COMPUTER RESEARCH AND MODELING
2023 VOL. 15 NO. 2 P. 369–379
DOI: 10.20537/2076-7633-2023-15-2-369-379

КиМ

MATHEMATICAL MODELING AND NUMERICAL SIMULATION

UDC: 519.8

Experimental comparison of PageRank vector calculation algorithms

D. A. Skachkov[a], S. I. Gladyshev[b], A. M. Raigorodskii[c]

Moscow Institute of Physics and Technology, MIPT,
1a/1 Kerchenskaya st., Moscow 117303, Russia

E-mail: [a] skachkov.da@phystech.edu, [b] gladyshev.si@phystech.edu, [c] mraigor@yandex.ru

Received 19.02.2023.
Accepted for publication 23.02.2023.

Finding PageRank vector is of great scientific and practical interest due to its applicability to modern search engines. Despite the fact that this problem is reduced to finding the eigenvector of the stochastic matrix P, the need for new algorithms is justified by a large size of the input data. To achieve no more than linear execution time, various randomized methods have been proposed, returning the expected result only with some probability close enough to one. We will consider two of them by reducing the problem of calculating the PageRank vector to the problem of finding equilibrium in an antagonistic matrix game, which is then solved using the Grigoriadis – Khachiyan algorithm. This implementation works effectively under the assumption of sparsity of the input matrix. As far as we know, there are no successful implementations of neither the Grigoriadis – Khachiyan algorithm nor its application to the task of calculating the PageRank vector. The purpose of this paper is to fill this gap. The article describes an algorithm giving pseudocode and some details of the implementation. In addition, it discusses another randomized method of calculating the PageRank vector, namely, Markov chain Monte Carlo (MCMC), in order to compare the results of these algorithms on matrices with different values of the spectral gap. The latter is of particular interest, since the magnitude of the spectral gap strongly affects the convergence rate of MCMC and does not affect the other two approaches at all. The comparison was carried out on two types of generated graphs: chains and d-dimensional cubes. The experiments, as predicted by the theory, demonstrated the effectiveness of the Grigoriadis – Khachiyan algorithm in comparison with MCMC for sparse graphs with a small spectral gap value. The written code is publicly available, so everyone can reproduce the results themselves or use this implementation for their own needs. The work has a purely practical orientation, no theoretical results were obtained.

Keywords: Grigoriadis – Khachiyan, Markov chain Monte Carlo, PageRank

Citation: *Computer Research and Modeling*, 2023, vol. 15, no. 2, pp. 369–379.

The research was supported by Russian Science Foundation (project No. 21-71-30005), https://rscf.ru/en/project/21-71-30005/.

КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2023 Т. 15 № 2 С. 369–379
DOI: 10.20537/2076-7633-2023-15-2-369-379

Ки&М

МАТЕМАТИЧЕСКИЕ ОСНОВЫ И ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

УДК: 519.8

Экспериментальное сравнение алгоритмов поиска вектора PageRank

Д. А. Скачков[a], С. И. Гладышев[b], А. М. Райгородский[c]

Московский физико-технический институт (национальный исследовательский университет),
Россия, 117303, Москва, ул. Керченская, 1а, корп. 1

E-mail: [a] skachkov.da@phystech.edu, [b] gladyshev.si@phystech.edu, [c] mraigor@yandex.ru

Получено 19.02.2023.
Принято к публикации 23.02.2023.

