A PageRank Algorithm based on Asynchronous Gauss-Seidel Iterations

Daniel Silvestre, João Hespanha and Carlos Silvestre

Abstract— **We address the PageRank problem of associating a relative importance value to all web pages in the Internet so that a search engine can use them to sort which pages to show to the user. This precludes finding the eigenvector associated with a particular eigenvalue of the link matrix constructed from the topology graph of the web. In this paper, we investigate the potential benefits of addressing the problem as a solution of a set of linear equations. Initial results suggest that using an asynchronous version of the Gauss-Seidel method can yield a faster convergence than using the traditional power method while maintaining the communications according to the sparse link matrix of the web and avoiding the strict sequential update of the Gauss-Seidel method. Such an alternative poses an interesting path for future research given the benefits of using other more advanced methods to solve systems of linear equations. Additionally, it is investigated the benefits of having a projection after all page ranks have been updated as to maintain all its entries summing to one and positive. In simulations, it is provided evidence to support future research on approximation rules that can be used to avoid the need for the projection to the n-simplex (the projection represents in some cases a threefold increase in the convergence rate over the power method) and on the loss in performance by using an asynchronous algorithm.**

I. Introduction

A search engine is a tool that allows users to supply a query and obtain web pages related to the information they are searching. A crucial task for these tools is some sort of ranking mechanism that selects meaningful links to be presented first. One of the most well-known algorithms is the PageRank from Google, which was initially proposed in [1], to rank pages based on their relative importance and on the number of links to each specific page. The interested reader is referred to [2] for a summary of tools, techniques and results that contribute to the PageRank algorithm and to the surveys in [3], [4] and [5] for the full description of how the search engine of Google operates.

D. Silvestre is with the Department of Electrical and Computer Engineering of the Faculty of Science and Technology of the University of Macau, Macau, China, and with the Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal, dsilvestre@umac.mo

João P. Hespanha is with the Dept. of Electrical and Computer Eng., University of California, Santa Barbara, CA 93106-9560, USA. J. Hespanha was supported by the U.S. Office of Naval Research under the MURI grant No. N00014-16-1-2710, the National Science Foundation under Grants No. ECCS-1608880. hespanha@ece.ucsb.edu

C. Silvestre is with the Department of Electrical and Computer Engineering of the Faculty of Science and Technology of the University of Macau, Macau, China, on leave from Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal, csilvestre@umac.mo

This work was partially supported by the project MYRG2016- 00097-FST of the University of Macau; by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UID/EEA/50009/2013.

The PageRank algorithm's main idea is that the ranking of a web page is dependent both on the number of links connecting to it and their corresponding ranking. An alternative way of perceiving the algorithm is to think of it as a random walk over the entire internet structure. Then, the rank of each page is the percentage of time this random surfer would spend in that particular page. Both views can be expressed mathematically as the problem of finding the eigenvector associated with the largest eigenvalue of a stochastic matrix, i.e., the stationary distribution of the corresponding Markov chain [6].

Computation of the page rank has been investigated by the research community using different techniques. In [7], the authors present a distributed randomized algorithm for the PageRank computation. In [8], by the same authors in a collaboration have studied ergodic randomized algorithms that can be applied to the PageRank problem. In essence, both algorithms resort to the power method and incorporate other research work in the field of consensus and distributed linear algorithms. In this setup, each page would compute its rank and send information to the neighbor web pages (i.e., those which it has a link to). A similar type of algorithms is discussed in [9]. In this paper, our aim is to tackle the convergence speed of the power method and show that using Gauss-Seidel iterations can yield a faster convergence in a scenario where a set of processors jointly computes the PageRank for a subset of the entire network structure.

The idea of using the Gauss-Seidel is not a novel idea as it can be found in [10]. However, to the best of our knowledge, the use of a projection step has not been proposed before. Such a step increases the convergence speed although the exact projection increases the communications required for each step. Nevertheless, a normalization is also studied as an alternative which can be performed without additional information or exchanges.

