m \in (0, 1)$. Note that x^* is a nonnegative vector, and the second equation above indicates that it is a probability vector. For m, it is common to use the value 0.15 as proposed by [7]; we follow this convention in this chapter.

The definition in (2) can be rewritten as

$$x^* = Mx^*, \ \ \mathbf{1}_n^T x^* = 1, \tag{3}$$

where the modified link matrix M is given by

$$M := (1-m)A + \frac{m}{n}\mathbf{1}_n\mathbf{1}_n^T.$$

Since M is a convex combination of two stochastic matrices A and $(1/n)\mathbf{1}_n\mathbf{1}_n^T$, it is stochastic as well. It is now clear that x^* is the eigenvector of the link matrix M corresponding to the eigenvalue 1. Such an eigenvector x^* exists and is unique; this follows from Perron's theorem [24] because the stochastic matrix M has the property of being positive.

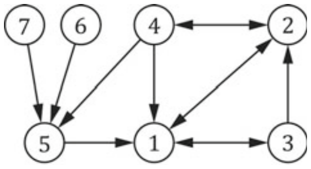
For its computation, the PageRank vector x^* can be obtained by solving the linear equation (2) or (3). The practical issue that requires serious attention is the size of the problem. Recall that the dimension of the PageRank vector is the same as the number of pages in the web. Hence, the computation must rely on algorithms that have simple structures.

A common approach, which is centralized, is to employ the power method. It is expressed by the iteration of the form

$$x(k+1) = (1-m)Ax(k) + \frac{m}{n}\mathbf{1}_n, \tag{4}$$

where $x(k) \in \mathbb{R}^n$ is the state whose initial value $x(0)$ can be taken as any probability vector. By Perron's theorem [24], it follows that $x(k) \to x^*$ as $k \to \infty$.

Fig. 1 An example graph with seven nodes



Another interesting interpretation of PageRank is that of the *random surfer* model. It follows from the expression in (3) that the PageRank vector x^* can be regarded as the stationary distribution of a Markov chain whose transition matrix is represented by the stochastic matrix M. We may imagine a person who surfs the web in a random manner: When he visits one page, with probability $1 - m$, he chooses one of the links with equal probability; otherwise, with probability m, he decides to jump to any of the pages in the web with equal probability, that is, $1/n$. Under this model, the PageRank of page i can be regarded as the probability that such a surfer visits there in the steady state. Clearly, the link structure of the web creates pages which are more likely to be visited by such an imaginary surfer.

We now present a simple example to illustrate the problem of PageRank.

Example 1 Consider the web consisting of seven pages depicted in Fig. 1. The hyperlink matrix A of this web is given by

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{3} & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

We can calculate the PageRank vector of this graph as

$$x^* = \left[0.316 \; 0.259 \; 0.156 \; 0.132 \; 0.0951 \; 0.0214 \; 0.0214\right]^T.$$

It is noted that the indices of the pages are set according to the order of their PageRanks. Pages 1 and 2 have, respectively, four and three incoming links, making their rankings high. Pages 6 and 7 have no incoming hyperlink and, as a result, take the lowest possible PageRank, which is equal to $m/n = 0.15/7 = 0.0214$. We should emphasize that the number of links is not the only factor that determines PageRank. Both pages 3 and 4 have only one incoming link, but take better rankings than page 5, which has three links. This is because the ranks also depend on the values of the pages from which the links originate. In this respect, pages 3 and 4 are more advantageous than page 5, whose links include those from pages 6 and 7, having minor impact on its importance.

3 Distributed Algorithms for PageRank

In this section, we discuss the recent studies on randomized distributed algorithms for the PageRank computation and their differences. Namely, we focus on the methods developed in Ishii and Tempo [26], You et al. [57], Dai and Freris [14], and Lagoa et al. [32].

The computation of PageRank may be costly if it is performed centrally because of the size of the problem determined by the number of pages in the entire web. Hence, distributed computation is one natural approach to resolve this issue. In the systems control community, this viewpoint is particularly motivated by the recent research on coordinated control of multi-agent systems (e.g., [8, 40]). In the setting of the web, the pages may act as agents interacting over the hyperlinks to compute their own PageRank values through an iterative algorithm. In practice, resources for computation and communication are available at the numerous web servers where the data regarding the pages connected via hyperlinks is available.

In what follows, we present several distributed algorithms for PageRank computation. We use the common notation for the value of page i at time k, which is expressed as $x_i(k)$. In view of the size of the system, one issue is how to coordinate the pages in terms of the timings for them to initiate updates and communication of their values. Here, we bring in randomization in the pages' decisions and employ the so-called *gossip*-type communication: At each time step, one of the pages is randomly chosen in an independently identically distributed (i.i.d.) manner. Denoting the index of the page chosen at time k by $\theta(k) \in \mathscr{V}$, we have

$$\mathrm{Prob}\big\{\theta(k) = i\big\} = \frac{1}{n} \text{ for } i \in \mathscr{V} \text{ and } k \in \mathbb{Z}_+. \tag{5}$$

All of the algorithms discussed here are equipped with this mechanism.

Among the distributed algorithms, there are differences in how the chosen page $\theta(k)$ interacts with its neighboring linked pages. Some approaches require that page $\theta(k)$ send its current value $x_{\theta(k)}(k)$ to its out-neighbors whereas other algorithms mandate communications with its in-neighbors. Another aspect that will become an important difference is the level of synchronization necessary among the pages in their clocks. In general, it is difficult to expect that a common clock exists, shared by all pages with perfect synchronization.

3.1 *Towards Distributed Computations*

First, we present the approach of [26], which introduced distributed algorithms from the viewpoint of coordinated control of multi-agent systems. The update law for this case is motivated by the centralized iterative one in (4). In comparison with other algorithms, a key feature is the reliance on the use of stochastic matrices.

The idea is to employ distributed link matrices A_i containing the ith column of the link matrix A given in (1), and the remaining columns are set so that A_i becomes a stochastic matrix. Here, we present the version from [27], which takes a slightly simpler form. More concretely, the distributed link matrices A_i, $i = 1, \ldots, n$, are defined by

(i) The ith column of A_i is equal to the ith column of A.
(ii) The diagonal entries of the columns other than the ith one are equal to one.
(iii) The remaining entries are chosen to be zero.

It is clear that these matrices A_i are column stochastic by construction.

As a consequence, according to the probability distribution of the process $\theta(k)$, the average matrix $\overline{A} := \mathbb{E}[A_{\theta(k)}]$ takes a special form as

$$\overline{A} = \frac{1}{n}\sum_{i=1}^{n} A_i = \frac{2}{n}A + \left(1 - \frac{2}{n}\right) I. \tag{6}$$

Notice that this matrix $\overline{A}$ is a convex combination of two column stochastic matrices.

The distributed update law using the link matrices can be represented as follows:

$$x(k+1) = (1-\hat{m})A_{\theta(k)}x(k) + \frac{\hat{m}}{n}\mathbf{1}_n, \tag{7}$$

where the initial vector $x(0)$ is a probability vector and $\hat{m} \in (0, 1)$ is a parameter to be determined, corresponding to $m = 0.15$ in the centralized algorithm (4). This update law is accompanied by the time average process $y(k)$ of the states $x(0), \ldots, x(k)$ given by

$$y(k) := \frac{1}{k+1}\sum_{\ell=0}^{k} x(\ell). \tag{8}$$

Observe that each page i can locally compute the average $y_i(k)$ of its own past states $x_i(\ell)$, $\ell = 0, \ldots, k$. It is also noted that the state $x(k)$ and hence the average $y(k)$ are both probability vectors at all k.