Задача поиска PageRank вектора представляет большой научный и практический интерес ввиду своей применимости к работе современных поисковых систем. Несмотря на то, что данная задача сводится к поиску собственного вектора стохастической матрицы P, потребность в новых алгоритмах для ее решения обусловлена большими размерами входных данных. Для достижения не более чем линейного времени работы применяются различные рандомизированные методы, возвращающие ожидаемый ответ лишь с некоторой достаточно близкой к единице вероятностью. Нами рассматриваются два таких способа, сводящие задачу поиска вектора PageRank к задаче поиска равновесия в антагонистической матричной игре, которая затем решается с помощью алгоритма Григориадиса – Хачияна. При этом данная реализация эффективно работает в предположении о разреженности матрицы, подаваемой на вход. Насколько нам известно, до сих пор не было ни одной успешной реализации ни алгоритма Григориадиса – Хачияна, ни его применения к задаче поиска вектора PageRank. Данная статья ставит перед собой задачу восполнить этот пробел. В работе приводится описание двух версий алгоритма с псевдокодом и некоторые детали их реализации. Кроме того, в работе рассматривается другой вероятностный метод поиска вектора PageRank, а именно Markov chain Monte Carlo (MCMC), с целью сравнения результатов работы указанных алгоритмов на матрицах с различными значениями спектральной щели. Последнее представляет особый интерес, поскольку значение спектральной щели сильно влияет на скорость сходимости MCMC, и не оказывает никакого влияния на два других подхода. Сравнение проводилось на сгенерированных графах двух видов: цепочках и d-мерных кубах. Проведенные эксперименты, как и предсказывает теория, демонстрируют эффективность алгоритма Григориадиса – Хачияна по сравнению с MCMC для разреженных графов с маленьким значением спектральной щели. Весь код находится в открытом доступе, так чтобы все желающие могли воспроизвести полученные результаты самостоятельно, или же использовать данную реализацию в своих нуждах. Работа имеет чисто практическую направленность, никаких теоретических результатов авторами получено не было.

Ключевые слова: Григориадис – Хачиян, Markov chain Monte Carlo, PageRank

Исследование выполнено за счет гранта Российского научного фонда (проект № 21-71-30005), https://rscf.ru/project/21-71-30005/.

Introduction

In this paper we consider the Grigoriadis – Khachiyan algorithm [Khachiyan, 2009], applied for the problem of finding the PageRank vector. The PageRank problem was first formulated by Brin and Page [Brin, Page, 1998], but remains to be important nowadays. The problem consists of finding a stationary distribution of Markov Chain defined by a web graph, where websites are considered as states with some probabilities of transition. More formally, given a stochastic matrix $P \in [0, 1]^{n \times n}$, we aim at calculating vector $\mathbf{v}$ satisfying the equality

$$P^T \mathbf{v} = \mathbf{v}.$$

This can be done by iterative methods like, for example, power iteration, introduced in [Mises, Pollaczek-Geiringer, 1929], which requires $O\left(n^3\right)$ time. To deal with large matrices, different randomized algorithms with better time complexity were proposed.

Two of them, described in [Gasnikov, Dmitriev, 2015] and [Anikin et al., 2022], use the Grigoriadis – Khachiyan algorithm. To our best knowledge, the only attempt at conducting experiments for the Grigoriadis – Khachiyan algorithm, which was made in [Anikin et al., 2022], was not particularly successful, as it demonstrated inaccurate results. We aim at feeling this annoying gap and present our own comparison of the Grigoriadis – Khachiyan algorithm to other approaches.

The structure of this paper is the following: in Section 2 we describe three randomized methods of finding the PageRank vector that we considered during our work, then, in Section 3, we describe the experiments and present the obtained results, finally, in Section 4, we make a conclusion based on the observations.

Methods

In this section we describe approaches that have been considered in our work, giving some necessary comments on their implementation. Throughout this section we use the following notation and conventions:

1) I_n is an $n \times n$ identity matrix;
2) $\mathbf{0}_n$ ($\mathbf{1}_n$) is a column-vector of zeros (ones) of length n;
3) $S_n(1) := \left\{ \mathbf{x} \in [0, 1]^n \mid \sum\limits_{i=1}^{n} \mathbf{x}_i = 1 \right\}$ is a set of stochastic vectors;
4) we say that vector $\mathbf{v}$ is an ε-optimal solution in the sense of two-norm if $\|\mathbf{v} - \mathbf{v}^*\|_2 \leqslant \varepsilon$, where $\mathbf{v}^*$ is the true PageRank vector;
5) we say that vector $\mathbf{v}$ is an ε-optimal solution in the sense of infinity norm if $\left\|\left(P^T - I\right)\mathbf{v}\right\|_\infty \leqslant \varepsilon$.

Markov chain Monte Carlo method

The idea of using the Markov chain Monte Carlo method (MCMC) for solving linear equations, as well as the Monte Carlo method [Ermakov, 2009], is not novel. Applied to PageRank problem, it can be implemented more efficiently as one can use the properties of matrix P to accelerate the step of random walk. We introduce a pseudocode of this algorithm, based on the one given in [Gasnikov, Dmitriev, 2015].

