ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2015, Vol. 55, No. 3, pp. 349–365. © Pleiades Publishing, Ltd., 2015.
Original Russian Text © A.V. Gasnikov, D.Yu. Dmitriev, 2015, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2015, Vol. 55, No. 3, pp. 355–371.

On Efficient Randomized Algorithms for Finding the PageRank Vector

A. V. Gasnikov and D. Yu. Dmitriev

Moscow Institute of Physics and Technology, Institutskii per. 9, Dolgoprudnyi, Moscow oblast, 141700 Russia
Institute for Information Transmission Problems, Russian Academy of Sciences, Bolshoi Karetnyi per. 19, str. 1, Moscow, 127051 Russia
e-mail: gasnikov@yandex.ru, dmitden@gmail.com

Received September 3, 2014

Abstract—Two randomized methods are considered for finding the PageRank vector; in other words, the solution of the system $\mathbf{p}^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}}P$ with a stochastic $n \times n$ matrix P, where $n \sim 10^7$–10^9, is sought (in the class of probability distributions) with accuracy ε: $\varepsilon \gg n^{-1}$. Thus, the possibility of brute-force multiplication of P by the column is ruled out in the case of dense objects. The first method is based on the idea of Markov chain Monte Carlo algorithms. This approach is efficient when the iterative process $\mathbf{p}_{t+1}^{\mathrm{T}} = \mathbf{p}_t^{\mathrm{T}}P$ quickly reaches a steady state. Additionally, it takes into account another specific feature of P, namely, the nonzero off-diagonal elements of P are equal in rows (this property is used to organize a random walk over the graph with the matrix P). Based on modern concentration-of-measure inequalities, new bounds for the running time of this method are presented that take into account the specific features of P. In the second method, the search for a ranking vector is reduced to finding the equilibrium in the antagonistic matrix game

$$\min_{\mathbf{p} \in S_n(1)} \max_{\mathbf{u} \in S_n(1)} \langle \mathbf{u}, (P^{\mathrm{T}} - I)\mathbf{p} \rangle,$$

where $S_n(1)$ is a unit simplex in $\mathbb{R}^n$ and I is the identity matrix. The arising problem is solved by applying a slightly modified Grigoriadis–Khachiyan algorithm (1995). This technique, like the Nazin–Polyak method (2009), is a randomized version of Nemirovski's mirror descent method. The difference is that randomization in the Grigoriadis–Khachiyan algorithm is used when the gradient is projected onto the simplex rather than when the stochastic gradient is computed. For sparse matrices P, the method proposed yields noticeably better results.

DOI: 10.1134/S0965542515030069

Keywords: mirror descent method, Markov chain Monte Carlo, stochastic optimization, randomization, PageRank.

1. INTRODUCTION

It is well known that the Google Search engine was created as a student research project at Stanford University (see [1]). In 1996, they worked over the BackRub search engine, which was used to create the new Google Search engine in 1998 (see [2, 3]). A certain method for ranking web pages was proposed in [1]. This method, as well as the rather large class of ranking problems arising, for example, in the computation of citation indices of researchers or journals [4] is reduced (see [5]) to finding the left eigenvector (normalized to unity: $\sum_{k=1}^{n} p_k = 1$) corresponding to the eigenvalue 1 of some stochastic (in rows) matrix $P = \|p_{ij}\|_{i,j=1}^{n,n}$: $\mathbf{p}^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}}P$.

Remark 1. The consensus-finding problem is reduced (for example, in the DeGroot model) to finding such a vector, which is sometimes called the Perron–Frobenius vector. A more detailed exposition of this subject and consensus models overall can be found in [6].

Remark 2. The solution of $\mathbf{p}^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}}P$ always exists by the Brouwer fixed-point theorem (the continuous (bounded) operator P maps a convex compact set (simplex) to itself) and is unique in the class of probability distributions if and only if there is a single class of interdependent (essential) states in the possible presence of nonessential states [7]. In other words, if the matrix P is associated with a directed graph such that the vertices i and j are joined by an edge if and only if $p_{ij} > 0$, then any vertex in this graph is of one of the following two types: nonessential (if, starting from this vertex and moving along edges according to

their orientations, we can always arrive at a vertex from which there is no path back) and essential (if, starting from any such vertex, we can reach any other essential vertex, including returning to the original vertex).

This method can be substantiated following [5]. Let $G = \langle V, E\rangle$ be a directed graph representing the Internet (vertices are web pages, edges are links, and the notation $(i, j) \in E$ means that, on the ith page, there is a link to the jth page), and let N be the number of Internet users (which does not vary with time). Let $n_i(t)$ be the number of visitors to web page i at the time t. Over one time cycle, each visitor to this web page goes independently (using the link) to the web page j with probability p_{ij}. The stochastic matrix P is assumed to be indecomposable and aperiodic (see [7]). Below, we present a result from [5] (which is substantiated in Section 4) that offers another interpretation of the vector $\mathbf{p}$ (PageRank), according to which web pages are ranked:

$$\exists \lambda_{0.99} > 0, \quad T > 0: \quad \forall t \geq T,$$

$$P\left(\left\|\frac{\mathbf{n}(t)}{N} - \mathbf{p}\right\|_2 \leq \frac{\lambda_{0.99}}{\sqrt{N}}\right) \geq 0.99,$$

where $\mathbf{p}^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}} P$ (the solution is unique in the class of probability distributions since the matrix is indecomposable).

In some cases, it is assumed that

$$P = (1 - \delta) I + \delta \tilde{P},$$

where $\delta \in (0, 1]$, $I = \mathrm{diag}\{1, \ldots, 1\}$ is the identity matrix, and

$$\tilde{p}_{ij} = \begin{cases} |\{k : (i,j) \in E\}|^{-1}, & i \neq j, \\ 0 \quad \text{otherwise.} \end{cases}$$

This specific property of P will be used in Section 4. Note also that I is frequently replaced by a stochastic matrix with identical elements, namely, by a teleportation matrix (see [2]). This immediately yields the lower bound δ on the spectral gap of P.

Let us briefly describe the structure of this paper. Section 2 gives an overview of most popular numerical methods for finding the PageRank vector. The novelty is that the overview is made simultaneously with discussing two new methods for finding the PageRank vector: one based on the Markov chain Monte Carlo (MCMC) algorithm and the other based on the Grigoriadis–Khachiyan algorithm. In Section 3, which is also an overview, we briefly describe the basic MCMC-related results necessary in what follows. There are no new results in this section. Nevertheless, the literature review presented can be of interest. In Section 4, the MCMC method is used to find the PageRank vector. New results are convergence rate estimates for the method and estimates for the total number of arithmetic operations. It should be emphasized that, in contrast to the general case, the efficiency of MCMC is analyzed for a special class of problems. As a result, we obtain substantially better estimates than those expected in the general case. Another novelty of the method consists in the organization of random walks. Note that we propose a well parallelizable version of the MCMC method. The MCMC method described in Section 4 performs well if the spectral gap of P is large enough. A new method for finding the PageRank vector without imposing any constraints on the spectral gap is proposed in Section 5. According to this method, finding a ranking vector is reduced to finding a Nash equilibrium in an antagonistic matrix game. The equilibrium is found by applying the Grigoriadis–Khachiyan algorithm, which takes into account the sparsity of P and is well parallelizable.

2. OVERVIEW AND DISCUSSION OF PREVIOUSLY KNOWN AND NEW METHODS FOR NUMERICALLY FINDING THE PAGERANK VECTOR

Various methods for numerically finding the PageRank vector $\mathbf{p}$ were proposed in [8–15].

Remark 3. Strictly speaking, other problems were solved in [11, 12], but the algorithms proposed are easy to extend to the problem of finding the PageRank vector: this is straightforward in the case of [11], but, in the case of [12], a somewhat more accurate convergence analysis yields the correction factor $\ln(\sigma^{-1})$, which appears because the expectation is eliminated from the cost (objective) functional.