We now discuss the convergence properties of the update scheme (7) and (8). The parameter $\hat{m}$ is to be set as

$$\hat{m} := \frac{2m}{n - m(n-2)}.$$

This choice allows us to represent the PageRank vector x^* based on the average matrix $\overline{A}$ in (6) as

$$x^* = \big(1-\hat{m}\big)\overline{A}x^* + \frac{\hat{m}}{n}\mathbf{1}.$$

Then, we can establish that the expected value $\mathbb{E}[x(k)]$ of the states converges to the PageRank vector x^*, that is, $\mathbb{E}[x(k)] \to x^*$ as $k \to \infty$. While it is not possible to show that the state vector $x(k)$ itself converges to the PageRank vector x^*, it follows that its time average $y(k)$ does so in the mean-square sense. More specifically, for any initial state $x(0)$ which is a probability vector, it holds that

$$\mathbb{E}\big[\|y(k) - x^*\|^2\big] \to 0 \ \text{as} k \to \infty.$$

This kind of convergence is referred to as ergodicity of random processes and the time average plays an important role. The original state $x(k)$ in general demonstrates persisting oscillations without convergence. Furthermore, the scheme converges with probability one, as was shown in [58] using methods from stochastic approximation.

We discuss a few issues that may be of concern about the update scheme (7) and (8). They become relevant because of the involvement of the time averaging in $y(k)$. One is that the speed of convergence is somewhat limited. It can be shown to be linear and, more specifically, of the order $1/k$. Another is the necessity for the pages to be synchronized; this is needed for correctly computing the time average $y_i(k)$ by all pages as pointed out in [57]. Finally, in this algorithm, each page i must store and update two variables, namely, $x_i(k)$ and $y_i(k)$. Interesting extensions of this class of algorithms have been made in [22, 46], where applications to sensor localization in wireless networks, social dynamics, and state estimation in power systems can be found.

3.2 Enhancement in Convergence Speed

Next, we proceed to discuss the alternative approach of the work [57] for distributed computation of the PageRank vector. Their approach is based on the viewpoint of distributed optimization, which in turn allows us to adopt existing algorithms from the area and to demonstrate its exponential convergence.

The starting point is to rewrite the PageRank vector x^* in its definition as the solution to the linear equation given by

$$[I - (1-m)A]\,x^* = \frac{m}{n}\mathbf{1}_n. \tag{9}$$

This implies that the vector can be obtained through an unconstrained optimization given by

$$x^* = \arg\min_x \left\| [I - (1-m)A]\,x - \frac{m}{n}\mathbf{1}_n \right\|^2. \tag{10}$$

Under this formulation, the PageRank computation problem can be further reduced to a form having a distributed nature more explicitly. Let

$$H := I - (1-m)A \ \text{ and } \ g := \frac{m}{n}\mathbf{1}_n. \tag{11}$$

Denote by $\tilde{h}_i^T \in \mathbb{R}^n$ the ith row of the matrix H. The optimization problem (10) can be expressed as

$$x^* = \arg\min_x \sum_{i=1}^{n} \left(\tilde{h}_i^T x - g_i \right)^2 . \tag{12}$$

For solving this optimization problem, the work [57] presents a distributed gradient-descent algorithm, which can be seen as an extension of the randomized Kaczmarz algorithm of [59]. An interesting feature is that the index $\theta(k)$ of the chosen page follows a Markov chain whose states correspond to the pages in the web; this is introduced to deal with the situation where the total number n of pages in the web is unknown.

We now present the simple case where $\theta(k)$ is an i.i.d. random process according to (5). This version requires the knowledge of the size n of the web. The update scheme can be given as follows:

$$\begin{aligned} x(k+1) &= x(k) - \frac{1}{2n} \cdot \frac{d}{dx} \left(g_{\theta(k)} - \tilde{h}_{\theta(k)}^T x \right)^2 \Big|_{x=x(k)} \\ &= x(k) + \frac{1}{n} \tilde{h}_{\theta(k)} \left(g_{\theta(k)} - \tilde{h}_{\theta(k)}^T x(k) \right), \end{aligned} \tag{13}$$

where the initial condition is set as $x(0) = 0$.

It is shown in [57] that the randomized algorithm in (13) has a guaranteed convergence rate and, in fact, it exponentially converges almost surely to the true PageRank vector x^*. This is an important characteristic, which is not attainable in the approach of [26] based on stochastic matrices and time averaging of the state. Another difference is that this algorithm involves only one variable per page for the PageRank computation.

On the other hand, it is important to note that the necessary communication load among the nodes may be high and requires the knowledge of the in-neighbors at each page. This can be confirmed since in the update rule (13), the row $\tilde{h}_{\theta(k)}$ of H corresponding to the chosen page $\theta(k)$ at time k appears twice. This indicates that (i) the values $x_j(k)$ of the in-neighbors $j \in \mathcal{N}_{\theta(k)}$ of page $\theta(k)$ must be collected for obtaining $\tilde{h}_{\theta(k)}^T x(k)$ and then (ii) the value $g_{\theta(k)} - \tilde{h}_{\theta(k)}^T x(k)$ is sent back to the same in-neighbors for the update of their own values $x_j(k)$.

3.3 Reduction in Communication Loads

The perspective of linear equations was the motivation also for the third approach for PageRank computation proposed in [14]. The distributed algorithm there employs the technique of matching pursuit algorithms from the area of signal processing (e.g., [39]). Matching pursuit is for approximating a signal with a finite number of functions (called atoms). As a consequence, this algorithm is guaranteed to possess exponential convergence as well.

In contrast to the approach of [57], which required the updating pages to communicate with their in-neighbors, the algorithm of [14] involves interactions only with the out-neighbors. Such neighbors are easily known to any page as they can be reached through its own hyperlinks.

To this end, we introduce the notations for the columns of the matrix H in (11). Let $h_i \in \mathbb{R}^n$ be the ith column of H. In this case, each page is equipped with two scalar variables denoted by $x_i(k)$ and $r_i(k)$, whose initial values are given by $x_i(0) = 0$ and $r_i(0) = m/n$. As in the algorithms discussed so far, let $\theta(k)$ be the page chosen at time k via the probability density in (5) in an i.i.d. fashion. Then, the two variables are updated as

$$x(k+1) = x(k) + \frac{h_{\theta(k)}^T r(k)}{\|h_{\theta(k)}\|^2} e_{\theta(k)}, \tag{14}$$

$$r(k+1) = r(k) - \frac{h_{\theta(k)}^T r(k)}{\|h_{\theta(k)}\|^2} h_{\theta(k)}, \tag{15}$$

where e_j is the jth column of the identity matrix I_n.

This scheme has the property that $Hx(k) + r(k)$ remains constant. This can be easily verified by multiplying H from the left of (14) and then adding it with (15), which yields

$$Hx(k+1) + r(k+1) = Hx(k) + r(k), \quad k \geq 0.$$

In particular, because the initial values have been chosen as $x_i(0) = 0$ and $r_i(0) = m/n$, this implies that

$$Hx(k) + r(k) = r(0) = g, \quad k \geq 0. \tag{16}$$

However, in general, in this scheme, the vector $x(k)$ is not consistent, meaning that $x(k)$ is not a probability vector. We can check this by multiplying $\mathbf{1}_n^T$ from the left of (16) and obtain

$$\mathbf{1}_n^T (Hx(k) + r(k)) = \mathbf{1}_n^T \big(mx(k) + r(k)\big) = \mathbf{1}_n^T g = m.$$

Thus, we have $\mathbf{1}_n^T x(k) = 1 - \mathbf{1}_n^T r(k)/m$.

It can be shown that this algorithm has exponential convergence in the mean-square sense, that is, it holds that $\mathbb{E}\big[\|x(k) - x^*\|^2\big] \to 0$ as $k \to \infty$. In view of (16), the convergence property can be attained by showing that $\mathbb{E}\big[\|r(k)\|^2\big]$ goes to zero exponentially fast.

