Shifted power-GMRES method accelerated by extrapolation for solving PageRank with multiple damping factors

Luca Lombardo

Abstract

In the years following its publication in 1998, the PageRank model has been studied deeply to be extended in fields such as chemistry, biology and social network analysis. The aim of this project is the implementation of a modified version of the Power method to solve the PageRank problem with multiple damping factors. The proposed method is based on the combination of the Power method with the shifted `GMRES` method.

Contents

1 Introduction

The PageRank model was proposed by Google in a series of papers to evaluate accurately the most important web-pages from the World Wide Web matching a set of keywords entered by a user. For search engine rankings, the importance of web-pages is computed from the stationary probability vector of the random process of a web surfer who keeps visiting a large set of web-pages connected by hyperlinks. The link structure of the World Wide Web is represented by a directed graph, the so-called web link graph, and its corresponding adjacency matrix $G \in \mathbb{N}^{n\times n}$ where n denotes the number of pages and G_{ij} is nonzero (being 1) only if the *jth* page has a hyperlink pointing to the *ith* page. The transition probability matrix $P \in \mathbb{R}^{n\times n}$ of the random process has entries as described in 1.

$$P(i,j) = \begin{cases} \dfrac{1}{\sum_{k=1}^{n} G_{kj}} & \text{if } G_{i,j} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The entire random process needs a unique stationary distribution. To ensure this propriety is satisfied , the transition matrix P is usually modified to be an irreducible stochastic matrix A (called the Google matrix) as follows:

$$A = \alpha\tilde{P} + (1-\alpha)\,ve^T \tag{2}$$

In 2 we have defined a new matrix called $\tilde{P} = P + vd^T$ where $d \in N^{n\times 1}$ is a binary vector tracing the indices of the damping web pages with no hyperlinks, i.e., $d(i) = 1$ if the *i-th* page has no hyperlink, $v \in \mathbb{R}^{n\times n}$ is a probability vector, $e = [1,1,...,1]^T$ and $0 < \alpha < 1$, the so-called damping factor that represents the probability in the model that the surfer transfer by clicking a hyperlink rather than other ways. Mathematically, the PageRank model can be formulated as the problem of finding the positive unit eigenvector x (the so-called PageRank vector) such that

$$Ax = x, \quad \|x\| = 1, \quad x > 0 \tag{3}$$

or, equivalently, as the solution of the linear system

$$(I - \alpha\tilde{P})x = (1-\alpha)\,v \tag{4}$$

The authors of the paper [1] emphasize how in the in the past decade or so, considerable research attention has been devoted to the efficient solution of problems 3 4, especially when n is very large. For moderate values of the damping factor, e.g. for $\alpha = 0.85$ as initially suggested by Google for search engine rankings, solution strategies based on the simple Power method have proved to be very effective. However, when α approaches 1, as is required in some applications, the convergence rates of classical stationary iterative methods including the Power method tend to deteriorate sharply, and more robust algorithms need to be used.

In the reference paper that we are using for this project, the authors focus their attention in the area of PageRank computations with the same network structure but multiple damping factors. For example, in the Random Alpha PageRank model used in the design of anti-spam mechanism [2], the rankings corresponding to many different damping factors close to 1 need to be computed

simultaneously. They explain that the problem can be expressed mathematically as solving a sequence of linear systems

$$(I - \alpha_i \tilde{P}) x_i = (1 - \alpha_i) v \quad \alpha_i \in (0,1) \quad \forall i \in \{1,2,...,s\} \tag{5}$$

As we know, standard PageRank algorithms applied to 5 would solve the s linear systems independently. Although these solutions can be performed in parallel, the process would still demand large computational resources for high dimension problems. This consideration motived the authors to search novel methods with reduced algorithmic and memory complexity, to afford the solution of larger problems on moderate computing resources. They suggest to write the PageRank problem with multiple damping factors given at once 5 as a sequence of shifted linear systems of the form:

$$\left(\frac{1}{\alpha_i} I - \tilde{P}\right) x^{(i)} = \frac{1 - \alpha_i}{\alpha_i} v \quad \forall i \in \{1,2,...,s\} \quad 0 < \alpha_i < 1 \tag{6}$$

We know from literature that the Shifted Krylov methods may still suffer from slow convergence when the damping factor approaches 1, requiring larger search spaces to converge with satisfactory speed. In [1] is suggest that, to overcome this problem, we can combine stationary iterative methods and shifted Krylov subspace methods. They derive an implementation of the Power method that solves the PageRank problem with multiple dumpling factors at almost the same computational time of the standard Power method for solving one single system. They also demonstrate that this shifted Power method generates collinear residual vectors. Based on this result, they use the shifted Power iterations to provide smooth initial solutions for running shifted Krylov subspace methods such as `GMRES`. Besides, they discuss how to apply seed system choosing strategy and extrapolation techniques to further speed up the iterative process.