Below is a list of complexity bounds for the algorithms from [8–15] and for the algorithms proposed in this paper. By complexity, we mean the number of arithmetic operations (multiplications of two numbers)

Table

Method	Condition	Complexity	Objective (min)
Nazin, Polyak [8]	No	$O\left(\frac{n\ln(n/\sigma)}{\varepsilon^2}\right)$	$\left\|P^T\mathbf{p}-\mathbf{p}\right\|_2^2$
Nesterov [9, 10]	$\bar{S}$	$O\left(\frac{s^2\ln n}{\varepsilon^2}\right)$	$\max\limits_{\mathbf{u}\in S_n(1)}\langle\mathbf{u}, P^T\mathbf{p}-\mathbf{p}\rangle$
Juditsky, Lan, Nemirovski, Shapiro [11]	No	$O\left(\frac{n\ln(n/\sigma)}{\varepsilon^2}\right)$	$\max\limits_{\mathbf{u}\in S_n(1)}\langle\mathbf{u}, P^T\mathbf{p}-\mathbf{p}\rangle$
Grigoriadis, Khachiyan [12]	$\bar{S}$	$O\left(\frac{s\ln n\ln(n/\sigma)}{\varepsilon^2}\right)$	$\max\limits_{\mathbf{u}\in S_n(1)}\langle\mathbf{u}, P^T\mathbf{p}-\mathbf{p}\rangle$
Nesterov, Nemirovski [13]	*G*, *S*	$\frac{sn}{\alpha}\ln\left(\frac{2}{\varepsilon}\right)$	$\left\|\mathbf{p}-\mathbf{p}_*\right\|_1$
Polyak, Tremba [14]	*S*	$\frac{2sn}{\varepsilon}$	$\left\|P^T\mathbf{p}-\mathbf{p}\right\|_1$
Spielman [15]	*G*, *S*	$O\left(\frac{s^2}{\alpha\varepsilon}\ln\left(\frac{1}{\varepsilon}\right)\right)$	$\left\|\mathbf{p}-\mathbf{p}_*\right\|_\infty$
MCMC	*SG*	$O\left(\frac{\ln n\ln(n/\sigma)}{\alpha\varepsilon^2}\right)$	$\left\|\mathbf{p}-\mathbf{p}_*\right\|_2$

that are required to produce a solution with probability no smaller than $1-\sigma$ that is accurate up to ε in terms of the objective functional.

Let us explain the notation used.

The *G-condition* means that there is a web page (for example, one corresponding to the Google Search itself) that can be reached from any another web page with probability no smaller than $\alpha > cn^{-1}$. Without loss of generality, we assume that the vertex corresponding to this web page has an index of 1 (see the MCMC algorithm).

The *S-condition* means that every web page has at most $s \ll n$ links to other pages, i.e., the matrix P has the sparsity property. If every web page has at most $s \ll n$ outgoing and incoming links, then we talk about the $\bar{S}$-condition.

Remark 4. In some cases (for example, for the method from [14]), the *S*-condition can be relaxed to "go out on average."

The *SG-condition* means that the spectral gap α of the matrix P satisfies the condition $\alpha \gg n^{-2}$, where α is the distance between the maximum eigenvalue in absolute value (Perron–Frobenius number) in the matrix P (equal to 1) and the absolute value of the next (in magnitude) eigenvalue. If the *G*-condition is satisfied, then the *SG*-condition holds with α no smaller than in the *G*-condition (see [2, 15]).

Note that the table slightly coarsens the results of the quoted works, specifically, of [9, 10]. This was done for illustrative purposes. Note also that the complexity bounds for the algorithms from [9, 10] are strongly overstated. In practice, these algorithms perform noticeably better. The algorithm from [8], on the contrary, performs not very quickly because of the large constant in $O(\cdot)$. A somewhat similar situation occurs with the algorithms from [11, 12]. They are not so fast as would be expected. This is also associated with the fact that the possibility of working with randomness in huge dimensions is rather poorly implemented in standard packages (we used MatLab). The algorithms from [13, 14] and MCMC work exactly as indicated in the bounds. The algorithm from [15] performs noticeably better that the complexity bound presented (here, we do not mean the constant in $O(\cdot)$). Nevertheless, the conditions under which this algorithm is competitive are rather restrictive and, even under these conditions, the method is generally outperformed by that from [13]. A practical analysis of all algorithms presented in the table (and some others) has shown that the methods from [12, 14] are efficient when there is no guarantee that the spectral gap is

not small. If the latter condition can be ensured, then the MCMC method performs quite well, provided that the randomization module is tailor-made rather than based on standard methods for discrete random number generation.

It should be noted that different algorithms have different goals. In this context, it is useful to mention the "typical" relations (see [16])

$$\|\cdot\|_1 \sim \sqrt{n}\|\cdot\|_2, \quad \|\cdot\|_2 \sim \sqrt{n}\|\cdot\|_\infty$$

for vectors with identical (in order) components. In our case, they are not typical (the components decrease according to the power law [17]), so it can be expected that the bounds presented involve a dependence weaker than $\sqrt{n}$, i.e., in the transition from one norm to the other, the factor $\sqrt{n}$ is replaced by something closer to $O(1)$. However, this observation has not been rigorously proved.

According to the classification in [18, Chapter 6], the methods from [8, 9, 10] are variational, i.e., the system $\mathbf{p}^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}}P$ in these methods is reduced to convex optimization problems, which are solved by applying various gradient descent algorithms (randomized mirror descent, the Polyak–Shor method, randomized componentwise descent).

The methods from [11, 12] are variational, but with a game aspect (an optimization problem is understood as finding a saddle point, i.e., a Nash equilibrium in a matrix game). Indeed, the problem under consideration can be rewritten (with the help of Perron–Frobenius theory [19]) as the problem

$$f(\mathbf{p}) = \max_{\mathbf{u} \in S_n(1)} \langle \mathbf{u}, A\mathbf{p} \rangle \longrightarrow \min_{\mathbf{p} \in S_n(1)},$$

where

$$A = P^{\mathrm{T}} - I, \quad A = \|a_{ik}\|, \quad S_n(1) = \left\{ \mathbf{p} \geq 0 : \sum_{k=1}^{n} p_k = 1 \right\}.$$

Note that $0 \leq f(\mathbf{p}) \leq \|A\mathbf{p}\|_\infty$, $\mathbf{p} \in S_n(1)$, and $f(\mathbf{p}) = 0$ for the PageRank vector (and only for it). In fact, this objective functional $f(\mathbf{p})$ turns out to be very close to $\|P^{\mathrm{T}}\mathbf{p} - \mathbf{p}\|_\infty$. Both game methods are underlain by randomized versions of mirror descent as applied to finding an equilibrium in matrix games.

The methods from [8, 11, 12] can be parallelized on $\sim\log_2(1/\sigma)$ processors. Specifically, $\sim\log_2(1/\sigma)$ independent trajectories are computed in parallel by any of these methods. Using tight bounds for the number of iterations that guarantee prescribed accuracy and confidence level, which is set equal to 1/2 in this parallel scheme, the computation of the trajectories is terminated after executing the required number of iterations and a smallness criterion for the objective functional is checked. With probability $1 - \sigma$, at least one of the trajectories yields a solution with the prescribed accuracy. Unfortunately, the verification of the smallness criterion for the objective functional involves the multiplication $P^{\mathrm{T}}\mathbf{p}$, which can be more expensive than finding a candidate solution. Accordingly, such parallelization is not always reasonable. Moreover, the verification of the smallness criterion requires that the number of iterations guaranteeing the prescribed accuracy and confidence level be accurately estimated (see, e.g., the estimate on the number of iterations in the theorem in Section 5). Note that the method from [12] without allowance for sparsity can be parallelized on $n/\ln n$ processors (see [12]).

The other methods in the table can be interpreted as versions of the stationary iteration method from [18, 20]. The following question arises: why not try to solve the system $\mathbf{p}^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}}P$ with a well-known method, for example, from [18]? For example, we could use (see [21]) $\mathbf{p}^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}}(0)P^\infty$, where P^∞ is the output of the next iterative process, which requires a finite number of iterations:

$$P(0) = I, \quad P(k) = I - k\frac{P(k-1)(I-P)}{\mathrm{tr}(p(k-1)(I-P))}, \quad k \in \mathbb{N}.$$

The iteration terminates as soon as $\mathrm{tr}(P(m)(I-P)) = 0$, where $\mathrm{tr}(B)$ is the trace of the matrix B, i.e., the sum of its diagonal elements. Moreover, $P^\infty = P(m)$.

Let us try to answer this question. First, many iterative methods require that the matrix P be symmetric and positive definite, which is generally not the case. However, even if all the required conditions hold and the matrix is additionally sparse, the convergence is the same as in [13] but the constant α is replaced by a certain analogue (from the *SG*-condition), which is generally much more difficult to estimate. Thus, instead of presenting a number of classical iterative methods with their estimates, we give only those of them (their applicability conditions are well interpreted) that are the best adapted to the (rather narrow)

class of problems under study. The method from [21, 22] cannot be used, because $n \gg 10^4$ (i.e., $n^3 \gg 10^{12}$ is the number of arithmetic operations that can be executed on a personal computer over one hour).

In connection with the stationary iteration mention, we note that the methods described lead to the accumulation of roundoff errors (finite length of the mantissa) when $\|P\| > 1$ (see, e.g., [18, Chapter 6, Section 4; 20, Chapter 5, Section 1]). (Moreover, the spectrum of the matrix may lie in the unit disk and, theoretically (without roundoff errors), the stationary iteration must converge exponentially). However, in the case under consideration, in the natural norm subordinated to l_1^n, $\|P\| = 1$, and no such problems arise.