A notable difference of this algorithm from that of [57] is the use of the columns $\{h_i\}$ of the matrix H instead of the rows $\{\tilde{h}_i\}$. In the networked system under consideration, the nonzero entries of the ith column h_i correspond to the out-neighbors of page i. In the update scheme (14) and (15) for page $\theta(k)$, first $h_{\theta(k)}^T r(k)$ must be computed, which requires the values $r_j(k)$ of the out-neighbor j be sent to page $\theta(k)$. Then, in (14), only $x_{\theta(k)}(k)$ is updated whereas in (15), the values $r_j(k)$ are updated for all out-neighbors j of page $\theta(k)$. This means that page $\theta(k)$ sends its own state value $r_{\theta(k)}(k)$ to all of its out-neighbors. Furthermore, it is clear that the norm $\|h_{\theta(k)}\|^2$ appearing in both (14) and (15) can be computed locally at each page in an offline manner before the execution of the algorithm.

3.4 Exponential Convergence with Consistency

All schemes that we have seen so far with exponential convergence do not have the property of consistency, that is, $x(k)$ is not a probability vector. This aspect is pointed out and then improved in the scheme introduced by [32]. Lack of consistency may be problematic in practice since the update schemes will terminate the updates in their states after a finite number of steps. Even at that point, there is no guarantee that the vector $x(k)$ is a stochastic vector. We skip the details of the update scheme; though it involves only two variables per page, the description of the algorithm tends to be complicated.

4 An Alternative Approach to PageRank

In this section, we present a new formulation of PageRank by transforming its original definition [50]. Then, novel distributed algorithms are developed where this formulation becomes the key. The idea itself is simple, but its advantage in the context of distributed computation of PageRank will become clear.

4.1 Reformulation of the PageRank Problem

The formula of PageRank in (2) can be transformed as

$$\begin{aligned} x^* = (1-m)Ax^* + \frac{m}{n}\mathbf{1}_n \quad &\Longleftrightarrow \quad x^* = [I-(1-m)A]^{-1}\,\frac{m}{n}\mathbf{1}_n \\ &\Longleftrightarrow \quad x^* = \sum_{t=0}^{\infty} [(1-m)A]^t\,\frac{m}{n}\mathbf{1}_n. \end{aligned} \tag{17}$$

In the last transformation, the Neumann series is applied. Notice that $(1-m)A$ is a Schur stable matrix because the link matrix A is stochastic and thus has the spectral radius equal to 1.

The formula in (17) suggests that the PageRank computation can be carried out iteratively in several ways. It is immediate to write down an equation for the state $x(k) \in \mathbb{R}^n$ given by

$$x(k) = \sum_{t=0}^{k} [(1-m)A]^t \, \frac{m}{n} \mathbf{1}_n. \tag{18}$$

The power method in (4) is a compact way to realize this using only $x(k)$ as the state. There, we can express the state $x(k)$ as the solution to the linear system. With a slight difference in the time index, it follows that

$$x(k) = [(1-m)A]^k \, x(0) + \sum_{t=0}^{k-1} [(1-m)A]^t \, \frac{m}{n} \mathbf{1}_n.$$

The contribution of the initial value $x(0)$ in the first term on the right-hand side attenuates asymptotically, but it is effective in maintaining consistency in the state and, thus, it always holds that $\mathbf{1}_n^T x(k) = 1$ for all k.

Another approach to the expression in (18) is to use a redundant iteration by having an additional state, denoted by $z(k) \in \mathbb{R}^n$. Set the initial states as $x(0) = z(0) = (m/n)\mathbf{1}_n$. Then, the update scheme of the two states is given as follows:

$$\begin{aligned} x(k+1) &= x(k) + (1-m)Az(k), \\ z(k+1) &= (1-m)Az(k). \end{aligned} \tag{19}$$

Through this alternative algorithm, we can obtain the PageRank vector x^*. We formally state this along with other properties of this algorithm as a proposition in the following. Similar properties will appear in our development of distributed algorithms.

Proposition 1 *In the update scheme in (19), the states $x(k)$ and $z(k)$ satisfy the following:*

(i) $z(k) \to 0$ *as* $k \to \infty$.
(ii) $x(k) \le x(k+1) \le x^*$ *for* k.
(iii) $x(k) \to x^*$ *as* $k \to \infty$.

Proof (i) As the link matrix A is stochastic, its spectral radius equals 1, and thus $(1-m)A$ is a Schur stable matrix. This implies that $z(k)$ converges to zero.

(ii) Note that $z(k) \ge 0$ because A is stochastic and $z(0) > 0$. Furthermore, we have $x(0) > 0$. Thus, it is clear that $x(k)$ is nondecreasing as a function of k. The fact that it is upper bounded by x^* follows from (iii).

(iii) From (19), we can write $x(k)$ as

$$x(k) = \sum_{t=1}^{k} z(t) + x(0) = \sum_{t=1}^{k} [(1-m)A]^t \, z(0) + x(0)$$
$$= \sum_{t=0}^{k} [(1-m)A]^t \, \frac{m}{n} \mathbf{1}_n. \tag{20}$$

This and (17) indicate that the state $x(k)$ converges to x^*. ∎

We have a few remarks on the alternative approach introduced above in comparison with the power method in (4). First, the computation uses the second state $z(k)$ in addition to $x(k)$. As seen in (20), this state $z(k)$ is integrated over time to compute $x(k)$ in (18). Second, the initial values of $x(k)$ and $z(k)$ are fixed to $(m/n)\mathbf{1}_n$, and there is no freedom in these choices. Hence, each time the computation takes place through the update scheme (19), the algorithm cannot, for example, make use of the PageRank values computed in the past as initial guesses. This point may be a limitation of this approach. Also, the initial states are not probability vectors as in the power method. In fact, $x(k)$ becomes a probability vector only asymptotically when converging to x^*. Third, notice that n/m is the minimum PageRank value, which will be assigned to pages having no incoming links. For such pages, the states will not change during the updates.

Though we do not discuss in this chapter, there is a generalized PageRank definition which uses a probability vector $v \in \mathbb{R}^n$ instead of $(1/n)\mathbf{1}_n$, that is, $x^* = (1-m)Ax^* + mv$ (e.g., [33]). In such a case, the proposed algorithm can be easily modified by replacing the initial states with $x(0) = z(0) = mv$.

We now turn our attention to distributed algorithms. From the perspective of such algorithms, one interpretation of (19) can be given as follows:

1. At time 0, all pages start with the value m/n.
2. At time k, each page attenuates its current value by $1-m$ and then sends it to its linked pages after equally dividing it. At that time, page i computes the weighted sum of the values received from the neighbors having links to the page.

We finally present a distributed algorithm based on (19) with synchronous communication.

Algorithm 1 (*Synchronous distributed algorithm*) For each page i, set the initial values as $x_i(0) = z_i(0) = m/n$. At each time k, page i transmits its value $z_i(k)$ to its neighbors along its outgoing hyperlinks and then makes updates for its two states $x_i(k)$ and $z_i(k)$ as

$$x_i(k+1) = x_i(k) + \sum_{j:\, i \in \mathscr{L}_j} \frac{1-m}{n_j} z_j(k),$$
$$z_i(k+1) = \sum_{j:\, i \in \mathscr{L}_j} \frac{1-m}{n_j} z_j(k).$$

Through simulations in Sect. 5, we will demonstrate that this synchronized algorithm may not be particularly fast, especially in comparison with the power method.

Moreover, due to the additional state $z(k)$, the algorithm requires more memory and computation. The advantage of the proposed reformulation however becomes evident in the asynchronous versions of this distributed algorithm, which will be presented in the next subsection.

4.2 Gossip-Type Distributed Algorithms

In this subsection, we extend the distributed algorithm discussed above so that the pages may interact with each other at different time instants. The algorithms are based on randomized gossip communication among the pages similarly to those presented in Sect. 3.

In the asynchronous update schemes, at each time k, one page $\theta(k) \in \mathscr{V}$ is randomly chosen, which transmits its current state value to the linked pages. We present two algorithms which differ in their probability distributions for selecting the updating pages. One uses the uniform distribution and the other is more general. In both cases, the distributions remain fixed throughout the execution of the algorithms; thus, the updating pages are chosen in an i.i.d. manner.

4.2.1 Algorithm Based on the Uniform Distribution

First, we consider the case where the selection of the updating pages follows the uniform distribution. The proposed distributed algorithm for this case is outlined below.