In a similar setup to the one in this paper, the authors in [11] and [12] have proposed the use of information aggregation to speed up a distributed randomized algorithm for the PageRank. In the literature, one can find similar earlier examples of this approach to deal with the convergence speed of randomized versions of the algorithm such as that in [13]. In [14], the authors also devise a novel technique to store the pages as to reduce the inter-communications. All such approaches work on partitioning the data across the different processors whereas our approach is based on having a different update for the pages that belong to the same processor.

In the literature, there are alternatives to the power method, namely finding the eigenvector as an optimization problem

or based on monte carlo experiments. The main challenge of these approaches is the computation of the gradient. Given that it is a linear combination of all states, it does not scale well with the dimension of the state-space. An alternative is to use an approximation such as in [15]. The execution can take advantage of the randomization inherent to the stochastic approximation and, therefore, the algorithm is distributed by nature.

The work in [16] exploits the definition of the PageRank as the result of a power series and provides results as how the errors of the approximation evolves after truncating the higher order terms. The main idea is that an approximation of the ranking vector can be found as the difference of two consecutive elements of the power series. In this paper, we shift our attention to distributed algorithms that preserve the sparsity of their associated transition matrix.

Another interesting direction is provided in [17], where the authors study the trend that the ranks for different pages converge at very distinct rates, with some being very fast whereas others take a considerable amount of additional time instants. The authors explore that behavior to avoid part of the arithmetic operations and save computational resources. Although targeting the problem using a different approach, it is an interesting issue to approach regarding the solution presented in this paper.

The main contribution of the this paper is to present a distributed algorithm to be run by a family of processors, each storing a subset of the entire network graph. This solution satisfies two criteria: i) it converges faster to the solution than the power method; ii) it minimizes the communications between different processors. Moreover, it can be cast in the format of a synchronous or asynchronous algorithm, with a deterministic or randomized selection of updating pages.

Notation : The transpose and the spectral radius of a matrix A are denoted by $A^\intercal$ and $\rho(A)$, respectively. We let $\mathbf{1}_n$ and $\mathbf{0}_n$ denote n-dimension vector of ones and zeros, and I_n the identity matrix of dimension n. Dimensions are omitted when no confusion arises. The vector $\mathbf{e}_i$ denotes the canonical vector whose components equal zero, except component i that equals one. The notation $\text{diag}(A)$ indicates a diagonal matrix with the diagonal being taken from the diagonal of A. The Euclidean norm for vector x is represented as $\|x\|_2 := \sqrt{x^\intercal x}$.

II. Preliminaries and Problem Statement

This section formally introduces the multiple alternative formulations of the PageRank, identifying the key issues that arise when selecting a specific formulation.

A. PageRank Problem

The PageRank problem described in the remainder of this section is based on the material found in [1], [4] and [5].

Consider a network representing the interconnection of n webpages defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V} := \{1, \cdots, n\}$ corresponds to all webpages numbered from 1 to n and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges representing the links between any two webpages. The edge $(i, j) \in \mathcal{E}$ if there exists an outgoing link from page with index i to that with index j.

The value produced by the PageRank algorithm for a page i is going to be denoted by the real number $x_i^\star \in [0, 1]$. We can stack all such variables and get $x^\star := \begin{bmatrix} x_1^\star & \cdots & x_n^\star \end{bmatrix}^\intercal$ which represents the ranks for all the pages.

PageRank searches for a normalized rank that depends on the number of neighbors in the graph $\mathcal{G}$ and those nodes rank. This translates into the rank of node i being the weighted sum of the ranks of its neighbors, i.e.,

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

where the matrix $A \in \mathbb{R}^{n \times n}$ (the link matrix) is defined as

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

The aforementioned problem of finding the eigenvector associated with the eigenvalue of 1 in the link matrix A is not easy given that the general web does not form a strongly connected graph. In the literature, it was proposed a slightly changed version of the problem as a walk-around to this issue. To ensure the existence of the eigenvalue 1, outgoing links are added to all the *dangling nodes*, i.e., all nodes with no out-going links such as jpeg or pdf files; for example links to their parent pages. In doing so, matrix A defined in (1) becomes a stochastic matrix.

