PhD

3.º CICLO

FCUP
2018

U.PORTO

PageRank: how to accelerate its computation

Isabel Cristina da Silva Barros Rodrigues Mendes Pinto

FC

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

PhD

PageRank: how to accelerate its computation

Isabel Cristina da Silva Barros Rodrigues Mendes Pinto

Tese de Doutoramento apresentada à
Faculdade de Ciências da Universidade do Porto,
Matemática Aplicada

2018

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

PhD

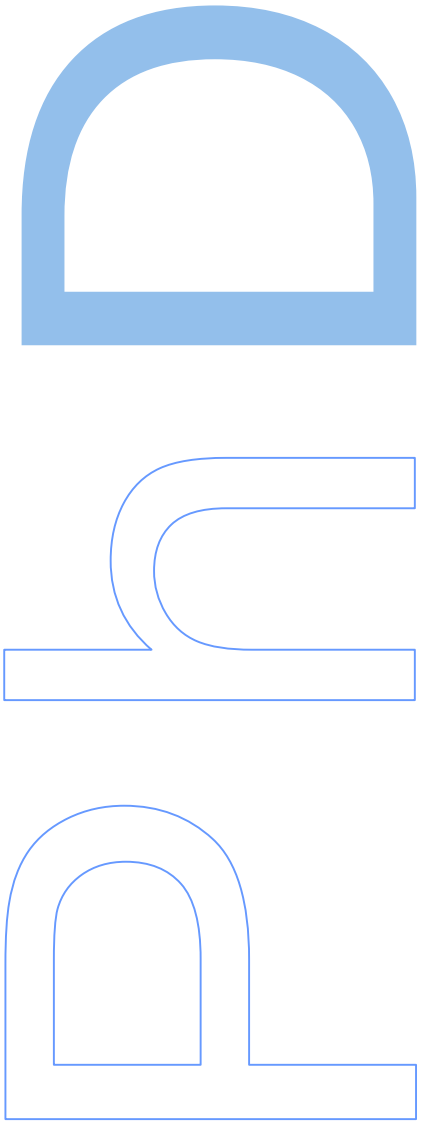
PageRank: how to accelerate its computation

Isabel Cristina da Silva Barros Rodrigues Mendes Pinto

Programa Doutoral em Matemática Aplicada
Departamento de Matemática
2018

Orientador
Paulo José Abreu Beleza de Vasconcelos,
Professor Auxiliar com Agregação, Faculdade de Economia da Universidade do Porto

Coorientador
Maria Filomena Guimarães Dias de Almeida,
Professor Catedrático, Faculdade de Engenharia da Universidade do Porto



To my daughters
Cláudia and Patrícia

Agradecimentos

Desejo expressar o meu mais vivo agradecimento a todos aqueles que contribuiram para a realização deste trabalho. No entanto, devo especial referência às seguintes individualidades e instituições:

Começo por agradecer ao meu orientador, Professor Doutor Paulo José Abreu Beleza de Vasconcelos, e à minha coorientadora, Professora Doutora Filomena Dias de Almeida, pela oportunidade que me concederam e pela orientação e compreensão durante a realização deste trabalho. Devo também agradecer a confiança em mim depositada, o apoio e o incentivo durante todo o tempo que levei a concluir este trabalho.

À Professora Doutora Alexandra Gavina pela preciosa ajuda na formatação do texto e sobretudo pelo incentivo e pela generosa disponibilização do seu tempo. Ao meu cunhado, Engenheiro Rui Fazenda pelo tempo disponibilizado e ajuda com a simulação numérica.

Ao Instituto Superior de Engenharia do Porto (ISEP), onde sou docente do Departamento de Matemática, pelas facilidades concedidas para a realização deste trabalho. Ao Laboratório de Engenharia Matemática (LEMA), onde sou investigadora, pela disponibilização de equipamento. À FCT (Formação para a Ciência e Tecnologia) pela atribuição de uma Bolsa de Doutoramento.

Às minhas colegas e amigas Doutora Alzira Faria, Engenheira Alzira Teixeira e Professora Doutora Ana Júlia Viamonte do Departamento de Matemática do ISEP pelo trabalho extra que suportaram para que eu pudesse terminar a escrita deste documento.

A todos os meus colegas e amigos que me incentivaram e apoiaram, sobretudo às Professoras Doutoras Isabel Figueiredo e Luísa Hoffbauer, ao Professor Doutor José Matos e à Doutora Sandra Aires.

Agradeço ainda a toda a minha família, aos meus pais, às minhas irmãs e, sobretudo, ao meu marido Arnaldo Pinto e às minhas duas filhas Cláudia e Patrícia pela motivação e pelo tempo que me concederam para conseguir concretizar este objetivo.

PageRank: como acelerar o cálculo

Resumo

A recuperação de informações na Web é extremamente desafiadora devido ao grande número de páginas da web. PageRank é um método numérico que o motor de busca Google usa para calcular a importância de uma página, atribuindo uma pontuação a cada página da web. O PageRank é, portanto, a base do sucesso do motor de busca Google e pode ser tratado matematicamente como um problema de valores próprios ou como a solução de um sistema linear homogéneo.

De um ponto de vista dos valores próprios, o vetor PageRank é o vetor próprio dominante à esquerda de uma matriz da web que está relacionada com a estrutura de hiperligações da web, a matriz Google. O método da Potência Iterada é um dos mais antigos e mais simples métodos iterativos para encontrar o valor próprio e o vetor próprio dominante de uma matriz e foi o método original proposto por Brin e Page para encontrar o vetor PageRank. O cálculo do vetor PageRank pelo método da Potência Iterada leva dias para convergir, uma vez que as matrizes envolvidas são grandes. O número de páginas da web está a aumentar rapidamente, portanto, torna-se necessário encontrar refinamentos ou alternativas para acelerar a computação.

A matriz do Google é grande e esparsa, portanto, algoritmos adaptados devem ser desenvolvidos para conseguir um menor custo computacional e requisitos de memória mínimos. Além disso, a precisão do vetor PageRank não necessita de ser grande, pelo que são preferidos métodos iterativos de baixo custo.

Entre as abordagens mais bem-sucedidas para reduzir o trabalho associado ao vetor PageRank estão as técnicas de extrapolação [78] e mais recentemente, os métodos Lumping [67, 93] que reordenam a matriz de acordo com o tipo de nós.

A primeira parte deste trabalho apresenta uma nova abordagem para a aceleração do cálculo do PageRank, combinando técnicas de reordenação e ex-

trapolação. São propostos dois algoritmos, denominados métodos LumpingE, considerando a extrapolação de Aitken dentro do método original de agregação. Simulações numéricas comparando os novos métodos LumpingE com o método da Potência e os métodos Lumping originais são ilustradas. Os resultados mostram o mérito desta nova proposta.

A segunda parte deste trabalho trata o problema do cálculo do vector do PageRank através da resolução de um sistema de equações linear homogéneo. O método iterativo recente denominado "Matrix Analogue of the Accelerated Overrelaxation (MAAOR) iterative method" [58], que contém como casos particulares, o método "Accelerated Overrelaxation (AOR)" [54] e a família de métodos "Generalized AOR (GAOR)" [69], é usado para o cálculo do vetor PageRank. Além disso, os métodos de Lumping que foram aplicados à formulação do problema de valores próprios também podem ser usados na formulação do sistema linear [67, 93]. Portanto, é proposta uma nova abordagem combinando os métodos Lumping e MAAOR para a solução do sistema linear. São apresentadas simulações numéricas que ilustram o método MAAOR e o método MAAOR combinados com técnicas de Lumping aplicadas aos cálculos do PageRank.

Palavras Chave: Métodos diretos para sistemas lineares; Métodos iterativos para sistemas lineares; Valores próprios, vetores próprios; Precondicionadores para métodos iterativos; Computação de cadeias de Markov; Matrizes esparsas.
MSC(2010): 65F05; 65F10; 65F10; 65F08; 65C40; 65F50.

PageRank: how to accelerate its computation

Abstract

Web information retrieval is extremely challenging due to the huge number of web pages. PageRank is a numerical method that Google uses to compute a page's importance, by assigning a score to every web page. PageRank is thus at the basis of Google's search engine success and can be mathematically regarded either as an eigenvalue problem or as the solution of a homogeneous linear system.

From an eigenvalue point of view the PageRank vector is the left dominant eigenvector of a web matrix that is related to the hyperlink structure of the web, the Google matrix. The Power method is one of the oldest and simplest iterative methods for finding the dominant eigenvalue and eigenvector of a matrix and it was the original method proposed by Brin and Page for finding the PageRank vector. The PageRank computation by the standard Power method takes days to converge since matrices involved are large. The number of web pages is increasing rapidly, so, it becomes necessary to find refinements or alternatives to speed up the computation.

The Google matrix involved is large and sparse, so tuned algorithms must be developed to tackle it with the lowest computational cost and minimum memory requirements. Furthermore, the accuracy of the ranking vector needs not to be very precise, so inexpensive iterative methods are preferred.

Among the most successful approaches for reducing the work associated with the PageRank vector are the extrapolation techniques [78], and the more recent Lumping methods [67, 93] that proceed with a matrix reordering according to dangling and nondangling nodes.

The first part of this work presents a novel approach for the acceleration of the PageRank computation by combining reordered and extrapolation techniques. Two algorithms, called LumpingE methods, considering standard

Aitken extrapolation within the original lumping method are proposed. Numerical experiments comparing the new LumpingE methods with standard Power method and original Lumping methods are illustrated. Results show the merits from this new proposal.

The second part of this work uses the homogeneous linear system approach of the PageRank vector computation problem. The recent Matrix Analogue of the Accelerated Overrelaxation (MAAOR) iterative method [58], which contains as particular cases the Accelerated Overrelaxation (AOR) [54] and the Generalized AOR (GAOR) [69] stationary family of methods is explored for the PageRank computation. Additionally, the Lumping methods that have been applied to the eigenproblem formulation can also be used in the linear system formulation [67, 93]. Therefore, a novel approach combining the Lumping and MAAOR methods for the solution of the linear system is proposed. Numerical experiments illustrating the MAAOR method and the MAAOR method combined with Lumping techniques applied to PageRank computations are presented.

Keywords: Direct methods for linear systems; Iterative methods for linear systems; Eigenvalues, eigenvectors; Preconditioners for iterative methods; Computational Markov chains; Sparse matrices.
MSC(2010): 65F05; 65F10; 65F10; 65F08; 65C40; 65F50.

PageRank: comment accélérer son calcul

Résumé

La récupération de l'information Web est un problème extrêmement difficile en raison du grand nombre de pages Web. PageRank est une méthode numérique utilisée par Google pour calculer l'importance d'une page, en attribuant à cette dernière un numéro. PageRank est donc à la base du succès des moteurs de recherche de Google et peut être vu mathématiquement soit comme un problème de valeurs propres, soit comme la solution d'un système linéaire homogène.

Du point de vue spectral, le vecteur PageRank est le vecteur propre dominant à gauche d'une matrice web liée à la structure de l'hyperlien du Web, la matrice de Google. La méthode de la puissance itérée est l'une des méthodes itératives les plus anciennes et les plus simples pour trouver la valeur propre dominante et un vecteur propre correspondant d'une matrice. C'est la méthode originale proposée par Brin et Page pour trouver le vecteur PageRank. Le calcul du vecteur PageRank par la méthode de la puissance itérée standard prend des jours pour converger puisque les matrices impliquées sont beaucoup trop grandes. Le nombre de pages web augmente rapidement, il devient donc nécessaire de trouver des améliorations ou des alternatives pour accélérer le calcul.

La matrice de Google dont il s'agit est grande et creuse, donc des algorithmes conséquents doivent être développ'es pour l'attaquer avec le coût de calcul le plus bas possible et des besoins de mémoire réduits au minimum. Comme la précision du vecteur de classement n'est pas très importante des méthodes itératives peu coûteuses sont préférées.

Les techniques d'extrapolation [78] et les méthodes Lumping plus récentes [67, 93] qui reordonnent la matrice selon si les noeuds sont dangling ou non-dangling, sont parmi les approches les plus efficaces pour réduire le travail associé au vecteur PageRank.

La première partie de ce travail présente une nouvelle approche pour accélérer le calcul du PageRank en combinant des techniques de réordination et d'extrapolation. Deux algorithmes, appelés méthodes LumpingE, considérant l'extrapolation standard d'Aitken dans la méthode d'agglomération originale sont proposés. Des expériences numériques comparant les nouvelles méthodes de LumpingE avec la méthode Power standard et les méthodes Lumping originales sont illustrées. Les résultats montrent les mérites de cette nouvelle idée.

La deuxième partie de ce travail regarde le calcul du vecteur de PageRank comme résolution d'un système linéaire homogène. L'analogue matriciel récent de la méthode itérative accélérée (MAAOR) [58], qui contient comme cas particuliers la méthode de relaxation stationnaire accélérée (AOR) [54] et la famille de méthodes stationnaires généralisée AOR (GAOR) [69] sont explorées pour le calcul du vecteur PageRank. De plus, les méthodes de Lumping qui ont été appliquées à la formulation du problème spectral peuvent également être utilisées dans la formulation de système linéaire [67, 93]. Par conséquent, une nouvelle approche combinant les méthodes Lumping et MAAOR pour la solution du système linéaire est proposée. Des expériences numériques illustrant la méthode MAAOR et la méthode MAAOR combinées aux techniques de Lumping appliquées aux calculs PageRank sont présentées.

Mots-clés: Les méthodes directes pour les systèmes linéaires; Méthodes itératives pour les systèmes linéaires; valeurs propres, vecteurs propres; Préconditionnements pour les méthodes itératives; Simulation de chaînes de Markov; Matrices creuses.
MSC(2010): 65F05; 65F10; 65F10; 65F08; 65C40; 65F50.

List of Figures

List of Tables

List of Algorithms

Contents

1

Introduction

1.1 The Google's search engine

Search engine use is one of the most popular internet activities. There are many internet search engines, but Google and Bing receive the large majority of all search requests. The search engine Google was founded in 1998 by Larry Page and Sergey Brin.

Google is, by far, the most popular search engine in the Internet. What set Google apart from its competitors in the first place? The answer is PageRank.

The importance of PageRank is emphasized in one of Google's web pages: *"The heart of our software is PageRank^{TM}, a system for ranking web pages developed by our founders Larry Page and Sergey Brin at Stanford University. And while we have dozens of engineers working to improve every aspect of Google on a daily basis, PageRank continues to provide the basis for all of our web search tools."*

Other search engines relied entirely on web page content to rank the results; due to that, web pages developers could easily manipulate the ordering of search by placing concealed information on web pages. Page and Brin developed a ranking algorithm (the PageRank) that solved that problem. They used the link structure of the Web to determine the importance of a web page. During the processing of a query, Google's search algorithm combined precomputed PageRank scores with text matching scores to obtain an overall ranking for each web page.

PageRank is a numerical algorithm that assigns a weight (from 0 to 10) to each element of a hyperlinked set of documents, based on a link analysis.

Its aim is to order the relative importance of web documents based on a web graph (within the World Wide Web or other set). The weight of a page, PageRank weight, is computed recursively and depends on the number and PageRank weight of all incoming links.

The business community knows that Google is the search engine of choice, so to maximize the PageRank score of its web page has become an important part of company marketing strategies.

During the last two decades the scientific community was very active in studying and presenting numerical approaches to solve the problem as fast as possible and with the less computational cost possible. A plethora of publications have been produced on this topic. The original paper [20], *"Anatomy of a Large-Scale Hypertextual Web Search Engine"*, written by Google founders Larry Page and Sergey Brin, was the seed that led to all this work. In fact, the PageRank algorithm is much more complex, involving other topics such as author rank, spammers' control, among others. Other link based ranking algorithms have been developed, and still are, but Google's PageRank is certainly, until now, the most successful and studied of them all.

1.2 Some notes about Google's history

Sergey Brin and Larry Page, Google founders, first met in 1995 when Page visited the computer science department of Stanford University during a recruitment weekend. Page has graduated from the University of Michigan and was considering Stanford for a PhD in computer science. His tour guide was Brin a second-year graduate student who was already pursuing his PhD in that department. Despite their common interests, they did not immediately hit it off; in fact, they discussed many topics during their first meeting and disagreed on nearly every issue.

Page elected to attend Stanford, and soon began working on a Web project, called BackRub, which investigated how sites linked back to other web pages. Brin found Page's work on BackRub interesting, so the two started working together on this project. They believed that they were creating a search engine that adapted to the ever-increasing size of the Web, and that finding pages with more incoming links (particularly from credible websites) would be a better way to search the Internet.

In 1997, Page and Brin decided to rename BackRub, so, "Google" was born. Google is a common misspelling of the word googol, a mathematical term for the number 10^{100}. The use of this term reflects their mission to

organize the huge amount of information on the web.

They wanted to finish their PhDs, so, they tried to sell their search innovations to another company. They did get offers, but they were not for much money. Since they could not sell their technology they decided to start their own company since they were certain that their technology was superior to any being used. With the financial assistance of a small group of initial investors, Brin and Page founded the Web search engine company Google, Inc. in September 1998.

Brin and Page's project would permanently change web search, Google has become one of the most successful dot-com businesses in history [122].

1.3 Contributions

The PageRank model can be mathematically seen as an eigenvalue problem or as a homogeneous linear system.

In this work we intend to contribute with new alternatives for the PageRank computations.

In the first part of this work, the eigenvector point of view presented in Chapter 3, we propose new acceleration methods for PageRank computations, the LumpingE methods. These methods combine Lumping methods with Aitken extrapolation. Several numerical experiments on some datasets that were extracted as subsets of the Web and on tuned matrices designed, for the purpose of exploiting certain aspects, comparing the new LumpingE methods with standard Power iteration method and original Lumping methods are presented. Results show the benefits of our proposal.

In the second part of this work, Chapter 4, a linear system point of view on the PageRank problem is followed.

We apply the new Matrix Analogue of the Accelerated Overrelaxation (MAAOR) family of methods developed by Hadjidimos in [58] on the PageRank problem for the first time. Several methods within the MAAOR family are compared.

We also developed new acceleration methods analogous to the ones presented in Chapter 3, but that use the linear system formulation and the MAAOR family of methods combined with lumping methods.

Several numerical experiments with the same matrices used in Chapter 3 show the good results obtained by these new methods.

The `MATLAB` codes developed for the various methods studied are available

at request.

2

The Mathematics behind the PageRank

Contents

2.1 The PageRank model

How does the search engine Google determine the order in which to display web pages? The major ingredient in determining this order is the PageRank vector, which assigns a score to every web page. Web pages with high scores are displayed first, and web pages with low scores are displayed later [68].

Google uses the notion of a random Web surfer to describe the behavior of a person that uses Google to answer a query. The internet user randomly chooses a web page to view from the listing of available web pages. Then, if the web page as several outlinks (links from that page to other pages), the surfer chooses one at random, hyperlinks to a new page, and continuous this random decision process indefinitely. The proportion of time the surfer spends on a given page is a measure of the relative importance of that page. Pages that the surfer revisits often must be important. Sometimes the random surfer find himself in a page with no outlinks (e.g. pdf files, image files). To get out of that page the surfer will hyperlink to any page at random. Although the surfer follows the hyperlink structure of the Web, at times it gets bored and abandons the hyperlink method of surfing by entering a new destination in the browser's URL line. So, the surfer "teleports" to a new page and begins surfing again [20, 88].

To model the activity of a random Web surfer, the PageRank algorithm represents the link structure of the Web as a massive directed graph. The nodes in the graph represent web pages and the directed arcs represent the hyperlinks. Inlinks point into nodes and outlinks point out from nodes.

Figure 2.1 presents an example of a tiny web with seven pages.

Google assumes that a hyperlink is a recommendation. The hyperlink from page i to page j means that page i is recommending page j (page i votes in page j). A page with more inlinks (votes) must be more important than a page with a few inlinks. However the status of the recommender is also important, a vote from a page with a higher PageRank (an important page) is more useful than a vote from a page with low PageRank. If a page has lots of outlinks it means that this page distributes votes indiscriminately. Several outlinks diminish the PageRank of a page.