Algorithm 2 (*Distributed randomized algorithm*) For page $i \in \mathscr{V}$, set the initial values as $x_i(0) = z_i(0) = m/n$. At time k, the following steps are executed:

1. Select one page $\theta(k)$ based on the uniform distribution as in (5).
2. Page $\theta(k)$ transmits its value $z_{\theta(k)}(k)$ over its outgoing links.
3. Each page i updates its values $x_i(k)$ and $z_i(k)$ as

$$x_i(k+1) = \begin{cases} x_i(k) + \frac{1-m}{n_{\theta(k)}} z_{\theta(k)}(k) & \text{if } i \in \mathscr{L}_{\theta(k)}, \\ x_i(k) & \text{otherwise,} \end{cases}$$
$$z_i(k+1) = \begin{cases} 0 & \text{if } i = \theta(k), \\ z_i(k) + \frac{1-m}{n_{\theta(k)}} z_{\theta(k)}(k) & \text{if } i \in \mathscr{L}_{\theta(k)}, \\ z_i(k) & \text{otherwise.} \end{cases} \tag{21}$$

This distributed algorithm has a simple structure, which can be seen to be efficient from both computational and communication viewpoints. Each page keeps track of its states $x_i(k)$ and $z_i(k)$ and when it is randomly chosen as $\theta(k) = i$, it transmits one of its states, namely $z_i(k)$, to its neighboring pages along its outgoing hyperlinks.

Such hyperlinks are clearly known to the pages, and the necessary communication is limited with only one value at a time, without any data sent back from the linked pages. Other pages not linked by page $\theta(k)$ will simply keep their states unchanged. Since the time index k is irrelevant and not involved in the computation, there is no synchronization required in time among the pages.

The resemblance of this algorithm to Algorithm 1 is obvious. The two states $x_i(k)$ and $z_i(k)$ play similar roles in both algorithms. The differences are that in the asynchronous case, the updates are made with one neighbor at a time, and also both $x_i(k)$ and $z_i(k)$ are integrated over time. For $z_i(k)$, this was not the case in Algorithm 1. The two variables are updated differently when page i is the selected page $\theta(k)$ at time k: In such cases, its own $z_i(k)$ is set to zero. By contrast, in Algorithm 1, $z_i(k)$ is zero only in the case where page i has no incoming link.

We now rewrite this algorithm in the vector form. First, let $Q := (1-m)A$. Denote the ith columns of the $(n \times n)$-identity matrix I_n and Q, respectively, by e_i and q_i. Then, we define the matrices Q_i, $R_i \in \mathbb{R}^{n\times n}$ by

$$\begin{aligned} Q_i &:= \begin{bmatrix} e_1 \; e_2 \; \cdots \; e_{i-1} \; q_i \; e_{i+1} \; \cdots \; e_n \end{bmatrix}, \\ R_i &:= \begin{bmatrix} 0_n \; 0_n \; \cdots \; 0_n \; q_i \; 0_n \; \cdots \; 0_n \end{bmatrix}, \end{aligned}$$

where in both matrices, it is the ith column that is equal to q_i. Note that the matrices Q, Q_i, and R_i are all nonnegative matrices for $i \in \mathcal{V}$.

Let the initial states be $x(0) = z(0) = (m/n)\mathbf{1}_n$. The update schemes in (21) for the two states can be written in a compact form as

$$\begin{aligned} x(k+1) &= x(k) + R_{\theta(k)} z(k), \\ z(k+1) &= Q_{\theta(k)} z(k). \end{aligned} \tag{22}$$

We are now ready to present the main result for this distributed algorithm for PageRank computation. It shows that the true PageRank values can be obtained almost surely.

Theorem 1 *Under Algorithm 2, the PageRank vector x^* is computed with $x(k) \to x^*$ as $k \to \infty$ with probability one. In particular, the following two properties hold:*

(i) $x(k) \le x(k+1) \le x^*$ holds for $k \ge 0$.
(ii) $\mathbb{E}[x(k)] \to x^*$ as $k \to \infty$, and the convergence speed is exponential.

This theorem guarantees that the proposed gossip-based algorithm computes the true PageRank almost surely in a fully distributed fashion. In particular, similar to the synchronous case, the state vector $x(k)$ is a nondecreasing function of time k elementwise. Furthermore, its convergence to the PageRank vector is shown to be exponential in the mean, that is, the mean $\mathbb{E}[x(k)]$ approaches x^* exponentially fast. These two properties indicate that despite the use of randomization in the updates, there will not be any oscillation in the trajectories of the states. We will see that this is a unique feature among the other distributed algorithms.

In comparison with the algorithms presented in Sect. 3, our method is based on a simple reinterpretation of the definition of PageRank from the systems viewpoint, and it seems well suited for the PageRank computation in terms of convergence. We also note that similarly to [14], our algorithm does not require the pages to know the incoming links. Different from [14], communication in our scheme is directed in the sense that page i must transmit its value $z_i(k)$ to its outgoing neighbors, but need not receive their values. We will make further comparisons among the different schemes later in Sect. 4.2.3.

4.2.2 Generalization to Nonuniform Distributions

We next generalize the gossip-type distributed algorithm to the case where the pages will be chosen from distributions not limited to the uniform one. This extension is an interesting feature of the proposed approach and makes the algorithm more suitable for its use in a distributed environment. For example, depending on the computational and communication resources, the pages or the servers that carry out the PageRank computation may like to update at different frequencies [12]. Even in such situations, this algorithm is capable of computing the correct values with probability one.

Consider an i.i.d. random sequence $\{\theta(k)\}$ for the page selections. Let p_i be the probability of page i to be chosen at each time k. Assume that all p_i are strictly positive and $\sum_{i=1}^{n} p_i = 1$. The distributed algorithm for this nonuniform case is outlined below.

Algorithm 3 (*Generalized distributed randomized algorithm*) For page $i \in \mathcal{V}$, set the initial values as $x_i(0) = z_i(0) = m/n$. At time k, execute the following steps:

1. Select one page $\theta(k)$ based on the distribution p_i:

$$\text{Prob}\big\{\theta(k) = i\big\} = p_i \ \text{ for } i \in \mathcal{V}. \tag{23}$$

2. Page $\theta(k)$ transmits its value $z_i(k)$ to pages over its outgoing links.
3. Each page i updates its values $x_i(k)$ and $z_i(k)$ as in (21) of Algorithm 2.

For this algorithm, we now state the main result.

Theorem 2 *Under Algorithm 3, the PageRank vector x^* is computed with $x(k) \to x^*$ as $k \to \infty$ with probability one. In particular, the following two properties hold:*

(i) $x(k) \le x(k+1) \le x^*$ *holds for* $k \ge 0$.
(ii) $\mathbb{E}\left[x(k)\right] \to x^*$ *as* $k \to \infty$*, and the convergence speed is exponential.*

This theorem can be established similarly to Theorem 1.

This gossip-type distributed algorithm can be carried out even if the probability distribution for the page selection is not uniform. Though other algorithms may be able to deal with nonuniform selection [12, 28, 36], in those cases, additional computations and adjustments are often required. In contrast, in our algorithm, no change is necessary and the update scheme performed by each page remains exactly the same. We have seen that the state values increase monotonically to reach the true PageRank. This might indicate that increasing the selection probabilities of pages with large values may lead to faster convergence. We will examine this idea in the context of a numerical example later.

Another idea for assigning the probabilities is to make them time varying. In particular, for pages having no hyperlink pointing to them, it is enough if they transmit their values to the neighbors once in the entire run of the algorithm. This can greatly reduce the amount of the overall communication required in the algorithm. As we have seen above, such pages are already given their PageRank values, equal to m/n, as their initial states. By examining the update scheme in (21), it is clear that once such a page i transmits the state $z_i(k)$ for the first time to its linked pages, this state $z_i(k)$ is set to zero and then will remain so for the rest of the time since it will not receive any data from others. The other state $x_i(k)$ will stay unchanged at its true PageRank value m/n.