The idea itself is very simple: we start a random walk on graph and count the frequency of visiting each vertex. Exactly this vector of frequencies will be the resulting one. To make it more

Algorithm 1. MCMC

Require: $T^0_{\varepsilon,\alpha,n}$; $T_{\varepsilon,\sigma,\alpha,n}$; $P \in \mathbb{R}^{n\times n}$: $\sum\limits_{j=1}^{n} p_{ij} = 1\ \forall i \in \{1, \ldots, n\}$

Ensure: $\mathbf{x} \in \mathbb{R}^n$: $\sum\limits_{i=1}^{n} \mathbf{x}_i = 1$

1: $\mathbf{X} \leftarrow \mathbf{0}$ ▷ Counter
2: $k \leftarrow 1$ ▷ We start our random walk from the vertex 1
3: $t \leftarrow 0$ ▷ Time
4: **while** $t \leqslant T_{\varepsilon,\sigma,\alpha,n}$ **do**
5: **if** $t \geqslant T^0_{\varepsilon,\alpha,n}$ **then**
6: $X_k \leftarrow X_k + 1$ ▷ where k is an id of the current vertex
7: **end if**
8: Choose k according to matrix P.
9: $t \leftarrow t + 1$
10: **end while**
11: return $\frac{\mathbf{X}}{T_{\varepsilon,\sigma,\alpha,n} - T^0_{\varepsilon,\alpha,n}}$

reliable, we start counting frequencies only from T_{start} iteration and continue till T_{end}, where T_{start} and T_{end} are the parameters of the algorithm and will be studied a little bit later.

We will give some details about step 8 of MCMC algorithm, as it is the most time-consuming part. While initializing a graph, for each vertex v we store the vector $\mathbf{w}^v$ of cumulative probabilities for the outgoing edges, i. e., $\mathbf{w}^v_k$ equals the probability of moving to one of the first $k-1$ neighbors as soon as we ordered them in any way.

Thus, located in vertex v, one can generate ξ from uniform distribution on (0, 1] and then, using a binary search, find such i that $\mathbf{w}^v_i < \xi \leqslant \mathbf{w}^v_{i+1}$, which certainly can be done in $O(\log n)$. This i defines the neighbor to move to.

Turning back to T_{start} and T_{end}, it is known [Gasnikov, Dmitriev, 2015] that to get a solution that is ε-optimal in the sense of the two-norm, with a probability not less that $1-\sigma$, it is sufficient to put

$$T_{start} := O\left(\frac{1}{\alpha}\ln\left(\frac{n}{\varepsilon}\right)\right)$$

and

$$T_{end} := O\left(\frac{\ln n \ln\left(\frac{n}{\sigma}\right)}{\alpha\varepsilon^2}\right).$$

Here α denotes a spectral gap of matrix P, i. e. if $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of matrix P and $|\lambda_1| \geqslant |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|$, then

$$\alpha := |\lambda_1| - |\lambda_2|.$$

Notice that in the case of a stochastic matrix λ_1 is always equal to 1 and $|\lambda_2| < 1$, therefore $\alpha > 0$ (for more details see, for example, [Seneta, 1981]).

With such T_{start} and T_{end} values, MCMC obtains a result in

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

arithmetic operations.

Grigoriadis – Khachiyan algorithm

We consider two different variations of applying the Grigoriadis – Khachiyan algorithm to the PageRank problem.

First approach. Recall that we aim at finding a vector $\mathbf{v}$ such that

$$\left(P^T - I_n\right)\mathbf{v} = 0.$$

This can be rewritten as

$$\left\|\left(P^T - I_n\right)\mathbf{v}\right\|_\infty = 0$$

and thus, calculating the PageRank vector is equivalent to the optimization problem

$$\left\|\left(P^T - I_n\right)\mathbf{v}\right\|_\infty \to \min,$$

as soon as we know that the PageRank vector exists. Defining

$$B := P^T - I_n,$$

we get the problem of finding equilibrium in the matrix antagonistic game (for more details see [Khachiyan, 2009]):

$$\max_{\mathbf{u}\in S_n(1)} \langle \mathbf{u}, B\mathbf{v}\rangle \to \min_{\mathbf{v}}. \tag{1}$$

To apply the Grigiriadis – Khachiyan algorithm, following [Khachiyan, 2009], (1) can be reduced to the symmetric case by building a new matrix

$$A := \begin{pmatrix} 0 & B & -\mathbf{1}_n \\ -B^T & 0 & \mathbf{1}_n \\ \mathbf{1}_n^T & -\mathbf{1}_n^T & 0 \end{pmatrix}$$

and by considering the optimization problem

$$\max_{\mathbf{u}\in S_{2n+1}(1)} \langle \mathbf{u}, A\mathbf{x}\rangle \to \min_{\mathbf{x}\in S_{2n+1}(1)}.$$

If $\mathbf{y} = (\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{2n})^T$, then $\mathbf{v} := \frac{\mathbf{y}}{\|\mathbf{y}\|_1}$ is the solution to the original problem.