Importantly, in some cases considered in the table, the goal is to obtain a probability distribution vector $\boldsymbol{p}$ that provides a small residual in terms of the function. However, if $\|P^{\mathrm{T}}\mathbf{p} - \mathbf{p}\|$ is small, this does not mean that $\|\mathbf{p} - \mathbf{p}_*\|$ is small as well.

Remark 5. Recall that the vector $\mathbf{p}_*$ is the solution of the equation $P^{\mathrm{T}}\mathbf{p} = \mathbf{p}$. This solution is assumed to be unique in the class of probability distributions.

On the contrary, it is quite natural to expect that $\|\mathbf{p} - \mathbf{p}_*\|$ is α^{-1} times as large (see [18, 23]). Although this is an upper estimate, it seems that, with a high probability (if the starting point is chosen equiprobably), this estimate turns into a nearly exact relation (see [23, 24]). Thus, ensuring that $\|P^{\mathrm{T}}\mathbf{p} - \mathbf{p}\|$ is small does not mean that we completely found the PageRank vector. This circumstance and also the fact that the solution is sought on a strongly constrained set (unit simplex) and we make additional simplifying assumptions explain in part why the complexity of the algorithms described is so low. For example, if $\|\mathbf{p} - \mathbf{p}_*\|_\infty \le \varepsilon$ for $\varepsilon \gg n^{-1}$ and the true distribution of $\mathbf{p}_*$ is close to a uniform one, it is not clear what the algorithm produces with accuracy ε in terms of the function and how this can be used to rank web pages. Fortunately, situations when the true distribution of $\mathbf{p}_*$ is close to a uniform one are not typical in practice (see [26]), especially if the *G*-condition (*SG*-condition) holds. As a rule, we select components of $\mathbf{p}_*$ that are sufficiently large. Since the problem is stated as one of ranking web pages, in fact, we need to reconstruct the first few (in magnitude) components of $\mathbf{p}_*$. In other words, there is no need to precisely recover small components of $\mathbf{p}_*$ if we know that they are sufficiently small and there are a sufficient number large components. We can say that the vector produced by the algorithm [15] has at most $3\varepsilon^{-1}$ nonzero components. These circumstances explain why a vector $\mathbf{p}$ providing the estimate $\|P^{\mathrm{T}}\mathbf{p} - \mathbf{p}\|_\infty \le \varepsilon$ in one of the "poorest" norms l_∞^n can be useful when $\varepsilon \gg n^{-1}$.

3. MARKOV CHAIN MONTE CARLO METHOD

The idea of solving linear equations with the help of MCMC is as old as the usual Monte Carlo method from [25]. However, we consider not a general linear equation but rather an equation with a stochastic matrix P that is populated in a special way. Due to these circumstances, a random walk on the graph corresponding to P can be organized more efficiently.

Before describing the algorithm, we present some auxiliary facts.

Knuth–Yao Algorithm (see [25, 27])

The task is, by tossing a symmetric coin, to generate the distribution of a given discrete random variable taking a finite number of values. Suppose that we need to generate the distribution of a random variable taking three values, 1, 2, and 3, with equal probabilities 1/3. This can be done as follows. The coin is tossed twice: if the outcome is 00, we set its value to 1; if the outcome is 01, we set 2; and, if the outcome is 11, we set 3. If the outcome is 10, then the coin is again tossed twice and the argument is repeated. It can be shown that this scheme can be generalized so that the distribution of a discrete random variable taking n different values with generally different probabilities can be generated on average with the help of at most $\log_2(n-1) + 2$ tosses of a symmetric coin. If these probabilities are identical, then the procedure for "preparing" such a generation algorithm also has logarithmic complexity in n.

MCMC Algorithm (see [7, 28–36])

To construct a homogeneous discrete Markov process with a finite number of states with a prescribed invariant (stationary) measure π, the transition probabilities are sought in the form

$$p_{ij} = p^0_{ij} b_{ij}, \quad i \neq j; \quad p_{ii} = 1 - \sum_{j:\, j \neq i} p_{ij},$$

where p^0_{ij} is a preliminary matrix, which is hereafter assumed to be symmetric. It is easy to see that the matrix p_{ij} has an invariant (stationary) measure π:

$$\frac{b_{ij}}{b_{ji}} = \frac{\pi_j p^0_{ji}}{\pi_i p^0_{ij}} = \frac{\pi_j}{\pi_i}, \quad p^0_{ij} > 0.$$

To find b_{ij}, it is sufficient to find a function $F\colon \mathbb{R}_+ \longrightarrow [0, 1]$ such that

$$F(z)/F(1/z) = z$$

and to set

$$b_{ij} = F\left(\frac{\pi_j p^0_{ji}}{\pi_i p^0_{ij}}\right) = F\left(\frac{\pi_j}{\pi_i}\right).$$

Perhaps, the best known example of such a function is $\tilde{F}(z) = \min\{z, 1\}$, i.e., the Metropolis–Hastings algorithm [7]. Note that, for any such function $F(z)$, we have $F(z) \leq \tilde{F}(z)$. Another example is the function $F(z) = z/(1 + z)$. Note also that p^0_{ij} is usually chosen equal to $p^0_{ij} = 1/M_i$, where M_i is the number of "neighboring" states for i, or

$$p^0_{ij} = 1/(2M), \quad i \neq j, \quad p^0_{ii} = 1/2, \quad i \neq j.$$

At long times t, by the ergodic theorem, the probability distribution is close to the stationary one π. Indeed, under the conditions described above, we have the detailed balance condition (Markov chains satisfying this condition are sometimes called reversible):

$$\pi_i p_{ij} = \pi_j p_{ji}, \quad i, j = 1, \ldots, n,$$

which implies the invariance of π, i.e.,

$$\sum_i \pi_i p_{ij} = \pi_j \sum_i p_{ji} = \pi_j, \quad j = 1, \ldots, n.$$

The basic application of this fact consists in the observation that, in many cases, the time required for reaching a stationary measure in a Markov process (i.e., mixing time [37]) is surprisingly short.

Remark 6. Moreover, the problem of finding a symmetric random walk (of uniform invariant measure because of the symmetry) on a graph of given structure that has the "least" mixing time (in other words, the largest spectral gap) is reduced to a semidefinite programming problem, which is known to be solvable in polynomial time (in the number of vertices of this graph) [38].

Note that, according to the Knuth–Yao algorithm, one time step of a random walk on the graph corresponding to the considered Markov chain can also be quickly performed. Thus, an efficient method for generating the distribution of a discrete random variable with probability distribution π can frequently be obtained in time polynomial in the logarithm of the number of components in the vector π.

To estimate mixing time, we need to estimate the spectral gap of the stochastic transition probability matrix specifying the Markov dynamics under study, i.e., we have to estimate the distance from the maximum eigenvalue of this matrix equal to 1 (Perron–Frobenius theorem) to the next in value modulus. It is this number that determines the common ratio of the geometric progression majorizing the studied sequence of norms of differences of distances (in variation) between the distribution at a given time and the stationary (final) distribution. Numerous methods are available for spectral gap estimation. Examples (see [29–36]) are the Poincaré inequality (canonical way), Cheeger's isoperimetric inequality (conductivity), the coupling technique [39] (which yields simple, but, as a rule, rather rough estimates), Matron

coupling [40], and the coarse Ricci curvature and concentration-of-measure theorems (of Milman and Gromov [36, 41]). Examples of MCMC applications (see [7]) include shuffling n cards by partitioning them into two roughly equal portions and mixing them (mixing time $\sim\log_2 n$); Hit and Run (mixing time $\sim n^3$); the Ising model: n spins on an interval, a stationary distribution, i.e., the Gibbs distribution, Glauber dynamics (mixing time $\sim n^{2\log_2 e/T}$, $0 < T \ll 1$); the problem for finding the shortest Hamiltonian path; simulated annealing for combinatorial optimization problems; and MCMC for enumerative combinatorics problems.

Remark 7. Perhaps, the contrast is most pronounced in the example of card shuffling. For a deck of 52 cards, the state space of a Markov chain has cardinality 52! (adding the life times in nanoseconds of all people who ever lived on Earth yields a number less than 52! by many orders of magnitude). At the same time, shuffling in which the card from the top of the deck is placed at random inside the deck according to a certain random walk, with very good accuracy, reaches a uniform measure corresponding to an intermixed deck after 200–300 steps. In the case of shuffling by partitioning into portions, the same can be achieved even more quickly after 8–10 steps (see [44]).