1.1 Overview of the classical PageRank problem

The Power method is considered one of the algorithms of choice for solving either the eigenvalue 3 or the linear system 4 formulation of the PageRank problem, as it was originally used by Google. Power iterations write as

$$x_{(k+1)} = A x_k = \alpha \tilde{P} x_{(k)} + (1 - \alpha) v \tag{7}$$

The convergence behavior is determined mainly by the ratio between the two largest eigenvalues of A. When α gets closer to 1, though, the convergence can slow down significantly.

As stated in [1] The number of iterations required to reduce the initial residual down to a tolerance τ, measured as $\tau = \|Ax_k - x_k\| = \|x_{k+1} - x_k\|$ can be estimated as $\frac{\log_{10}\tau}{\log_{10}\alpha}$. The authors provide an example: when $\tau = 10^{-8}$ the Power method requires about 175 steps to converge for $\alpha = 0.9$ but the iteration count rapidly grows to 1833 for $\alpha = 0.99$. Therefore, for values of the damping parameter very close to 1 more robust alternatives to the simple Power algorithm should be used.

2 The shifted power method for PageRank computations

In this section we'll see the extensions of stationary iterative methods for the solution of PageRank problems with multiple damping factors, as presented in [1]. We are interested in knowing if, for each method, there exists an implementation such that the computational cost of solving the PageRank problem with multiple damping factor is comparable to that of solving the ordinary PageRank problem with single damping factor.

2.1 The implementation of the shifted power method

Inspired by the reason why shifted Krylov subspaces can save computational cost, the authors of [1] investigate whether there are duplications in the calculations of multiple linear systems in this problem class by the stationary iterative methods, so that the duplications in the computation can be deleted and used for all systems. It's some sort of dynamic programming approach. Firstly, they analyze the Power method applied to the sequence of linear systems in 4. It computes at the *k-th* iteration approximate solutions $x_k^{(i)} (1 \leq i \leq s)$ of the form

$$x_k^{(i)} = \alpha_i^k \tilde{P}^k x_k^{(i)} + (1 - \alpha_i^k) \sum_{j=0}^{k-1} \alpha_i^j \tilde{P}^j v \tag{8}$$

If the s systems in 4 are solved synchronously, this means that all the $x_k^{(i)}$ are computed only after all previous approximations $x_{k-1}^{(j)}$ are available. We can now rearrange the computation efficiently as reported in [1]:

- at the first iterations
  - compute and store $\mu_1 = \tilde{P} x_0$ and $\mu_2 = v$;
  - compute and store $x_1^{(i)} = \alpha_i \mu_1 + (1 - \alpha_i)\mu_2$;
- at any other subsequent iteration $k > 1$
  - compute and store $x_k^{(i)} := (1 - \alpha_i) \sum_{j=0}^{k-2} \alpha_i^j \tilde{P}^j v = x_{k-1}^{(i)} - \alpha_i^{k-1} \mu_1$;
  - compute and store $\mu_1 = \tilde{P}\mu_1$ and $\mu_2 = \tilde{P}\mu_2$;
  - compute and store $x_k^{(i)} = \alpha_i \mu_1 + x_k^{(i)} + (1 - \alpha_i)\alpha_i^{k-1}\mu_2$.

This implementation requires at most 2 matrix-vector products at each step, which is a significant gain compared to the s matrix-vector products required by the standard Power method to compute $x_{k+1}^{(i)}$, especially when $s \gg 2$.

This was of course still a theoretical explanation. An efficient implementation can be written to compute and store $\mu = \tilde{P} v - v$ at the first iteration and then store

$$\mu = \tilde{P}^{k-1}(\tilde{P} v - v) = \tilde{P} \cdot (\tilde{P}^{k-2}(\tilde{P} v - v))$$

at each *k-th* iteration ($k > 1$), and then from each approximate solution as $x_k^{(i)} = \alpha_i^k \mu + x_{k-1}^{(i)}$. The residual vector $r_k^{(i)}$ associated with the approximate solution $x_k^{(i)}$ has the following expression

$$r_k^{(i)} = A x_k^{(i)} - x_k^{(i)} = x_{k+1}^{(i)} - x_k^{(i)} = \alpha_i^{k+1} \tilde{P}^k (\tilde{P} v - v) \tag{9}$$

Since in general each of the s linear systems may require a different number of Power iterations to converge, the s residual norms have to be monitored separately to test the convergence.