We present a pseudocode of this approach based on the one from [Gasnikov, Dmitriev, 2015]. The main idea is rather simple: starting with a vector $\mathbf{w} := \mathbf{1}_n$, we generate a column with probabilities proportional to $\mathbf{w}$ and then iterate through this generated column, updating vector $\mathbf{w}$.

The most challenging part of this algorithm is generating column and updating vector $\mathbf{w}$. To deal with it, we use a structure based on a balanced binary tree, where leaves correspond to columns. Every node of this tree contains the sum of weights of its child nodes. Thus, one can easily generate a leaf starting from the top node and moving down, every step doing a choice between two child nodes according to their weights. Updating the weight of a leaf is also quite simple procedure: starting from this leaf, one moves up to the top, changing weights in order to maintain the invariant. Both of these require $O(\log n)$ time and so each step of the Grigoriadis – Khachiyan algorithm can be done in $O(d \log n)$ time, where d is the number of nonzero elements in column. In [Gasnikov, Dmitriev, 2015] it is shown that given ε and σ

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

Algorithm 2. Grigoriadis – Khachiyan for the PageRank Problem (1^{st} approach)

Require: P, T, ε
Ensure: $\mathbf{x} \in \mathbb{R}^n$: $\sum\limits_{i=1}^{n} x_i = 1$
  Build matrix A from P.
  $\mathbf{X} \leftarrow \mathbf{0}_n$ ▹ Counter
  $\mathbf{w} \leftarrow \mathbf{1}_{2n+1}$ ▹ All weights are equal
  for $t = 1, \ldots, T$ **do**
    Choose k randomly from $\{1, \ldots, 2n + 1\}$ with weights $\mathbf{w}$.
    if $k \in \{n + 1, \ldots, 2n\}$ **then**
      $X_{k-n} \leftarrow X_{k-n} + 1$
    end if
    for $i = 1, \ldots, 2n + 1$ **do**
      $w_i \leftarrow w_i \exp\left(\frac{\varepsilon}{2} A_{ik}\right)$
    end for
  end for
  return $\mathbf{v} = \frac{\mathbf{X}}{T_{\varepsilon,\sigma,\alpha,n} - T^0_{\varepsilon,\alpha,n}}$

steps is enough to achieve an ε-optimal solution in the sense of infinity norm with probability not less than $1 - \sigma$. This fact implies total time-complexity equal to

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

where d is the maximum number of nonzero elements both per rows and columns of matrix P.

Second approach. This approach was taken from [Anikin et al., 2022]. Just as before, we consider an optimization problem

$$\|(P - I)v\|_\infty \to \min.$$

Following [Nemirovski et al., 2009], this can be set up as

$$\min_{x \in S_n(1)} \max_{y \in S_{2n}(1)} \langle x, Ay \rangle$$

with

$$A := (P - I_n)J$$

where $J := [I_n, -I_n]$.

We give the pseudocode based on the one from [Anikin et al., 2022]. The main idea of it is a simulation of a 2-player game, where on every step both player 1 and player 2 choose independently a column and a row, respectively, and then update their probabilities vectors according to the choice of the opponent, using the parameters l^1 and l^2. To generate values and change weights, we use exactly the same structure as the one mentioned in the description of the Grigoriadis – Khachiyan algorithm. The vector of frequencies for the rows to be chosen by player 2 is exactly the required PageRank vector.

It is claimed in [Anikin et al., 2022] that

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

steps is enough to obtain a solution that is ε-optimal in the sense of infinity norm, with probability not less then $1 - \sigma$, if the parameters l^1 and l^2 are chosen in an appropriate way. Moreover, the total complexity is equal to

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

where d is the maximum number of non-zero elements both per rows and columns of matrix P.