To avoid the problem of multiple eigenvalues equal to one, the work in [1] proposes finding the eigenvector of the following matrix $M \in \mathbb{R}^{n \times n}$:

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

where $m \in (0, 1)$ is a parameter defining the convex combination of the matrix A with the matrix $S := \mathbf{1}_n \mathbf{1}_n^\intercal$. A typical choice is $m = 0.15$ [1]. Following the discussion in [7], a greater value of m results in a faster convergence towards the PageRank for the standard power method but also averages out the ranks for the pages. The problem is summarized in Definition 1.

Definition 1 (PageRank problem): The PageRank value vector $x^\star$ is defined as the eigenvector of matrix M that satisfies:

$$x^\star = Mx^\star, \quad x^\star \in [0, 1]^n, \quad \sum_{i=1}^{n} x_i^\star = 1$$

B. Network Model

In this paper, the underlying assumption for the network topology is that there are N processors each of them holding a subset of the rows in matrix M and the entries of vector x. In reality, this is a fairly reasonable assumption given that the state space is on the order of 10 billion as reported in a 2012 document [11]. Thus, our assumption is that $N \ll n$ as opposed to other models such in [7]. Moreover, we assume that each processor can communicate to other processors if the web data it holds is linked in the original graph $\mathcal{G}$. In a more formal way, we define a graph $\mathcal{G}_\mathcal{P} := (\mathcal{P}, \mathcal{L})$

representing the network topology for the processors $\mathcal{P} := \{p_1, \cdots, p_N\}$ with the link set $\mathcal{L} \subseteq \mathcal{P} \times \mathcal{P}$. The link between processors p_i and p_j exists, i.e., $(p_i, p_j) \in \mathcal{G}_{\mathcal{P}}$ if it exists an out-going link between two pages belonging to those processors, i.e., $\exists q \in \mathcal{R}_i, \ell \in \mathcal{R}_j : (q, \ell) \in \mathcal{E}$, where the sets $\mathcal{R}_i$ and $\mathcal{R}_j$ represent the web pages belonging to processor p_i and p_j, respectively.

As a consequence of the assumption on the network model, the adopted method should favor computations involving variables such that their indices belong to the same $\mathcal{R}_i$ set. This minimizes inter-processors communication which reduces the communication time. A similar idea can be found in [14] where the redefinition of the tuples being stored saves inter-communication between clusters belonging to different processors.

C. Equivalent formulations for the PageRank

The PageRank problem can take different formulations that may impact on the design of a distributed algorithm. The standard format to compute the eigenvector for this problem as in Definition 1 is to use the power method, leading to the following iteration:

$$x(k+1) = Mx(k) = (1-m)Ax(k) + \frac{m}{n}1_n \qquad (2)$$

where $x(k) \in \mathbb{R}^n$ and $\forall k \geq 0 : 1_n^\intercal x(k) = 1$. We emphasize that the above formulation is only possible given the assumption that the eigenvector sums to one and that the iteration of the method preserves this property. Also that (2) involves a computation according to the sparse matrix A and the sum of a constant. However, this method convergence rate is only $|\lambda_2(M)/\lambda_1(M)|^k$, which might be slow depending on the second largest eigenvalue of M.

The PageRank problem can also be formulated as an optimization problem or as the solution to a linear equation. The results in this paper focus on the latter where the objective is to solve

$$(I_n - (1-m)A)x = \frac{m}{n}1_n. \qquad (3)$$

In the next section, emphasis is placed on the fact that the power method applied to the PageRank problem is equivalent to the Jacobi iteration for solving (3) and, then using the Gauss-Seidel algorithm to achieve an improved performance in the network model assumed in this work.

III. PageRank solution using Gauss-Seidel iterations

Any attempt to solve the problem in (3) must take into consideration two main requirements, namely, being a distributed iteration and a particular focus on the storage of vector x due to its dimension. In the literature, many approaches have been taken to solve linear equations in a distributed fashion but many resort to having an estimate of x per processor (see an example in a recent publication in [18]) Such an approach is clearly not suitable for the problem at hand due to the large state space size n.