4.2.3 Comparison of Different Methods

So far, we have introduced five different methods for the computation of PageRank by randomized distributed algorithms. We have seen that they have different features in terms of convergence speed, necessary computation and communication resources, and so on. In Table 1, we summarize the various aspects of these algorithms. The five algorithms are listed in the chronological order that they appeared in the literature.

The aspects that are shown here are the following:

(i) Data received from: Each time the page $\theta(k)$ is chosen at time k for initiating an update, it may use for updating its own state the data received from other pages. These pages are linked either by the incoming hyperlinks (in-neighbors) or the outgoing ones (out-neighbors).

(ii) Data sent to: The updating page $\theta(k)$ sends its own state, which will be used for the updates by the pages that receive it. Again, such pages may be the in-neighbors or out-neighbors, depending on the algorithms.

(iii) Time synchronization: In the distributed update schemes, the pages may require time synchronization among them. This is in fact needed only in the scheme of [26] for accurately computing the time average of the states.

(iv) Consistency: The state vector $x(k)$ is said to be consistent if it is a probability vector, i.e., $\sum_{i=1}^{n} x_i(k) = 1$ at all times k. As discussed in [57], this property may not be critical and may not be possible to achieve especially if the total number of pages in the network is unknown.

(v) Convergence speed: The method of [26] is not exponential in its convergence speed. This is because it uses the time average and thus becomes linear. Other algorithms all have exponential rates for their convergence.

(vi) Simulation result: We will see in the next section that the five algorithms exhibit different performances in numerical simulations. This point will be further discussed there.

Table 1 Comparison of randomized distributed algorithms

Method	Data received from	Data sent to	Time synch.	Consistent	Conv. speed	Simulation result
Ishii and Tempo [26]	None	Out-neighbors	Yes	Yes	Linear	Slow
You et al. [57]	In-neighbors	In-neighbors	No	No	Exponential	Medium
Dai and Freris [14]	Out-neighbors	Out-neighbors	No	No	Exponential	Fast
Lagoa et al. [32]	In-neighbors	In-neighbors	No	Yes	Exponential	Medium
Algorithm 2	None	Out-neighbors	No	No	Exponential	Very fast

5 Numerical Examples

To illustrate the performance of the distributed algorithms discussed so far in Sects. 3 and 4, we present results obtained through numerical simulations. We apply the different update schemes to two types of graphs and compare their properties including convergence speeds.

5.1 *Small Graph*

The first case that we consider is the simple graph with seven pages shown in Fig. 1 from Example 1.

5.1.1 Synchronized Algorithms

As an initial step, we examine the performance of the following two synchronous algorithms: The power method in (4) and Algorithm 1 from Sect. 4. These algorithms may be more suited for centralized implementation, but if proper synchronization can be introduced, distributed implementation should be possible as discussed in Sect. 4.

Their differences can be summarized as follows: (i) They have been derived from different viewpoints. The power method follows the original definition of (3) while Algorithm 1 is based on the interpretation expressed as the Neumann series (18) and has not been studied elsewhere. (ii) The numbers of variables per page are one for the power method and two for Algorithm 1. (iii) For the initial states, the power method can take any initial value as long as it is a probabilistic vector; in this simulation, we used uniform values, i.e., $(1/n)\mathbf{1}_n$. In the meantime, Algorithm 1 requires $x(0)$ to be fixed as $(m/n)\mathbf{1}_n$, which is also uniform, but not a probabilistic vector. On the other hand, these two algorithms share the property of being deterministic. Thus, the responses of pages 6 and 7 become exactly the same since both of them have no incoming link due to the structure of the graph.

The time response of the PageRank value for each page is shown in Fig. 2 for the two algorithms. We observe that the power method converges faster and, for most of the nodes, it takes less than 10 time steps. In the responses of the proposed algorithm, the convergence is slower and takes about 30 time steps. It is noticeable that they are nondecreasing with respect to time, a property shown in Proposition 1(i). Also, recall that for pages 6 and 7, in the proposed algorithm, the PageRank values of these pages are equal to the assigned initial values m/n. Hence, for these pages, the proposed algorithm requires no update.

5.1.2 Distributed Algorithms via Gossiping

Next, we discuss the simulation results for the gossip-based distributed algorithms using the simple network.

We make comparisons of the convergence performance of the five algorithms shown in Table 1. All five algorithms select one page at each time k based on the uniform distribution as shown in (5), and we applied the same sequence $\{\theta(k)\}$ to them for each run. As discussed earlier, in the two algorithms of [14, 57], the total number n of pages in the web may be unknown; here, we assume that n is known by all pages. Concerning the initial states, only our proposed algorithm requires that the pages take fixed values, equal to m/n. Other algorithms have some freedom in the choices. Here, however, we set them so that all pages are given the same initial values: For the algorithm of [14], it was set to 0, and in the remaining two algorithms, we took $1/n$.

The time responses of the calculated PageRank values of the pages are plotted in Fig. 3. We omit the result for page 7 as its behavior is similar to that of page 6. It is observed that the responses for most algorithms are oscillatory or noisy due to the randomization in the gossiping for updates and communication. On the other hand, there are certain levels of differences in the speeds of convergence among the algorithms. The responses of [26] appear to be the slowest and the most oscillatory with high peaks, possibly reflecting the fact of being the only non-exponential algorithm among the five.

In view of this, the proposed algorithm, namely Algorithm 2, is characteristic in that despite the randomization, the profile of the responses is smooth and again nondecreasing as in Fig. 2. This behavior is most visible in the plot for page 5. It is also clear that the proposed algorithm is the fastest in terms of convergence time for all pages in comparison with other algorithms. This is more evident in Fig. 4 where the total errors in the states from the true PageRank are displayed for all five algorithms in the logarithmic scale.

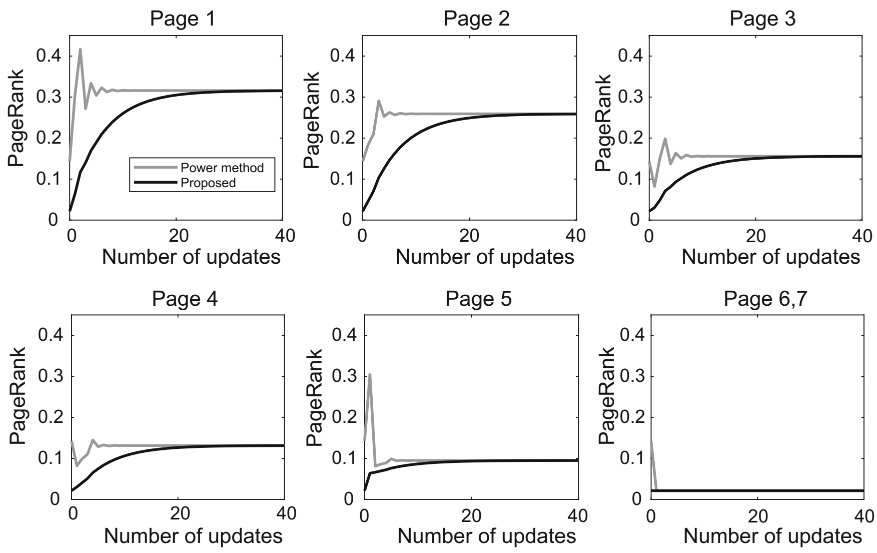


Fig. 2 Time responses of the synchronous algorithms for the small graph: The power method and the proposed Algorithm 1

5.1.3 Comparison of Distributions in Page Selection

In this part of the simulation, we illustrate how the convergence speed can be improved by employing Algorithm 3 with a nonuniform distribution for the random selection of $\theta(k)$. As discussed in Sect. 4.2.2, to improve convergence of the algorithm, it seems reasonable to increase the selection probability of pages which are expected to take larger PageRank values. We adjusted the probabilities so that pages having more incoming links are more likely to be selected, and each page's probability of selection is larger than 0. In particular, we assigned each page the probability proportional to its in-degree plus 1.