Let us illustrate what was said above by two examples (which were used, for example, to obtain the mixing time bounds in [42]), which will be useful in what follows.

Example 1 (Cheeger's approach [30, 33, 35]). Recalling that $\pi(\cdot)$ is an invariant measure and $P = \|p_{ij}\|_{i,j=1}^{n,n}$ is the Markov-chain transition probability matrix, let

$$h(G) = \min_{S\subseteq V_G:\,\pi(S)\le 1/2} P(S \longrightarrow \bar{S}\,|\,S) = \min_{S\subseteq V_G:\,\pi(S)\le 1/2} \frac{\sum\limits_{(i,j)\in E_G:\, i\in S, j\in \bar{S}} \pi(i)p_{ij}}{\sum\limits_{i\in S}\pi(i)} \quad \text{(Cheeger constant)},$$

$$T(i,\varepsilon) = \Theta(h(G)^{-2}(\ln(\pi(i)^{-1}) + \ln(\varepsilon^{-1}))) \quad \text{(mixing time)}.$$

Then

$$t \ge T(i,\varepsilon) \longrightarrow \left\|P^t(i,\cdot) - \pi(\cdot)\right\|_1 = \sum_{j=1}^{n}\left|P^t(i,j) - \pi(j)\right| \le \varepsilon \qquad \forall i = 1,\ldots,n,$$

where $P^t(i,j)$ is the conditional probability that, starting from the state i, the Markov process reaches the j after t steps. Note that a reasonable assumption is that the components of the PageRank vector π ranked in increasing order decrease according to the power law (see [26]). Therefore,

$$\max_{i=1,\ldots,n} \ln(\pi(i)^{-1}) = O(\ln n).$$

Example 2 (coarse Ricci curvature [36]). The Monge–Kantorovich distance between two (discrete) probability distributions μ and ν is defined as

$$W_1(\mu,\nu) = \min_{\substack{\xi\ge 0:\ \sum_j \xi(i,j) = \mu(i)\\ \sum_i \xi(i,j) = \nu(j)}} \sum_{i,j} d(i,j)\xi(i,j),$$

where each pair of vertices is assigned a nonnegative number $d(i,j)$ (with distance (metric) properties). We say that κ is a coarse Ricci curvature if

$$\exists t_0 > 0 : \forall i,j = 1,\ldots,n,$$

$$W_1(P^{t_0}(i,\cdot), P^{t_0}(j,\cdot)) \le (1-\kappa)d(i,j).$$

Suppose that there is such $\kappa > 0$. Then

$$W_1(P^t(i,\cdot), \pi(\cdot)) \le \left[\kappa^{-1}\sum_{j=1}^{n} d(i,j)p_{ij}\right](1-\kappa)^{1/t_0},$$

$$E\left[\left\|\frac{1}{T}\sum_{t=T_0}^{T+T_0}\mathbf{x}_t-\boldsymbol{\pi}\right\|_2^2\right]=(\text{Bias})^2+\text{Var}=\left(O\left(n\frac{(1-\kappa)^{T_0}}{\kappa T}\right)\right)^2+O\left(\frac{1}{\kappa T}\right),$$

where $\mathbf{x}_t$ is a random vector with all zero components, except for the only one corresponding to the vertex at which the Markov stochastic process was at the step t. Assuming that

$$\text{Var}\gg(\text{Bias}),$$

we have

$$\exists c>0:P\left(\left\|\frac{1}{T}\sum_{t=T_0}^{T+T_0}\mathbf{x}_t-\boldsymbol{\pi}\right\|_2>c\sqrt{\frac{\ln n+\Omega}{\kappa T}}\right)\le\exp(-\Omega)$$

(this result is not contained in explicit form in [36], but it can be derived using some ideas of [40]).

The result

$$\frac{1}{T}\sum_{t=T_0}^{T+T_0}\mathbf{x}_t\xrightarrow[T\to\infty]{}\boldsymbol{\pi}$$

is the ergodic theorem for Markov processes (a rather delicate convergence rate estimate was given above). This result is quite expected (see [14]) and corresponds to classical ergodic theorems for dynamical systems (Birkhoff–Khinchin, von Neumann). However, in contrast to dynamical systems, the convergence rate can be estimated in this case.

Note that the approach described in Example 2 is related to a result stating that positive linear operators (in the space of all possible rays centered at the origin in the nonnegative orthant) are contractions in the Birkhoff metric (see [43]); specifically, these operators can be defined by a stochastic matrix P such that some of its powers has all elements positive (this is equivalent to the Markov chain being indecomposable and nonperiodic (see [44])). Incidentally, relying on this understanding of the ergodic theorem for Markov chains, we can interpret it as a contraction mapping theorem, which is somewhat unusual in the context of comparing this theorem to ergodic theorems for dynamical systems.

4. MCMC ALGORITHM

Now we can describe the required version of the MCMC algorithm.

Step 1. Initialization: $\mathbf{X}=\mathbf{0}$, vertex $=1$, $t=0$.

Step 2. Iteration counter: $t:=t+1$.

Step 3. Modification of $\mathbf{X}$: if $t\ge T^0_{\varepsilon,\alpha,n}$, then $X_k:=X_k+1$, where k is the index of the current vertex.

Step 4. Modification of a vertex: "pass" at random from the current vertex to a "neighboring" one according to the matrix P.

Step 5. Termination: if $t<T_{\varepsilon,\sigma,\alpha,n}$, then go to Step 2, otherwise, go to Step 6.

Step 6. Output: $\mathbf{p}=\mathbf{X}/(T_{\varepsilon,\sigma,\alpha,n}-T^0_{\varepsilon,\alpha,n})$.

The most expensive step among the first five is Step 4, but even in the worst case (when the number of edges leaving the current vertex is on the order of n) this step requires $O(\ln n)$ operations (see, e.g., the Knuth–Yao algorithm). (Note that other methods can also be used. Moreover, the complexity $O(\ln n)$ can be understood not in the average case, but as usual (see [25]). The Knuth–Yao algorithm is mentioned only as an illustrative example.) Examples 1 and 2 (with the spectral gap α substituted for its bounds, which can be done in this context for reversible Markov chains (see [40]) but requires imposing certain conditions in the general case) imply that, for

$$T^0_{\varepsilon,\alpha,n}=O\left(\frac{1}{\alpha}\ln\left(\frac{n}{\varepsilon}\right)\right)$$

after

$$T_{\varepsilon,\sigma,\alpha,n} = O\left(\frac{\ln(n/\sigma)}{\alpha\varepsilon^2}\right)$$

iterations, with probability no smaller than $1-\sigma$, the MCMC algorithm produces an ε-optimal (in terms of the function) solution of the original problem. Overall, the MCMC algorithm requires

$$O\left(n + \frac{\ln n \ln(n/\sigma)}{\alpha\varepsilon^2}\right)$$

elementary arithmetic operations (of multiplication of two floating-point numbers); here, the term $O(n)$ "reflects" the cost of Step 6.

A method for efficient parallelization of this algorithm is easy to propose. Specifically, instead of a single trajectory, we use

$$N_{\varepsilon,\sigma} = O\left(\frac{\ln(\sigma^{-1})}{\varepsilon^2}\right)$$

independent random-walk trajectories that start at vertices chosen randomly and independently. Each of the trajectories is monitored over the time $T^0_{\varepsilon/2,\alpha,n}$. Taking the average (see below) over all these trajectories yields an answer (with possibility of organizing parallel computations), but the number of operations somewhat increases: instead of the multiplier $\ln(n/\sigma)$, we have $\ln(n/\varepsilon)\ln(\sigma^{-1})$, which is not bad.

Remark 8. In practice, this circumstance can be partially compensated as follows. For example, a single trajectory is issued first. At random times, the randomly walking particle produces descendants, which also walk randomly starting at their "birthplace" and are also prone to spontaneous divisions. An important issue is to choose a suitable division (multiplication) rate.