The idea presented in this paper is to use standard algorithms for distributed solution of linear equations and leverage those according to the network model to gain in performance. Our analysis starts by presenting the format for the Jacobi method. For a general system $Ax = b$, where A can be partitioned as $A = D + R$ for $D = \text{diag}(A)$ and $R = A - \text{diag}(A)$, the Jacobi iteration takes the following form:

$$x(k+1) = D^{-1}(b - Rx(k)) \qquad (4)$$

which is distributed given that matrix D^{-1} is diagonal. The following lemma asserts that the Jacobi method for the PageRank is equivalent to the power method.

Lemma 2: The power method iteration in (2) is equivalent to the Jacobi method in (4) applied to the problem in (3).

Proof: For the problem (3) we have $D = I_n$, $R = -(1-m)A$ and $b = \frac{m}{n}1_n$ in (4), which leads to the conclusion. ∎

The natural step is to introduce the Gauss-Seidel iteration and apply it to the problem. For a general system $Ax = b$, decomposing $A = L + D + U$ as the sum of lower, diagonal and upper matrices, the method has the following update rule:

$$x(k+1) = (L+D)^{-1}(b - Ux(k)) \qquad (5)$$

which by taking advantage of the triangular form of $L + D$ can be sequentially updated for each i using forward substitution as

$$x_i(k+1) = \frac{1}{A_{ii}}\left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j(k+1) - \sum_{j=i+1}^{n} A_{ij}x_j(k)\right). \qquad (6)$$

In comparison, the Jacobi method in (4) has the following update for each i

$$x_i(k+1) = \frac{1}{A_{ii}}\left(b_i - \sum_{j \neq i} A_{ij}x_j(k)\right). \qquad (7)$$

Thus, the iteration (6) can be viewed as a form of (7) where the entries of x that have already been updated are being used in subsequent updates instead of using older values. However, the iteration in (6) is sequential which might represent an issue for its implementation. Nevertheless, there are randomized and asynchronous (i.e., where any order of the updates can be followed during each iteration) versions of the Gauss-Seidel iteration, such as the one in [19]. The main conclusion is that each processor can employ the Gauss-Seidel iteration to compute the PageRank using updated values sent by the other processors if the web data does not belong to that processor and use the past iteration values for all the remaining ones.

The next theorem is going to be useful in showing the main result. The proof can be found on page 446 in [20].

Theorem 3 (Jacobi vs. Gauss-Seidel): Suppose a matrix $A \in \mathbb{R}^{n \times n}$ in a general system of linear equations $Ax = b$. If $\forall i : A_{ii} > 0$ and $\forall i \neq j : A_{ij} \leq 0$, then one and only one of the following statements holds:

i) $\rho(T_j) = \rho(T_{gs}) = 0$;
ii) $\rho(T_j) = \rho(T_{gs}) = 1$;
iii) $0 < \rho(T_{gs}) < \rho(T_j) < 1$;

iv) $1 < \rho(T_j) < \rho(T_{gs})$.

One of the main claims of this article is presented in the following result, where the notation T_p identifies the iteration matrix for the power method.

Theorem 4: Denoting by T_{gs} and T_p the matrices corresponding to the discrete-time linear iterations (5) and (2), respectively, we have that

$$\rho(T_{gs}) < \rho(T_p).$$

The following definitions for the iteration matrices are going to be used $T_{gs} = -(L+D)^{-1}U$ and $T_j = -D^{-1}(L+U)$.

Proof: Given Lemma 2, it holds that $T_p = T_j = (1-m)A$.

For the problem in (3), the diagonal matrix $D = I_n$, implying that $\forall i : D_{ii} > 0$. Since $L + U = -(1-m)A$ and that $\forall i, j : A_{ij} \geq 0$, it also holds that all elements in L and U are negative or zero. Therefore, the conditions of Theorem 3 are satisfied. Moreover, matrix $I_n - (1-m)A$ is strictly diagonally dominant [21] and therefore $\rho(T_j) < 1$ and $\rho(T_{gs}) < 1$ by Theorem 2.1 in [20]. As a consequence, using Theorem 3 leads to $0 < \rho(T_{gs}) < \rho(T_j) < 1$, which finalizes the proof. ■

The consequence of Theorem 4 is that solving the PageRank using a Gauss-Seidel iteration is faster than resorting to the power method. As discussed in this section, the proposed algorithm for the PageRank computation is not going to take full advantage of the Gauss-Seidel iteration to prevent the need for synchronization in the updates. In the next section, preliminary simulation results are presented as to illustrate the impact of varying the number of processors and a comparison against the randomized Gauss-Seidel.