In Fig. 5, the time responses of the pages are shown for two algorithms, Algorithm 2 using the original uniform distribution in (5) and Algorithm 3 using this nonuniform distribution. As in Fig. 3, the responses of page 7 are omitted. We confirm that the nonuniform distribution is capable to further accelerate the convergence by a certain margin. It remains to be investigated what kind of distribution can in general be beneficial in improving the convergence rate.

5.2 *Large Graph*

We proceed to apply the five distributed algorithms to a larger web data. Specifically, we randomly generated a graph with 60 nodes. Figure 6 displays the network

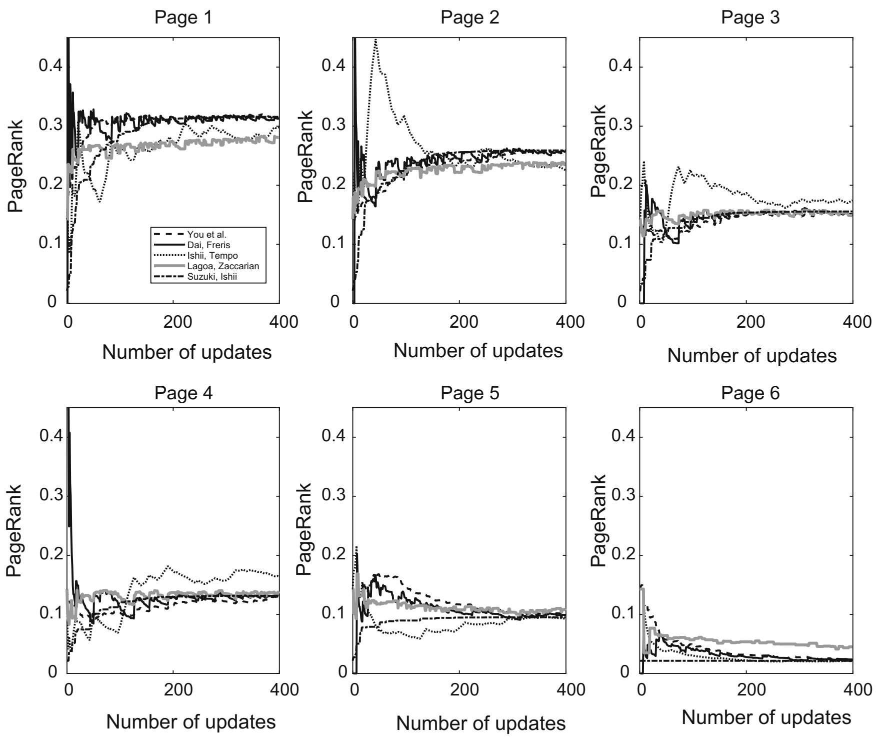


Fig. 3 Time responses of the asynchronous algorithms for the small graph: Ishii and Tempo [26], You et al. [57], Dai and Freris [14], Lagoa et al. [32], and the proposed Algorithm 2

structure where the dots indicate the nonzero entries of the hyperlink matrix A. The first eight pages are designed to be popular and receive hyperlinks from roughly one-third of the remaining nodes. In addition, each node has up to two hyperlinks to randomly selected nodes. In total, there are 223 hyperlinks and no dangling node in the network.

We applied the five randomized distributed algorithms with similar initial conditions. The responses of the sum of the errors are shown in Fig. 7. Here, we confirm that the performance of the proposed algorithm is the fastest and the error reduces exponentially. While the response of [26] is the slowest, the three methods of [14, 32, 57] exhibit exponential convergence. It is noted that we made simulations with other graphs of various sizes and observed similar results in general.

6 Discussion on Randomization in Multi-agent Systems

In this section, we would like to discuss, from a more general perspective, the roles that randomization plays in distributed algorithms and control for multi-agent sys-

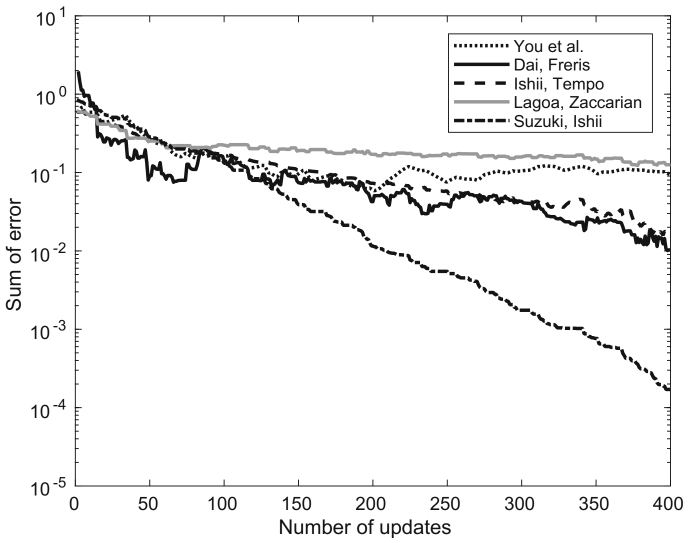


Fig. 4 Time responses of the errors in the asynchronous algorithms for the small graph: Ishii and Tempo [26], You et al. [57], Dai and Freris [14], Lagoa et al. [32], and the proposed Algorithm 2

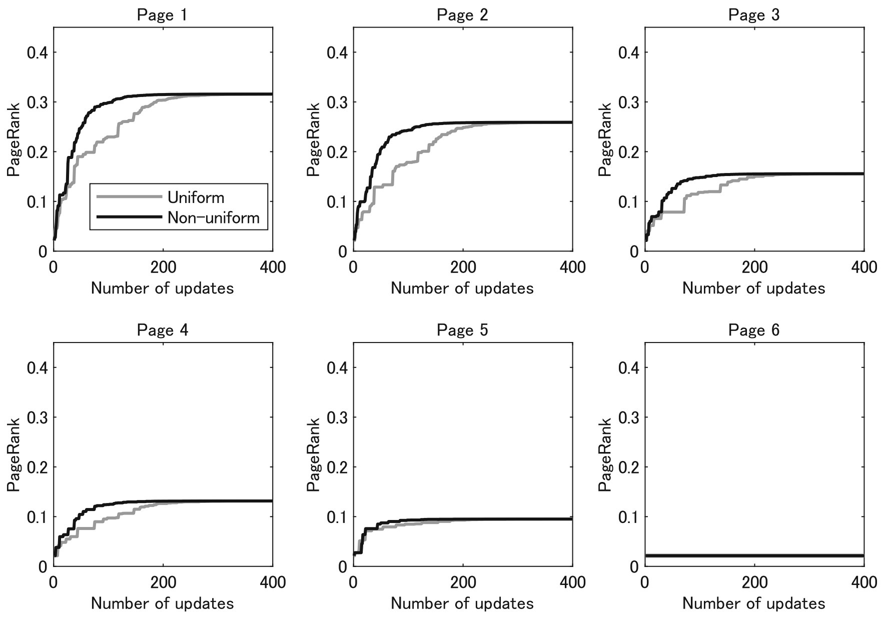


Fig. 5 Time responses in the asynchronous algorithms for the small graph: Algorithm 2 (uniform distribution) and Algorithm 3 (nonuniform distribution)

tems. This is in fact a broad subject as randomized techniques can now be found to be employed in many ways and we do not intend to be exhaustive. Our discussion hence will be limited to the recent research that we conducted and the works that are related.

Randomization techniques have received a significant level of attention within the community of systems control in the last two decades or so. In the early times, the motivation for employing such techniques originated from the need to address the issue of computational complexity arising in the context of uncertain and hybrid systems. For such systems, many control analysis and design problems are known to be computationally difficult to solve and can even be NP-hard (e.g., [4]). Application of probabilistic techniques to such problems has been found to be useful in developing computationally efficient algorithms. Recent developments can be found in the monograph [52]; see also the survey paper [53].