Indeed, we obtain a set of $N_{\varepsilon,\sigma}$ independent identically distributed random vectors $\mathbf{x}^k_{T^0_{\varepsilon,\alpha,n}}$, $k = 1, \ldots, N_{\varepsilon,\sigma}$, where all components of $\mathbf{x}^k_{T^0_{\varepsilon,\alpha,n}}$ are zero, except for one equal to 1. This component corresponds to the state at which the kth trajectory is at the step $T^0_{\varepsilon/2,\alpha,n}$. The vectors are identically distributed; moreover

$$\left\| E[\mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}} - \boldsymbol{\pi}] \right\|_1 \le \varepsilon/2$$

(recall that $\mathbf{p}_* = \boldsymbol{\pi}$ is an invariant measure). Applying a Hoeffding-type inequality in the Hilbert space l_2^n (see [48]) to

$$\frac{1}{N_{\varepsilon,\sigma}} \sum_{k=1}^{N_{\varepsilon,\sigma}} \mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}}$$

(this method was proposed together with Klochkov [45]), we obtain

$$P\left(\left\| \frac{1}{N_{\varepsilon,\sigma}} \sum_{k=1}^{N_{\varepsilon,\sigma}} \mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}} - \boldsymbol{\pi} \right\| \ge \varepsilon\right) = P\left(\left\| \sum_{k=1}^{N_{\varepsilon,\sigma}} (\mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}} - \boldsymbol{\pi}) \right\|_2 \ge \varepsilon N_{\varepsilon,\sigma}\right)$$

$$\le \exp\left(-\frac{1}{4N_{\varepsilon,\sigma}}\left(\varepsilon N_{\varepsilon,\sigma} - E\left(\left\| \sum_{k=1}^{N_{\varepsilon,\sigma}} (\mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}} - \boldsymbol{\pi}) \right\|_2\right)\right)^2\right),$$

where

$$E\left(\left\| \sum_{k=1}^{N_{\varepsilon,\sigma}} (\mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}} - \boldsymbol{\pi}) \right\|_2\right) \le \sqrt{E\left(\left\| \sum_{k=1}^{N_{\varepsilon,\sigma}} (\mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}} - \boldsymbol{\pi}) \right\|_2^2\right)} \le \sqrt{2\sqrt{2}N_{\varepsilon,\sigma} + (\varepsilon^2/4)N^2_{\varepsilon,\sigma}}.$$

Next, we choose $N_{\varepsilon,\sigma}$ such that

$$P\left(\left\|\frac{1}{N_{\varepsilon,\sigma}}\sum_{k=1}^{N_{\varepsilon,\sigma}}\mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}}-\boldsymbol{\pi}\right\|_2\geq\varepsilon\right)\leq\sigma.$$

For this purpose, it is sufficient that

$$\varepsilon N_{\varepsilon,\sigma}-\sqrt{2\sqrt{2}N_{\varepsilon,\sigma}+(\varepsilon^2/4)N^2_{\varepsilon,\sigma}}\geq\sqrt{4N_{\varepsilon,\sigma}\ln(\sigma^{-1})},$$

where

$$N_{\varepsilon,\sigma}=\frac{4+6\ln(\sigma^{-1})}{\varepsilon^2}=O\left(\frac{\ln(\sigma^{-1})}{\varepsilon^2}\right).$$

Note that, up to a multiplicative constant, this bound cannot be improved, which follows from the Chebyshev inequality

$$P(X\geq EX-\varepsilon)\geq 1-\frac{\mathrm{Var}(X)}{\varepsilon^2}$$

for

$$X=\left\|\frac{1}{N_{\varepsilon,\sigma}}\sum_{k=1}^{N_{\varepsilon,\sigma}}\mathbf{x}^k_{T^0_{\varepsilon/2,\alpha,n}}-\boldsymbol{\pi}\right\|_2,\quad \boldsymbol{\pi}=(n^{-1},\ldots,n^{-1})^{\mathrm{T}},\quad N_{\varepsilon,\sigma}\gg n.$$

Remark 9. When the components of the ranked vector $\boldsymbol{\pi}$ decrease exponentially, the estimates obtained below in the 2-norm can be replaced by similar estimates (with different constants and an additional logarithmic penalty) in the 1-norm. This can be understood taking into account that, in fact, we have the problem of reconstructing the parameters of a multinomial distribution. More precisely, we discusses the influence exerted by the dimension of the parameter space on the convergence rate estimates in the Fisher theorem on the maximum likelihood method in the nonasymptotic case [46] (sample means are the estimates produced by this method). In this example, the multinomial distribution leads (in this place, unfortunately, we have to use the Stirling formula, i.e., impose some asymptotic conditions) to the Kullback–Leibler divergence between the sample mean and true distributions in the exponent [47]. In turn, this divergence is estimated, according to Pinsker's inequality, by the 1-norm squared [48].

A slightly surprising result is that the total complexity depends weakly on the size of the matrix P, especially, if we take into account that no assumptions were made about the sparsity of P. The catch is that, in fact, unless random walks are considered on special graphs (for example, on expanders [49]), the condition that α is uniformly in n separated from zero is unnatural. For example, the (apparently sharp) estimate $\alpha\sim n^{-1}$ was given in [42] for the class of frequently encountered Markov processes arising in the description of various macroscopic systems "living" in the mean field approximation (their dynamics involve equal rights of interacting agents—the Guldberg–Waage law of mass action). It can also be noted that the gain in the complexity bound depending on n is due to the rather poor dependence on ε. Generally, a complexity dependence on ε better than ε^{-2} cannot be expected in randomized algorithms (see [50]), so randomization makes sense, as a rule, only for $\varepsilon\gg n^{-1}$. This becomes especially clear if we compare the obtained complexity bound with that of, say, the algorithm from [13]. In other words, it is reasonable to use MCMC when $\varepsilon\gg n^{-1}$.

The basic shortcoming of the MCMC method is that the exact values of $T_{\varepsilon,\sigma,\alpha}$ and $T^0_{\varepsilon,\alpha,n}$ are unknown even if α is given. Moreover, the efficiency of the algorithm is directly related to the bound for α, which is generally a priori not known. Nevertheless, α can be estimated in Cn arithmetic operations (unfortunately, with a rather large constant C), for example, with the help of the δ^2-process for practical estimation of the spectral gap (see, e.g., [18]). Another method for estimating α and $T^0_{\varepsilon,\alpha,n}$ can be found in [51]. The problem of determining $T_{\varepsilon,\sigma,\alpha}$ is solved by monitoring the difference $\|\mathbf{p}_{t+\tau}-\mathbf{p}_t\|_2$. To find $T^0_{\varepsilon,\alpha,n}$, the algorithm can first be run with $T^0_{\varepsilon,\alpha,n}=0$ and the output can then be corrected by setting, say, $T^0_{\varepsilon,\alpha,n}=T_{\varepsilon,\sigma,\alpha}/5$.

Note also that, initially, the matrix P must be properly stored in computer memory. If this matrix is treated in the usual manner, then the random choice of a neighboring vertex at every step in the MCMC

algorithm takes time on the order of n rather than $\ln n$, as would be desired. To avoid this, prior to running the algorithm, the graph has to be represented as a list of links. It will borrow time (storage) of exponent of vertices, but to make it is necessary only once.

5. GRIGORIADIS–KHACHIYAN ALGORITHM

The problem under consideration is rewritten (with the help of Perron–Frobenius theory [19]) as the problem of finding a saddle point (equilibrium in an antagonistic matrix game):

$$f(\mathbf{p}) = \max_{\mathbf{u} \in S_n(1)} \langle \mathbf{u}, A\mathbf{p} \rangle \longrightarrow \min_{\mathbf{p} \in S_n(1)}, \quad (1)$$

where

$$A = P^T - I, \quad S_n(1) = \left\{ \mathbf{p} \geq \mathbf{0} : \sum_{k=1}^{n} p_k = 1 \right\}.$$

Note that $f(\mathbf{p}) \geq 0$, $\mathbf{p} \in S_n(1)$, and $f(\mathbf{p}) = 0$ at the PageRank vector (and only at it).

With accuracy ε (in terms of the function) and with probability no smaller than $1 - \sigma$, the equilibrium can be found using the stochastic mirror descent algorithm from [11] in

$$O\left(\frac{n \ln(n/\sigma)}{\varepsilon^2}\right)$$

operations.

However, the specific features of the problem in the case of a sparse matrix P are not fully taken into account in these methods. To solve problem (1), we use the method from [12] for finding equilibria in symmetric antagonistic matrix games (note that this method is, in fact, a randomized version of the mirror descent method from [50]). For this purpose, following Dantzig [12], the problem (matrix game) is preliminarily reduced to a symmetric form (this stage is avoided if we use the randomized online mirror descent method from Theorem 2 in [52]):

$$\max_{\mathbf{u} \in S_{2n+1}(1)} \langle \mathbf{u}, A\mathbf{x} \rangle \qquad \min_{\mathbf{x} := (\mathbf{y}, \mathbf{p}', u) \in S_{2n+1}(1)}, \quad A := \begin{bmatrix} 0 & A & -\mathbf{e} \\ -A^{\mathrm{T}} & 0 & \mathbf{e} \\ \mathbf{e}^{\mathrm{T}} & -\mathbf{e}^{\mathrm{T}} & 0 \end{bmatrix},$$

where $\mathbf{e} = (1, \ldots, 1)^{\mathrm{T}}$ and $A = \|a_{ik}\|$. Then $f(\mathbf{p}) \leq 2\varepsilon$, where $\mathbf{p} = \mathbf{p}'/(\mathbf{e}^{\mathrm{T}}\mathbf{p}')$; moreover, $\mathbf{e}^{\mathrm{T}}\mathbf{p}' \leq 1/2 - \varepsilon$ if $A\mathbf{x} \leq \varepsilon\mathbf{e}$.