Using (6) to solve the PageRank problem has an additional issue. The formulation in (3) is only accurate if the vector x sum remains equal to one since this fact was used to avoid the use of the full matrix S. For the power method, this is kept in all iterations whereas the Gauss-Seidel does not enforce that. The second main claim of this paper is that simulations suggest a speed up in convergence (in some cases a threefold increase which is higher than what can be found in the literature [10]) by using a projection after a complete round of updates from all the processors. Intuitively, the objective is to ensure that equation (3) is the accurate representation of our problem. In the next section, we also simulate the exact projection and a simpler version, and leave for future work to incorporate this projection in the updates.

IV. Simulation results

The main objective of this section is to present preliminary results regarding the various issues related to the proposal of this paper of using the Gauss-Seidel iteration, in a network model of distributed processors, to solve the PageRank problem. The simple example of a four page network in [7] is used to illustrate the need for a projection step to increase the convergence rate. The graph for the interconnection is depicted in Fig. 1. In the simulations using the processor network, a complete graph is assumed with $N = 2$ where p_1 has $\mathcal{R}_1 = \{1, 2\}$ and p_2 with $\mathcal{R}_2 = \{3, 4\}$.

The first simulation illustrates the advantage of the Gauss-Seidel iteration (6) with a n-simplex projection against the power method (2). In this setup, both processors are working in the synchronous problem, which by intuition should be the best performance of the proposed algorithm. It is used $\|(M - I_n)x(k)\|_2$ as a measure for the error and the projection onto the n-simplex defined as

$$\Delta^n := \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, 1, \cdots, n, \mathbf{1}_n^\intercal x = 1\}$$

using the method given in [22].

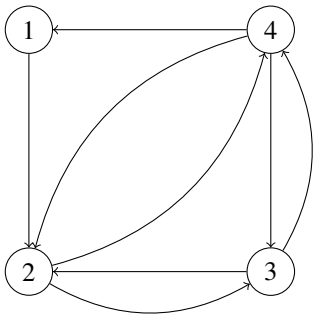

Fig. 1. A network composed of four pages.

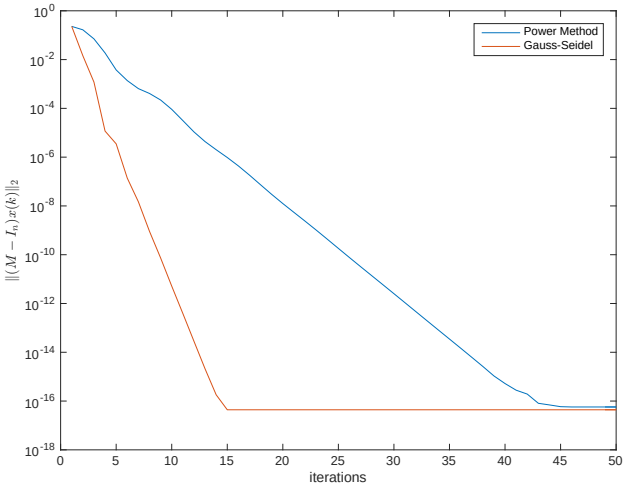

Fig. 2. Evolution of the error of the power method and Gauss-Seidel in a logarithmic scale.

Figure 2 depicts the error of both approaches using a logarithmic scale and when the eps variable in matlab is set to 10^{-16}. There is a substantial improvement in terms of number of iterations, going from around 45 from the power method to 15 using the Gauss-Seidel, thus representing a factor of 3 decrease in the necessary number of iterations for that error level.