In large-scale network systems, randomized algorithms have been widely employed, but the distributed nature of such systems calls for the exploitation of randomization with a purpose different from that of relaxing computational complexity as discussed above. Here, we would like to highlight three essential roles in multi-agent systems that randomized techniques can play. Those are related to (i) communication, (ii) decision-makings through dithering, and (iii) cybersecurity. In the following, we briefly describe recent progress along these directions.

(i) In multi-agent systems, communication among the agents must be initiated by the individual agents since there is often no centralized entity that would command them to synchronize. Thus, as we have seen in this chapter, communicating at

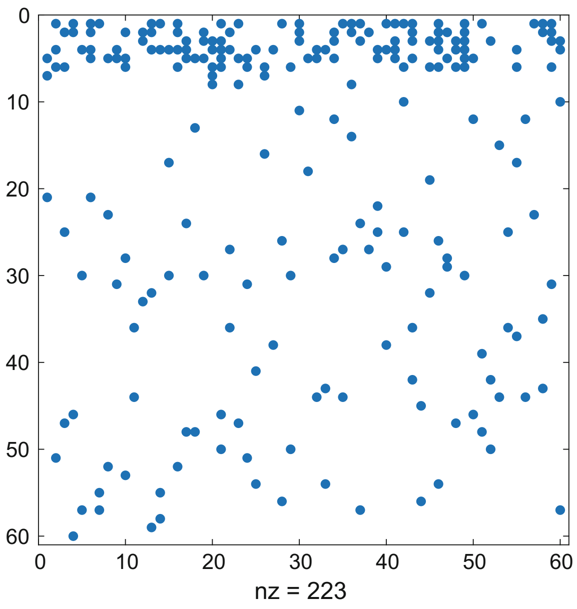


Fig. 6 Network structure of the large graph with 60 web pages

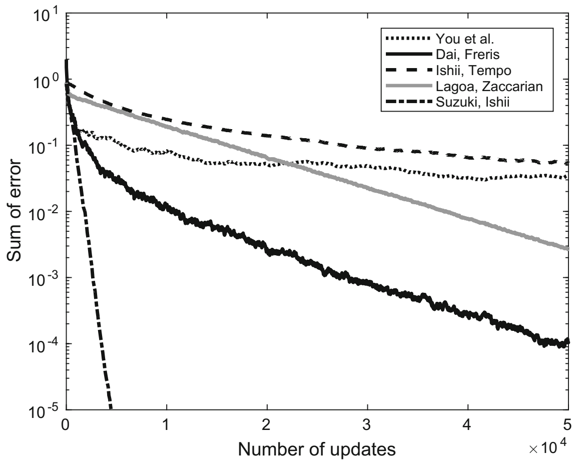


Fig. 7 Time responses of the errors in the asynchronous algorithms for the large graph: Ishii and Tempo [26], You et al. [57], Dai and Freris [14], Lagoa et al. [32], and the proposed Algorithm 2

randomly chosen time instants can be a useful option. It is also a realistic model in the case of wireless communication; if collisions occur due to simultaneous transmissions, retransmissions will be made after some waiting times, whose lengths are randomly chosen. Communication at random times is sometimes referred to as gossiping [6] and has been exploited in a number of works in multi-agent systems including [9–11, 15, 30, 34, 46, 51, 56].

(ii) In distributed algorithms, randomized algorithms can help the process of decision-makings by introducing a certain level of noise or perturbation in the system. In signal processing, dithering is a well-known probabilistic method in quantization [55]. It introduces random noise before the operation of quantizing a real-valued signal. In audio signals, for instance, dithering is commonly used for reducing unnatural sounds in quantized signals that can result from certain periodicity introduced through the analog-to-digital conversion.

This method has been found useful in the context of multi-agent systems as well. In particular, in the so-called quantized consensus problems, agents take integer values in their states. There, some update schemes employ randomized quantization so that, for example, the state may be rounded up or down randomly. Such a method has the effect of introducing perturbation in the consensus process so as to avoid the states being stuck before reaching consensus. For related studies, see, e.g., [2, 9, 10, 15, 21, 30].

(iii) In potentially hazardous environments where malicious attackers may exploit the vulnerabilities in systems and communication networks (e.g., [18, 44, 47]), randomization can be a viable method in raising the security level. For example, intruders interested in the data exchanged among agents may need more resources to attack or to eavesdrop on the communications when the times are chosen randomly. Such a stochastic scheme is proposed and analyzed in a multi-agent consensus problem in [31]; the agents' communication is disrupted by jamming attacks, but the energy for emitting jamming signals is constrained as in [49]. Making the transmission times unpredictable becomes the key to realize consensus even under a less stringent condition for the attackers.

On the other hand, in the literature of distributed algorithms in computer science, multi-agent consensus has been long studied. An important class of problems there includes fault-tolerant consensus for multi-agent systems in the presence of faulty agents or even those which are driven by malicious attackers. Such agents may not follow the a priori given update rules. The non-faulty, regular agents are equipped with a resilient version of the consensus algorithm, which determines the neighbors taking suspicious values and thus to be ignored in the updates. Such problems have been studied in, e.g., [3, 5, 54] in computer science, and more recently in, e.g., [16, 17, 35] in the systems control literature.

In computer science, probabilistic algorithms have been known to improve resilience. In distributed decision-making problems, various "impossibility results" have been derived, showing that deterministic approaches are insufficient for achieving the desired goal with certain scalability properties [20, 37, 45]. We would like to mention that recently, in [15], it was established that in asynchronous update schemes for resilient consensus, probabilistic gossip-based communication among agents can be superior to deterministic approaches in terms of the necessary network structures; this may be seen as a form of an impossibility result.

More generally, probabilistic techniques have been extensively studied in the area of algorithms in computer science; see, e.g., the monographs [41, 42] and the references therein. As discussed in [53], randomized algorithms can be classified into two categories: The Monte Carlo type and the Las Vegas type. Roughly speaking, algorithms of the Monte Carlo type may produce incorrect outputs with limited probabilities whereas the Las Vegas types are guaranteed to provide correct solutions with probability one. Many of the algorithms in the studies of uncertain and hybrid control systems belong to the Monte Carlo type. They often rely on random sampling in the uncertain sets, which are continuous sets. On the other hand, those discussed in this chapter are of the Las Vegas type. Indeed, in Theorems 1 and 2 of Sect. 4, we have seen that the convergence of the randomized algorithms is guaranteed almost surely.

7 Conclusion

In this chapter, we have introduced the problem of PageRank computation from the perspective of systems and control and then provided a short overview on the recent developments on distributed algorithms. We have also proposed a new class of distributed algorithms for the computation of PageRank using a new interpretation of its definition. Specifically, two types of distributed algorithms have been obtained: One is synchronous in that all agents update their state values at the same time, while in the other, randomization is used for determining the page that initiates an update at each time step. Regarding their convergence properties, it has been established that they are exponential. The relation of the proposed algorithms to those in the literature has been discussed as well. One characteristics of our approach making it suitable for distributed implementation is that it does not need to follow the uniform distribution. We have shown through simulations that our algorithms exhibit superior performance in both a simple web and a large-scale web. Finally, a general discussion from a broader perspective on the advantages that randomization may bring to distributed algorithms has been given.

In future research, we will further analyze the convergence speeds of the proposed algorithms and employ other schemes for page selections. We are also interested in studying other problems where our approach can be useful in developing distributed algorithms.

Acknowledgements The authors are sincerely grateful to Roberto Tempo for our long-term collaboration, which led us to the research on randomized algorithms for large-scale network systems presented in this chapter. He will be remembered as a wonderful collaborator, mentor, and friend.