Step 1. Initialization: $\mathbf{X} = \mathbf{0}$, $\mathbf{p}^{\mathrm{T}} = \frac{1}{2n+1} \underbrace{(1, \ldots, 1)^{\mathrm{T}}}_{2n+1}$, $t = 0$.

Step 2. Iteration counter: $t := t + 1$.

Step 3. Random number generator: choose $k \in \{1, \ldots, 2n+1\}$ with probability p_k.

Step 4. Modification of $\mathbf{X}$: $X_k := X_k + 1$.

Step 5. Modification of $\mathbf{p}$: $i = 1, \ldots, 2n+1$, $p_i := p_i \exp\left(\frac{\varepsilon a_{ik}}{2}\right)$.

Step 6. Termination: if $t < T_{\varepsilon, \sigma, n}$, then go to Step 2, otherwise go to Step 7.

Step 7. Output: $\mathbf{x} = \mathbf{X}/t$.

Remark 10 (on Step 5). Later, it will be shown that the probability distribution can be specified up to a normalization factor.

Assuming that P is a sparse matrix with at most s nonzero elements in a column or row (see [9]), we see that the most expensive steps is the recalculation (regeneration) of the probability distribution (Steps 3 and 5). Using the method of generating a discrete random variable with the help of a balanced binary tree, we find that a change in the probability weight of a single outcome is, in fact, equivalent to changing the weights of those tree vertices that lie on the path to the tree leaf to be changed.

Remark 11. Let us describe this procedure more precisely. We are given a balanced binary tree of height $O(\log_2 n)$ with $O(n)$ vertices and $2n + 1$ leaves (to avoid going into technical details, assume that $2n + 1$ is

a power of two; clearly, this is not the case, but we can take, say, the least positive integer m such that $2^m > 2n + 1$ and consider a tree with 2^m leaves). Each vertex of the tree (other than a leaf) has a nonnegative weight equal to the sum of the weights of its two descendants. The original procedure for preparing the tree with leaves of identical weights requires $O(n)$ operations, but it has to be run only once. The procedure for generating a discrete random variable with a distribution corresponding (up to a normalization factor) to the leaf weights can be implemented using a random walk from the root to one of the leaves. Since the tree is binary, each vertex has two edges toward vertices with weights $a > 0$ and $b > 0$ and the passage of each vertex in a random walk is performed by tossing a coin ("prepared" so that the probability of passing the vertex with weight a is $a/(a + b)$). Clearly, this procedure does not need the additional normalization condition $a + b = 1$. If the algorithm changes the weight of some leaf in the course of its performance, then the weights of those and only those vertices that lie on the path from the root to this leaf also have to be properly changed. This is necessary to preserve the property that each tree vertex (other than a leaf) has a weight equal to the sum of the weights of its two descendants.

Note that there is no need to renormalize the probability distribution (which would cost $O(n)$), i.e., to change the weights of vertices other than those lying on the path to the leaf to be changed. All these tasks (generation with the help of the tree and the update of the tree) can be done in $O(\ln n)$ operations of comparing two numbers. In the ("typical") case when this has to be done $\sim s$ times, $O(s\ln n)$ operations are required (and only when $k = 2n + 1$, we need to perform $\sim 2n$ operations). Later, it will be shown that, after

$$T_{\varepsilon,\sigma,n} = O\left(\frac{\ln(n/\sigma)}{\varepsilon^2}\right)$$

iterations, with probability no smaller than $1 - \sigma$, the algorithm produces an ε-optimal (in terms of the function) solution of the original problem. Overall, the algorithm performs

$$O\left(n + \frac{s\ln n\ln(n/\sigma)}{\varepsilon^2}\right)$$

elementary arithmetic operations (of multiplication of two floating-point numbers). In the case $s \ll n$, this is noticeably better than in all known optimization analogues. The term $O(n)$ "reflects" the cases when $k = 2n + 1$ and the cost of final Step 7. Thus, we have the following result, whose proof is given in the Appendix.

Theorem. *After*

$$T_{\varepsilon,\sigma,n} = 12(\ln(2n + 1) + \ln(\sigma^{-1}))\varepsilon^{-2}$$

iterations, with probability no smaller than $1 - \sigma$, *the Grigoriadis–Khachiyan algorithm produces* $\mathbf{p}$ *such that* $0 \le f(\mathbf{p}) \le \varepsilon$. *Overall, the algorithm performs*

$$O\left(n + \frac{s\ln n\ln(n/\sigma)}{\varepsilon^2}\right)$$

operations (*of multiplication of two double-precision numbers*).

It was earlier noted that the ε-dependence of the complexity cannot be improved without substantially worsening the dependence of the complexity on n. More precisely, for methods involving randomization with the variance of the stochastic gradient D, the dependence of the number of iterations $\sim D\varepsilon^{-2}\ln(\sigma^{-1})$ is typical and cannot be improved. Nevertheless, if we take the expectation of a strongly convex functional (which is not the case in this study; on the contrary, we got rid of this situation to make the functional more informative), then the convergence can be improved up to $\sim D\varepsilon^{-1}\ln(\varepsilon^{-1})$ (see [53, 54]). For example, if we use the (deterministic) fast gradient method (see [55, 56]), then the ε-dependence of the complexity can be obtained in the form ε^{-1}, but the number of operations then increases by at least n times. Since the relation $\varepsilon \gg n^{-1}$ is quite natural for such problems, the resulting advantage seems doubtful. Note also [12] that, in the class of deterministic algorithms, the dependence on n cannot be better than $\sim n^2$ (when the matrix P is not sparse).

If the usual mirror descent method (see [57, 58]) is used to find the minimum of a nonsmooth convex function $f(\mathbf{p}) + \left(\sum_{k=1}^{n} p_k - 1\right)^2$, then, applying the theorem on the subdifferential of the maximum from [59, 60] and assuming that A is a weakly dense matrix with an average density characteristic (in rows) $\chi \ll 1$

(specifically, the number of nonzero elements in the matrix is $\leq \chi n^2$), we conclude that the total complexity is on average

$$O\left(\chi n^2 + \frac{(n + \chi^2 n^2)\ln n}{\varepsilon^2}\right).$$

In certain situations (for example, when $\chi \sim n^{-1/2}$), this bound is fairly competitive.

Note that the algorithm described is related to online optimization. Roughly speaking, algorithms of this type are asymptotically most effective in online learning based on expert experience even in the case of opposing Nature (see [50, 52–54, 58, 60–63]).

Note also that the Grigoriadis–Khachiyan algorithm is related to the concept of bounded rationality in the context of discrete choice theory (see [52, 61, 64]).

Let us summarize the results obtained in this section. In fact, the Grigoriadis–Khachiyan algorithm corresponds to the mirror descent method for finding an equilibrium in a matrix game with a skew-symmetric matrix A. For a non-skew-symmetric matrix, see [52].

The difference of the considered mirror descent from that presented, for example, in [11] is that, in our work (as in [12]), the iterative process for finding the saddle point $\langle \mathbf{u}, A\mathbf{x}\rangle$ on the product of two unit simplexes performs only with respect to $\mathbf{x}$, while the usual mirror descent method (designed for a non-skew-symmetric matrix A) performs with respect to $\mathbf{u}$ as well. Moreover, it is extremely important how randomization is implemented. In this paper, as in [12], randomization is introduced in selecting the component of the vector $\mathbf{x}$ used for componentwise motion. In all the other works, randomization is introduced at the stage of computing $A\mathbf{x} = E[A^{\langle x\rangle}]$, where x is a discrete random variable with distribution $\mathbf{x}$ and $A^{\langle x\rangle}$ is the x-column of the matrix A. Both randomization methods require

$$O\left(\frac{\ln(n/\sigma)}{\varepsilon^2}\right)$$

steps for achieving the accuracy ε (in terms of the function), but sparsity is taken into account noticeably better in the Grigoriadis–Khachiyan randomization. The cause is simple: at every step, the randomized mirror descent from [8, 11] requires the update of $\mathbf{x}$ by adding a vector with $2n + 1$ nonzero components, irrespective of the sparsity of A. Therefore, no less than $2n + 1$ operations per step are required. At the same time, as was shown above, the number of operations per step in the Grigoriadis–Khachiyan algorithm in the sparse case is typically $O(s\ln n)$, which can be noticeably less than n in the case $s \ll n$.