An interesting question arose in the previous section regarding the necessity of a projection and also on whether there might be simple algorithms for the case being studied. In Fig. 3, it is presented the error evolution for 3 different cases: the standard simplex projection, a division by the sum of the vector, and no projection. Two main ideas can be inferred from the simulations that are relevant to the problem at hand. First, the projection is a necessary operation given that if the sum of the approximation vector x to the solution $x^\star$ is not 1, the representation in (3) becomes a source

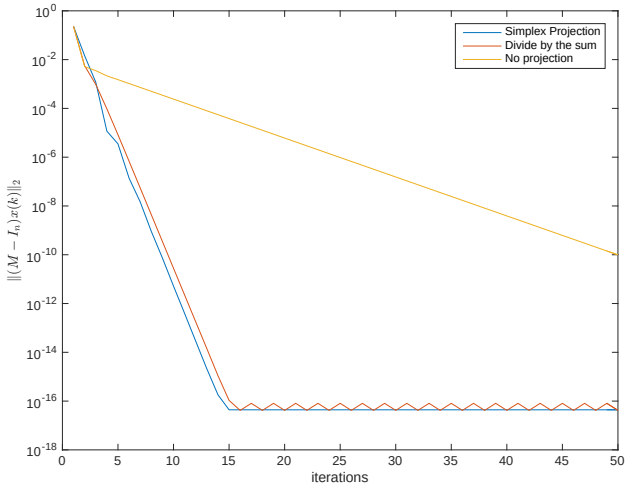

Fig. 3. The impact of the projections on the convergence of the algorithm.

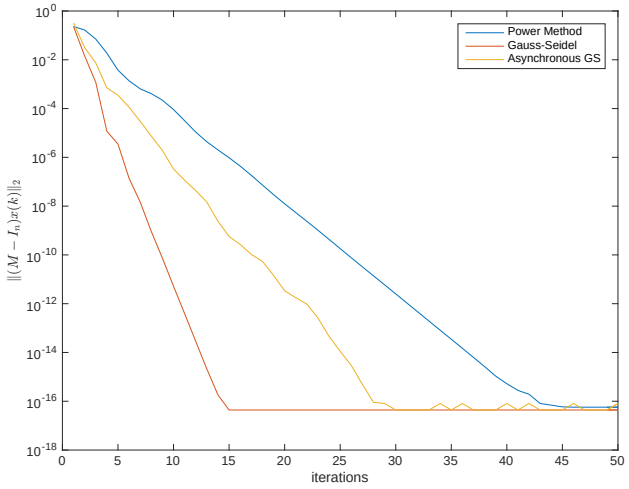

Fig. 4. Sequential Gauss-Seidel against Asynchronous Gauss-Seidel errors.

of error. In those circumstances, the Gauss-Seidel might be beneficial only to solve the problem for a small error tolerance. The second remark is that a simpler to implement projection such as dividing by the sum of the vector has a very similar convergence behavior. This motivates the question for future work of whether the projection step can be avoided by forcing a constant sum to the vector by a modified version of the iteration in (6).

Figure 4 shows the plot for the errors for the sequential and the asynchronous version of the Gauss-Seidel algorithm. In this setup, for each iteration all nodes are updated in an arbitrary fashion following the directions $\mathsf{e}_1, \cdots, \mathsf{e}_n$ selected in a random order. The observed behavior points towards a compromise in performance to allow for a non-sequential algorithm. The exact evolution of the error curve for the Asynchronous Gauss-Seidel depends on the chosen sequences of directions but the simulations seem to point towards a performance that is between the power method and the sequential Gauss-Seidel.

A last setup was designed to study the future possibility

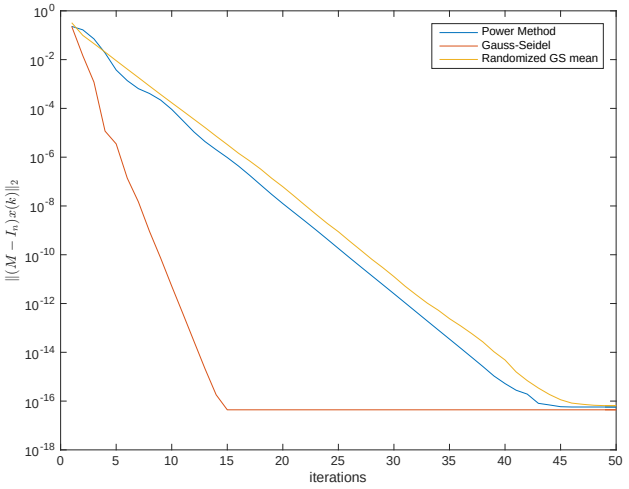

Fig. 5. Sequential Gauss-Seidel against Randomized Gauss-Seidel errors.

of investigating the benefits of a randomized version of the Gauss-Seidel method. Figure 5 depicts the mean value for the error in each iteration for 10^4 random runs of the algorithm. The direction of update is selected based on a uniform discrete distribution. An iteration in this setup is considered a group of n updates regardless of repetition. The mean value for the errors shows that the randomized version suffers in performance when compared to the Asynchronous or Sequential version and follows the same trend of the power method (although underperforming).