The PageRank's thesis is that *a web page is important if it is pointed to by other important pages.*

Comparing the PageRank scores for two pages gives an indication of the

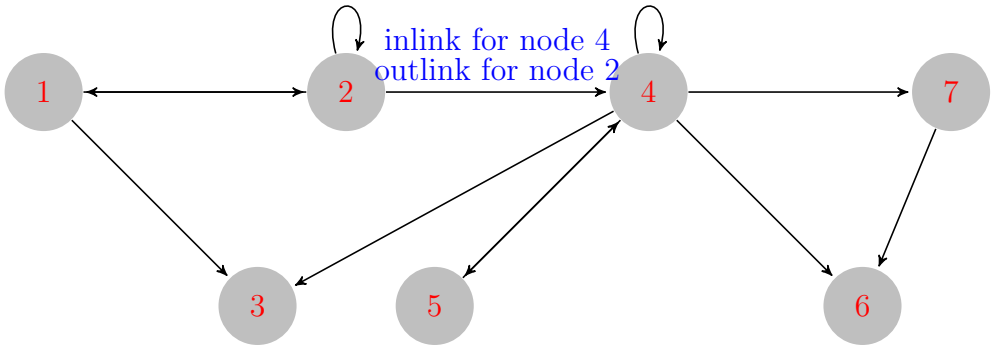


Figure 2.1: Directed graph representing a web of seven pages. The seven nodes represent seven web pages and the twelve directed arcs the hyperlinks.

relative importance of the two pages.

The PageRank of a page i, $\pi(i)$, is defined as the probability that at some particular time step the surfer is at page i.

PageRank is query-independent, which means it produces a global ranking of the importance of all pages in Google's index.

Web information retrieval is extremely challenging due to the huge number of web pages. The success of a search engine relies on its capacity to deploy fast, accurately and in order, a set of results satisfying a particular query.

To determine the order of importance in which to display web pages after a query, Google's search engine computes the PageRank vector, the left principal eigenvector of a web matrix that is related to the hyperlink structure of the web, the Google matrix.

PageRank importance scores are the stationary values of an enormous Markov chain, therefore many properties of the PageRank model are explained with Markov Theory [105].

Next it will be explained how Google translate the PageRank's thesis into mathematical equations [88].

Consider that n is the number of pages in Google's index of the Web.

The process for determining PageRank begins by expressing the directed web graph as the $n \times n$ hyperlink matrix, H. The matrix H reflects the link structure of the web and is sparse and row normalized with $h_{i,j} = \frac{1}{n_i}$ if there is a link from page i to page j and $h_{i,j} = 0$ otherwise. The number of outlinks of page i is denoted by n_i.

The elements of matrix H are probabilities: $h_{i,j}$ represents the likelihood

that a surfer follows the link from page i to page j. The nonzero elements of row i correspond to the outlinking pages of page i. The nonzero elements of column i correspond to the inlinking pages of page i.

The pages with no outlinks are called *dangling nodes*. These pages create zero rows in the H matrix. All the other rows are *stochastic* (the sum of elements in each row is 1). Thus, H is called *substochastic*.

A *nondangling node* is a page with, at least, one outlink.

As an example, let us consider the hyperlink matrix, H, for the tiny graph of Figure 2.1.

$$H = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 \end{array} \\ \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Row 3 and row 6 are zero rows, they correspond to pages without outlinks. So, page 3 and page 6 are dangling nodes.
Page 1 has two outlinks, one to page 2 ($h_{1,2} = \frac{1}{2}$) and another to page 3 ($h_{1,3} = \frac{1}{2}$). Page 5 only has one outlink to page 4 ($h_{5,4} = 1$).

Brin and Page began the calculation of the PageRank vector π (the PageRank scores for all pages of the web) considering that the PageRank of a page i is obtained by the sum of the PageRank scores of all pages that point to page i.

However, there is a problem, the PageRank of the pages inlinking to page i are unknown. To sidestep this problem it is necessary to use an iterative process:

$$\pi^{(k+1)^T} = \pi^{(k)^T} H \tag{2.1}$$

The row vector $\pi^{(k)^T}$ is the PageRank vector at the k^{th} iteration.

The iterative process is initialized (iteration 0) considering all pages of the web with the same PageRank ($\pi^{(0)}(i) = \frac{1}{n}$). The iterative process is applied until the PageRank scores converge to some final stable values.

For a dense square matrix of size n, each iteration of (2.1) requires $\mathcal{O}(n^2)$ computation because it involves one vector-matrix multiplication. A vector-matrix multiplication involving a sparse matrix requires less effort than the

$\mathcal{O}(n^2)$ for dense computation, it requires $\mathcal{O}(nnz(H))$ computation being $nnz(H)$ the number of nonzeros in H.

Matrix H is very sparse, it has about $10n$ nonzeros because the average web page has about 10 outlinks [88]. So, for matrix H, the vector-matrix multiplication of (2.1) reduces to $\mathcal{O}(n)$ effort.

The iterative process of equation (2.1) is the known Power method applied to H.

In presence of an iterative process it is necessary to know if the properties of the H matrix guarantee the convergence, if convergence occurs for just one or multiple vectors and if the convergence depends on the starting vector.

The use of H in equation (2.1) causes problems: for instance, in case of *rank sinks* (pages that monopolize the scores because they accumulate more and more PageRank at each iteration and do not share) and in case of *cycles* (pages that create an infinite loop).

From the Markov theory it is known that, for any starting vector, the Power method applied to a Markov matrix converges to a unique positive vector, the stationary vector, as long as the matrix is stochastic, irreducible and aperiodic.

If the H matrix could be altered and transformed into a Markov matrix, the convergence problems caused by cycles and rank sinks would be solved. Brin and Page did exactly that, they made adjustments to the matrix H that solved the initial convergence problems. However, they did not use the Markov theory as the justification for the adjustments; they used the notion of a random surfer described early.

Nevertheless, the Markov theory supports the random surfer idea.
If node i is a dangling node this means that the probability of a random surfer move from node i to any other node in the directed graph is zero. The majority of web pages are dangling nodes (pdf files, image files, data tables or protected pages are dangling nodes). Since H does not model the possibility of moving from dangling node web pages to other web pages, the long term behavior of web surfers cannot be determined from H alone.

This leads to the first adjustment on the H matrix.

To solve this problem the 0^T rows of the H matrix were replaced with

$$w^T = \frac{1}{n}e^T \ , \ e^T = (1, \ldots, 1)^T$$

Vector $w \geq 0$ is known as the *dangling node vector* and it is a probability

vector.

The vector d is used to identify the pages that are dangling nodes: $d_i = 1$ if page i is a dangling node ($n_i = 0$) and $d_i = 0$ otherwise.

A new matrix

$$S = H + dw^T \tag{2.2}$$

with no 0 rows is formed.

Proposition 2.1.1. *The stochastic adjustment* (2.2) *guarantees that S is stochastic, and thus, is the transition probability matrix for a Markov chain.*

Proof. S is the sum of a stochastic matrix with a real non-negative matrix. □

With this adjustment, the random surfer has a way of jumping to any web page at random after entering in a dangling nodes.

As an example of this adjustments let us return to the tiny web of figure 2.1 and modify the corresponding matrix H.

The modified stochastic matrix S for this example is:

$$S = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 \end{array} \\ \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 0 & 0 & 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Row 3 and row 6 (corresponding to dangling nodes) were modified. They are no longer 0^T rows. If a random surfer is visiting page 3 and wants to leave, he can jump to any of the seven pages of the web (each one with probability of $\frac{1}{7}$).

However, the basic model is not yet complete. Even when web pages have links to other web pages, a random web surfer might grow tired of following links and decide to move to a different web page some other way (entering a new URL in the address line). The matrix S does not consider this possibility.

When this situation happens, the random surfer jumps ("teleports") to a new page and begins hyperlink surfing again. This behavior can be represented by introducing a new adjustment in the matrix.

In terms of Markov theory, the use of the S matrix does not guarantees the convergence results desired. It is not mandatory that the equation (2.2) will rapidly converge to a unique positive vector π^T that exists.

So, the H matrix has been modified for the second time. With this new adjustment, called a *primitivity adjustment*, the matrix becomes primitive, that is, irreducible and aperiodic. This new modified matrix, G, is called *Google matrix*.

The PageRank algorithm uses the G matrix which is given by

$$G = \alpha S + (1-\alpha)E \quad \text{with} \quad E = ev^T \tag{2.3}$$

E is a rank-1 matrix (the *teleportation matrix*) and e is the vector of all ones.

The vector $v \geq 0$ is a probability vector known as *personalization* or *teleportation vector*.
It is usual to assume that $v = w = \frac{1}{n}e$. This way, the *teleportation matrix* E is uniform and the teleporting is random. That is, the surfer as the same probability to jump to any page when teleporting. This stochastic matrix E mimics the random behavior of web surfers and can be also used to combat link spamming [53, 68].

The parameter α ($\alpha \in [0, 1[$), the *damping factor*, controls the portion of time the random surfer follows the hyperlinks as opposed to teleporting.

It is generally assumed that Google uses $\alpha = 0.85$. This means that 85% of the time the random surfer follows the hyperlink structure of the Web and the other 15% of the time he teleports to a random new page.

Proposition 2.1.2. *G is a stochastic matrix (it is a convex combination of two stochastic matrices E and S), it is aperiodic ($g_{ij} > 0$, $\forall i$) and it is irreducible (every page is directly connected to every other page).*

So, G is primitive which implies that the vector π^T exists, is unique and the Power method converges to π^T. However, with these adjustments the matrix G becomes completely dense.

Proof. See [88]. □

The fact of G matrix being completely dense could constitute a problem for computation.
However, matrix G can be written as a *rank-one update* to the very sparse matrix H:

$$\begin{aligned} G &= \alpha S + (1-\alpha)E \\ &= \alpha\left(H + \tfrac{1}{n}de^T\right) + (1-\alpha)\tfrac{1}{n}ee^T \\ &= \alpha H + (\alpha d + (1-\alpha)e)\,\tfrac{1}{n}e^T \end{aligned}$$

As we will see in section 2.2, this strategy brings a substantial computational advantage to the computations.

As an example we use, again, the web of Figure 2.1.
Considering $\alpha = 0.85$,

$$G = 0.85S + 0.15E$$

$$\Leftrightarrow G = 0.85S + 0.15 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \end{bmatrix}$$

$$\Leftrightarrow G = \begin{bmatrix} 3/140 & 25/56 & 25/56 & 3/140 & 3/140 & 3/140 & 3/140 \\ 32/105 & 32/105 & 3/140 & 32/105 & 3/140 & 3/140 & 3/140 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 3/140 & 3/140 & 67/350 & 67/350 & 67/350 & 67/350 & 67/350 \\ 3/140 & 3/140 & 3/140 & 61/70 & 3/140 & 3/140 & 3/140 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 3/140 & 3/140 & 3/140 & 3/140 & 3/140 & 61/70 & 3/140 \end{bmatrix}$$

The vector $\pi^T = (\pi(1), \cdots, \pi(n))^T$ is the *PageRank vector*, and contains the PageRank scores for every web page in the Google's index. $\pi(i)$ is the PageRank of page i and $\left\|\pi^T\right\|_1 = 1$.

The PageRank vector is calculated with

$$\pi^T = \pi^T G. \tag{2.4}$$

As mentioned before, using Markov matrix G, the PageRank algorithm avoids convergence problems.

With the two adjustments made, the PageRank vector (stationary vector of the Markov chain) exists, is unique, and can be found with the Power method.

Applying the Power method (2.4), for the example in Figure 2.1 whose G matrix is shown above, we obtain Google's PageRank vector π^T

$$\begin{array}{cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \pi^T = & (0,1025 & 0,1461 & 0,1430 & 0,2254 & 0,0995 & 0,1840 & 0,0995) \end{array}$$

The result $\pi(6) = 0,1840$ means that $18,40\%$ of the time the random surfer visits page 6. So, for this web, pages are ranked by their importance as (4 6 2 3 1 5 7). Page 4 is the most important one because it has the higher PageRank and page 7 is the least important page because has the lowest PageRank.

There are two ways of solving the PageRank problem:
(i) as an *eigenvector problem*

$$\pi^T = \pi^T G \quad \pi^T e = 1$$

or (ii) as a *linear homogeneous system*

$$\pi^T (I - G) = 0^T \quad \pi^T e = 1.$$

In the first approach, the eigenvector problem, the goal is to find the normalized dominant left eigenvector, π^T, of the matrix G that corresponds to the dominant eigenvalue λ_1. G is a stochastic matrix, so $\lambda_1 = 1$. Also, $\lambda_1 = 1$ is not a repeated eigenvalue of G and is greater in magnitude than any other eigenvalue. The eigensystem, $\pi^T = \pi^T G$, has a unique solution.

In the second approach, the linear homogeneous system problem, the goal is to find the normalized left null vector of $I - G$.

In both approaches, the normalization equation $\pi^T e = 1$ ensures that π^T is a probability vector.

2.2 PageRank as an eigenvector problem

The PageRank of one page, $\pi(i)$, is given by the PageRank of the pages that link to it, and so, as mentioned before, a recursive process must be used for the evaluation,

$$\pi^{(k+1)^T} = \pi^{(k)^T} G \tag{2.5}$$

The PageRank vector is the stationary vector of a Markov chain with transition matrix G, and much research has been done on computing the stationary vector for a general Markov chain, see [127]. However, the specific features of the PageRank matrix make one numerical method, the Power method, the clear favorite.

The Power method is one of the oldest and simplest iterative methods for finding the dominant eigenvalue and associate eigenvector of a matrix.

Therefore, it can be used to find the stationary vector of a Markov chain. The stationary vector is simply the dominant left-hand eigenvector of the Markov matrix.

2.2.1 Power Method

The Power method was the original method proposed by Brin and Page for finding the PageRank vector.

Given a starting vector

$$\pi^{(0)^T} = v^T = \frac{1}{n} e^T,$$

the Power method calculates successive iterates

$$\pi^{(k+1)^T} = \pi^{(k)^T} G, \quad \text{where} \quad k = 0, 1, \ldots$$

until some convergence criterion is satisfied.

The intuition behind the convergence of the Power method is as follows. For simplicity, consider $A = G^T$, so

$$\pi^{(k+1)^T} = \pi^{(k)^T} G \quad \Leftrightarrow \quad \pi^{(k+1)} = A\pi^{(k)}$$

Assuming that matrix A has n distinct eigenvectors u_i:

$$Au_i = \lambda_i u_i$$

Then, we can write any n-dimensional vector as a linear combination of the eigenvectors of A,

$$\pi^{(0)} = u_1 + \sum_{i=2}^{n} \alpha_i u_i$$

Since the first eigenvalue of a Markov matrix is $\lambda_1 = 1$,

$$\pi^{(1)} = A\pi^{(0)} = u_1 + \sum_{i=2}^{n} \alpha_i \lambda_i u_i$$

and,

$$\pi^{(k)} = A^k \pi^{(0)} = u_1 + \sum_{i=2}^{n} \alpha_i \lambda_i^k u_i$$

Proposition 2.2.1. *Since* $1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$, $A^k\pi^{(0)}$ *approaches* u_1 *as* k *grows. Therefore, the Power method converges to the principal eigenvector of the Markov matrix* A.

Brin and Page chose the Power method to compute the PageRank vector knowing that this iterative method converges very slowly. Why did they and why is it still the predominant method in PageRank research?

Google uses the Power method for several reasons: it is simple to implement and program; it is storage-friendly (matrices S and G are not formed nor stored) because it allows the use of the sparse matrix H instead of the dense matrix G; it only stores the H matrix (and the vectors d and π); typically for the problem at hands it only needs about 50-100 power iterations to compute the PageRank vector before it converges [20]; it is matrix-free (no manipulation of the matrix is needed during each step, the coefficient matrix is only used in a vector-matrix multiplication).

The Power method applied to matrix G (equation (2.5)) can be expressed using the H matrix instead. The iteration

$$\begin{aligned} {\pi^{(k+1)}}^T &= {\pi^{(k)}}^T G \\ &= \alpha {\pi^{(k)}}^T S + \tfrac{1-\alpha}{n} {\pi^{(k)}}^T e e^T \\ &= \alpha {\pi^{(k)}}^T H + \left(\alpha {\pi^{(k)}}^T d + 1 - \alpha\right) \tfrac{e^T}{n} \end{aligned} \tag{2.6}$$

only execute vector-matrix multiplications on a sparse H matrix (each vector-matrix multiplication is $\mathcal{O}(n)$ since H has about 10 nonzeros per row).

The theory of Markov chains justify why the Power method applied to G requires only about 50 iterations to converge. Indeed, as will be explained in a while, 50 iterations already provides the necessary accuracy.

The ratio of the two eigenvalues largest in magnitude (λ_1 and λ_2) for a given matrix determines how quickly the Power method converges [50]. So, the *asymptotic convergence rate* of the Power method depends on $\left|\frac{\lambda_2}{\lambda_1}\right|^k \to 0$.

For stochastic matrices such as G, $|\lambda_2|$ governs the convergence because $\lambda_1 = 1$. G is primitive, so $|\lambda_2| < 1$.

Next is presented a theorem and proof for the second eigenvalue of the Google matrix.

The theorem proves the relationship between the spectrum of S and the spectrum of G and, to maintain generality, it uses a generic personalization

vector v^T ($v^T > 0$ is a probability vector) rather than the uniform teleportation vector $\frac{1}{n}e^T$ [84, 88].

Theorem 2.2.2. *If the spectrum of the stochastic matrix S is,*

$$\sigma(S) = \{1, \mu_2, \ldots, \mu_n\},$$

then the spectrum of the Google matrix $G = \alpha S + (1-\alpha)E$ with $E = ev^T$ is

$$\sigma(G) = \{1, \lambda_2, \ldots, \lambda_n\},$$

where

$$\lambda_k = \alpha\mu_k \quad \textit{for} \quad k = 2, 3, \ldots, n$$

and v^T is a probability vector.

Proof. To clarify the proof vectors and matrices are written in bold.

$\mathbf{S}$ is a stochastic matrix, so $(1, \mathbf{e})$ is an eigenpair of $\mathbf{S}$.

Let $\mathbf{Q} = (\mathbf{e} \quad \mathbf{X})$ be a nonsingular matrix in which the first column is the eigenvector $\mathbf{e}$.