In the future, we intend to examine how to organize the random componentwise gradient descent for convex optimization in large-scale problems so that sparsity can be taken into account as much as possible. Now other approaches are popular (see [9, 10]). However, we believe that, in particular situations, the randomization method described (Grigoriadis–Khachiyan algorithm) will produce better results. Additionally, we intend to address the robust estimation of the PageRank vector (see [65, 66]) and to develop some ideas associated with distributed computations of the PageRank vector (see [67]).

APPENDIX

Proof of the theorem. For simplicity, $2n + 1$ is replaced by n. Determine $r > 0$ from the relation $\sigma \simeq n^{-r}$. Generally following [12], we show that, after

$$T_{\varepsilon,\sigma,n} = 3(\ln n + \ln(\sigma^{-1}))\varepsilon^{-2} = 3(1 + r)\varepsilon^{-2}\ln n$$

iteration steps, the Grigoriadis–Khachiyan algorithm produces $\mathbf{x}$ such that, with probability no smaller than $1 - \sigma$,

$$A\mathbf{x} \leq \varepsilon\mathbf{e}.$$

First, following [12, 60, 61], we set

$$p_i(t) = P_i(t)\left(\sum_{j=1}^{n} P_j(t)\right)^{-1}, \quad P_i(t) = \exp(\varepsilon U_i(t)/2),$$

and

$$\Phi(t) = \sum_{i=1}^{n} P_i(t), \quad \mathbf{U}(t) = A\mathbf{X}(t).$$

Then, by analogy with [12], we have

$$\Phi(t+1) = \sum_{i=1}^{n} P_i(t)\exp(\varepsilon a_{ik}/2) = \Phi(t)\sum_{i=1}^{n} p_i(t)\exp(\varepsilon a_{ik}/2),$$

$$E[\Phi(t+1)|\mathbf{P}(t)] = \Phi(t)\sum_{i,k=1}^{n} p_i(t)p_k(t)\exp(\varepsilon a_{ik}/2)$$

and ($\|a_{ik}\| \le 1$)

$$\exp(\varepsilon a_{ik}/2) \le 1 + \varepsilon a_{ik}/2 + \varepsilon^2/6.$$

However, since

$$\sum_{i,k=1}^{n} p_i(t)p_k(t) = \left(\sum_{i=1}^{n} p_i(t)\right)^2 = 1, \quad \sum_{i,k=1}^{n} p_i(t)p_k(t)\frac{\varepsilon^2}{6} = \frac{\varepsilon^2}{6},$$

$$\sum_{i,k=1}^{n} p_i(t)p_k(t)a_{ik} = \langle \mathbf{p}(t), A\mathbf{p}(t)\rangle = 0,$$

we conclude that

$$E[\Phi(t+1)|\mathbf{P}(t)] \le \Phi(t)(1+\varepsilon^2/6),$$

$$E[\Phi(t+1)] \le [\Phi(t)](1+\varepsilon^2/6).$$

Combining this inequality with

$$E[\Phi(0)] = \Phi(0) = n$$

yields

$$E[\Phi(t)] \le n(1+\varepsilon^2/6)^t.$$

Therefore,

$$E[\Phi(t)] \le n\exp(t\varepsilon^2/6) \quad \text{and} \quad E[\Phi(t^*)] \le n^{3/2+r/2}.$$

Combining this result with the Markov inequality

$$\forall \text{ eigenvalue } \xi \ge 0, \quad t > 0 \longrightarrow P(\xi \ge t) \le E\xi/t,$$

we have

$$P(\Phi(t^*) \le n^{3/2(1+r)}) = 1 - P(\Phi(t^*) \ge n^{3/2(1+r)}) \ge 1 - E[\Phi(t^*)]/n^{3/2(1+r)} \ge 1 - \sigma.$$

Taking the logarithm of both sides of the inequality

$$\exp(\varepsilon U_i(t^*)/2) = P_i(t^*) \le \sum_{i=1}^{n} P_i(t^*) = \Phi(t^*) \le n^{3/2(1+r)},$$

which holds with probability no smaller than $1-\sigma$, we obtain

$$P(\varepsilon U_i(t^*)/2 \le 3/2(1+r)\ln n, i = 1, \ldots, n) \ge 1 - \sigma,$$

$$P(U_i(t^*)/(3(1+r)\varepsilon^{-2}\ln n) \le \varepsilon, i = 1, \ldots, n) \ge 1 - \sigma,$$

whence

$$P(A\mathbf{x}(t^*) \le \varepsilon\mathbf{e}) \ge 1 - \sigma,$$

as required.

This paper is accompanied with a mini-lecture course (with a video) that was delivered by one of the authors at the Summer School of Modern Mathematics (LShSM-2013) in July, 2013: http://www.mathnet.ru/php/presentation.phtml?option_lang=rus&presentid=7259.

ACKNOWLEDGMENTS

We thank MIPT students I. Konovalenko, who found an important misprint in this work, and E. Klochkov, who contributed much to Section 4. We are also grateful to V.V. V'yugin, V.V. Vysotskii, A.V. Nazin, Y.E. Nesterov, A.S. Nemirovski, B.T. Polyak, A.M. Raigorodskii, V.G. Spokoiny, S.P. Tarasov, R. Tempo, A. Tremba, and P.Yu. Chebotarev for their valuable remarks. Additionally, we thank MIPT student A. Imeeva, who performed a series of numerical experiments with the algorithms discussed in this paper.

This work was supported by the Russian Foundation for Basic Research (project no. 14-01-00722-à), by the Laboratory of Structural Methods for Data Analysis in Predictive Modeling of the Moscow Institute of Physics and Technology, and by a grant from the President of the Russian Federation (project no. MK-5285.2013.9). The study of the MCMC method was supported by the Russian Science Foundation, project no. 14-50-00150.

REFERENCES

1. S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," Comput. Network ISDN Syst. **30** (1–7), 107–117 (1998).
2. A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings* (Princeton Univ. Press, New York, 2006).
3. P. Baldi, P. Frasconi, and P. Smyth, *Modeling the Internet and the Web: Probabilistic Methods and Algorithms* (Wiley, New York, 2003); http://ibook.ics.uci.edu/.
4. M. Franceschet, "PageRank: Standing on the shoulders of giants," Commun. ACM **54** (6), 92–101 (2011); arXiv:1002.2858v3.
5. A. V. Gasnikov, E. V. Gasnikova, and O. S. Fed'ko, "On possible dynamics in Google's PageRank model and a modified model of computation correspondence matrix," Tr. Mosk. Fiz.-Tekh. Inst. **4** (2(14)), 101–120 (2012).
6. R. P. Agaev and P. Yu. Chebotarev, "Convergence and stability in characteristics-matching problems (overview of basic results)," Upr. Bol'sh. Sist. **30** (1), 470–505 (2010).
7. D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chain and Mixing Times* (AMS, 2009); http://pages.uoregon.edu/dlevin/MARKOV/markovmixing.pdf.
8. A. V. Nazin and B. T. Polyak, "Randomized algorithm to determine the eigenvector of a stochastic matrix with application to the PageRank problem," Autom. Remote Control **72** (2), 342–352 (2011).
9. Y. E. Nesterov, "Subgradient methods for huge-scale optimization problems," CORE Discussion Paper; 2012/2, 2012; http://dial.academielouvain.be/handle/boreal:107876.
10. Y. E. Nesterov, "Efficiency of coordinate descent methods on large scale optimization problem," SIAM J. Optim. **22** (2), 341–362 (2012); http://dial.academielouvain.be/handle/boreal:121612.
11. A. Juditsky, G. Lan, A. Nemirovski, and A. Shapiro, "Stochastic approximation approach to stochastic programming," SIAM J. Optim. **19** (4), 1574–1609 (2009).
12. L. G. Khachiyan, *Selected Works* (MTsNMO, Moscow, 2009), pp. 38–48 [in Russian].
13. Y. Nesterov and A. Nemirovski, "Finding the stationary states of Markov chains by iterative methods," CORE Discussion Paper; 2012/58, 2012; http://dial.academielouvain.be/handle/boreal:122163.
14. B. T. Polyak and A. A. Tremba, "Regularization-based solution of the PageRank problem for large matrices," Autom. Remote Control **73** (11), 1777–1894 (2012).
15. D. Spielman, http://www.cse.cuhk.edu.hk/~chi/csc5160/notes/L07.pdf.
16. B. S. Kashin, "Diameters of some finite-dimensional sets and classes of smooth functions," Math. USSR Izv. **11** (2), 317–333 (1977).
17. G. Pandurangan, P. Raghavan, and E. Upfal, "Using PageRank to characterize web structure," Internet Math. **3** (1), 1–20 (2006).
18. N. S. Bakhvalov, N. P. Zhidkov, and G. M. Kobel'kov, *Numerical Methods* (Binom, Moscow, 2002).
19. H. Nikaido, *Convex Structures and Economic Theory* (Academic, New York, 1968; Mir, Moscow, 1972).
20. V. S. Ryaben'kii, *Introduction to Computational Mathematics* (Fizmatlit, Moscow, 2000) [in Russian].
21. P. Chebotarev and R. Agaev, "Forest matrices around the Laplacian matrix," Linear Algebra Appl. **356**, 253–274 (2002).
22. D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra* (Freeman, San Francisco, 1963; Fizmatgiz, Moscow, 1963).