The previous example illustrated some of the features of the proposed algorithm. In order to simulate more realistic scenarios, a set of randomly selected network topologies was used. Given that World-Wide Web follows a scale-free network [23], i.e., a topology where the degree distribution of the nodes follows a power law asymptotically, the algorithm of Barabási-Albert (BA) [24] was used to generate the links between pages in a network with $n = 500$ pages with a minimum of 2 outgoing links. In this scenario, 4 different methods are simulated for 15 iterations (it is enough to have a convergence with error smaller than 10^{-4}), namely: power method, standard Gauss-Seidel, Gauss-Seidel with the simplex projection, and Gauss-Seidel with the normalization. A 1000 points monte carlo run is considered and the means and standard deviations over time are reported.

Figure 6 depicts the evolution of the mean error for all four algorithms. One of the main points of interest is that the Gauss-Seidel algorithm is faster to converge on average as expected by the results in this paper. The second feature is that it converges faster to an *acceptable* accuracy of the page rank $x^\star$ although slowing down for very small error requirements (under 10^{-3}). For this simulation, the approximation of the projection by the normalization was not very successful, obtaining mean errors with a negligible improvement. One of the topics of future research will be on understanding what exactly impacts the quality of such an approximation and, if possible, what other approximations are useful.

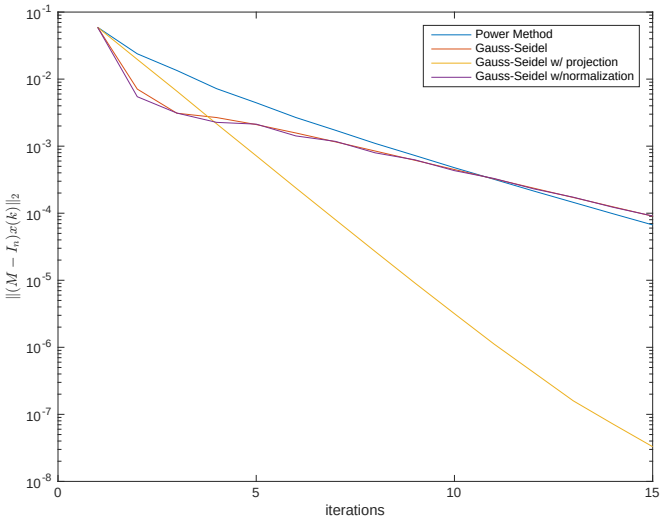

Fig. 6. Mean error for each of the algorithms for the 1000 point monte carlo simulation on the random scale-free network topologies.

Lastly, this larger simulation confirmed the main trend found for the toy example, specifically that the addition of the projection method has a major impact on the convergence speed of the algorithm. Interestingly, in the initial iterations (the first two), the standard Gauss-Seidel outperform the proposed method and a more detailed investigation about the reasons that cause that might benefit in developing heuristic-based algorithms that might switch between different version of the algorithm depending on some structural property of the topology.

V. Discussion and Future Work

In this paper, the PageRank problem was addressed as the solution of a system of linear equations. The main issue arising in this formulation is the fact that either the problem involving the altered M matrix is considered, in which case the sparsity of communication is lost; or, it is needed a method that maintains the sum of the solution vector equal to one. The first step in our study was to show that the power method, when applied to the PageRank, coincides with the Jacobi method for the correspondent system of linear equations. That suggest that other methods can be implemented that are considerably faster than the Jacobi iteration. The sequential version of the Gauss-Seidel algorithm was shown to always be faster and simulations corroborated that result. Other setups including the asynchronous and randomized version were studied in order to have a comparison with the power method performance.