Consider the inverse matrix

$$\mathbf{Q}^{-1} = \begin{pmatrix} \mathbf{y}^T \\ \mathbf{Y}^T \end{pmatrix}.$$

Then,

$$\mathbf{Q}^{-1}\mathbf{Q} = \begin{pmatrix} \mathbf{y}^T\mathbf{e} & \mathbf{y}^T\mathbf{X} \\ \mathbf{Y}^T\mathbf{e} & \mathbf{Y}^T\mathbf{X} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

which gives two useful identities, $\mathbf{y}^T\mathbf{e} = 1$ and $\mathbf{Y}^T\mathbf{e} = \mathbf{0}$. As a result, the similarity transformation

$$\mathbf{Q}^{-1}\mathbf{S}\mathbf{Q} = \begin{pmatrix} \mathbf{y}^T\mathbf{e} & \mathbf{y}^T\mathbf{S}\mathbf{X} \\ \mathbf{Y}^T\mathbf{e} & \mathbf{Y}^T\mathbf{S}\mathbf{X} \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{y}^T\mathbf{S}\mathbf{X} \\ 0 & \mathbf{Y}^T\mathbf{S}\mathbf{X} \end{pmatrix},$$

reveals that $\mathbf{Y}^T\mathbf{S}\mathbf{X}$ contains the remaining eigenvalues of $\mathbf{S}$, $\mu_2, \ldots, \mu_n$. Applying the similarity transformation to $\mathbf{G} = \alpha\mathbf{S} + (1-\alpha)\mathbf{E}$ with $\mathbf{E} = \mathbf{e}\mathbf{v}^T$, $\mathbf{v}^T > \mathbf{0}$ a probability vector, gives

$$\begin{aligned}
\mathbf{Q}^{-1}\mathbf{G}\mathbf{Q} &= \mathbf{Q}^{-1}\left(\alpha\mathbf{S}+(1-\alpha)\mathbf{e}\mathbf{v}^T\right)\mathbf{Q} \\
&= \alpha\mathbf{Q}^{-1}\mathbf{S}\mathbf{Q}+(1-\alpha)\mathbf{Q}^{-1}\mathbf{e}\mathbf{v}^T\mathbf{Q} \\
&= \alpha\begin{pmatrix} 1 & \mathbf{y}^T\mathbf{S}\mathbf{X} \\ 0 & \mathbf{Y}^T\mathbf{S}\mathbf{X} \end{pmatrix}+(1-\alpha)\begin{pmatrix} \mathbf{y}^T \\ \mathbf{Y}^T \end{pmatrix}\mathbf{e}\mathbf{v}^T\begin{pmatrix}\mathbf{e} & \mathbf{X}\end{pmatrix} \\
&= \begin{pmatrix} \alpha & \alpha\mathbf{y}^T\mathbf{S}\mathbf{X} \\ 0 & \alpha\mathbf{Y}^T\mathbf{S}\mathbf{X} \end{pmatrix}+(1-\alpha)\begin{pmatrix} \mathbf{y}^T\mathbf{e} \\ \mathbf{Y}^T\mathbf{e} \end{pmatrix}\begin{pmatrix}\mathbf{v}^T\mathbf{e} & \mathbf{v}^T\mathbf{X}\end{pmatrix} \\
&= \begin{pmatrix} \alpha & \alpha\mathbf{y}^T\mathbf{S}\mathbf{X} \\ 0 & \alpha\mathbf{Y}^T\mathbf{S}\mathbf{X} \end{pmatrix}+(1-\alpha)\begin{pmatrix} \mathbf{y}^T\mathbf{e}\mathbf{v}^T\mathbf{e} & \mathbf{y}^T\mathbf{e}\mathbf{v}^T\mathbf{X} \\ \mathbf{Y}^T\mathbf{e}\mathbf{v}^T\mathbf{e} & \mathbf{Y}^T\mathbf{e}\mathbf{v}^T\mathbf{X} \end{pmatrix} \\
&= \begin{pmatrix} \alpha & \alpha\mathbf{y}^T\mathbf{S}\mathbf{X} \\ 0 & \alpha\mathbf{Y}^T\mathbf{S}\mathbf{X} \end{pmatrix}+\begin{pmatrix} (1-\alpha) & (1-\alpha)\mathbf{v}^T\mathbf{X} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \\
&= \begin{pmatrix} 1 & \alpha\mathbf{y}^T\mathbf{S}\mathbf{X}+(1-\alpha)\mathbf{v}^T\mathbf{X} \\ \mathbf{0} & \alpha\mathbf{Y}^T\mathbf{S}\mathbf{X} \end{pmatrix}.
\end{aligned}$$

Therefore, the eigenvalues of $\mathbf{G}=\alpha\mathbf{S}+(1-\alpha)\mathbf{e}\mathbf{v}^T$ are $\{1,\alpha\mu_2,\alpha\mu_3,\ldots,\alpha\mu_n\}$.

This theorem provides a more compact proof than that found in [63] by Haveliwala and Kamvar. □

Due to the link structure of the web, $|\mu_2|=1$ or $|\mu_2|\approx 1$. So, $|\lambda_2(G)|=\alpha$ or $|\lambda_2(G)|\approx\alpha$. So, the convex combination parameter α explains the convergence after just 50 iterations.

Proposition 2.2.3. *Considering $\alpha^{50}=0,85^{50}\approx 0,000296$ then, at the 50^{th} iteration, one can expect roughly 2-3 decimal places of accuracy in the approximate PageRank vector (which is sufficient for Google's ranking needs).*

For the Google matrix G, the modulus of the subdominant eigenvalue of the Google matrix is $|\lambda_2(G)|\leq\alpha$.

In the case of $|\lambda_2(S)|=1$ then $|\lambda_2(G)|=\alpha$. Therefore, the asymptotic rate of convergence of the PageRank Power method of equation (2.6) is the rate at which $\alpha^k\longrightarrow 0$.

Proposition 2.2.4. *To produce PageRank scores with approximately β digits of accuracy, about $-\frac{\beta}{log_{10}\alpha}$ iterations must be completed.*

Proof. See [88]. □

A damping factor far less than 1 allows for convergence (linear) of the Power method. Usually $\alpha=0.85$ is the reference value and a higher value closer to 1 makes the computation more difficult. For the former cases, more sophisticated iterative methods, such as Implicit Arnoldi or Krylov-Schur are the only answer to solve these (large) eigenvalue problems [49].

An excellent overview of Google's PageRank and other search engines can be found in [88].

2.2.2 Other methods

The computation of PageRank is a specific instance of the computation of a stationary distribution vector, with two distinct features. On one hand, the matrices involved are so large that only a small set of computational tools can be applied. That is, algorithms based on factorization techniques cannot be used. On the other hand, the PageRank incorporates a damping factor, α, that defines a convex combination of the original matrix (after modifying zero-sum rows) and a rank-1 matrix. This determines the magnitude of the second largest eigenvalue, and consequently, the level of difficulty of the computation of the PageRank vector.

The damping factor is a real positive number smaller than 1. It was reported that the damping factor was originally set to 0.85 [111].

The question of what value of α is the "correct" value to give meaningful rankings is subject to ongoing investigation [14]. In [84] it was argued that the PageRank vector derived from larger α, such as 0.99, may give a "truer" PageRank than $\alpha = 0.85$ does. Also, the computation of many PageRank vectors, with different values of α, seems promising for the design of anti-spam mechanisms [67, 138].

The smaller α is, the easier is to compute the PageRank vector by simple means like the Power method. The closer the damping factor is to 1 the closer the matrix is to the original web link graph.

If the largest eigenvalue ($\lambda_1 = 1$) is not well separated from other eigenvalues the power method does not work and it is necessary to employ more efficient and sophisticated techniques.

So, some researchers dropped the restriction to the Power method and used different methods to compute the PageRank vector. The Arnoldi method is one of those methods.

Arnoldi's method has been popular for computing a small number of selected eigenvalues and the associated eigenvectors for large unsymmetric matrices.

Imposing the Galerkin condition to $Au = \lambda u$ [1], A $n \times n$, a new problem is built $B_m y = \tilde{\lambda} y$, where $B_m = V^T A V$, B_m $m \times m$, V an orthogonal basis. Each eigenvalue $\tilde{\lambda}_i$ of B_m is a Ritz value and $V y_i$ a Ritz vector, where y_i is

[1] we are considering $A = G^T$

the eigenvector of B_m associated with $\tilde{\lambda}_i$.

However, there is no guarantee that the Ritz vectors obtained by the Arnoldi's method converge, even when Ritz values (approximate eigenvalues) do. In order to find a way around this obstacle, Jia, in [72, 73], computed refined approximate eigenvectors by small sized singular value decompositions (SVD).

This proposal key idea is that, after computing the Ritz values by Arnoldi's method, it is chosen a refined approximate eigenvector in the Krylov subspace involved which minimizes the norm of the residual formed with the Ritz value. So, rather than using the Ritz vectors Vy_i as approximate eigenvectors, for each $\tilde{\lambda}_i$, we now seek a unit norm vector v_i satisfying the condition

$$\left\| \left(A - \tilde{\lambda}_i I \right) v_i \right\| = \min_{v, \|v\|=1} \left\| \left(A - \tilde{\lambda}_i I \right) v \right\|$$

and use it to approximate u_i. The v_i are called refined approximate eigenvectors corresponding to λ_i.

This refined approximate eigenvectors converge to the eigenvectors if the Ritz values do. In fact, the resulting refined algorithms converge more rapidly. The numerical experiments show that Jia's algorithms are more efficient than their counterparts, the iterative Arnoldi and Arnoldi-Chebyshev algorithms [72].

Golub and Greif applied the Arnoldi method to the PageRank problem [48, 49].

In [49] is proposed a variant of the restarted refined Arnoldi method, based in [72], which does not involve Ritz value computations. That is, since the largest eigenvalue of the Google matrix is known, it is used as a shift. This strategy improves the algorithm performance and circumvents drawbacks of complex arithmetic.

The strength of the Arnoldi-type methods lies in obtaining orthogonal vectors. It allows for effective separation of the PageRank vector from other approximate eigenvectors.

Golub and Greif experimented with a few web matrices and proved that the Arnoldi-type algorithm used in [49] works well for values of the damping factor towards the right part of the spectrum (close to 1). For high values of α, the method yields substantial computational savings (at the price of higher memory requirements).

The authors also comment on the sensitivity of the PageRank problem to the choice of the damping parameter α. An analysis of the sensitivity of PageRank to the removal, addition, change of pages and α parameter, can be found in [68, 88, 110].

Lately, several advanced strategies to accelerate the Arnoldi-type methods emerged. Wu and Wei, in [139], presented a thick restarted Arnoldi method for computing PageRank. They concluded that the algorithm was not satisfactory due to its heavy cost per iteration. They dealt with this problem by periodically knitting the cheap Power method together with the thick restarted Arnoldi algorithm.

This idea was due to the fact that the Arnoldi method can be viewed as an accelerated Power method. The resulting new algorithm, called Power-Arnoldi algorithm, can be understood as a thick restarted Arnoldi algorithm preconditioned with power iteration, or a Power method accelerated with thick restarted Arnoldi.

The authors studied the sensitivity and stability of the PageRank vector and they compared the convergence speed of the Power-Arnoldi method with the convergence speed of the Power method, Quadratic-extrapolation method and Arnoldi method for Google matrices.

The experimental results showed that the Power-Arnoldi algorithm is often superior to its counterparts for computing the PageRank. However, the experiments have also illustrated that the Power method and the quadratic-extrapolation methods may be better than the Power-Arnoldi method in many cases, especially when the damping factor is not high.

Two advantages of the Power-Arnoldi algorithm are flexibility and easy parallelization. On the other hand, the authors noticed that in the Power-Arnoldi process, some parameters are difficult to handle.

More recently, the same investigators, Wu and Wei, used another strategy to accelerate the Arnoldi method. In [140], they presented an extrapolation procedure based on Ritz values and periodically knitted this extrapolation procedure together with the Arnoldi-type algorithm. The resulting algorithm is called the Arnoldi-Extrapolation algorithm.

The Arnoldi-type algorithm provides better and better approximate eigenvalues to the extrapolation procedure, and the extrapolation procedure provides better and better initial guess to the Arnoldi iteration.

The extrapolation procedure is based in the work developed in [16, 19, 18, 64, 78], but with a difference, the Arnoldi-Extrapolation algorithm exploits some approximate eigenvalues which are available from the Arnoldi-type algorithm.

In the Arnoldi-type algorithm the Arnoldi process is restarted every m steps. By altering m, one can change the number of iterations and also the execution time. Since the size of the Google matrix is very large, it is desirable to choose m as small as possible.

The idea is to construct an improved starting vector using the extrap-

olation procedure and compute a new Arnoldi factorization with the new starting vector. This leads to combine the Arnoldi-type algorithm with the extrapolation procedure periodically (the Arnoldi-Extrapolation algorithm).

In [140] the authors compared the performance of the Power method, the Quadratic extrapolation algorithm, the Power-Arnoldi algorithm and the Arnoldi-Extrapolation algorithm. Numerical experiments illustrate that the new Arnoldi-Extrapolation algorithm is often more powerful than the Arnoldi-type algorithm and the Power method.

One advantage of this new Arnoldi-Extrapolation algorithm is that the dimension of the Krylov subspace can be chosen very moderate, allowing the memory costs to be reasonable. Other advantages are the potential use on parallel architectures and its applicability for a wide range of the parameter α which can be set close to 1.

As mentioned early, the Google matrix can be reordered according to dangling and nondangling nodes and the PageRank problem can be applied to a much smaller matrix, which reduces the size of the problem [67, 77, 84, 86, 93]. So, Wu and Wei suggest that the two new algorithms they developed [139, 140] can also be used in combination with these reordering schemes.

2.2.3 Acceleration methods for the Power Method

The PageRank computation by the standard Power method takes days to converge since matrices involved are large.

Even though Power method is slow to converge, the size and sparsity of the web matrix made it the method of choice to compute the PageRank.

The Web is big. After looking at different sources it becomes clear that the number of web pages indexed by Google is not consensual, different web sites report different numbers. However, the most reliable source for this kind of information must be the Google company itself. The first Google index in 1998 already had 26 million pages, by year 2000, they already hit the one billion mark. In 2008, Google claimed that the Web had 1 trillion unique URLs (short scale - 1 trillion is 1 000 000 000 000) [2]. According to Statistic Brain Research Institute, Google's search engine found 30 trillion web pages in 2014 [15]. In April 2016, the estimated size of Google's index is 45 trillion of web pages [30].

Due to the fact that the number of web pages is increasing rapidly, it became necessary to find methods to speed up the computation. Some of the

most important contributions have come from researchers in Stanford University. They proposed several methods for accelerating the Power method. As the computation of just one iteration of the Power method on a web-sized matrix is so expensive, reducing the number of iterations by a handful can save hours of computation.

There are two ways to reduce the work involved in any iterative method: either reduce the work per iteration or reduce the total number of iterations. The problem is that reducing the number of iterations usually comes at the expense of a slight increase in the work per iteration, and vice-versa. As long as the overhead is minimal, the proposed acceleration is considered beneficial [88].

Some of the most successful approaches for reducing the work associated with the PageRank vector are: Adaptive Power method [76], Extrapolation techniques [16, 64, 78, 124], Aggregation [78] and Lumping methods [67, 93] that proceed with a matrix reordering according to dangling and nondangling nodes.

Adaptive Methods

As mentioned above, various methods to accelerate the simple Power method iterations have been developed. Stanford researchers S. Kamvar, T. Haveliwala and G. Golub noticed that most pages in the web converge to their true PageRank quickly, while relatively few pages take much longer to converge [76]. However, the standard Power method with its macroscopic view does not notice this, and blindly makes unnecessary calculations. The pages that converge slowly generally have high PageRank and the pages that converge quickly generally show low PageRank. The Power method is forced to drag on because a small proportion of pages (pages with high PageRank) take longer to converge to their final PageRank values. As elements of the PageRank vector converge, the idea is to "lock" them and do not recompute leading to a reduction of effort in subsequent iterations. Two algorithms were developed in [76], Adaptive PageRank and Modified Adaptive PageRank, which allowed a speed up in computation of 18% and 28% respectively.

Aggregation Methods

BlockRank In [77] Kamvar et al. produced another contribution to the acceleration of PageRank. This method, called BlockRank, works on both

acceleration goals simultaneously, it reduces both the number of iterations and the work per iteration.

This method is an aggregation method that lumps sections of the web by hosts. Hosts are high-level web pages under which lots of other pages sit.

The web link graph has a nested block structure, that is, most hyperlinks link (intralink) pages on a host to other pages on the same host. Only a few links (interlinks) are links between hosts. So, the webgraph (nodes are web pages) is compressed into a hostgraph (nodes are hosts).

In [77], a 3-stage algorithm, that explores this structure, is used to speed up the computation of the PageRank vector. First, the local PageRanks of pages for each host are computed independently using that host link structure; second, these local PageRanks are then weighted by the "importance" of the corresponding host; third, the standard PageRank algorithm is run using the weighted aggregate of the local PageRanks as its starting vector.

Intralinks are ignored in the global hostgraph. When the PageRank model is applied to a small hostgraph, the output is a HostRank vector. The relative importance of host i is given by the HostRank for host i. Although the HostRank problem is much smaller than the original PageRank problem, it gives the importance of individual hosts (instead of individual pages).

It is necessary to compute many local PageRank vectors (for the pages in each individual host) in order to obtain one global PageRank vector. So, the PageRank model is applied to each host with the interlinks being ignored and only the intralinks being used. This computation is inexpensive because hosts generally have less than a few thousand pages. Considering that $|H|$ is the number of hosts and $|H_i|$ the number of pages in host H_i, there is one global $1 \times |H|$ HostRank vector and $|H|$ local PageRank vectors each with size $1 \times |H_i|$.

The expansion step gives an approximation to the global PageRank vector, by multiplying the local PageRank vector for host H_i by the probability of being in that host, given by the i^{th} element of the HostRank vector.

In each step of the compression (aggregation step) some links are ignored causing the loss of some valuable information. So, only an approximation to the true PageRank vector (computed by the Power method) is obtained. However, this approximation can be improved if the collapsing/expanding process is repeated until convergence.

This method speeds up the computation of PageRank by a factor of 2 on some datasets used in [77].

A variant of this algorithm, that efficiently computes many different personalized PageRanks, was also developed.

Page and Brin suggested that by changing the random teleportation vector v to be nonuniform, the resultant PageRank vector can be biased to prefer certain kinds of pages. However, personalized PageRank generally requires computing a large number of PageRank vectors.

In [77] it was used the BlockRank algorithm and a simple restriction on the jump behavior of the random surfer to obtain a significantly reduction of the computation time for personalized PageRank. The restriction is that instead of being able to choose a distribution over pages to which he jumps when he is bored, the random surfer may choose hosts.

With this restriction, the local PageRank vectors are the same for different personalizations, only the BlockRank vector changes.

It was also presented another variant of the BlockRank algorithm that efficiently recomputes PageRank after node updates. It uses the strategy of reusing local PageRank vectors when the goal is to recompute PageRank after several pages have been added or removed from the web. It is only necessary to recompute the local PageRank of those hosts to which pages have been added or removed at each update.

IAD Method / SIAD Method The web is volatile; thousands of modifications (node modifications and link modifications) are made daily. A link modification occurs when a hyperlink is added or deleted, a node modification occurs when a web page is added to or deleted from the web.

These modifications on the structure of the web induce changes on the PageRank vector.

This is one of the major bottlenecks associated with web-based information retrieval systems, the need to update importance rankings of pages to account for the constant changes occurring in the web's structure.

As the classical Power method applied to the PageRank problem takes several days to converge, updating cannot be done constantly.

The PageRank vector is recomputed periodically (say, once a month) and it is very expensive due to the size of the Google matrix.

One way to reduce the computational work of each recomputation is to exploit the previous PageRank computation. The intuition here is that, during, say, one month (period between recomputations of the PageRank vector), several changes in the web may be made, but the vast majority of the web's structure remain unchanged.

Chien et al. in [24] developed an algorithm that exploits the unchanged part of the web to approximate a new PageRank vector without the computation of the full PageRank vector of the updated web.

The Iterative Aggregation/Disaggregation method (IAD method) was first proposed by Takahashi in 1975 [128] and has been widely used to accelerate convergence of iterative methods for solving linear systems and minimization problems.

In [82], Langville and Meyer improved Chien's work providing three new updating algorithms for the PageRank problem. The first two (an algorithm for exact updating of PageRanks using the group inverse and an algorithm based on stochastic complementation) are of theoretical value only because they are computationally expensive. The third, an iterative aggregation/disaggregation updating method (IAD PageRank updating method) proved to be very promising and computationally practical.

The IAD method significantly reduces the number of power iterations and only adds the minor additional expense of a stationary vector solve on a very small Markov chain [82]. Also, the IAD algorithm handles both link updates and nodes updates.

Extensive analysis of the properties of the algorithm in [82] has been done by Ipsen ans Kirkland in [65]. The authors show that the Power method applied to the Google matrix G always converges and that the convergence rate of the IAD method applied to G is at least as good as that of the Power method.

The IAD algorithms were further explored in [83, 85, 87] by the same authors as [82]. The IAD algorithm uses iterative aggregation techniques to focus on the slow-converging states of the Markov chain. This algorithm can be used jointly with other PageRank acceleration methods such as the ones showed in [76, 78].

In [66] Ipsen and Kirkland used the updating algorithm proposed by Langville and Meyer in [82, 83, 85, 87] and analyzed the convergence, in exact arithmetic, of the method. In this paper, Ipsen refers to the method developed by Langville as SIAD for Special Iterative Aggregation/Disaggregation method. The SIAD method is expressed as the Power method preconditioned by partial LU factorization.

Ipsen showed that the asymptotic convergence rate of the SIAD method is at least as good as that of the Power method. Furthermore, by exploiting the hyperlink structure of the web it was shown that the asymptotic convergence rate of the SIAD method applied to the Google matrix can be strictly better

than of the Power method and computations can be saved.