23. M. A. Krasnosel'skii and S. G. Krein, "Remark on the distribution of errors in the solution of a system of linear equations by means of an iterative process," Usp. Mat. Nauk **7** (4(50)), 157–161 (1952).
24. R. A. Horn and C. R. Johnson, *Matrix Analysis* (Cambridge Univ. Press, Cambridge, 1985; Mir, Moscow, 1989).
25. S. M. Ermakov, *Monte Carlo Method in Computational Mathematics: An Introductory Course* (Binom, Moscow, 2009).
26. A. M. Raigorodskii, *Internet Models* (Intellekt, Dolgoprudnyi, 2013).
27. D. Knuth and A. Yao, "The complexity of nonuniform random number generation," *Algorithms and Complexity: New Directions and Recent Results* (Academic, New York, 1976), pp. 357–428.
28. A. Sinclair and M. Jerrum, "Approximate counting, uniform generation and rapidly mixing Markov chains," Inf. Comput. **82** (1), 93–133 (1989).
29. M. Dyer, A. Frieze, and R. Kannan, "A random polynomial-time algorithm for approximating of the volume of convex bodies," J. ACM **38** (1), 1–17 (1991).
30. M. Jerrum and A. Sinclair, "The Markov chain Monte Carlo method: An approach to approximate counting and integration," *Approximation Algorithms for NP-Hard Problems*, Ed. by D. S. Hochbaum (PWS, Boston, 1996), pp. 482–520; http://citeseerx.ist.psu.edu/viewdoc/summary?doi = 10.1.1.32.9553.
31. P. Lezaud, "Chernoff-type bound for finite Markov chains," Ann. Appl. Probab. **8** (3), 849–867 (1998).
32. D. Aldous and J. Fill, *Reversible Markov Chains and Random Walks on Graphs* (2002); http://stat-www.berkeley.edu/users/aldous/RWG/book.html.
33. Fan Chung. "Laplacians and the Cheeger inequality for directed graphs," Ann. Combin., No. 9, 1–19 (2005); http://math.ucsd.edu/~fan/.
34. P. Diaconis, "The Markov chain Monte Carlo revolution," Bull. AMS (New Ser.) **49** (2), 179–205 (2009); http://math.uchicago.edu/~shmuel/Network-course-readings/MCMCRev.pdf.
35. D. Spielman, Lecture No. 2, 2009; http://www.cse.cuhk.edu.hk/~chi/csc5160/notes/L02.pdf.
36. A. Joulin and Y. Ollivier, "Curvature, concentration and error estimates for Markov chain Monte Carlo," Ann. Probab. **38** (6), 2418–2442 (2010).
37. R. Montenegro and P. Tetali, *Mathematical Aspects of Mixing Times in Markov Chains* (2006); http://people.math.gatech.edu/~tetali/PUBLIS/survey.pdf.
38. S. Boyd and L. Vandenberghe, *Convex Optimization* (Cambridge Univ. Press, Cambridge, 2004).
39. S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability* (Springer, London, 2005); http://probability.ca/MT/.
40. D. Paulin, "Concentration inequalities for Markov chains by Marton couplings," e-print, arXiv:1212.2015v2, 2013.
41. M. Ledoux, *Concentration of Measure Phenomenon* (Am. Math. Soc., Providence, RI, 2001).
42. A. V. Gasnikov and E. V. Gasnikova, "On entropy similar functionals arising in stochastic chemical kinetics at invariant measure concentration and as Lyapunov functions of quasi-mean dynamics," Mat. Zametki **93** (6), 816–824 (2013).
43. M. A. Krasnosel'skii, E. A. Lifshits, and A. V. Sobolev, *Positive Linear Systems: Method of Positive Operators* (Nauka, Moscow, 1985) [in Russian].
44. M. Ya. Kel'bert and Yu. M. Sukhov, *Probability and Statistics with Examples and Problems* (MTsNMO, Moscow, 2010), Vol. 2 [in Russian].
45. E. Yu. Klochkov, "Parametric estimation in Transportation Problems," Bachelor's Thesis (Moscow Inst. Phys. Eng., Dolgoprudnyi, 2013).
46. V. Spokoiny, "Parametric estimation: Finite sample theory," Ann. Stat. **40** (6), 2877–2909 (2012).
47. I. N. Sanov, "On the probability of large deviations of random variables," Mat. Sb. **42** (1), 11–44 (1957).
48. S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities: A Nonasymptotic Theory of Independence* (Oxford Univ. Press, London, 2013).
49. A. Romashchenko, *Expanders: Construction and Applications* (Computer Science Club, Mosk. Gos, Univ., Moscow, 2009); http://www.mccme.ru/~anromash/courses/expanders2009.pdf.
50. A. S. Nemirovski and D. B. Yudin, *Complexity of Problems and Efficiency of Optimization Methods* (Nauka, Moscow, 1979) [in Russian].
51. D. Paulin, "Nonasymptotic confidence interval for MCMC," e-print, arXiv:1212.2016v4, 2013.
52. A. V. Gasnikov, Yu. E. Nesterov, and V. G. Spokoiny, "On the efficiency of a randomization method for mirror descent in online optimization problems," Comput. Math. Math. Phys. (2015) [in press].
53. Y. Mansour, *Algorithmic Game Theory and Machine Learning* (2011); www.tau.ac.il/~mansour/advanced-agt+ml/.
54. S. Bubeck, *Introduction to Online Optimization: Lecture Notes* (Princeton University, New York, 2011); http://www.princeton.edu/~sbubeck/BubeckLectureNotes.pdf.
55. Y. Nesterov, "Smooth minimization of nonsmooth function," Math. Program. Ser. A **103** (1), 127–152 (2005).

56. Yu. E. Nesterov, *Introduction to Convex Optimization* (Mosk. Tsentr Neprer. Mat. Obrazovan., Moscow, 2010) [in Russian].
57. http://www2.isye.gatech.edu/~nemirovs/.
58. Y. Nesterov, "Primal-dual subgradient methods for convex problems," Math. Program. Ser. B **120**, 221–259 (2009).
59. G. G. Magaril-Il'yaev and V. M. Tikhomirov, *Convex Analysis: Theory and Applications* (Am. Math. Soc., Providence, 2003; URSS, Moscow, 2011).
60. V. V. V'yugin, *Mathematical Foundations of Machine Learning Theory and Prediction* (Mosk. Tsentr Neprer. Mat. Obrazovan., Moscow, 2013); http://www.iitp.ru/upload/publications/6256/vyugin1.pdf.
61. G. Lugosi and N. Cesa-Bianchi, *Prediction, Learning, and Games* (Cambridge Univ. Press, New York, 2006).
62. A. B. Juditsky, A. V. Nazin, A. B. Tsybakov, and N. Vayatis, "Recursive aggregation of estimators by the mirror descent algorithm with averaging," Probl. Inf. Transmission **41** (4), 368–384 (2005).
63. A. Juditsky, A. V. Nazin, A. B. Tsybakov, and N. Vayatis, "Gap-free bounds for stochastic multi-armed bandit," *Proceedings of the 17th IFAC World Congress, Seoul, Korea, July 6–11 2008* (IFAC, 2008), pp. 11560–11563.
64. S. P. Andersen, A. de Palma, and J.-F. Thisse, *Discrete Choice Theory of Product Differentiation* (MIT Press, Cambridge, 1992).
65. A. Juditsky and B. Polyak, "Robust eigenvector of stochastic matrix with application to PageRang," *Proceedings of the 51st IEEE Conference on Decision and Control, December 10–13, 2012, Maui, Hawaii, USA* (IEEE, 2012), pp. 3171–3176 (arXiv:1206.4897).
66. A. Tremba and A. Nazin, "Extension of a saddle point mirror descent algorithm with application to robust PageRank," *Proceedings of the 52nd IEEE Conference on Decision and Control, December 10–13, 2013, Florence, Italy* (IEEE, 2013), pp. 3691–3696.
67. R. Tempo, http://staff.polito.it/roberto.tempo/.

Translated by I. Ruzanova