Now we can summarize the efficient implementation of the Power method presented in this section for solving problem 4 in Algorithm 1, as reported in [1]. From now on, we'll refer to this implementation as the *Shifted-Power method*.

Algorithm 1 Shifted-Power method for PageRank with multiple damping factors

Require: $\tilde{P}$, v, τ, $\max_{mv}$, α_i $(1 \leq i \leq s)$
Ensure: mv, $x^{(i)}$, $r^{(i)}$ $(1 \leq i \leq s)$
  Compute $\mu = \tilde{P}v - v$
  Set $mv = 1$
  for $i = 1 : s$ **do**
    Compute $r^{(i)} = \alpha_i \mu$
    Compute $Res(i) = \|r^{(i)}\|$
    if $Res(i) \geq \tau$ **then**
      Compute $x^{(i)} = r^{(i)} + v$
    end if
  end for
  while $\max(Res \geq \tau)$ and $mv \leq \max_{mv}$ **do**
    compute $\mu = \tilde{P}\mu$
    $mv = mv + 1$
    for $i = 1 : s$ **do**
      if $Res(i) \geq \tau$ **then**
        Compute $r^{(i)} = \alpha_i^{k+1}\mu$
        Compute $Res(i) = \|r^{(i)}\|$
        if $Res(i) \geq \tau$ **then**
          Compute $x^{(i)} = r^{(i)} + x^{(i)}$
        end if
      end if
    end for
  end while

Where mv is an integer that counts the number of matrix-vector products performed by the algorithm. The algorithm stops when either all the residual norms are smaller than the tolerance τ or the maximum number of matrix-vector products is reached. An implementation of this algorithm written in Python is available in the github repository of this project.

3 Shifted power-GMRES method

In this section we'll cover the approach that the authors in [1] used to combine the shifted power method with the fast shifted GMRES method to create an hybrid algorithm for solving complex PageRank problems with multiple damping factors.

3.1 Restarted GMRES method

The Restarted GMRES method (hereafter referred to as GMRES in short) is a non-symmetric Krylov subspace solver based on the Arnoldi decomposition procedure, that the authors sketch in the following algorithm

Algorithm 2 Arnoldi

Require: A, v_0, m
Ensure: $V_m, H_m, v_{m+1}, h_{m+1,m}, \beta, j$
Compute $\beta = \|v_0\|$
$v_1 = v_0/\beta$
for $j = 1 : m$ **do**
    Compute $w = Av_j$
    for $i = 1 : j$ **do**
        Compute $h_{i,j} = v_i^T w$
        Compute $w = w - h_{i,j} v_i$
    end for
    $h_{j+1,j} = \|w_i\|$
    if $h_{j+1,j} = 0$ **then**
        $m = j$,
        $v_{m+1} = 0$
        break
    else
        $v_{j+1} = w/h_{j+1,j}$
    end if
end for

Where $A \in \mathbb{R}^{n\times n}$ and $v_0 \in \mathbb{R}^{n\times 1}$ is the initial vector. After m iterations, the Arnoldi procedure produces the orthogonal basis $V_m = [v_1, \ldots, v_m]$ and the upper Hessenberg matrix $H_m \in \mathbb{R}^{m\times m}$, and the residual vector $v_{m+1} \in \mathbb{R}^{n\times 1}$ and the residual norm $h_{m+1,m} \in \mathbb{R}$. Starting from $v_0 = b - Ax_0$ with an initial guess x_0, after running m steps of the algorithm 2, the GMRES method produces the approximate solution $\tilde{x}$ of the linear system $Ax = b$ that minimizes the residual norm $\|b - Ax\|$ in the Krylov subspace of dimension m.

We know that the accuracy of the approximate solution $\tilde{x}$ of GMRES depends heavily on the dimension m of the search space. The authors in [1] propose to use the GMRES method as a preconditioner for the shifted power method presented in the previous section. The core idea of the method is to run standard GMRES on a seed system and to approximate the other solutions as by products. The theoretical basis is the shift-invariance property of the Krylov subspace that enables us to use only one Krylov subspace for all the shifted systems, provided that the residual vectors

are collinear to one other. The algorithm proposed by the authors is presented in Algorithm 3.