Also, in [149] Zhu proposed a distributed PageRank computation algorithm, DPC algorithm, based on Iterative Aggregation-disaggregation (IAD) method with Block Jacobi smoothing. The basic idea is divide-and-conquer. Each node in the distributed system computes a PageRank vector for local pages. Unlike parallel computations, distributed algorithms require simple mechanisms of interaction between nodes and low volume of communication traffic.

In this distributed approach the block structure of hyperlinks is explored. Local PageRank is computed by each node itself and then updated with a low communication cost with a coordinator.

Experiments on three real web graphs show that this method converges 5-7 times faster than the traditional Power method.

A survey of the aggregation methods above, including algorithms, can be seen in [7].

Extrapolation Methods

Aitken extrapolation A practical extrapolation method for accelerating PageRank computations was first presented by Kamvar et al. in [78]. The aim is to reduce the number of power iterations. This method is referred to as Aitken extrapolation because it is derived from classic Aitken Δ^2 method for accelerating linearly convergent sequences.

The size of the subdominant eigenvalue λ_2 governs the expected number of power iterations. The idea of Aitken extrapolation is "if the subdominant eigenvalue λ_2 causes the Power method to sputter, cut it out and throw it away".

Unfortunately, if λ_2 and λ_3 are complex conjugates, then $|\lambda_2| = |\lambda_3|$ and Aitken extrapolation performs poorly.

This method will be explored in Chapter 3.

Quadratic extrapolation Kamvar et al. developed an improved extrapolation method, called Quadratic extrapolation, based on the same idea as

Aitken extrapolation. That is "if λ_2 and λ_3 cause you problems, cut them both out and throw them away".

Quadratic extrapolation assumes that the iterate can be expressed as a linear combination of the first three eigenvectors in order to find an approximation to the principal eigenvector of the Google matrix.

On the datasets tested in [78], Quadratic extrapolation reduced PageRank computation time by 25-300% with minimal overhead .

However, Quadratic extrapolation is expensive and can only be done periodically.

Power extrapolation In Aitken extrapolation and Quadratic extrapolation it was assumed that none of the nonprincipal eigenvalues of the hyperlink matrix were known.

In [63] Haveliwala and Kamvar proved that the modulus of the second eigenvalue of G is given by the damping factor α, $|\lambda_2(G)| \approx \alpha$. Furthermore, the eigengap $1 - |\lambda_2|$ for the web Markov matrix G is given exactly by the teleport probability $1 - \alpha$. When the teleport probability is large (that is, α is small), the Power method works reasonably well.

Increasing α slows down convergence.

A high teleport probability means that a fixed bonus rank is given to every web page. However, when $1 - \alpha$ is large the effect of link spam is increased and web pages can achieve unfairly high rankings.

The web graph can have many eigenvalues with modulus α (i.e., one of α, $-\alpha$, αi and $-\alpha i$). So, when there are more than one (two) eigenvalue with modulus α the Aitken extrapolation (Quadratic extrapolation) performs poorly.

By exploiting known eigenvalues of the hyperlink matrix, Haveliwala and Kamvar derived a new extrapolation method called Power extrapolation [64]. They developed a series of algorithms, namely, Simple extrapolation, A^2 extrapolation and A^d extrapolation (Power extrapolation).

The Power extrapolation accelerates the convergence of the Power method by subtracting off the first few nonprincipal eigenvectors from the current iterate. It uses the fact that the first eigenvalue of G is 1 and the modulus of the second eigenvalues is α to compute estimates of the error along nonprincipal eigenvectors using two iterates of the Power method.

In the experiments, considering $\alpha = 0.85$, A^d extrapolation (Power ex-

trapolation) performed the best for $d = 6$.

The Quadratic extrapolation and the Power extrapolation have similar speedup in number of iterations. However, Power extrapolation has negligible overhead, so that its wallclock-speedup is higher. The Power extrapolation is simpler to implement and more efficient than Quadratic extrapolation (which is more widely applicable).

Also, Power extrapolation needs to be applied only once to achieve full benefit whereas Quadratic extrapolation must be applied several times to achieve maximum benefit.

Chebyshev and ϵ-extrapolation Other researchers, such as extrapolation expert Claude Brezinski, experimented with other classic extrapolation methods, such as Chebyshev and ϵ-extrapolation. In [16] Brezinski generalize the Quadratic extrapolation and interpret it on the basis of the method of Moments of Vorobyev and as a Krylov subspace method. He also gives a theoretical justification to several extrapolation methods and constructs Padé style rational approximations of the PageRank vector.

Minimal polynomial extrapolation and Reduced rank extrapolation In [124], Avram Sidi proposes two polynomial-type vector extrapolation methods that have proved to be very efficient convergence accelerators. The methods were used in the computation of the PageRank vector and are called Minimal polynomial extrapolation (MPE) and Reduced rank extrapolation (RRE). In this article, Sidi also generalizes the Quadratic extrapolation proposed by Kamvar in [78] and proves that the resulting generalization is very closely related to MPE.

Lumping Methods

Lumping is a special permutation of the hyperlink matrix that induces a special structure on the solution. With lumping computing the PageRank vector involves solving a core problem on a smaller matrix.

There are two advantages in reducing the matrix dimension: faster computation and smaller round-off error [137].

Lumping Google's PageRank via reordering the web matrix have been investigated by several authors in [4, 34, 37, 67, 77, 86, 90, 93].

In [4] Arasu proposed a strong component decomposition and in [34] Del Corso et al. used permutations to make iterative computations faster.

There are web pages without links to other pages (dangling nodes). The number of web pages that are dangling nodes may exceed the number of nondangling (pages with outlinks).

Lee at al. [90, 91] developed a fast two-stage algorithm for computing the PageRank vector that lumps the dangling nodes and aggregate the nondangling nodes.

Inspired by Lee's strategy Langville and Meyer [86] proposed a reordered algorithm that uses the lumping procedure recursively.

With this algorithm the 0^T rows of the top left submatrix are recursively placed in the bottom, until there is no 0^T rows in the new top left submatrix.

Applying this procedure, the top left submatrix gets smaller and smaller.

The stationery vector of this submatrix is less costly to compute because its dimension is much smaller compared to the origin matrix. After, forward substitutions are carried on to get the full PageRank vector.

However, this algorithm may suffer from the overhead of the recursively reordering procedure.

In [22] Bu and Huang tried to solve this problem and found a compromise between getting a reduced submatrix and saving the calculation spent on reordering. They developed an Adaptive reordered method that uses a stopping criterion that once the criterion is reached the reordering procedure stops.

This method gets rid of the overhead brought by the recursively reordering procedure and keeps its fast computational speed due to the reduction of computation.

Ipsen and Selee [67] expressed lumping as a similarity transformation of the Google matrix and proved that the dangling nodes can be lumped into a single node (resulting in a reduced matrix) and the PageRank of the nondangling nodes can be computed separately from that of the dangling nodes. The stochastic reduced matrix has the same nonzero eigenvalues as the full Google matrix.

In [93] Lin, Shi and Wei extended the results of [67] and showed that the reduced matrix can be further reduced by lumping a class of nondangling

nodes, called *weakly nondangling nodes*, to another single node. The further reduced matrix is also stochastic with the same nonzero eigenvalues as the Google matrix.

In both articles [67, 93] the authors considered the PageRank problem from the eigenvalue point of view and applied the Power method to the smaller lumped matrices.

The efficiency of the algorithm increases as the number of dangling nodes increases.

These two methods will be explored in Chapter 3.

In [146] Yu et al. introduced five type nodes for lumping the web matrix. In this article the authors explored the effect of the *unreferenced pages*, i.e., pages without inlinks.

Compared with the lumping strategies proposed in [67, 93], the Yu's new lumping strategy can reduce the original PageRank matrix to a much smaller one, while the overhead of the three reordering schemes is comparable.

In [146], instead of the eigenvector approach used in [67] and [93], the authors used a linear system approach. They applied an mathematically equivalent algorithm, the Jacobi method, to the smaller matrix to compute the PageRank vector.

Also, this new strategy is much cheaper than the recursively reordering strategy proposed by Langville and Meyer [86].

Regularization

The idea of iterative regularization begins with paper [9], where it was proposed for solving variational inequalities, with applications to ill-posed linear equations and optimization problems.

The traditional approach to the PageRank problem goes back to the pioneering paper [20] of Brin and Page. The Power method applied to the original matrix can diverge or it can converge slowly. To overcome this difficulty the matrix was modified. The matrix of the original problem was replaced with a modified matrix which allowed that the task of finding the principal eigenvector could be effectively solved using the Power method. For the modified matrix the Power method always converges linearly with ratio α to the dominant eigenvector π of the modified matrix. This convergence is fast enough but π can differ strongly from the eigenvector of the original matrix.

In [114], Polyak and Timonina demonstrate that the solution of the modified problem can be far enough from the solution of the original problem

and they propose an Iterative Regularization method which allows to find the desired solution. It has the form

$$\pi^{(k+1)} = A_k \pi^{(k)}, \quad A_k = \alpha_k S + (1 - \alpha_k)\, E, \tag{2.7}$$

$$\alpha_k \to 1, \qquad \sum_{k=0}^{\infty} (1 - \alpha_k) = \infty$$

where α_k varies at each iteration.

The iterative regularization process (2.7) converges to the principal eigenvector of S independently of $\pi^{(0)}$.

In [114] two simple models are proposed which are convenient for experiments with PageRank problem. For them, two different regularizations are tested. The first regularization (2.7) is able to find the principal eigenvector in situations when the power method does not converge or converge slowly. The second method, l^1-regularization, allows to get rid of low-ranking nodes which are considered of no interest for search engines.

In a posterior paper [115], Polyak discusses the Power method applied in the solution of the PageRank problem for large matrices and its modifications based on regularization. These new methods are based on the ones presented in [114].

Independently of the initial approximation, for $\alpha = 0.85$, the convergence of the Power method is relatively fast. For α close to one the rate of convergence drops dramatically. Moreover, for $\alpha \to 1$, the solution of the PageRank problem becomes numerically unstable, and a singular behavior is observed in $\alpha = 1$.

So, in [115], the authors consider an iterative process where the regularization parameter α (damping factor) varies during the power iterations

$$\hat{\pi}^{(k+1)} = A_{\alpha_k} \hat{\pi}^{(k)} \qquad k = 1, 2, \ldots$$

Two methods of varying α_k, Iterative regularization and Residual regularization, both relying on successive variation of the regularization parameter, were first proposed in [114].

The first method, Iterative regularization, is based on the iterative variation of the parameter according to a given law, such as

$$\alpha_k = 1 - \frac{1}{(k+2)^p}, \qquad 0 < p < 1.$$

As the number of iterations, k, grows the *damping factor*, α_k, also grows.

In this case, for the block irreducible original matrix, it was proved that the vectors $\pi^{(k)}$ converge in norm to the so-called dominant eigenvector of the original matrix.

The second method of regularization is represented by an Adaptive variation of the regularization parameter. In this case, α_k approaches 1 discontinuously as a certain condition for the small residual of the current matrix is satisfied. That is, in the second method the parameter varies dynamically depending on the current residual.

In [115] is presented another method of solving the PageRank problem. This method is represented by Averaging the power iterations and considers a vector constructed from the iterations of the Power method:

$$\bar{\pi}_k = \frac{1}{k+1} \sum_{i=0}^{k} \pi_i.$$

This vector always converges to a real nonnegative eigenvector for an arbitrary matrix S, and at that there always exists an estimate of the residual $\|S\bar{\pi}_k - \bar{\pi}_k\| \leq \frac{2}{k+1}$.

The authors presented numerical experiments with three examples of large sparse matrices of which two matrices were constructed from the model graphs presented in [114] and the third one was obtained from the physical graph of links between sites.

The numerical examples demonstrated good performance of the regularization method with variation of the parameter in residual. The authors conclude that, although there exists no theoretical substantiation of the rate of convergence of the regularization method with variation of the parameter in residual, this method seems attractive, especially if it is not known in advance whether a matrix is regular or not.

2.3 PageRank as a linear homogeneous system

As mentioned before, there are two ways of solving the PageRank problem: as an *eigenvector problem* ($\pi^T = \pi^T G$, $\pi^T e = 1$) or as *linear homogeneous system* ($\pi^T (I - G) = 0^T$, $\pi^T e = 1$).

The goal of the linear homogeneous system approach is to find the normalized left-hand null vector of $I - G$.

In their original paper [20] Page and Brin conceived of the PageRank problem as an eigenvector problem. So, the first approach became more popular and most investigation was done using that approach.

The normalized eigenvector problem

$$\pi^T G = \pi^T \quad \Leftrightarrow \quad \pi^T \left(\alpha S + (1 - \alpha) e v^T\right) = \pi^T$$

can be rewritten, with some algebra, as,

$$\pi^T (I - G) = 0^T \quad \Leftrightarrow \quad \pi^T (I - \alpha S) = (1 - \alpha) v^T.$$

This linear system is always accompanied by the normalization equation $\pi^T e = 1$.

Bianchini et al [13] and Cleve Moler [26] appear to have been the first to suggest the linear system in the form

$$\pi^T (I - \alpha S) = (1 - \alpha) v^T. \tag{2.8}$$

In this section we will discuss the linear system formulation of the PageRank problem.

There are good reasons to do that. Unlike what happens with the Power method, the value of the parameter α does not affect the computation time of the direct method. When the Power method is used, if $\alpha \to 1$, the amount of time that the method needs to converge increases. So, increasing the value of α ($\alpha \approx 1$), the weight given to the artificial teleportation matrix is smaller, and the problem approaches the true essence of the Web.

However, there are some sensitive issues to take into account. The PageRank vector is sensitive as $\alpha \to 1$ regardless of the problem formulation [75].

Some research has been done using the linear system formulation: particularly related with stationary iterative methods, Jacobi and Gauss-Seidel,

nonstationary iterative methods, such as BiCGSTAB and GMRES, along with the use of preconditioners and reorderings [33, 43, 86].

To study the linear system formulation of the PageRank problem it is necessary to refer some properties of the coefficient matrix $(I - \alpha S)$ of equation (2.8) and some definitions [88].

The proofs of the following properties can be consulted in Golub and Van Loan [50] and Meyer [106].

Definition 2.3.1. *If $A \in \mathbb{R}^{n\times n}$ satisfies $A \geq 0$ (> 0), i.e. all a_{ij} are nonnegative (positive), then A is said to be nonnegative (positive).*

The hyperlink matrix H and the stochastic matrix S are nonnegative matrices and the Google matrix G is a positive matrix.

Definition 2.3.2. *A real matrix $A \in \mathbb{R}^{n\times n}$ is called an M-matrix whenever there exists a matrix $B \geq 0$ and a real number $s \geq \rho(B)$, the spectral radius of B, such that $A = sI - B$.*

Definition 2.3.3. *If $s > \rho(B)$ in the above definition then A is a nonsingular M-matrix.*

M-matrices can be either nonsingular or singular.

Definition 2.3.4. *A matrix $A \in \mathbb{R}^{n\times n}$ is called a H-matrix if its comparison matrix $H = H(A)$ defined by*

$$h_{ij} = \begin{cases} |a_{ij}|, & i = j \\ -|a_{ij}|, & i \neq j \end{cases}$$

is an M-matrix (i.e. $H^{-1} > 0$).

Proposition 2.3.5. *Some properties of M-matrices:*

- *A is a nonsingular M-matrix if and only if $a_{ij} \leq 0$, for all $i \neq j$ and $A^{-1} \geq 0$.*
- *An M-matrix A has $a_{ij} \leq 0$, for all $i \neq j$ and $a_{ii} \geq 0$ for all i.*
- *If A is a nonsingular M-matrix, then all eigenvalues have positive real parts.*
- *Principal submatrices of nonsingular M-matrices are also nonsingular M-matrices.*

- *If A is an M-matrix, then all of its principal minors are nonnegative.If A is a nonsingular M-matrix , then all principal minors are positive.*

- *All matrices with nonpositive off-diagonal entries whose principal minors are nonnegative are M-matrices. All matrices with nonpositive off-diagonal entries whose principal minors are positive are nonsingular M-matrices.*

A nonsingular matrix A is ill-conditioned if a small relative change in A can produce a large relative change in A^{-1}. The condition number of A, given by $k = \|A\| \|A^{-1}\|$, measures the degree of ill-conditioning.

Condition numbers can be defined for each matrix norm.

The ∞-matrix norm ($\|A\|_\infty$) is the maximum absolute row sum.

Proposition 2.3.6. *Matrix $I - \alpha S$ has the following properties:*

- *$I - \alpha S$ is nonsingular*

- *$I - \alpha S$ is an M-matrix*

- *$\|I - \alpha S\|_\infty = 1 + \alpha$, provided at least one nondangling node exists*

- *The row sums of $(I - \alpha S)$ are $1 - \alpha$*

- *$I - \alpha S$ is an M-matrix so $(I - \alpha S)^{-1} \geq 0$*

- *The row sums of $(I - \alpha S)^{-1}$ are $(1 - \alpha)^{-1}$, so, $\left\|(I - \alpha S)^{-1}\right\|_\infty = (1 - \alpha)^{-1}$*

- *The condition number $k_\infty (I - \alpha S) = \frac{1+\alpha}{1-\alpha}$.*

Proof. See [50, 106].

□

As mentioned before, $I - \alpha S$ is dense. It is possible, and much better, to operate with the very sparse H matrix. Due to this, we must analyze if similar properties of $I - \alpha S$ hold for $I - \alpha H$.

As $S = H + dw^T$ and taking $w^T = v^T$ we have $S = H + dv^T$.

The linear system approach of the PageRank problem can be written in terms of the sparse matrix H as

$$\begin{aligned} & \pi^T (I - \alpha S) = (1 - \alpha) v^T \\ \Leftrightarrow \quad & \pi^T \left(I - \alpha H - \alpha d v^T\right) = (1 - \alpha) v^T \\ \Leftrightarrow \quad & \pi^T (I - \alpha H) - \alpha \pi^T d v^T = (1 - \alpha) v^T \\ \Leftrightarrow \quad & \pi^T (I - \alpha H) = \left(1 - \alpha + \alpha \pi^T d\right) v^T \end{aligned}$$

Let $\pi^T d = \gamma$, then the linear system becomes

$$\pi^T (I - \alpha H) = (1 - \alpha + \alpha\gamma) v^T$$

The scalar γ holds the aggregate PageRank for all the dangling nodes. Since the normalization equation $\pi^T e = 1$ will be applied at the end, we can choose a convenient value for γ, like $\gamma = 1$ [33, 43, 86].

So, we can obtain the linear system $\pi^T (I - \alpha H) = v^T$.

The matrix $I - \alpha H$ has many of the same properties as $I - \alpha S$.

Proposition 2.3.7. *Matrix $I - \alpha H$ has the following properties:*

- *$I - \alpha H$ is nonsingular*
- *$I - \alpha H$ is an M-matrix*
- *$\|I - \alpha H\|_\infty = 1 + \alpha$, provided H is nonzero*
- *The row sums of $I - \alpha H$ are either $1 - \alpha$ for nondangling or 1 for dangling nodes*
- *$I - \alpha H$ is an M-matrix so $(I - \alpha H)^{-1} \geq 0$*
- *The row sums of $(I - \alpha H)^{-1}$ are equal to 1 for dangling nodes and less than or equal to $\frac{1}{(1-\alpha)}$ for nondangling nodes*
- *The condition number $k_\infty (I - \alpha H) \leq \frac{1+\alpha}{1-\alpha}$*
- *The row of $(I - \alpha H)^{-1}$ corresponding to dangling node i is e_i^T where e_i is the i^{th} column of the identity matrix.*

Proof. See [50, 106].

□

The last property of $(I - \alpha H)^{-1}$ does not apply to $(I - \alpha S)^{-1}$.

The next theorem shows that a linear system formulation, related to the eigenvector formulation, exists.

Theorem 2.3.8 (Sparse linear system for the PageRank problem)**.** *Solving the sparse linear system*

$$x^T (I - \alpha H) = v^T$$

and letting $\pi^T = \frac{x^T}{x^T e}$ *produces the PageRank vector.*

Proof. [88] If $\pi^T G = \pi^T$ and $\pi^T e = 1$ then π^T is the PageRank vector.