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. T. C. Aysal, M. J. Coates, and M. G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Trans. Signal Processing*, 56:4905–4918, 2008.
3. M. H. Azadmanesh and R. M. Kiechafer. Asynchronous approximate agreement in partially connected networks. *Int. J. Parallel Distrib. Networks*, 5:26–34, 2002.
4. V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36:1249–1274, 2000.
5. Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil. Optimal Byzantine-resilient convergence in uni-dimensional robot networks. *Theoretical Computer Science*, 411:3154–3168, 2010.
6. S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Trans. Inform. Theory*, 52:2508–2530, 2006.
7. S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks & ISDN Systems*, 30:107–117, 1998.
8. F. Bullo. *Lectures on Network Systems*. CreateSpace, 2018.
9. K. Cai and H. Ishii. Quantized consensus and averaging on gossip digraphs. *IEEE Trans. Autom. Contr.* 56:2087–2100, 2011.
10. K. Cai and H. Ishii. Convergence time analysis of quantized gossip consensus on digraphs. *Automatica*, 48:2344–2351, 2012.
11. R. Carli, F. Fagnani, P. Frasca, and S. Zampieri. Gossip consensus algorithms via quantized communication. *Automatica*, 46:70–80, 2010.
12. T. Charalambous, C. Hadjicostis, M. Rabbat, and M. Johansson. Totally asynchronous distributed estimation of eigenvector centrality in digraphs with application to the PageRank problem. In *Proc.* 55*th IEEE Conf. on Decision and Control*, pages 25–30, 2016.
13. B. C. Csáji, R. M. Jungers, and V. D. Blondel. PageRank optimization by edge selection. *Discrete Applied Mathematics*, 169:73–87, 2014.
14. L. Dai and N. Freris. Fully distributed PageRank computation with exponential convergence. *arXiv:1705.09927*, 2017.
15. S. M. Dibaji, H. Ishii, and R. Tempo. Resilient randomized quantized consensus. *IEEE Trans. Autom. Contr.* 63:2508–2522, 2018.
16. S.M. Dibaji and H. Ishii. Resilient multi-agent consensus with asynchrony and delayed information. In *Proc.* 5*th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, pages 28–33, 2015.

17. S.M. Dibaji and H. Ishii. Resilient consensus of second-order agent networks: Asynchronous update rules with delays. *Automatica*, 81:123–132, 2017.
18. H. Fawzi, P. Tabuada, and S. Diggavi. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Trans. Autom. Contr.* 59:1454–1467, 2014.
19. O. Fercoq, M. Akian, M. Bouhtou, and S. Gaubert. Ergodic control and polyhedral approaches to PageRank optimization. *IEEE Trans. Autom. Contr.* 58:134–148, 2013.
20. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32:374–382, 1985.
21. P. Frasca, R. Carli, F. Fagnani, and S. Zampieri. Average consensus on networks with quantized communication. *Int. J. Robust & Nonlinear Control*, 19:1787–1816, 2009.
22. P. Frasca, H. Ishii, C. Ravazzi, and R. Tempo. Distributed randomized algorithms for opinion formation, centrality computation and power systems estimation: A tutorial overview. *European J. Control*, 24:2–13, 2009.
23. D. F. Gleich. PageRank beyond the Web. *SIAM Review*, 57(3):321–363, 2015.
24. R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge Univ. Press, 1985.
25. H. Ishii and R. Tempo. Computing the PageRank variation for fragile web data. *SICE J. Control, Measurement, and System Integration*, 2:1–9, 2009.
26. H. Ishii and R. Tempo. Distributed randomized algorithms for the PageRank computation. *IEEE Trans. Autom. Contr.* 55:1987–2002, 2010.
27. H. Ishii and R. Tempo. The PageRank problem, multi-agent consensus and web aggregation: A systems and control viewpoint. *IEEE Control Systems Magazine*, 34:34–53, 2014.
28. H. Ishii, R. Tempo, and E.-W. Bai. PageRank computation via a distributed randomized approach with lossy communication. *Syst. & Cont. Lett.* 61:1221–1228, 2012.
29. H. Ishii, R. Tempo, and E.-W. Bai. A web aggregation approach for distributed randomized PageRank algorithms. *IEEE Trans. Autom. Contr.* 57:2703–2717, 2012.
30. A. Kashyap, T. Başar, and R. Srikant. Quantized consensus. *Automatica*, 43:1192–1203, 2007.
31. K. Kikuchi, A. Cetinkaya, T. Hayakawa, and H. Ishii. Stochastic communication protocols for multi-agent consensus under jamming attacks. In *Proc. 56th IEEE Conf. on Decision and Control*, pages 1657–1662, 2017.
32. C. M. Lagoa, L. Zaccarian, and F. Dabbene. A distributed algorithm with consistency for PageRank-like linear algebraic systems. In *Proc. 20th IFAC World Congress*, pages 5339–5344, 2017.
33. A.N. Langville and C.D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.
34. J. Lavaei and R. M. Murray. Quantized consensus by means of gossip algorithm. *IEEE Trans. Autom. Contr.* 57:19–32, 2012.
35. H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram. Resilient asymptotic consensus in robust networks. *IEEE J. Selected Areas Comm.* 31:766–781, 2013.
36. J. Lei and H.-F. Chen. Distributed randomized PageRank algorithm based on stochastic approximation. *IEEE Trans. Autom. Contr.* 60:1641–1646, 2015.
37. N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, CA, 1997.
38. J. M. Maestre, H. Ishii, and E. Algaba. Node aggregation for enhancing PageRank. *IEEE Access*, 5:19799–19811, 2017.
39. S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41:3397–3415, 1993.
40. M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.
41. M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
42. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
43. A.V. Nazin and B.T. Polyak. Randomized algorithm to determine the eigenvector of a stochastic matrix with application to the PageRank problem. *Automation and Remote Control*, 72:342–352, 2011.

44. F. Pasqualetti, F. Dörfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Trans. Autom. Contr.* 58:2715–2729, 2013.
45. M. Rabin. Randomized Byzantine generals. In *Proc. 24th IEEE Symp. Foundations of Comp. Sci.* pages 403–409, 1983.
46. C. Ravazzi, P. Frasca, R. Tempo, and H. Ishii. Ergodic randomized algorithms and dynamics over networks. *IEEE Trans. Control of Network Syst.* 2:78–87, 2015.
47. H. Sandberg, S. Amin, and K.H. Johansson, guest editors. Special issue on cyberphysical security in networked control systems. *IEEE Control Systems Magazine*, 35(1), 2015.
48. A. Sarma, A. Molla, G. Pandurangan, and E. Upfal. Fast distributed PageRank computation. *Theoretical Computer Science*, 561:113–121, 2015.
49. D. Senejohnny, P. Tesi, and C. De Persis. A jamming-resilient algorithm for self-triggered network coordination. *IEEE Trans. Control of Network Syst.* 5:981–990, 2018.
50. A. Suzuki and H. Ishii. Distributed randomized algorithms for PageRank based on a novel interpretation. In *Proc. American Control Conf.* pp. 472–477, 2018.
51. A. Tahbaz-Salehi and A. Jadbabaie. A necessary and sufficient condition for consensus over random networks. *IEEE Trans. Autom. Contr.* 53:791–795, 2008.
52. R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems, with Applications,* Second Edition. Springer, London, 2013.
53. 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.
54. L. Tseng and N. H. Vaidya. Fault-tolerant consensus in directed graphs. In *Proc. ACM Symp. Principles of Distributed Comput.* pages 451–460, 2015.
55. R. A. Wannamaker, S. P. Lipshitz, J. Vanderkooy, and J. N. Wright. A theory of nonsubtractive dither. *IEEE Trans. Signal Processing*, 48:499–516, 2000.
56. K. You, Z. Li, and L. Xie. Consensus condition for linear multi-agent systems over randomly switching topologies. *Automatica*, 49:3125–3132, 2013.
57. K. You, R. Tempo, and L. Qiu. Distributed algorithms for computation of centrality measures in complex networks. *IEEE Trans. Autom. Contr.* 62:2080–2094, 2017.
58. W. Zhao, H.-F. Chen, and H. Fang. Convergence of distributed randomized PageRank algorithms. *IEEE Trans. Autom. Contr.* 50:1177–1181, 2013.
59. A. Zouzias and N. M. Freris. Randomized extended Kaczmarz for solving least squares. *SIAM J. Matrix Analysis and Applications*, 34:773–793, 2013.