Algorithm 3 Shifted GMRES

Require: $\tilde{P}, v, m, \alpha_i, maxit, x_0^i \ \ (1 \le i \le s)$
Ensure: $x^i, res_i \ \ (1 \le i \le s), mv$
1: Set $\frac{i}{0} = \frac{1-\alpha_i}{\alpha_i} v - \left(\frac{1}{\alpha_i} I - \tilde{P}\right) x_0^i$, iter = 1
2: Set $res_i = \alpha_i \|v\| \ \ (1 \le i \le s)$
3: Set mv = 0
4: **while** max$(res_i) \ge \tau$ && $iter \le maxit$ **do**
5: Find k that satisfies $res_k = \max(res_i)$
6: Compute $\gamma^i = \frac{res_i \alpha_k}{res_k \alpha_i}$ for all $i \ne k$
7: Run Arnoldi by $[V_m, \bar{H}_m^k, v_{m+1}, \bar{h}_{m+1,m}, \beta, j] = Arnoldi(\frac{1}{\alpha_k} I - \tilde{P}, r_0^k, m)$
8: Set $mv = mv + j$
9: Compute y_k, the minimizer of $\|\beta e_1 - \bar{H}_m^k y_k\|_2$
10: Compute $x^k = x_0^k + V_m y_k$
11: Compute $res_k = \alpha_k \|\beta e_1 - \bar{H}_m^k y^k\|$
12: **for** i = 1, 2, ..., k-1, k+1, ..., s **do**
13: **if** $res_i \ge \tau$ **then**
14: Set $\bar{H}_m^i = \bar{H}_m^k + \left(\frac{1-\alpha_i}{\alpha_i} - \frac{1-\alpha_k}{\alpha_k}\right) I_m$
15: Solve y_i and γ_i from $\begin{bmatrix} \bar{H}_m^i & z \end{bmatrix} \begin{bmatrix} y^i \\ \gamma^i \end{bmatrix} = \gamma^i \beta e_1$
16: Set $x^i = x_0^i + V_m y^i$
17: Set $res_i = \frac{\alpha_i}{\alpha_k} \gamma_k^i res_k$
18: **end if**
19: **end for**
20: Set $iter = iter + 1$
21: Set $x_0^i = x^i$
22: **end while**

Where $z = \beta e_1 - H_m^1 y_m^1$. In line 15, by solving this small size system, we can obtain the vector y_m^i and scalar γ_m^i that ensures the collinearity of the shifted results.

Problems: The implementation of this algorithm has been very problematic. The key of this algorithm is the use of the *seed choosing strategy* described in [1]. However, during my tests, after the second iteration, the k value remains the same and the res vector does not change. This leads obviously to a stall situation, where the program runs without updating the values until it reaches the maximum number of iterations allowed. This problem is still under investigation. I have provided anyway a notebook in the github repository with the code of the algorithm for completeness, even if it's still not working. I think that the problem is related to some misunderstanding of the algorithm provided in the pseudo-code, but I have not been able to find it yet. For this reason, there won't be any tests results for this algorithm in the following section.

4 Numerical experiments

In this experiment, we test the performance of the shifted Power method against the conventional Power method for solving PageRank problems with multiple damping factors, namely $\{\alpha_1 = 0.85, \alpha_2 = 0.86, \ldots, \alpha_{15} = 0.99\}$ on the `web-stanford` and `web-BerkStan` datasets. The `web-stanford` dataset is a directed graph with $|V| = 281,903$ nodes and $|E| = 1,810,314$ edges, and the `web-BerkStan` dataset is a directed graph with $|V| = 1,013,320$ nodes and $|E| = 5,308,054$ edges. The datasets are available at `http://snap.stanford.edu/data/web-Stanford.html` and `http://snap.stanford.edu/data/web-BerkStan.html` respectively. The datasets are stored in the `.txt` edge-list format. The characteristics of the datasets are summarized in Table 1.

Dataset	Nodes	Edges	Density
`web-Stanford`	281,903	2,312,497	2.9099×10^{-5}
`web-BerkStan`	685,230	7,600,595	1.6187×10^{-5}

Table 1: Summary of the datasets used in the experiments.

The personalization vector v has been set to $v = [1,1,...,1]^T/n$. All the experiments are run in Python 3.10 on a 64-bit Arch Linux machine with an AMD Ryzen™ 5 2600 Processor and 16 GB of RAM.