Algorithm 3. Grigoriadis – Khachiyan for the PageRank Problem (2^{nd} approach)

Require: P, T, l^1, l^2
Ensure: $\mathbf{x} \in \mathbb{R}^n$: $\sum\limits_{i=1}^{n} x_i = 1$
1: Build matrix A from P.
2: $w \leftarrow \mathbf{1}_{2n}$
3: $w' \leftarrow \mathbf{1}_n$
4: **for** $t = 1, \ldots, T$ **do**
5: Choose j randomly from $\{1, \ldots, 2n\}$ with weights w.
6: Choose i randomly from $\{1, \ldots, n\}$ with weights w'.
7: $X_i \leftarrow X_i + 1$
8: **for** $s = 1, \ldots, 2n$ **do**
9: $w_s \leftarrow w_s \exp\left(l^1 A_{is}\right)$
10: **end for**
11: **for** $s = 1, \ldots, n$ **do**
12: $w'_s \leftarrow w'_s \exp\left(l^2 A_{sj}\right)$
13: **end for**
14: **end for**
15: return $\frac{X}{T}$

Experimental results

In this section we present the obtained experimental results. All of the experiments can be reproduced with the help of the code from our GitHub: https://github.com/gladyshSI/Grigoriadis_Khachiyan.

We tested the approaches from Section 3 in the following cases:

1) simple chain,
2) d-dimensional cube.

We'll describe their generation in more detail.

Simple chain. To build a chain, we first generate an unordered chain of length n and then replace each edge with a pair of oriented edges, each having the probability of $\frac{1}{2}$. Since two boundary vertices have one outgoing edge, their weight should be equal to 1. We give an example of such a graph in Fig. 1.

For a simple chain the PageRank vector is $\mathbf{v} = \left(\frac{1}{2(n-1)} \quad \frac{1}{n-1} \quad \frac{1}{n-1} \quad \cdots \quad \frac{1}{n-1} \quad \frac{1}{2(n-1)}\right)^T$.

As can be seen from Table 1, the values of the spectral gap for chain graphs are extremely small, which is fatal for MCMC. But we will give another, more intuitive, look at the reasons why chain graphs can be a challenge for MCMC. According to the Law of the iterated logarithm [Khintchine, 1924], with very high probability MCMC wouldn't visit all vertices of chain faster than in n^2 steps, thus MCMC can't be expected to obtain a proper result in linear time.

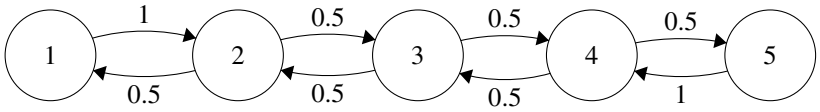

Figure 1. An example of a chain web-graph consisting of 5 vertices. The numbers inside the vertices mean the index of the vertex. The numbers near the edges indicate the probability of transition between the corresponding vertices

Table 1. Values of the spectral gap for chains and d-dimensional cubes, where n denotes the number of vertices

n	*chain*	d-cube
2^{14}	$2 \cdot 10^{-9}$	0.14
2^{17}	$3 \cdot 10^{-11}$	0.12
2^{20}	$5 \cdot 10^{-13}$	0.10

d-dimensional cube. We set all weights of edges equal to $\frac{1}{d}$, since each vertex has exactly d neighbors. The PageRank vector is equal to $\mathbf{v} = \left(\frac{1}{2^d} \quad \frac{1}{2^d} \quad \dots \quad \frac{1}{2^d}\right)^T$, which follows from the uniformity of the graph.

The values of the spectral gap are presented in Table 1. As they are not too small, one can expect a proper work of MCMC.

All approaches were tested on graphs of size $2^{14} = 16{,}384$, $2^{17} = 131{,}072$ and $2^{20} = 1{,}048{,}576$. The parameters for MCMC are presented in Table 2. Following [Gasnikov, Dmitriev, 2015], we set T_{start} equal to $\frac{T_{end}}{5}$. For the Grigoriadis – Khachiyan algorithm we took $\varepsilon = 0.01$. Figure 3 shows the results of the algorithms on chain graphs of different sizes and Figure 4 those on cube graphs.