Knowing that $\pi^T G = \pi^T \Leftrightarrow \pi^T (I - G) = 0^T \Leftrightarrow x^T (I - G) = 0^T$, we will prove that $x^T (I - G) = 0^T$:

$$\begin{aligned} x^T (I - G) &= x^T \left[I - \alpha H - \alpha d v^T - (1 - \alpha) e v^T \right] \\ &= x^T (I - \alpha H) - x^T \left[\alpha d + (1 - \alpha) e \right] v^T \end{aligned}$$

We have $x^T (\alpha d - (1 - \alpha) e) = 1$ due to the fact that v^T is a probability vector and

$$\begin{aligned} 1 &= v^T e \\ &= x^T (I - \alpha H) e \\ &= x^T e - \alpha x^T H e \quad \text{and} \quad He = e - d \\ &= x^T e - \alpha x^T (e - d) \\ &= x^T e - \alpha x^T e + \alpha x^T d \\ &= (1 - \alpha) x^T e + \alpha x^T d \\ &= x^T \left[(1 - \alpha) e + \alpha d \right] \end{aligned}$$

So, it results that

$$\begin{aligned} x^T (I - G) &= x^T (I - \alpha H) - x^T \left[\alpha d + (1 - \alpha) e \right] v^T \\ &= v^T - v^T \\ &= 0^T. \end{aligned}$$

□

Therefore, PageRank is both the stationary distribution of a Markov chain and the solution of a linear system. Although these two approaches are inherently linked, they require different numerical methods for finding the PageRank vector [86], which should be exploited.

Let us review the linear systems and the numerical methods for solving them.

Given an $n \times n$ matrix A ($A \in \mathbb{C}^{n\times n}$) and a n-vector b ($b \in \mathbb{C}^n$), the problem considered is: find $x \in \mathbb{C}^n$ such that

$$Ax = b \tag{2.9}$$

Equation (2.9) is a *linear system*, A is the *coefficient matrix* (nonsingular matrix with nonvanishing diagonal elements, $det(A) \neq 0$), b is the *right-hand side vector*, and x is the vector of *unknowns*.

The system is called *homogeneous* if all elements of b are zero; otherwise is called *nonhomogeneous*.

The numerical methods used to solve a linear system of equations can be divided into two categories: *direct methods* and *iterative methods*.

A *direct method* is one that produces the solution in a prescribed, finite number of steps. All versions of Gaussian elimination (Crout's, Doolittle's, Cholesky's method) are direct methods.

In contrast, an *iterative method* is based on recurrence relations starting from an approximation to the true solution (initial guess) and, if successful, obtain better and better approximations for a required accuracy. As examples of iterative methods we have: Jacobi method, Gauss-Seidel method, SOR method, GMRES method, BiCG method, and many others.

Iterative methods are ideally suited for problems involving large sparse matrices where the use of direct methods would be very expensive.

The order n of the Google matrix is so large that we cannot afford to spend about n^3 operations to solve the system by Gaussian elimination.

The use of iterative methods has the advantage that the matrix A is not altered during the computations. Hence, though the computation may be long, the problem of accumulation of rounding errors is less serious than for those methods, such as most direct methods, where the matrix changes during the computation process.

Besides, sometimes the iterative methods produce good approximations with relatively few iterations.

An *iterative method* for solving (2.9) has the form

$$x^{(k+1)} = H_k x^{(k)} + d_k \qquad k = 0, 1, \ldots \tag{2.10}$$

where H_k is the *iteration matrix* and d_k is a vector, and both can depend on iteration count k.

There are two types of iterative methods: *stationary methods* and *non-stationary methods*.

If neither H_k nor d_k depends upon the iteration count k then the iterative method is said to be *stationary*. Otherwise it is *nonstationary*.

Stationary methods are older, simpler to understand and implement, but usually not as effective. On the other hand, nonstationary methods are more recent, are harder to understand and implement, but they can be highly effective.

The rate at which an iterative method converges depends greatly on the spectrum of the coefficient matrix. Hence, iterative methods usually involve a second matrix that transforms the coefficient matrix into one with a more favorable spectrum. The transformation matrix is called a *preconditioner*. A good preconditioner can improve the convergence of the iterative method sufficiently to overcome the extra cost of constructing and applying the preconditioner [10, 11].

For a general review of iterative methods for linear systems see [8, 10, 50].

2.3.1 Stationary methods

Web search applications require solving systems of linear equations, but the magnitude of n is too large for direct solution methods based on Gaussian elimination to be effective. So, iterative methods are often the choice, and, due to the size, sparsity, and memory considerations, the preferred algorithms are the methods based on matrix-vector products that are simpler and that require no additional storage beyond that of the original data. The most used and explored are the *linear stationary iterative methods*.

Consider, again, the linear system of algebraic equations of (2.9), $Ax = b$, where A is a $n \times n$ matrix, $det(A) \neq 0$, b and x are $n \times 1$.

The first step for the construction of an iterative method usually begins with the splitting of A.

Writing A as

$$A = M - N \tag{2.11}$$

with M nonsingular is called *splitting* of A. M is taken to be invertible and cheap to invert, meaning that a linear system with matrix coefficient M is much more economical to solve than (2.9).

In this conditions the system (2.9) can be written as

$$Mx = Nx + b \quad \Leftrightarrow \quad x = M^{-1}Nx + M^{-1}b$$

that suggests the iterative scheme

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b.$$

Considering the product

$$H = M^{-1}N = I - M^{-1}A \tag{2.12}$$

with H called the associated *iteration matrix* and the vector,

$$d = M^{-1}b. \tag{2.13}$$

For an arbitrary initial vector $x^{(0)}$, ($x^{(0)} \in \mathbb{C}^n$), the sequence defined by

$$x^{(k+1)} = Hx^{(k)} + d \quad k = 0, 1, 2, \ldots \tag{2.14}$$

is called a *linear stationary iteration*. The general expression for all the *linear stationary iterative methods* is (2.14).

A sufficient and necessary condition for (2.14) to convergence, to the solution of (2.9), is $\rho(H) < 1$, where $\rho(H)$ denotes the *spectral radius* of H (i.e., the largest of the moduli of its eigenvalues). So, if $\rho(H) < 1$, then A is nonsingular, and $\lim_{k\to\infty} x^{(k)} = x = A^{-1}b$ (the solution to 2.9) for every $x^{(0)}$.

A sufficient condition for convergence is $\|H\| < 1$, where $\|\cdot\|$ denotes matrix norm induced by a vector norm [57, 88].

The *asymptotic convergence rate* for (2.14) is the number $R = -log_{10}\rho\,(H)$ and is used to compare different stationary iterative algorithms because it is an indication of the number of digits of accuracy that can be expected to be eventually gained on each iteration of $x^{(k+1)} = Hx^{(k)} + d$.

In the next methods we will consider the matrix A divided in the sum of three matrices

$$A = D - L - U, \tag{2.15}$$

where $D = diag(A)$ is the diagonal part of A (assuming each $a_{ii} \neq 0$, $det(D) \neq 0$), $-L$ is the strictly lower triangular part of A and $-U$ is the strictly upper triangular part of A.

Each different splitting (2.11), produces a different iterative algorithm.

There are three particular splittings that are widely used and produce three well-known stationary iterative methods: the Jacobi method, the Gauss-Seidel method and the SOR method.

Casting PageRank as a linear system was suggested by Arasu et al [4] where Jacobi, Gauss-Seidel and the SOR iterative methods were considered.

In [34] Del Corso et al proved that the PageRank vector can be computed as the solution of an sparse linear system by exploiting the effectiveness of stationary methods such as Jacobi, Gauss-Seidel and Reverse Gauss-Seidel methods for the solution of the system.

Many different permutation schemes were applied to the web matrix in order to increase data-locality and reduce the time necessary for computing the PageRank vector. The experiments performed in [34] showed that these strategies allowed a gain of up to 90% of the computational time needed to compute PageRank compared to the Power method.

The three different splittings mentioned are presented in Table 2.1 and Table 2.2.

$H = M^{-1}N$		Method
$M = D$	$N = L + U$	Jacobi
$M = D - L$	$N = U$	Gauss-Seidel
$M = \omega^{-1}(D - \omega L)$	$N = (\omega^{-1} - 1)D + U$	SOR

Table 2.1: Iterative methods for specific values of M and N.

Iteration scheme / in terms of components	Method
$x^{(k+1)} = D^{-1}Lx^{(k)} + D^{-1}Ux^{(k)} + D^{-1}b$ $x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right)$	Jacobi
$x^{(k+1)} = D^{-1}Lx^{(k+1)} + D^{-1}Ux^{(k)} + D^{-1}b$ $x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)$	Gauss-Seidel
$x^{(k+1)} = (1-\omega)\,x^{(k)} + \omega D^{-1}\left(b - Lx^{(k+1)} - Ux^{(k)}\right)$ $x_i^{(k+1)} = (1-\omega)\,x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)$	SOR

Table 2.2: Iteration schemes and iterative methods.

The Jacobi method The Jacobi method is based on solving for every variable locally with respect to the other variables. One iteration corresponds to solving for every variable once. The resulting method is simple to understand and implement, but has convergence [10].

The order in which the equations are examined is irrelevant, since the Jacobi method treats them independently. For this reason, the Jacobi method is also known as the method of Simultaneous Displacements, because the updates could in principle be done simultaneously [10].

The pseudocode for the Jacobi method is given in Algorithm 1.

Algorithm 1 Jacobi Method

Given matrix A and vector b;
Choose an initial guess $x^{(0)}$;
Compute $M := D$, where D is the diagonal of A;
Compute $N := M - A$;
Let $k := 0$;
while no convergence **do**
    Solve $Mx^{(k+1)} := Nx^{(k)} + b$;
    Update $k := k + 1$;
end while
Normalize $x^{(k)}$;

The Gauss-Seidel method If we proceed as with the Jacobi method, but now assume that the equations are examined one at a time, and the previously computed results are used as soon as they are available, we obtain the Gauss-Seidel method.

In the Gauss-Seidel method each component of the new iterate depends upon all previously computed components, therefore, the updates cannot be done simultaneously as in the Jacobi method. Also, the new iterate depends upon the order in which the equations are examined.

So, the Gauss-Seidel method is similar to the Jacobi method, except that it uses updated values as soon as they are available.

Due to the dependence of the iterates on the ordering, the Gauss-Seidel method is also called method of Successive Displacements. If this ordering is changed, the components of the new iterate will also change (and not just their order).

Generally, if the Jacobi method converges, the Gauss-Seidel method will converge faster than the Jacobi method, although still relatively slowly.

In the case of A being a sparse matrix we can observe that the presence of zeros may remove the influence of some of the previous components, that is, it is not absolute the dependency of each component of the new iterate on the previous components. A smart ordering of the equations may allow a reduction of such dependence.

The reordering of the equations can alter the rate at which the Gauss-Seidel method converges. A good choice of ordering can enhance the rate of convergence, a bad choice can degrade the rate of convergence [10].

The pseudocode for the Gauss-Seidel method is given in Algorithm 2.

Algorithm 2 Gauss-Seidel Method

Given matrix A and vector b;
Choose an initial guess $x^{(0)}$;
Compute D, the diagonal of A;
Compute L, the strict lower triangular part of A;
Compute U, the strict upper triangular part of A;
Compute $M := D + L$;
Compute $N := -U$;
Let $k := 0$;
while no convergence **do**
    Solve $Mx^{(k+1)} := Nx^{(k)} + b$;
    Update $k := k + 1$;
end while
Normalize $x^{(k)}$;

The Successive Overrelaxation (SOR) method The Successive Overrelaxation method (SOR method) seems to have appeared in the 1930s. However, formally its theory was established almost simultaneously by Frankel [40] and Young [144].

Thus, most of the early results of the SOR method can be found in the books by Varga [130] and Young [145]. To study the SOR method and related methods see [57].

The SOR method can be derived from the Gauss-Seidel method by introducing an *relaxation* (or *overrelaxation*) *parameter* $\omega \neq 0$ ($\omega \in \mathbb{C}\setminus\{0\}$).

This extrapolation takes the form of a weighted average between the previous iterate and the computed Gauss-Seidel iterate successively for each component:

$$x_i^{(k+1)} = \omega \bar{x}_i^{(k)} + (1-\omega)\, x_i^{(k)},$$

where $\bar{x}$ denotes de Gauss-Seidel iterate. The goal is to choose a value for the extrapolation factor ω that will accelerate the rate of convergence of the iterates to the solution.

For the optimal choice of ω, SOR may converge faster than Gauss-Seidel by an order of magnitude.

In the development of the SOR theory one seeks values of $\omega \in \mathbb{C}\setminus\{0\}$ for which the SOR method converges, the set of which defines the region of convergence, and, if possible, the best of ω ($\hat{\omega}$), for which the convergence is asymptotically optimal, that is $\rho(H_{\hat{\omega}}) = \min_{\omega\in\mathbb{C}\setminus\{0\}} \rho(H_\omega)$.

In general, it is not possible to compute in advance the value of ω that is optimal with respect to the rate of convergence of SOR. Even when the computation is possible the expense of such computation is usually prohibitive [10].

To find regions of convergence is a problem generally much easier than to define $\hat{\omega}$ [57].

Theorem 2.3.9 (Kahan)**.** *A necessary condition for the SOR method to converge is* $|\omega - 1| < 1$. *For* $\omega \in \mathbb{R}$ *this condition becomes* $\omega \in (0, 2)$.

Note that if $0 < \omega < 1$, the iterative method is known as a Successive Underrelaxation and can be used to obtain convergence when the Gauss-Seidel scheme is not convergent. For choices of $\omega > 1$ the scheme is a Successive Overrelaxation and is used to accelerate convergent Gauss-Seidel iterations. If $\omega = 1$ the SOR method is simply the Gauss-Seidel method.

However, for convenience, the term overrelaxation is used for any value of $\omega \in (0, 2)$. [57, 145]

It can be shown that Jacobi's method as well as the Gauss-Seidel method converges when A is *diagonally dominant* (i.e., when $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for each $i = 1, 2, \ldots, n$).

The pseudocode for the SOR method is given in Algorithm 3. The pseudocodes were obtained from [10].

Algorithm 3 The SOR Method

Given matrix A, vector b and parameter ω;
Choose an initial guess $x^{(0)}$;
Compute D, the diagonal of A;
Compute L, the strict lower triangular part of A;
Compute U, the strict upper triangular part of A;
Compute $M := \omega^{-1}(D + \omega L)$;
Compute $N := (\omega^{-1} - 1) D - U$;
Let $k := 0$;
while no convergence **do**
    Solve $Mx^{(k+1)} := Nx^{(k)} + b$;
    Update $k := k + 1$;
end while
Normalize $x^{(k)}$;

2.3.2 Extrapolation principle and AOR method

The Accelerated Overrelaxation (AOR) method is another one of the linear stationary iterative methods and was first introduced by Hadjidimos in [54].

Since the introduction of the AOR method, many properties as well as numerical results have been given [5, 6, 56, 62].

Several results about the AOR convergence theory in the $(r, \omega)-$plane can be consulted in four papers of Martins [98, 99, 100, 101].

Other results about convergence of the AOR method and its variants are in [27, 62, 112, 125, 126].

The idea of extrapolating an iterative scheme to accelerate its convergence rates goes back to Richardson [117] and has been exploited by several researchers [1].

The AOR method is related with the extrapolation principle, since, as proved in [54], it is an Extrapolation of either the Jacobi method or the SOR method, when the two parameters take on special values [61].

The Principle of Extrapolation and its relation with the AOR method are described next.

The Principle of Extrapolation

Let $\omega \in \mathbb{C}\setminus\{0\}$ be the *extrapolation parameter*, and based on $A = M - N$ (2.11), consider the splitting

$$A = M_\omega - N_\omega \tag{2.16}$$

with

$$M_\omega = \omega^{-1} M \quad \text{and} \quad N_\omega = \omega^{-1}\left[(1-\omega)\, M + \omega N\right] \tag{2.17}$$

M_ω is the new preconditioner matrix.

The next scheme is the extrapolated version of the original (2.14), $x^{(k+1)} = Hx^{(k)} + d$, which is the completely consistent with the system (2.9), $Ax = b$.

Considering that $H_\omega = M_\omega^{-1} N_\omega$ and $d_\omega = M_\omega^{-1} b$, the new Extrapolated method is given by

$$\begin{aligned} & x^{(k+1)} = H_\omega x^{(k)} + d_\omega \\ \Leftrightarrow \quad & x^{(k+1)} = \left[(1-\omega)\, I + \omega H\right] x^{(k)} + \omega d \quad k = 0, 1, 2, \ldots \end{aligned} \tag{2.18}$$

with iterative matrix

$$\Leftrightarrow \quad H_\omega = (1-\omega)\, I + \omega H \tag{2.19}$$

and vector

$$\Leftrightarrow \quad d_\omega = \omega d \tag{2.20}$$

which is also completely consistent with the original system (2.9).

Note that $\omega \neq 0$ (if $\omega = 0$ then $H_0 = I$ and the extrapolated method (2.18) has no meaning).

It is well-known that for nonsingular systems the iterative method (2.14) is convergent if and only if the spectral radius $\rho(H)$ is less than 1.

The smaller spectral radius is the faster the convergence is.

Therefore, to solve the system (2.18) it is necessary to find ω's for which $\rho\left(H_\omega\right) < 1$ and among them to choose the one $(\hat{\omega})$ which minimizes $\rho\left(H_\omega\right)$ ($\rho\left(H_\omega\right)$ is the spectral radius of H_ω). [56]

Theorem 2.3.10 (Extrapolation Theorem)**.** *If the scheme (2.14) converges ($\rho(H) < 1$) and $0 < \omega < \frac{2}{1+\rho(H)}$, then the extrapolated scheme (2.18) converges ($\rho\left((1-\omega)I + \omega H\right) < 1$).*

More specifically,

$$\rho\left((1-\omega)I + \omega H\right) \leq |1-\omega| + \omega\rho(H) < 1. \tag{2.21}$$

Proof. See [61].

□

The Accelerated Overrelaxation (AOR) Method

Next, two formulations of the AOR method are presented.

Consider the system $Ax = b$ (2.9), the splitting of matrix A in (2.15) $A = D - L - U$ and the assumptions made in section 2.3.1.

The AOR iterative scheme is given by

$$x^{(k+1)} = H_{r,\omega}x^{(k)} + d_{r,\omega} \qquad k = 0, 1, 2, \ldots \tag{2.22}$$

with the AOR iteration matrix

$$H_{r,\omega} = (D - rL)^{-1}\left[(1-\omega)D + (\omega - r)L + \omega U\right] \tag{2.23}$$

and vector

$$d_{r,\omega} = \omega\left(D - rL\right)^{-1} b, \tag{2.24}$$

where $\omega \in \mathbb{C}\backslash\{0\}$, is the *overrelaxation parameter* and r is the *acceleration parameter*.

To simplify the notation we set

$$\tilde{L} = D^{-1}L,\ \tilde{U} = D^{-1}U, \text{ and } \tilde{b} = D^{-1}b.$$

Then, the original system (2.9) is equivalent to $\tilde{A}x = \tilde{b}$ and the splitting of matrix A in (2.15) is equivalent to $\tilde{A} = I - \tilde{L} - \tilde{U}$.

With this notation, the AOR iteration matrix becomes

$$H_{r,\omega} = \left(I - r\tilde{L}\right)^{-1}\left[(1-\omega)I + (\omega - r)\tilde{L} + \omega\tilde{U}\right] \tag{2.25}$$

and the vector

$$d_{r,\omega} = \omega \left(I - r\tilde{L}\right)^{-1} \tilde{b}. \tag{2.26}$$

So, the AOR iterative scheme (2.22) is given by

$$x^{(k+1)} = \left(I - r\tilde{L}\right)^{-1} \left[(1-\omega)\, I + (\omega - r)\, \tilde{L} + \omega\tilde{U}\right] x^{(k)} + \omega \left(I - r\tilde{L}\right)^{-1} \tilde{b} \tag{2.27}$$

For specific values of parameter r, the AOR method reduces to other well-known methods.

If we distinguish the two cases $r = 0$ and $r \neq 0$ we can observe the following.