4.1 Technical details

GitHub repository of this project

`https://github.com/lukefleed/ShfitedPowGMRES`

In the project github repository we can find an `algo.py` file where all the functions used in the experiments are implemented. The `algo.py` file contains the following functions:

load_data This function loads the datasets from the `.txt` edge-list format and returns a networkx graph object. It takes as input a literal, the options are `web-stanford` and `web-BerkStan`.

pagerank This function computes the PageRank vector of a given graph. It takes as input the following parameters:

- `G:` a networkx graph object.
- `alpha:` Damping parameter for PageRank, default=0.85.
- `personalization:` The "personalization vector" consisting of a dictionary with a key some subset of graph nodes and personalization value each of those. At least one personalization value must be non-zero. If not specified, a nodes personalization value will $1/N$ where N is the number of nodes in `G`.
- `max_iter:` The maximum number of iterations in power method eigenvalue solver. Default is 200.

- `nstart`: Starting value of PageRank iteration for each node. Default is $None$.
- `tol`: Error tolerance used to check convergence in power method solver. Default is 10^{-6}.
- `weight`: Edge data key corresponding to the edge weight. If None, then uniform weights are assumed. Default is $None$.
- `dangling`: The outedges to be assigned to any "dangling" nodes, i.e., nodes without any outedges. The dict key is the node the outedge points to and the dict value is the weight of that outedge. By default, dangling nodes are given outedges according to the personalization vector (uniform if not specified).

This function is strongly based on the `pagerank_scipy` function of the networkx library.

shifted_pow_pagerank : This is the implementation of the algorithm 1 with the difference that I am using the $l1$ norm since the $l2$ norm is still not implemented for sparse matrices in SciPy.

There are is also another function called `pagerank_numpy`. The eigenvector calculation uses NumPy's interface to the `LAPACK` eigenvalue solvers. This will be the fastest and most accurate for small graphs. Unfortunately, the eigenvector calculation is not stable for large graphs. Therefore, the `pagerank_numpy` function is not used in the experiments.

4.2 Convergence results for the Shifted Power method

In the PageRank formulation with multiple damping factors, the iterative solution of each $i-th$ linear system is started from the initial guess $x_0^{(i)} = v$ and it's stopped when either the solution $x_k^{(i)}$ satisfies

$$\frac{\|(1-\alpha_i)v - (I-\alpha_i \tilde{P} x_k^{(i)}\|_2}{\|x_k^{(i)}\|_2} < 10^{-6}$$

or the number of matrix-vector products exceeds 200.

In this experiment we test the performance of the shifted Power method against the conventional Power method for solving PageRank problems with multiple damping factors.

Dataset	Method	CPU Time (s)	mv
web-Stanford	Power	71.7	70
web-Stanford	Shifted Power	665.4	56
web-BerkStan	Power	202.1	49
web-BerkStan	Shifted Power	1342.9	73

Table 2: Summary of the experiments.

The results presented on table 2 are a bit in contrast compared to what the paper [1] reports. In their experiment the CPU time of the shifted power method is lower then the one of the standard power method. However, in our experiments the CPU time of the shifted power method is far higher then the one of the standard power method. Furthermore, theoretically, the number of matrix-vector products should be lower for the shifted power method, in particular it should be

equal to the one of the standard PageRank algorithm with the biggest damping factor. However, in our experiments the number of matrix-vector products is higher for the shifted power method for the dataset `web-BerkStan` and lower for the dataset `web-Stanford`.

The reasons to those differences in results may be a lot. I think that the most plausible reason is the difference in programming language and implementation, combined with a possibility of misunderstanding of the pseudo-code presented in [1]. My standard PageRank function is a slightly modified version of the network library function `pagerank_scipy`, so I suppose that is better optimized in comparison to the shifted power method implementation that I wrote. Also, the network `Web-BerkStan` is very different from the `web-stanford` one. The adjacency matrix relative to the first one, has a lot of rows full of zeros in comparison to the second one (4744 vs 172). This might effect negatively the shifted power method for this specific cases of networks with a lot of dangling nodes.

References

[1] Zhao-Li Shen, Meng Su, Bruno Carpentieri, and Chun Wen. Shifted power-gmres method accelerated by extrapolation for solving pagerank with multiple damping factors. *Applied Mathematics and Computation*, 420:126799, 2022.

[2] Paul G. Constantine and David F. Gleich. Random alpha pagerank. *Internet Mathematics*, 6(2), 1 2009.