Table 2. Parameters of the MCMC algorithms, where T_{start} is the number of iteration to start counting frequencies from, and T_{end} is the total number of iterations

n	T_{start}	T_{end}
2^{14}	$2 \cdot 10^5$	10^6
2^{17}	$2 \cdot 10^5$	10^6
2^{20}	$2 \cdot 10^6$	10^7

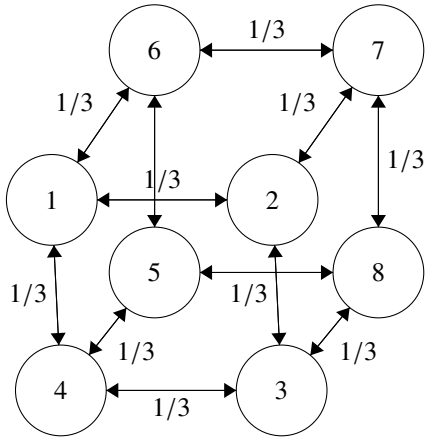

Figure 2. An example of a cube web-graph consisting of 8 vertices. The numbers inside the vertices mean the index of the vertex. The numbers near the edges indicate the probability of transition between the corresponding vertices

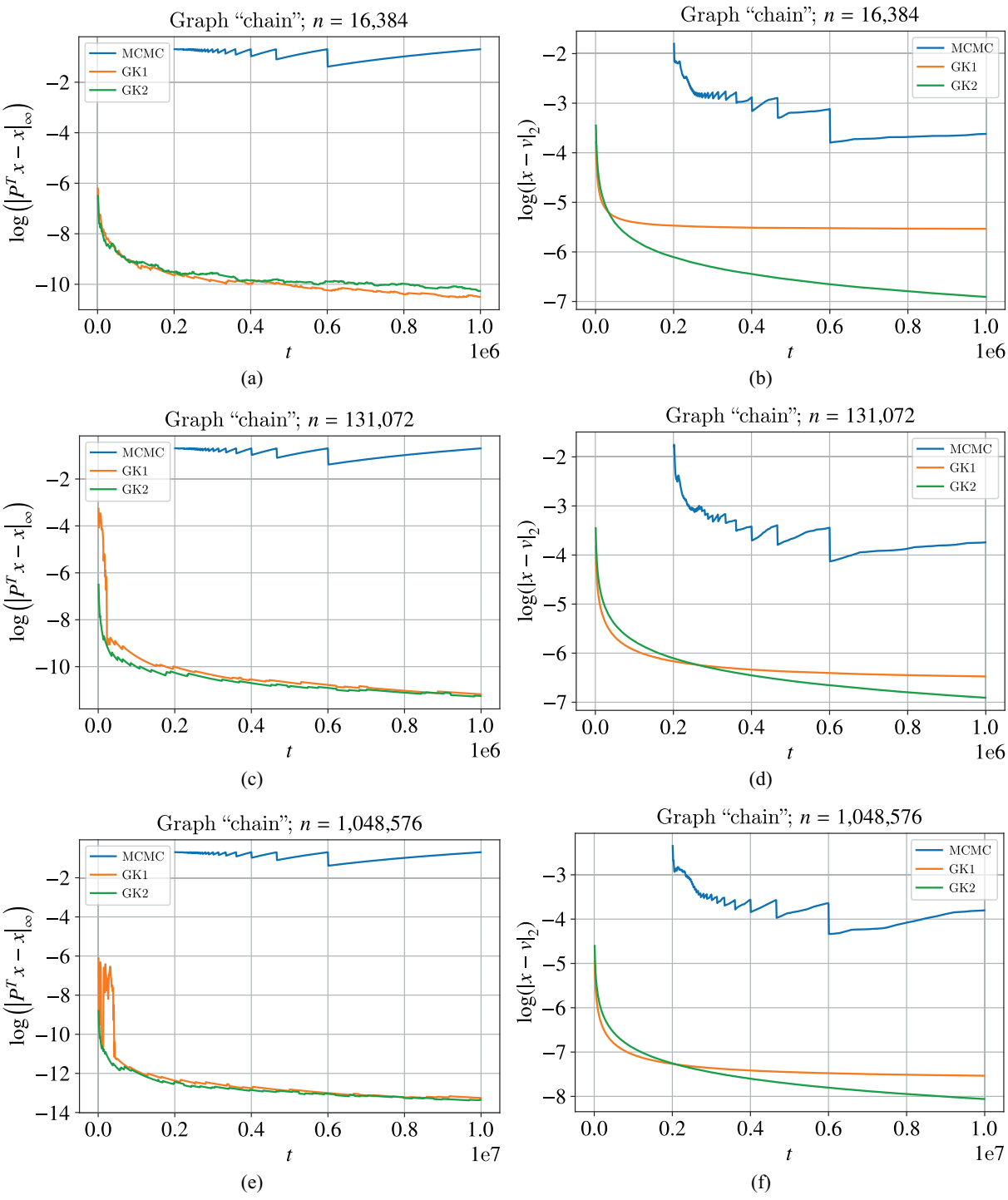

Figure 3. Comparison of the convergence rate of algorithms for chain graphs with n vertices, where a) $n = 2^{14}$; b) $n = 2^{14}$; c) $n = 2^{17}$; d) $n = 2^{17}$; e) $n = 2^{20}$; f) $n = 2^{20}$. On the Ox axis it is t — the number of iterations, Oy — $\|v - v^*\|_2$ (cases (b), (d), (f)) and $\log \left\|\left(P^T - I\right)v\right\|_\infty$ (cases (a), (c), (e)), where v is the obtained vector, and v^* is the optimal one