For $r = 0$ the AOR iteration scheme (2.27) reduces to

$$x^{(k+1)} = \left[(1-\omega)\, I + \omega \left(\tilde{L} + \tilde{U}\right)\right] x^{(k)} + \omega\tilde{b} \quad k = 0, 1, 2, \ldots \tag{2.28}$$

which is of the form (2.18) with $H = \tilde{L} + \tilde{U} = D^{-1}(L + D)$ and $H_\omega = (1-\omega)\, I + \omega H$.

Therefore, (2.28) constitutes the extrapolation scheme of

$$x^{(k+1)} = \left(\tilde{L} + \tilde{U}\right) x^{(k)} + \tilde{b} \quad k = 0, 1, 2, \ldots \tag{2.29}$$

with *extrapolation parameter* ω.

Equation (2.29) is the Jacobi scheme and (2.28) is the Extrapolated Jacobi scheme corresponding to the original system (2.9).

So, for $r = 0$, the AOR method is an Extrapolated Jacobi method with *extrapolation parameter* ω.

For $r \neq 0$, the iteration matrix in (2.25) can be written as

$$\begin{aligned}
& H_{r,\omega} = \left(I - r\tilde{L}\right)^{-1} \left[(1-\omega)\, I + (\omega - r)\, \tilde{L} + \omega\tilde{U}\right] \\
\Leftrightarrow \quad & H_{r,\omega} = \left(I - r\tilde{L}\right)^{-1} \left[\left(1 - \tfrac{\omega}{r} + \tfrac{\omega}{r} - \omega\right) I - r\left(1 - \tfrac{\omega}{r}\right) \tilde{L} + \omega\tilde{U}\right] \\
\Leftrightarrow \quad & H_{r,\omega} = \left(I - r\tilde{L}\right)^{-1} \left[\left(1 - \tfrac{\omega}{r}\right) I + \left(\tfrac{\omega}{r} - \omega\right) I - r\left(1 - \tfrac{\omega}{r}\right) \tilde{L} + \omega\tilde{U}\right] \\
\Leftrightarrow \quad & H_{r,\omega} = \left(I - r\tilde{L}\right)^{-1} \left[\left(1 - \tfrac{\omega}{r}\right) \left(I - r\tilde{L}\right) + \tfrac{\omega}{r}(1-r)\, I + \tfrac{\omega}{r} r\tilde{U}\right] \\
\Leftrightarrow \quad & H_{r,\omega} = \left(1 - \tfrac{\omega}{r}\right) I + \tfrac{\omega}{r} \left(I - r\tilde{L}\right)^{-1} \left[(1-r)\, I + r\tilde{U}\right]
\end{aligned}$$

and the vector (2.26) is equivalent to

$$d_{r,\omega} = \omega \left(I - r\tilde{L}\right)^{-1} \tilde{b} \;\Leftrightarrow\; d_{r,\omega} = \frac{\omega}{r} r \left(I - r\tilde{L}\right)^{-1} \tilde{b}$$

So, for $r \neq 0$, the *AOR scheme* (2.27) can be written as

$$x^{(k+1)} = \left\{ \left(1 - \frac{\omega}{r}\right) I + \frac{\omega}{r} \left(I - r\tilde{L}\right)^{-1} \left[(1-r)\, I + r\tilde{U}\right] \right\} x^{(k)} + \frac{\omega}{r} r \left(I - r\tilde{L}\right)^{-1} \tilde{b} \tag{2.30}$$

$k = 0, 1, 2, \ldots$

Comparing the form of (2.30) with that of the scheme (2.18), we can conclude that (2.30) is the extrapolated scheme of

$$x^{(k+1)} = \left(I - r\tilde{L}\right)^{-1} \left[(1-r)\, I + r\tilde{U}\right] x^{(k)} + r \left(I - r\tilde{L}\right)^{-1} \tilde{b} \tag{2.31}$$

$k = 0, 1, 2, \ldots$

with *extrapolation parameter* $\frac{\omega}{r}$, and that (2.31) corresponds to the SOR method with *overrelaxation parameter* r.

Then, for $r \neq 0$, the AOR method is an extrapolated scheme (Extrapolated SOR method with *extrapolation parameter* $\frac{\omega}{r}$) of the SOR method (with *overrelaxation parameter* r) [61].

Next we will show that, for $r \neq 0$, the AOR method given by (2.23) and (2.24) is an Extrapolated SOR method with *extrapolation parameter* $\frac{\omega}{r}$ and *overrelaxation parameter* r.

In one hand, the SOR method with *overrelaxation parameter* ω has iteration matrix

$$H_\omega = (D - \omega L)^{-1} \left[(1-\omega)\, D + \omega U\right]$$

and vector

$$d_\omega = \omega \left(D - \omega L\right)^{-1} b$$

On the other hand, the SOR method with *overrelaxation parameter* r has iteration matrix

$$H_r = (D - rL)^{-1} \left[(1-r)\, D + rU\right]$$

and vector

$$d_r = r \left(D - rL\right)^{-1} b$$

Then, the Extrapolated SOR iteration matrix with *extrapolation parameter* ω and *overrelaxation parameter* r is given by:

$$\begin{aligned} & H_{r,\omega} = (I - \omega)\, I + \omega H_r \\ \Leftrightarrow \quad & H_{r,\omega} = (I - \omega)\, I + \omega \left(D - rL\right)^{-1} \left[(1-r)\, D + rU\right] \end{aligned}$$

with vector

$$d_{r,\omega} = \omega d_r \quad \Leftrightarrow \quad d_{r,\omega} = \omega r \left(D - rL\right)^{-1} b.$$

Finally, the Extrapolated SOR iteration matrix with *extrapolation parameter* $\frac{\omega}{r}$ and *overrelaxation parameter* r is given by:

$$\begin{aligned}
& H_{r,\omega} = \left(1-\tfrac{\omega}{r}\right) I + \tfrac{\omega}{r} (D - rL)^{-1} [(1-r) D + rU] \\
\Leftrightarrow\ & H_{r,\omega} = \left(1-\tfrac{\omega}{r}\right) (D - rL)^{-1} (D - rL) + \tfrac{\omega}{r} (D - rL)^{-1} [(1-r) D + rU] \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1} \left\{ \left(1-\tfrac{\omega}{r}\right) (D - rL) + \tfrac{\omega}{r} [(1-r) D + rU] \right\} \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1} \left[\left(1-\tfrac{\omega}{r}\right) D - (r-\omega) L + \left(\tfrac{\omega}{r} - \omega\right) D + \omega U \right] \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1} \left[\left(1-\tfrac{\omega}{r} + \tfrac{\omega}{r} - \omega\right) D + (\omega - r) L + \omega U \right] \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1} [(1-\omega) D + (\omega - r) L + \omega U] \\
\Leftrightarrow\ & (2.23) \text{ which is the } \textit{AOR iteration matrix}
\end{aligned}$$

and vector

$$\begin{aligned}
& d_{r,\omega} = \tfrac{\omega}{r} r (D - rL)^{-1} b \\
\Leftrightarrow\ & d_{r,\omega} = \omega (D - rL)^{-1} b \\
\Leftrightarrow\ & (2.24) \text{ which is the } \textit{AOR iteration vector.}
\end{aligned}$$

By distinguishing the two cases of the AOR method ($r = 0$ and $r \neq 0$) the theorem 2.3.10 leads to the following two theorems.

Theorem 2.3.11. *[61] Sufficient conditions for the extrapolated Jacobi scheme (2.28) to converge ($\rho(H_{0,\omega}) < 1$) are: the Jacobi scheme (2.29) converges ($\rho(H_{0,1}) < 1$) and $0 < \omega < \frac{2}{1+\rho(H_{0,1})}$.*

Moreover we have $\rho(H_{0,\omega}) \leq |1-\omega| + \omega\rho(H_{0,1}) < 1$.

Theorem 2.3.12. *[61] Sufficient conditions for the extrapolated SOR scheme (2.30) to converge ($\rho(H_{r,\omega}) < 1$) are: the SOR scheme (2.31) converges ($\rho(H_{r,r}) < 1$) and $0 < \omega < \frac{2r}{1+\rho(H_{r,r})}$.*

Moreover we have $\rho(H_{r,\omega}) \leq \left|1-\frac{\omega}{r}\right| + \frac{\omega}{r}\rho(H_{r,r}) < 1$.

It is known that an M-matrix is also an H-matrix; hence, theorems for H-matrices are valid for M-matrices. [125]

The next theorem applies when A is an M-matrix.

Theorem 2.3.13. *[61] Let A be an M-matrix. Then $\rho(H_{r,\omega}) < 1$ for $0 \leq r \leq 1$ and $0 < \omega \leq max\left\{1, \frac{2r}{1+\rho(H_{r,r})}\right\}$, and also for $1 < r < \frac{2}{1+\rho(H_{0,1})}$ and $0 < \omega < \frac{2r}{1+\rho(H_{r,r})}$.*
(Note: If the largest of the two quantities in the braces is not the number 1 then the second inequality as regards ω must be a strict one.)

For more detailed information about the AOR method see section 4.1.3.

2.3.3 Nonstationary methods

Unlike nonstationary methods, the trouble with stationary schemes is that they do not make use of information that has accumulated throughout the iteration.

The computations of nonstationary methods involve information that changes at each iteration. Typically, constants are computed by taking inner products of residuals or other vectors arising from the iterative method.

Most of the nonstationary methods are based on the idea of sequences of orthogonal vectors. Chebyshev method, which is based on orthogonal polynomials, is an exception.

Methods based on orthogonalization were developed by a number of authors in the early '50s [10].

In [43], Gleich et al. proved that PageRank can be successfully computed using linear system iterative solvers. The iterative linear solvers used were chosen because they worked with nonsymmetric matrices and were easily parallelizable.

Thus, from the stationary methods, the authors used Jacobi iterations and from the nonstationary methods they used several Krylov subspace methods: Generalize Minimum Residual (GMRES); Biconjugate Gradient (BiCG); Quasi-Minimal Residual (QMR); Conjugate Gradient Squared (CGS); Biconjugate Gradient Stabilized (BiCGSTAB) and Chebyshev Iterations. These methods are based on certain minimization procedures and only use the matrix through matrix-vector multiplication. They also used preconditioners to improve the convergence of the Krylov methods: Parallel Jacobi; Block Jacobi and Adaptive Schwarz preconditioners.

Detailed description of this algorithms can be consulted in [10] and [119].

One main advantage of the Krylov subspace methods applied in PageRank is that they are parallelizable [12, 43].

In [43] the methods were implemented in parallel and the numerical results showed that GMRES and BiCGSTAB were overall the best choice and, for the majority of the matrices used, provided faster convergence than power iterations.

The GMRES algorithm [119, 120] ranks among the most popular methods to solve sparse, large and nonsymmetric linear systems.

Experiments with an application of Krylov subspace methods, such as GMRES for PageRank linear system, can also be found in Corso et al. [34, 35].

In [35] Del Corso et al also compared the effectiveness of stationary and nonstationary methods on some portion of real web matrices for different choices of α. They concluded that stationary methods were very reliable and more competitive for small values of α (when the problem is well conditioned). However, for large values of α nonstationary methods, such as Preconditioned BiCGSTAB or Restarted preconditioned GMRES, became competitive with stationary methods in terms of Mflops count as well as in number of iterations necessary to reach convergence. From this study it is possible to see that many nonstationary methods such as CGS, BiCG and Quasi-minimal Residual (QMR) are not reliable since statistically they have many chances to fail.

The Arnoldi-type algorithm [49], referred in section 2.2.2, deals with the PageRank problem from an eigenproblem point of view, while the GMRES algorithm [34, 35, 43] does so from a linear system point of view. Both approaches are Krylov subspace methods that rely on the Arnoldi process.

In [141], Wu and Wei focus on a theoretical and numerical comparison of the two approaches. They developed some Krylov subspace techniques, such as preconditioned GMRES, to solve the linear system above.

Although the two approaches are not mathematically equivalent, in that GMRES is an oblique projection method, while the Arnoldi-type algorithm can be viewed as a classical orthogonal projection method, Wu and Wei, in [141], show that by choosing appropriate initial vectors, the search subspaces of the two approaches are essentially the same. The authors present a computable formula for the difference between the approximations by the two approaches. They conclude that the better method, the Arnoldi or the GMRES algorithm, is problem-dependent; that the restarted GMRES is prone to stagnate and the restarted Arnoldi-type algorithm may also stagnate.

Recently, in [147], Zhang formulates the PageRank problem as a singular linear system. In this paper, the PageRank problem $A\pi = \pi$ is formulated as a consistent singular linear system $(I - A)\,\pi = 0$ and the Full Orthogonalization method (FOM) is applied to solve it.

As mentioned above, approaching the PageRank problem as linear system in not new, but, this paper is the first in literature to solve the singular linear system for the PageRank problem.

The coefficient matrix $I - A$ is singular since 1 is the largest eigenvalue of A. When some specific Krylov subspace methods are used for solving such

a singular linear system, it is possible to encounter unexpected breakdowns caused by rank deficiencies. Fortunately, the singular matrix $I - A$ is characterized by index one, namely $index\,(I - A) = 1$, from which unexpected breakdowns can survive when FOM is applied. Here $index$ denotes the size of the largest Jordan block corresponding to zero eigenvalues of a matrix.

In [147], the breakdown performance of the FOM on a general singular linear system is analyzed, and the authors conclude that FOM can determine a solution if it converges, without any unfortunate breakdowns for the target problem used. A vector extrapolation method, based on Ritz values which directly stems from the Arnoldi-Extrapolation algorithm in [140], referred in section 2.2.2, is also proposed to speed up the convergence of FOM. This extrapolation is based on power iterations and Ritz values and it begins with the current approximation $\pi^{(0)}$ being a linear combination of the first three largest eigenvectors of the Google matrix G. The resulting algorithm is called FOM-EXT.

One advantage of this extrapolation is the exact eigenvalues can be approximated by Ritz values. That is, in spite of unknown values of λ_2 and λ_3, they can be estimated along with FOM.

Numerical experiments, on Google web matrices of different sizes, are carried out to evaluate the proposed algorithms. Computational results confirm that the new algorithms are always effective and much better than the refined Arnoldi-type algorithm. This approach achieves a significant saving in computing time, especially when the damping factor is close to one. In one example, the FOM-EXT algorithm saved more than ninety percent of time of the refined Arnoldi-type algorithm.

When $\alpha = 0.99$ and $\alpha = 0.995$, the refined Arnoldi-type algorithm stagnates for two of the examples without determining any approximate solutions. But FOM and FOM-EXT removed the stagnation.

2.4 Multilinear PageRank

In [47], Gleich et al. extended the PageRank problem to a higher-order Markov chain. In the higher-order Markov chains, the stochastic process depends on more history than just the previous state. For instance, in a second-order chain, the choice of state at the next step depends on the last two states.

However, this system that has attractive theoretical properties it is, in many cases, computationally intractable. Gleich et al., motivated by the

"spacey random surfer" model, developed a computationally tractable approximation to the higher-order PageRank vector that involves a system of polynomial equations and called it Multilinear PageRank.

In the "spacey random surfer" model, the surfer continuously forgets its own immediate history but remembers bits and pieces of history and is influenced by this information. The surfer combines the current state with this aggregate history to determine the next state.

This process is the motivation for the Multilinear PageRank.

The higher-order PageRank problem behaves much like the standard PageRank problem in terms of guaranteed uniqueness and fast convergence. However, the Multilinear PageRank problem only has uniqueness and fast convergence in a narrower regime. Outside that regime, the existence of a solution is guaranteed, but uniqueness is not.

For the Multilinear PageRank problem, outside the uniqueness regime, the convergence of simple iterative methods is highly dependent on the data. Five different algorithms were used to solve the multilinear system: a fixed-point method (as Power method and Richardson method); a shifted fixed-point method; a nonlinear inner-outer iteration; an inverse iteration (as the Inverse Power method) and a Newton iteration.

All of these algorithms are fast in the unique regime. Outside that range, the author used exhaustive enumeration and random sampling to build a repository of problems that do not converge with the methods above. Among the test cases, the Newton's method and the inner-outer algorithm presented the most reliable convergence properties.

Due to this result, a two-phase approach for solving the problems emerged: first experiment with the simple shifted method and, if it does not converge, apply either a Newton or an inner-outer iteration.

The Multilinear PageRank problem is only interesting for massive problems, the higher-order PageRank approach should be used when there is $O(n^2)$ memory available, except if there is a modeling reason to use the multilinear approach. This is the case for some applications, like in modern bioinformatics and social network analysis, where the large problems involved make the higher-order approach difficult to scale.

2.5 Parallel PageRank Computations

Another way of accelerating the computation of the PageRank vector is parallel processing. Obviously the computation of the PageRank is performed using parallel processing. The amount of storage required as well as the amount of calculations needed are not compatible with simple serial computations.

Since PageRank is a popular benchmark for parallel programming models, various versions of PageRank have been implemented in different parallel platforms.

Kollias et al. [80] have conducted experiments that executed the PageRank calculations in parallel with very little overhead communication between processors. Adaptive schemes for the asynchronous computation of PageRank were explored.

There have been a few implementations of PageRank on CPU based infrastructures, including PC cluster [20, 96, 118] or P2P architectures [97].

In data mining communities PageRank has been extensively studied, and many different approximate algorithms [3, 70], have been developed over the years. Interesting research work that include parallel PageRank can be found in [12].

Extensive studies have been conducted to shape PageRank suitable for distributed computation [29, 43, 44, 134, 149].

Internet search engines use Web crawlers to download data to their central servers to process queries. However, using crawlers bring some disadvantages, mainly, crawlers do not scale.

In [134] Wang and DeWitt described and evaluated an distributed approach in which every Web server acts as an individual search engine on its own pages, eliminating the need for crawlers and centralized servers. In the paper, a series of PageRank variants, including Local PageRank, ServerRank, Local PageRank Refinement, and Result Fusion, is proposed.

A real Web data set was used in the experiments, which showed that a distributed approach can produce PageRank vectors that are comparable to the results of the centralized PageRank algorithm.

Gleich et al. [43, 44], considered the PageRank linear system formulation to investigate the convergence of iterative stationary and Krylov subspace methods, including the convergence dependency on teleportation, using parallel PageRank computing and compared the methods efficiency.

The random teleportation used in PageRank strongly affects the conver-

gence of power iterations [63]. High teleportation can help spam pages to accumulate PageRank [53], but a redution in teleportation difficults the convergence of standard Power method. In [43] the authors investigated how the convergence of the linear system is affected by a reduction in a degree of teleportation.

To use an iterative linear solver for PageRank problems it is necessary that it works with nonsymmetric matrices and it must be easily parallelizable. Thus, from the stationary methods, the authors in [43, 44] used Jacobi iterations and from the non-stationary methods they chose several Krylov subspace methods: Generalize Minimum Residual (GMRES); Biconjugate Gradient (BiCG); Quasi-Minimal Residual (QMR); Conjugate Gradient Squared (CGS); Biconjugate Gradient Stabilized (BiCGSTAB) anf Chebyshev Iterations.

The Krylov methods can be improved by the use of preconditioners. So, in [43, 44], it was used parallel Jacobi, Block Jacobi and Adaptive Schwarz preconditioners.

Manaskasemsak et al. [96] presented a parallel PageRank computation on a Gigabit PC cluster and achieved significant improvement. They stored a piece of the Web graph on separate hard disks for each processor and iterated through these files as necessary.

In [43, 44], an alternative approach was used, the entire Web graph was kept in memory, on a distributed memory parallel computer, while computing the PageRank vector.

The parallel PageRank codes use the Portable, Extensible Toolkit for Scientific computation (PETSc) which contains parallel implementations of many linear solvers, including GMRES, BiCGSTAB and BiCG.

In [43] the authors used six Web related directed graphs, being the largest data set used the "av" graph (the Alta Vista 2003 crawl of the Web with 1.4 billion nodes). The numerical results showed that: GMRES and BiCGSTAB were overall the best choice of solution methods for the PageRank class of problems, and for most graphs, provided faster convergence than power iterations; Power and Jacobi methods have approximately the same rate and the most stable convergence pattern; convergence of Krylov methods strongly depends on the graph and is non-monotonic; the QMR, CGS and Chebyshev methods did not converge. They also demonstrated that the linear system PageRank can converge for much larger values of the teleportation coefficient (damping factor α) than standard power iterations.