Simulations indicate that the need for the exact projection on the n-simplex can be relaxed and a normalization of the sum can be performed instead on some cases. Such observation motivates the idea that it can be possible to incorporate the projection on the iteration itself and thus obtaining an algorithm with the same type of communication constraints as the power method but considerably faster.

References

[1] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107 – 117, 1998.

[2] M. Franceschet, "Pagerank: Standing on the shoulders of giants," *Communications of the ACM*, vol. 54, no. 6, pp. 92–101, Jun. 2011.

[3] A. N. Langville and C. D. Meyer, "Deeper inside pagerank," *Internet Mathematics*, vol. 1, no. 3, pp. 335–380, 2004.

[4] K. Bryan and T. Leise, "The $25,000,000,000 eigenvector: The linear algebra behind google," *SIAM Review*, vol. 48, no. 3, pp. 569–581, 2006.

[5] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011.

[6] K. You, R. Tempo, and L. Qiu, "Randomized incremental algorithms for the pagerank computation," in *54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 431–436.

[7] H. Ishii and R. Tempo, "Distributed randomized algorithms for the pagerank computation," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 1987–2002, Sept 2010.

[8] C. Ravazzi, P. Frasca, R. Tempo, and H. Ishii, "Ergodic randomized algorithms and dynamics over networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 1, pp. 78–87, March 2015.

[9] W. Zhao, H. F. Chen, and H. T. Fang, "Convergence of distributed randomized pagerank algorithms," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3255–3259, Dec 2013.

[10] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin, "Pagerank computation and the structure of the web: experiments and algorithms," in *In The Eleventh International WWW Conference*. ACM Press, 2002.

[11] H. Ishii, R. Tempo, and E. W. Bai, "A web aggregation approach for distributed randomized pagerank algorithms," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2703–2717, Nov 2012.

[12] H. Ishii and R. Tempo, "The pagerank problem, multiagent consensus, and web aggregation: A systems and control viewpoint," *IEEE Control Systems*, vol. 34, no. 3, pp. 34–53, June 2014.

[13] A. N. Langville and C. D. Meyer, "Updating pagerank with iterative aggregation," in *13th International World Wide Web Conference on Alternate Track Papers & Posters*, ser. WWW Alt. '04. New York, NY, USA: ACM, 2004, pp. 392–393.

[14] C. Kohlschütter, P.-A. Chirita, and W. Nejdl, "Efficient parallel computation of pagerank," in *ECIR*, vol. 3936. Springer, 2006, pp. 241–252.

[15] J. Lei and H. F. Chen, "Distributed randomized pagerank algorithm based on stochastic approximation," *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1641–1646, June 2015.

[16] P. Boldi, M. Santini, and S. Vigna, "Pagerank: Functional dependencies," *ACM Transactions on Information Systems*, vol. 27, no. 4, pp. 19:1–19:23, Nov. 2009.

[17] S. Kamvar, T. Haveliwala, and G. Golub, "Adaptive methods for the computation of pagerank," *Linear Algebra and its Applications*, vol. 386, no. Supplement C, pp. 51 – 65, 2004, special Issue on the Conference on the Numerical Solution of Markov Chains 2003.

[18] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, Nov 2015.

[19] H. Avron, A. Druinsky, and A. Gupta, "Revisiting asynchronous linear solvers: Provable convergence rate through randomization," *J. ACM*, vol. 62, no. 6, pp. 51:1–51:27, Dec. 2015.

[20] A. Ralston and P. Rabinowitz, *A first course in numerical analysis*. Courier Corporation, 2001.

[21] B. C. Csáji, R. M. Jungers, and V. D. Blondel, "Pagerank optimization by edge selection," *Discrete Applied Mathematics*, vol. 169, pp. 73 – 87, 2014.

[22] Y. Chen and X. Ye, "Projection Onto A Simplex," *ArXiv e-prints*, Jan. 2011.

[23] A.-L. Barabási, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: the topology of the world-wide web," *Physica A: Statistical Mechanics and its Applications*, vol. 281, no. 1, pp. 69 – 77, 2000.

[24] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002.