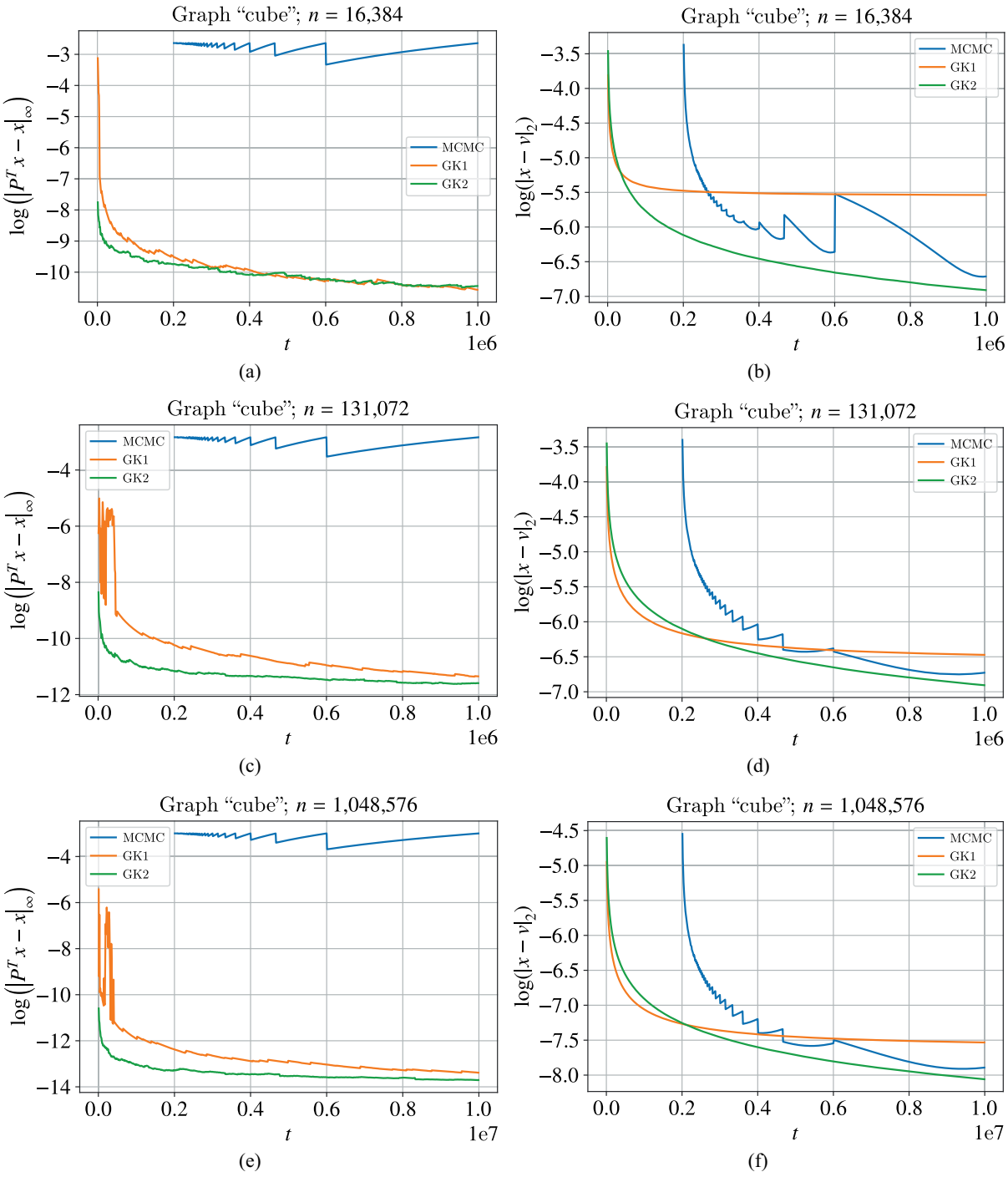

Figure 4. Comparison of the convergence rate of algorithms for cubes graphs with n vertices, where a) $n = 2^{14}$; b) $n = 2^{14}$; c) $n = 2^{17}$; d) $n = 2^{17}$; e) $n = 2^{20}$; f) $n = 2^{20}$. On the Ox axis it is t — the number of iterations, Oy — $\|v - v^*\|_2$ (cases (b), (d), (f)) and $\log \left\| \left(P^T - I\right) v \right\|_\infty$ (cases (a), (c), (e)), where v is the obtained vector, and v^* is the optimal one

In the first columns in Fig. 3 (chain graphs) and Fig. 4 (cube graph) one can see the dependence of the $\left\|\left(P^T - I\right)v\right\|_\infty$ on the number of iterations and in the second columns, the dependence of the $\|v - v^*\|_2$ on the number of iterations, where v is the obtained vector and v^* is the true PageRank vector.

Conclusion

The experiments have shown the difference between the Grigoriadis – Khachiyan algorithm and other methods: they confirmed that on sparse graphs with extremely small value of spectral gap the Grigoriadis – Khachiyan algorithm works much better than MCMC. To our best knowledge, we are the first to implement the Grigoriadis – Khachiyan algorithm, at least in the scientific sphere, and we hope that our experience may be helpful to others.

References

Гасников А. В., Дмитриев Д. Ю. Об эффективных рандомизированных алгоритмах поиска вектора PageRank // Журнал вычислительной математики и математической физики. — 2015. — Т. 55, № 3. — С. 355–371.