In [44] the same authors used a power law Web graph with 1.5 billion nodes on a distributed memory 140 processor RLX cluster. The numerical

results showed that power iterations and BiCGSTAB are overall the best choice of solution methods. With the parallel implementation it was possible to reduce the time to compute the PageRank vector on a full Web graph from 10 hours to 5.5 minutes.

Zhu et al. [149] proposed a distributed PageRank computation (DPC) algorithm based on Iterative Aggregation-disaggregation (IAD) method with Block Jacobi smoothing. The basic idea was divide-and-conquer. They treated each Web site as a node to explore the block structure of the Web [77]. Each node, in the distributed system, computed a PageRank vector for its local pages by links within sites and and then updated its local PageRank through low volume communication with a given coordinator.

Experiments on three real Web graphs showed that this method converged 5-7 times faster than the traditional Power method and that the DPC algorithm achieved better approximation than the work presented in [134].

Existing approaches to PageRank parallelization can be divided into two classes: Exact computations and Approximations.

In case of the Exact computations the Web graph is initially partitioned into blocks: grouped randomly [121], lexicographically sorted by page [96, 123], or balanced according to the number of links [43]. Then, standard iterative methods such as Jacobi or Krylov subspace [43] are performed over these pieces in parallel. The partitions, periodically, must exchange information.

The main idea behind PageRank Approximations approaches is that it might be sufficient to get a rank vector which is comparable, but not equal, to PageRank. Instead of ranking pages, higher-level formations are used. The inner structure of these formations can then be computed in an independently parallel manner, as in BlockRank [77], the U-Model [21], SiteRank [142], ServerRank [134], or HostRank [37].

In [79] Kohlschutter et al. tried to take the best out of both approaches: the exactness of a straight PageRank computation but the speed of an approximation, without any centralized re-ranking.

The two-dimensional view of the Web had the host ID as the only discriminator and the Gauss-Seidel method, for solving linear systems, was used to calculate the PageRank vector in parallel.

Partitioned PageRank was implemented in Java using a P2P network with a central coordinator instance. This coordinator was only responsible for arranging the iteration process at partition-level and did not know anything about the rank scores or the link structure.

This approach produced the same ranks as the original PageRank, while

being more scalable than other parallel PageRank algorithms.

Buehrer et al. [23] implemented PageRank algorithm on Cell BE. Cell BE processor is a multi-core processor designed for computational intensive algorithms. However, due to the large number of random memory writes, and data transfer between PPE (power processing element) and SPE (synergistic processor element) required by PageRank algorithm, implementation took more time than on single processor Xeon.

In [81], Kumar et al. tried to solve this problem and presented a new approach which reduces the data transfer between PPE and SPE drastically and lead to a better performance.

The introduction of Multi-core Processors has proved to be an alternative for solving computational intensive algorithms in efficient ways in terms of time. Kumar et al. used two multi-core architectures, STI Cell BE and CUDA (Computer Unified Device Architecture), to implement the parallelization of PageRank algorithm and compared the two approaches for standard Web graphs. It was found that implementations on CUDA performed much better than on Cell.

Another parallel implementation of PageRank algorithm on a CUDA platform can be viewed in [36]. Linear vectors were used to store the computed PageRank values and the nodes were stored in a special data structure, the Binary Link Structure.

In [25] it was also proposed a parallel PageRank algorithm that uses the architectural benefits that CUDA brings.

The MapReduce distributed programming model [31] provides a new solution to large data sets parallel computing.

Recently, Liu et al. [94] developed two different algorithms. The first algorithm was a Parallel PageRank algorithm based on Power method combined with MapReduce concepts. The second algorithm, the Power Iteration Acceleration method, was a distributed iteration acceleration model based in vector computing.

The Parallel PageRank algorithm based on power iteration acceleration needs more computations in every iteration compared to the Parallel PageRank algorithm based on Power method, however reaches convergence faster since the number of iterations required is smaller.

Also, Whang et al. [136] studied general approaches for designing scalable data-driven graph algorithms using the PageRank algorithm as a case study. In particular, they used three different algorithm design axes (i.e., work activation, data access pattern, and scheduling) to present eight dif-

ferent formulations and in-memory parallel implementations of PageRank algorithm.

It was shown that, by considering data-driven formulations, it is possible to have more flexibility in processing the active nodes, which enables the development of work-efficient algorithms.

The authors showed that data-driven push-based algorithms are able to achieve more than 28× the performance of standard PageRank implementations.

2.6 PageRank today

Although the Google's PageRank method was developed to evaluate the importance of web pages via their link structure, the mathematics of PageRank are entirely general and can be applied to any graph or network in any domain.

Today, the PageRank algorithm is regularly used in bibliometrics, social and information network analysis, link recommendation and prediction and system analysis of road networks. It is possible to encounter applications of PageRank to biology, chemistry, neuroscience, ecology, physics, computer systems and sports [45].

Two uses underlie the majority of PageRank applications. In the first, PageRank is used as a network centrality measure. A network centrality score determines the importance of each node in terms of the entire graph structure. These applications usually use global and near-uniform teleportation behaviors. In the second, PageRank is used as a localized measure. That is, it is used to illuminate a region of a large graph around a target set of interest. In this case, personalized teleportation behaviors allow the random surfer to teleport only to pages that are interesting to him.

In the centrality case, the input is a graph that represents relationships or flows among a set of things (such as documents, genes, proteins, roads) and the goal is to determine the expected importance of each member in light of the full set of relationships and the teleporting behavior.

In the localized case, the input is also the same type of graph, but the goal is to determine the expected importance relative to a small subset of the objects.

In both cases it is necessary to build a stochastic or substochastic matrix from a graph.

Gleich, in [45], presents a review of some of the common constructions

that produce a PageRank or pseudo-PageRank system, such as standard random walk; strongly preferential PageRank; weakly preferential PageRank; sink preferential PageRank; reverse PageRank; Dirichlet PageRank; weighted PageRank and PageRank on an undirected graph.

When PageRank is used within applications, it tends to acquire a new name, such as TrustRank, BookRank, CiteRank, VisualRank, etc.

Some of the most interesting applications of PageRank can be seen in Biology and Bioinformatics (GeneRank, ProteinRank, IsoRank). Microarray experiments measure whether or not a gene's expression is repressed or promoted in an experimental condition. Although microarrays estimate the outcomes for thousands of genes simultaneously under a few experimental conditions, the results are extremely noisy. GeneRank [136] helps to denoise them. The GeneRank algorithm uses a graph of known relationships between genes to find genes that are highly related to those promoted or repressed in the experiment, but were not themselves promoted or repressed. The microarray expression results are used as the teleportation distribution vector for a PageRank problem on a network of known relationships between genes, that is, undirected and unweighted with a few thousand nodes. The GeneRank uses a localized teleportation behaviour.

Another example is the ProteinRank [79]. Its goal is to find proteins that may share a functional annotation given an undirected network of protein-protein interactions and human-curated functional annotations about what these proteins do. This problem is also a localized use.

Another use of PageRank is for road and urban space networks, where it helps to predict both traffic flow and human movements. In [33], Jiang et al. show that PageRank is the best network measure in terms of predicting traffic on the individual roads.

The PageRank approach is also used in Sports [11]. The winner network is one of the natural network constructions for sports. In the winner network, each team is a node and node i points to node j if j won the match between i and j. These networks are often weighted by the score by which team j beat team i. Govan et al. [108] used the centrality sense of PageRank with uniform teleportation to rank football teams. Also, Radicchi [41] used PageRank on a network of tennis players. It used a uniform teleportation in a weighted network.

The BookRank is another example of a PageRank application, in this case, to Literature. Traditional library catalogues use a carefully curated set of terms to indicate the contents of books. Social cataloguing sites, such as Librarything and Shelfari, indicate the users which are the topics of books

using as data *books* and *tags*. BookRank, a localized PageRank on the bipartite book-tag graph [74], produces accurate suggestions for what books to read next.

The field of bibliometrics also uses ranking methods. Generally, PageRank is used as a centrality measure to reveal the most important journals, papers and authors. An example is the citation network among individual papers in which each node is an individual article and the edges are directed based on the citation.

TimedPageRank is an algorithm that weights the edges of the stochastic matrix in PageRank such that more recent citations are considered more important. CiteRank is a subsequent idea that uses teleportation to increase the rank of recent articles [89]. The goal of both methods is to obtain temporally relevant orderings that remove the bias of older articles in acquiring citations.

The coauthorship graph is another type of bibliographic network. For each paper, insert edges among all coauthors, so that each paper becomes a clique in the coauthorship network. This undirected network gives a practical ranking of the most important authors [52]. In [116] and [104], examples of combination of citation analysis and coauthorship analysis are presented.

The ItemRank is an example of the use of PageRank in recommender systems. A recommender system attempts to predict what its users will do based on their past behavior. Amazon and Netflix have some of the most famous recommendation systems that predict products and movies, respectively, that their users will enjoy [131].

PageRank can be even used in a social network. In this kind of network the nodes are people and the edges are some type of social relationship. The PageRank helps to solve link prediction problems to find individuals who will become friends soon. It also serves a classic role in evaluating the centrality of the people involved to estimate their social status and power. Finally, it can help to evaluate the potential influence of a node on the opinions on the network.

One of the important questions in social network analysis is finding influential individuals, that is, nodes that can spread their influence widely. To understand the origins of influence the correct model is Reverse PageRank instead of traditional PageRank. These ideas also extend to finding topical authorities in social networks by using the teleportation vector and topic-specific transition probabilities to localize the PageRank vector in TwitterRank [95].

PageRank and Reverse PageRank also provide information on the "spaminess" of particular pages through metrics such as TrustRank and BadRank. As the commercial value of websites grew, it became highly profitable to cre-

ate *spam* sites that do not contain new information content but attempt to capture Google search results by appearing to contain information. BadRank [135] and TrustRank [92] are link analysis tools to combat this problem.

Considering all that was said, an important question remains. Does Google still use PageRank?

Google reportedly uses over 200 types of ranking metrics, or signals, to determine the final order in which results are returned [39]. These evolve continuously and vary depending on when and where the search is being made. The exact role that PageRank plays in Google's search ordering is a closely guarded secret. However, based on statements from Google, the PageRank is still believed to play a kernel role in Google's search engine. For instance, in [71], Mat Cutts, a Google engineer, wrote about how Google uses PageRank to determine crawling behavior. He later wrote about how Google moved to a full substochastic matrix in terms of their PageRank vector [51]. Thus, Google's PageRank computation is a pseudo-PageRank problem now.

2.7 Conclusions

In the beginning of this chapter the PageRank model and the mathematics behind the model is explained.

There are two ways of solving the PageRank problem: as an eigenvector problem or as a linear homogeneous system.

On one hand, for the eigenvector approach, the Power method is the clear favorite. However, several authors explored different methods to compute the PageRank vector and, also, several methods to accelerate the Power method were analyzed.

On the other hand, the PageRank as a linear homogeneous system and three stationary iterative methods (Jacobi, Gauss-Seidel and SOR) to solve it were studied.

The extrapolation principle and the AOR method were also explored.

A review on nonstationary methods used for PageRank computations, on multilinear PageRank and on parallel PageRank computations is presented.

Nowadays, the PageRank problem developed by Page and Brin in [20] has evolved and new applications of the PageRank model abound. Several of these applications are discussed at the end of this chapter.

3

Acceleration of PageRank as an eigenvector problem

Contents

3.1 The Power Method

In this chapter we explore the eigenvector point of view of the PageRank problem. We propose new acceleration methods for the Power method, the LumpingE methods. These methods combine Lumping methods with Aitken extrapolation.

Several numerical experiments on datasets that were extracted as subsets of the Web and on tuned matrices are presented. In these experiments the performance of the new LumpingE methods with standard Power method and original Lumping methods are compared. Results show the advantages obtained with our new hybrid proposal.

The Power method is the oldest method for computing the principal eigenvector of a matrix and it is at the heart of both the motivation and implementation of the original PageRank algorithm [78].

As mentioned in section 2.2.1 the Power method algorithm for computing the PageRank vector begins with the uniform distribution ${\pi^{(0)}}^T = v^T = \frac{1}{n}e^T$ and computes successive iterates ${\pi^{(k+1)}}^T = {\pi^{(k)}}^T G$ where $k = 0, 1, \ldots$, until some convergence criterion is satisfied.

The Power method for computing the PageRank vector is presented in Algorithm 4. The relative rate of convergence of the algorithm is measured using the norm of the residual vector, $\left\|x^{(k+1)} - x^{(k)}\right\|$ or $\frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{\left\|x^{(k+1)}\right\|}$. Also, *tol* represents the convergence tolerance and matrix $A = G^T$.

Algorithm 4 Power Method

Given matrix A, vector v and tolerance *tol*;
Choose an initial guess $x^{(0)} := v$;
Let $k := 1$;
repeat
  Compute $x^{(k+1)} := Ax^{(k)}$;
  Compute $\delta := \left\|x^{(k+1)} - x^{(k)}\right\|$;
  Update $k := k + 1$;
until $\delta < tol$;

Proposition 3.1.1. *Let G be an $n \times n$ matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ satisfying $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$.*

The convergence rate of the Power method is $k = \left|\frac{\lambda_2}{\lambda_1}\right|$ with λ_1 the dominant eigenvalue of the Google matrix and λ_2 the subdominant eigenvalue.

Proof. Let $u_1, u_2, \ldots, u_n$ the eigenvectors that form a basis of $\mathbb{R}^n$, with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ respectively, $Au_i = \lambda_i u_i$, for $A = G^T$.

Expressing the initial vector as a linear combination of the eigenvectors, as $\pi^{(0)} = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n$ for some coefficients α_i, and applying power iterations yields

$$\begin{aligned}
\pi^{(k+1)} &= A\pi^{(k)} \\
&= A^{k+1}\pi^{(0)} \\
&= \sum_{i=1}^{n} \alpha_i A^{k+1} u_i \\
&= \sum_{i=1}^{n} \alpha_i \lambda_i^{k+1} u_i \\
&= \lambda_1^{k+1} \left[\alpha_1 u_1 + \sum_{i=2}^{n} \alpha_i \left(\frac{\lambda_i}{\lambda_1} \right)^{k+1} u_i \right]. \qquad (3.1)
\end{aligned}$$

Since $|\lambda_1| > |\lambda_2|$, as the number of iterations $k \to \infty$, the first terms on the right-hand-side will dominate. Since λ_1 is dominant, the fractions $\frac{\lambda_2}{\lambda_1}, \ldots, \frac{\lambda_n}{\lambda_1}$, all have absolute values that are strictly less than 1, and hence $\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1}, \ldots, \left(\frac{\lambda_n}{\lambda_1}\right)^{k+1}$ will tend to zero as k increases.

Furthermore, equation (3.1) reveals that the rate of convergence of the Power method will depend on the ratio of the two largest eigenvalues, $\frac{\lambda_2}{\lambda_1}$. That is, the convergence rate is $k \leq \left|\frac{\lambda_2}{\lambda_1}\right|$ and exactly $k = \left|\frac{\lambda_2}{\lambda_1}\right|$ as long as $\alpha_2 \neq 0$.

If the ratio $\left|\frac{\lambda_2}{\lambda_1}\right|$ is small, the convergence rate is fast. If λ_2 is almost as large as λ_1, $\left|\frac{\lambda_2}{\lambda_1}\right|$ is only a little smaller than 1 and then convergence will be much slower. □

Since $\lambda_1 = 1$, if λ_2 is close to 1, then the Power method is slow to converge.

Also $|\lambda_2| = \alpha$ (damping factor) and the eigengap $1 - |\lambda_2|$ for the Web Markov matrix G is given exactly by the teleport probability $1 - \alpha$ [63].

Thus, when the teleport probability is large and the personalization vector v is uniform over all pages, the Power method works reasonably well. However, this increases the effect of link spam and pages can achieve unfairly high rankings. Indeed, a high teleport probability (i.e., small values of α) means that every page is given a fixed "bonus" rank (equal to all pages when v is uniform).

On the other hand, for small values of $1 - \alpha$ (i.e., values of α close to 1) the convergence of PageRank slows down dramatically.

As an example we show the tiny web of seven pages presented in Section 2.1 .

Figure 3.1 compares the convergence rate of the Power Method for different values of α, $\alpha \in \{0.80, 0.85, 0.90, 0.95, 0.99\}$ using a uniform v.

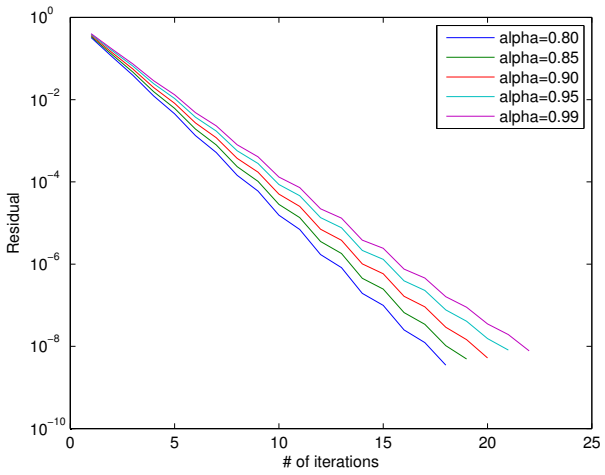


Figure 3.1: Comparison of convergence rate for the Power method on a web of seven pages for $\alpha \in \{0.80, 0.85, 0.90, 0.95, 0.99\}$.

Note that increasing the damping factor α slows down convergence. For $\alpha = 0.80$ 18 iterations are necessary for convergence, while for $\alpha = 0.99$ 22 are required.

3.2 The Power method with Aitken Extrapolation

Aitken's process is a well-known sequence transformation to accelerate of the rate of convergence of a slowly converging sequence [17].

Since $1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$, it is known that the Power method, ${\pi^{(k)}}^T G$, converges to the principal eigenvector of the Markov matrix G. Assuming that the starting vector lies in the subspace spanned by the eigenvectors of G, the Power method can be expanded has

$$\begin{aligned} {\pi^{(k)}}^T &= {\pi^{(k-1)}}^T G \\ &= {\pi^{(0)}}^T G^k \\ &= {\pi^{(0)}}^T \left(e\pi^T + \textstyle\sum_{i=2}^{n} \lambda_i^k x_i y_i^T \right) \\ &= \pi^T + \alpha_2 \lambda_2^k y_2^T + \textstyle\sum_{i=3}^{n} \alpha_i \lambda_i^k y_i^T \end{aligned}$$

where x_i and y_i, $i = 1, \ldots, n$, are, respectively, the right-hand and left-hand eigenvectors corresponding to λ_i, and $\alpha_i = {\pi^{(0)}}^T x_i$, $i = 2, \ldots, n$. The size of the subdominant eigenvalue λ_2 governs the number of power iterations. Note that π^T is hidden until $\lambda_2^k \to 0$, which takes a while when $|\lambda_2|$ is large. The extrapolation technique removes the spoiler.

Proposition 3.2.1. *To improve the convergence, if $|\lambda_2| > |\lambda_3|$, then ${\pi^{(k)}}^T - \alpha_2 \lambda_2^k y_2^T$ will be a better approximation to π^T, and $\alpha_2 \lambda_2^k y_2^T$ can be estimated by Aitken's delta square process based on three steps [78]:*

$$\alpha_2 \lambda_2^k y_2^T \approx \frac{\left(\Delta {\pi^{(k)}}^T\right)^2}{\Delta^2 {\pi^{(k)}}^T}$$

$$\Delta {\pi^{(k)}}^T = {\pi^{(k+1)}}^T - {\pi^{(k)}}^T$$

$$\Delta^2 {\pi^{(k)}}^T = {\pi^{(k+2)}}^T - 2{\pi^{(k+1)}}^T + {\pi^{(k)}}^T$$

where $(*)^2$ indicates component-wise squaring of the vector elements.

So, by subtracting $\alpha_2 \lambda_2^k y_2^T$ to ${\pi^{(k)}}^T$ we can accelerate the convergence of the Power method.