Gasnikov A. V., Dmitriev D. Yu. Ob effektivnykh randomizirovannykh algoritmakh poiska vektora PageRank [About effective randomized algorithms for searching for the PageRank vector] // Journal of Computational Mathematics and Mathematical Physics. — 2015. — Vol. 55, No. 3. — P. 355–371 (in Russian).

Ермаков С. М. Метод Монте-Карло в вычислительной математике: ввод. курс. — СПб.: Невский Диалект, Бином, Лаборатория знаний, 2009.
Ermakov S. M. Metod Monte-Karlo v vychislitel'noi matematike: vvod. kurs [The Monte Carlo method in computational mathematics: an introduction course]. — St. Petersburg: Nevsky Dialect, Binom, Laboratory of Knowledge, 2009 (in Russian).

Хачиян Л. Г. Избранные труды / сост. С. П. Тарасов. — М.: МЦНМО, 2009. — С. 38–48.
Khachiyan L. Izbrannye trudy [Selected works] / comp. S. P. Tarasov. — Moscow: MCNMO, 2009. — P. 38–48 (in Russian).

Anikin A. et al. Efficient numerical methods to solve sparse linear equations with application to pagerank // Optimization Methods and Software. — 2022. — Vol. 37, No. 3. — P. 907–935.

Brin S., Page L. The anatomy of a large-scale hypertextual web search engine // Computer networks and ISDN systems. — 1998. — Vol. 30, No. 1–7. — P. 107–117.

Khintchine A. Über einen satz der wahrscheinlichkeitsrechnung // Fundamenta Mathematicae. — 1924. — Vol. 6, No. 1. — P. 9–20.

Mises R. V., Pollaczek-Geiringer H. Praktische verfahren der gleichungsauflösung // ZAMM-Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik. — 1929. — Vol. 9, No. 1. — P. 58–77.

Nemirovski A. et al. Robust stochastic approximation approach to stochastic programming // SIAM Journal on optimization. — 2009. — Vol. 19, No. 4. — P. 1574–1609.

Seneta E. Non-negative matrices and Markov chains. — 2nd ed. — 1981.