Next, as in section 2.2.1, consider $A = G^T$, ${\pi^{(k)}}^T = {\pi^{(k-1)}}^T G \Leftrightarrow \pi^{(k)} = A\pi^{(k-1)}$ and that matrix A has n distinct eigenvectors u_i: $Au_i = \lambda_i u_i$.

We will assume that the starting vector $\pi^{(0)}$ lies in the subspace spanned by the eigenvectors of A, $\pi^{(0)} = u_1 + \sum_{i=2}^{n} \alpha_i u_i$ and will use $\alpha_2 \lambda_2^k u_2$ instead of $\alpha_2 \lambda_2^k y_2^T$.

Proposition 3.2.2. *The quantity $\alpha_2 \lambda_2^k u_2$ can be estimated by Aitken's delta square process by using the two subsequent iterates of the Power method $\left(\pi^{(k+1)}, \pi^{(k+2)}\right)$:*

$$\alpha_2 \lambda_2^k u_2 \approx \frac{\left(\Delta \pi^{(k)}\right)^2}{\Delta^2 \pi^{(k)}},$$

with $\Delta \pi^{(k)} = \pi^{(k+1)} - \pi^{(k)}$ *and* $\Delta^2 \pi^{(k)} = \pi^{(k+2)} - 2\pi^{(k+1)} + \pi^{(k)}$ *[78].*

Proof. Beginning by assuming that $\pi^{(k)}$ can be expressed as a linear combination of the first two eigenvectors.

$$\pi^{(k)} = u_1 + \alpha_2 u_2 \tag{3.2}$$

We will calculate an estimate of the principal eigenvector u_1 in closed form using the successive iterates. This approximation becomes increasingly accurate as k becomes larger.

Applying the Power method and knowing that (λ_1, u_1) and (λ_2, u_2) are eigenpairs of A:

$$\pi^{(k+1)} = A\pi^{(k)} = \lambda_1 u_1 + \alpha_2 \lambda_2 u_2$$

Since $\lambda_1 = 1$,

$$\pi^{(k+1)} = u_1 + \alpha_2 \lambda_2 u_2. \tag{3.3}$$

Applying another iteration of the Power method,

$$\pi^{(k+2)} = A\pi^{(k+1)} = u_1 + \alpha_2 \lambda_2^2 u_2 \tag{3.4}$$

Let us define,

$$g_i = \left(\Delta \pi_i^{(k)}\right)^2 = \left(\pi_i^{(k+1)} - \pi_i^{(k)}\right)^2,$$

$$h_i = \Delta^2 \pi_i^{(k)} = \pi_i^{(k+2)} - 2\pi_i^{(k+1)} + \pi_i^{(k)}$$

where π_i represents the component i of vector π.

Using simple algebra and equations (3.3) and (3.4),

$$g_i = \alpha_2^2 \left(\lambda_2 - 1\right)^2 \left(u_2\right)_i^2, \quad h_i = \alpha_2 \left(\lambda_2 - 1\right)^2 \left(u_2\right)_i.$$

Considering f_i as the quotient $\frac{g_i}{h_i}$:

$$f_i = \alpha_2 \left(u_2\right)_i .$$

Therefore, $f = \alpha_2 u_2$.

Hence, from equation (3.2), we have a closed-form solution for u_1:

$$u_1 = \pi^{(k)} - \alpha_2 u_2 = \pi^{(k)} - f.$$

□

The pseudocode for the Aitken Extrapolation method is given in Algorithm 5 and for the Power method with Aitken Extrapolation in Algorithm 6. The pseudocodes were obtained from [78].

Algorithm 5 Aitken Extrapolation

Given vectors $\pi^{(k+2)}$, $\pi^{(k+1)}$, $\pi^{(k)}$;
for $i := 1 : n$ **do**
  Compute $g_i := \left(\pi_i^{(k+1)} - \pi_i^{(k)}\right)^2$;
  Compute $h_i := \pi_i^{(k+2)} - 2\pi_i^{(k+1)} + \pi_i^{(k)}$;
  Compute $\pi_i := \pi_i^{(k)} - \frac{g_i}{h_i}$;
end for

Algorithm 6 Power method with Aitken Extrapolation

Given matrix A, vector v and tolerance tol;
Choose an initial guess $\pi^{(0)} := v$;
Let $k := 0$;
repeat
  Compute $\pi^{(k+1)} := A\pi^{(k)}$;
  Compute $\delta := \left\|\pi^{(k+1)} - \pi^{(k)}\right\|_1$;
  periodically, $\pi^{(k+1)} := \text{AITKEN}\left(\pi^{(k+1)}, \pi^{(k)}, \pi^{(k-1)}\right)$;
  Update $k := k + 1$;
until $\delta < tol$;
Set $\pi := \pi^{(k+1)}$

3.3 The Lumping1 Method

The basic idea of the Lumping methods is to perform the PageRank computation for the nondangling nodes separately.

In [67] Ipsen and Selee developed an algorithm for computing PageRank that lumps all dangling nodes into a single node. This algorithm should become more competitive as the web expands and the number of dangling nodes increases.

The algorithms in [86, 91] are special cases of the Ipsen algorithm [67] because they allow the personalization vector v and dangling node vector w to be different which facilitates the implementation of TrustRank [53].

This algorithm can also be extended to a general Google matrix that contains several different dangling node vectors.

Let n be the number of web pages and k the number of nondangling nodes among the web pages, $1 \leq k < n$. If the rows and columns of the hyperlink matrix H are permuted (i.e., the indices reordered) so that the rows corresponding to dangling nodes are at the bottom of the hyperlink matrix H, then H is of the following form:

$$H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}$$

where the matrix H_{11}, $k \times k$, represents the links among the nondangling nodes (ND), and H_{12}, $k \times (n-k)$, represents the links from nondangling nodes to dangling nodes (D). The $(n-k)$ zero rows in H are associated with the $(n-k)$ dangling nodes, see Figure 3.2.

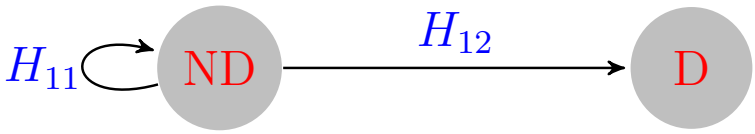


Figure 3.2: A simple model of the link structure of the Web divided in D and ND nodes. ND represents the set of nondangling nodes, and D represents the set of dangling nodes. The submatrix H_{11} represents all the links from nondangling nodes to nondangling nodes and H_{12} represents all the links from nondangling nodes to dangling nodes.

The elements in the nonzero rows of H (k first rows) are nonnegative and

sums one,

$$H_{11} \geq 0, \quad H_{12} \geq 0, \quad H_{11}e + H_{12}e = e, \quad \text{where} \quad e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix},$$

and the inequalities are to be interpreted element wise. To obtain a stochastic matrix, it is necessary to add artificial links to the dangling nodes. That is, each zero row in H is replaced by the same dangling node vector

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \quad w \geq 0, \quad \|w\|_1 = w^T e = 1.$$

Here w_1 is $k \times 1$, w_2 is $(n-k) \times 1$, $\|\cdot\|_1$ denotes the ℓ_1-norm.

The resulting matrix $S = H + dw^T = \begin{bmatrix} H_{11} & H_{12} \\ ew_1^T & ew_2^T \end{bmatrix}$, where $d = \begin{bmatrix} 0 \\ e \end{bmatrix}$, is stochastic, that is, $S \geq 0$ and $Se = e$.

With the personalization vector, v, the matrix has a unique stationary distribution

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \quad v \geq 0, \quad \|v\|_1 = v^T e = 1$$

where v_1 is $k \times 1$, v_2 is $(n-k) \times 1$.

As mentioned before, the Google matrix is defined as the convex combination

$$G = \alpha S + (1-\alpha)\, ev^T, \quad 0 \leq \alpha < 1,$$

and G has a unique stationary distribution,

$$\pi^T G = \pi^T, \quad \pi \geq 0, \quad \|\pi\|_1 = 1.$$

The element i of the stationary distribution π, represents the PageRank for web page i.

If we partition the PageRank conformally with G,

$$\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix},$$

then π_1 represents the PageRank associated with the nondangling nodes and π_2 represents the PageRank of the dangling nodes.

Thus, the Google matrix has the following block structure:

$$G = \begin{bmatrix} G_{11} & G_{12} \\ eu_1^T & eu_2^T \end{bmatrix},$$

where

$$G_{1i} = \alpha H_{1i} + (1-\alpha)\, ev_i^T, \quad u_i = \alpha w_i + (1-\alpha)\, v_i, \quad i = 1, 2.$$

$$u = \alpha w + (1-\alpha)\, v = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \ G_{11} \text{ is } k \times k, \text{ and } G_{12} \text{ is } k \times (n-k).$$

Note that $u_1 = v_1$ and $u_2 = v_2$ if the dangling vector equals the personalization vector.

Ipsen and Selee [67] have shown that all of the dangling nodes can be lumped into a single node and the PageRank of the nondangling nodes can be computed separately from that of the dangling nodes. They presented a simple algorithm, which applies the Power method to the smaller lumped matrix and has the same convergence rate as that of the Power method applied to the full matrix G, for computing the PageRank vector π.

Theorem 3.3.1 shows a similarity transformation that reduces the matrix G to block upper triangular form and illustrates how the PageRank π^T can be given in terms of the stationary distribution σ of the small matrix $G^{(1)}$.

Theorem 3.3.1. *Let* $X = \begin{bmatrix} I_k & 0 \\ 0 & L \end{bmatrix}$, *with* $L = I_{n-k} - \frac{1}{n-k}\hat{e}e^T$, $\hat{e} = e - e_1 = [0, 1, 1, \ldots, 1]^T$ *the first canonical basis vector, and* $I_n = [e_1 \cdots e_n]$ *the identity matrix of order* n.

Then, $XGX^{-1} = \begin{bmatrix} G^{(1)} & * \\ 0 & 0 \end{bmatrix}$, *where* $G^{(1)} = \begin{bmatrix} G_{11} & G_{12}e \\ u_1^T & u_2^T e \end{bmatrix}$.

The matrix $G^{(1)}$ *is stochastic of order* $k+1$ *with the same nonzero eigenvalues as* G.

Furthermore, let $\sigma^T G^{(1)} = \sigma^T$, $\sigma^T \geq 0$, $\|\sigma\| = 1$, *with partition* $\sigma^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma_{k+1} \end{bmatrix}$, *where* σ_{k+1} *is a scalar.*

Then the PageRank vector π^T *equals* $\pi^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{pmatrix} G_{12} \\ u_2^T \end{pmatrix} \end{bmatrix}$.

Proof. See [102] based on [67] □

In [67] it was presented an algorithm, based on Theorem 3.3.1 for computing the PageRank π from the stationary distribution σ of the lumped matrix $G^{(1)}$.

The input to the Algorithm 7 is the personalization vector v, the dangling node vector w, the damping factor α and the nonzero elements of the hyperlink matrix H. The output is an approximation $\hat{\pi}$ to the PageRank π, which is computed from an approximation $\hat{\sigma}$ of σ.

Algorithm 7 Lumping 1 method

Given H_{11}, H_{12}, v_1, v_2, w_1, w_2, α, tol;
Choose an initial vector $\hat{\sigma}^T := \left[\hat{\sigma}_{1:k}^T \;\; \hat{\sigma}_{k+1}\right]$ with $\hat{\sigma} \geq 0$ and $\|\hat{\sigma}\| := 1$;
while no convergence **do**
  Compute $\hat{\sigma}_{1:k}^T := \alpha \hat{\sigma}_{1:k}^T H_{11} + (1-\alpha)\, v_1^T + \alpha \hat{\sigma}_{k+1} w_1^T$;
  Compute $\hat{\sigma}_{k+1} := 1 - \hat{\sigma}_{1:k}^T e$;
end while
Compute $\hat{\pi}^T := \left[\hat{\sigma}_{1:k}^T \;\; \alpha \hat{\sigma}_{1:k}^T H_{12} + (1-\alpha)\, v_2^T + \alpha \hat{\sigma}_{k+1} w_2^T\right]$;

The following comments are in order.

1. This algorithm has all advantages of the power method, that is, it is simple to implement and requires minimal storage. Unlike Krylov subspace methods, this algorithm is insensitive to changes in the matrix and exhibits predictable convergence behavior [43].

2. Each iteration of the Power method applied to $G^{(1)}$ involves a sparse matrix vector multiply with the matrix H_{11} and several vector operations.

3. The dangling nodes are not included in the Power method computation.

4. The convergence rate applied to G is α [68]. This algorithm has the same convergence rate, because $G^{(1)}$ has the same nonzero eigenvalues as G. However, algorithm 7 is much faster because it operates on a smaller matrix whose dimension does not depend on the number of dangling nodes.

5. In the final step of the algorithm, π is recovered via a single sparse matrix vector multiply with the matrix H_{12} and several vector operations.

6. Algorithm 4 can be extended to the situation when the Google matrix has several different dangling node vectors [67].

7. The Power method in Algorithm 7 corresponds to stage 1 of the algorithm in [91].

8. The methods [86, 91] are special cases of Algorithm 7. In those methods the personalization vector can be different from the dangling node vector. The interest of using a different personalization vector v, for instance, one that contains zero elements, may be to diminish the harm done by link spamming. Using $v_i = 0$ it allows to diminish the PageRank of a spam page i.

Algorithm 7 and Theorem 3.3.1 show that the PageRank of the nondangling nodes, π_1, are computed separately from the PageRank of the dangling nodes, π_2, and that π_2 depends directly on π_1.

The next theorem shows exactly that.

Theorem 3.3.2. *With the previous notation,*

$$\pi_1^T = \left[(1-\alpha)\, v_1^T + \rho w_1^T\right] (I - \alpha H_{11})^{-1},$$

$$\pi_2^T = \alpha \pi_1^T H_{12} + (1-\alpha)\, v_2^T + \alpha\,(1 - \|\pi_1\|)\, w_2^T,$$

where

$$\rho = \alpha \frac{1 - (1-\alpha)\, v_1^T (I - \alpha H_{11})^{-1} e}{1 + \alpha w_1^T (I - \alpha H_{11})^{-1} e} \geq 0.$$

Proof. See [67] □

With the theorem 3.3.2 we can see how the PageRank of the dangling and nondangling nodes influence each other and how the dangling node vector affects the PageRank.

Because the PageRank of the nondangling nodes, π_1, are computed separately, without knowledge of π_2, it does not depend on the PageRank of the dangling nodes, π_2.

Vector π_1 does not depend on the personalization vector for the dangling nodes (elements of v_2), on the connectivity among dangling nodes (elements of w_2) or links from nondangling nodes to dangling nodes (elements of H_{12}). Instead, the dependence is on the norms, that is, $\|w_2\| = 1 - \|w_1\|$, $\|v_2\| = 1 - \|v_1\|$ and $H_{12}e = e - H_{11}e$.

The PageRank π_1 of the nondangling nodes is obtained from v_1 and w_1 distributed through the links H_{11}.

On the other hand, the PageRank π_2 of the dangling nodes is obtained from v_2, w_2 and the PageRank π_1 of the nondangling nodes filtered through the connecting links H_{12}.

The amount of PageRank that flows from nondangling to dangling nodes is determined by the links H_{12}.

In conclusion, the PageRank of the dangling nodes depends strongly on that of the nondangling nodes but not vice-versa.

Figure 3.3 illustrates these conclusions.

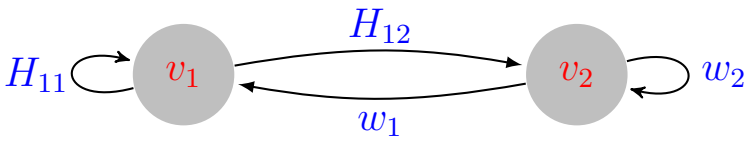


Figure 3.3: Sources of PageRank: nondangling nodes (left circle) receive their PageRank from v_1 and w_1, distributed through the links H_{11}. The PageRank of the dangling nodes (right circle) comes from v_2 and w_2 and the PageRank of the nondangling nodes through the links H_{12}.

3.4 The Lumping2 Method

The nodes are usually classified into two classes, nondangling nodes and dangling nodes. In [67] it was shown that the dangling nodes can be lumped into a single node and the PageRank of the nondangling nodes can be computed separately from that of the dangling nodes, reducing the number of operations necessary to compute the PageRank.

Lin, Shi and Wei divided the nondangling nodes into two classes. In [93] they show that one of the two classes of nondangling nodes can be also lumped into a single node and the PageRank of the other class of nondangling nodes can be computed separately. They proved that the small matrix obtained by lumping the dangling nodes can be further reduced by lumping a class of nondangling nodes and the further reduced matrix is stochastic with the same nonzero eigenvalues as the Google matrix G. Moreover, the full PageRank vector π can be easily recovered from the stationary distribution of the further reduced matrix.

This alternative more complex approach considers a refined division of the nondangling nodes (ND) in *weakly nondangling nodes* (WND) and *strongly nondangling nodes* (SND). The *weakly nondangling nodes* are pages that are not dangling but point to only dangling nodes (D). Pages with links to pages that are not dangling nodes are called *strongly nondangling nodes*. Thus, a node is either dangling, weakly nondangling, or strongly nondangling.

Let k_1 be the number of strongly nondangling nodes and k_2 the number of weakly nondangling nodes. Then, $k = k_1 + k_2$.

The rows and columns of $[H_{11} \quad H_{12}]$ are permuted so that the rows corresponding to weakly nondangling nodes are at the bottom of $[H_{11} \quad H_{12}]$, then this division leads to a matrix H of the following form:

$$H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} H_{11}^{11} & H_{11}^{12} & H_{12}^{1} \\ 0 & 0 & H_{12}^{2} \\ 0 & 0 & 0 \end{bmatrix}$$

where the $k_1 \times k_1$ matrix H_{11}^{11} represents the links among SND, the $k_1 \times k_2$ matrix H_{11}^{12} represents the links from SND to WND, the $k_1 \times (n-k)$ matrix H_{12}^{1} represents the links from SND to D, and the $k_2 \times (n-k)$ matrix H_{12}^{2} represents the links from WND to D, see Figure 3.4. Note that $H_{12}^{2} e = e$, which will be used in the following analysis of the PageRank.

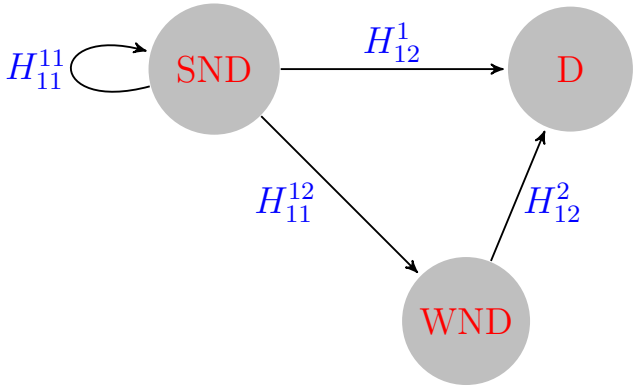


Figure 3.4: A simple model of the link structure of the Web divided in SND, WND and D nodes. SND represents the set of strongly nondangling nodes, WND represents the set of weakly nondangling nodes and D represents the set of dangling nodes. The submatrix H_{11}^{11} represents the links among strongly nondangling nodes, H_{11}^{12} represents the links from strongly nondangling nodes to weakly nondangling nodes, H_{12}^{1} represents the links from strongly nondangling nodes to dangling nodes and H_{12}^{2} represents the links from weakly nondangling nodes to dangling nodes.

Partitioning $w_1^T = \left[w_1^{(1)^T} \;,\; w_1^{(2)^T} \right]$ and $v_1^T = \left[v_1^{(1)^T} \;,\; v_1^{(2)^T} \right]$ with $w_1^{(1)}$, $v_1^{(1)}$ being $k_1 \times 1$ and $w_1^{(2)}$, $v_1^{(2)}$ being $k_2 \times 1$, the Google matrix has the following 3×3 block structure:

$$G = \begin{bmatrix} G_{11}^{11} & G_{11}^{12} & G_{12}^{1} \\ (1-\alpha)\, e v_1^{(1)^T} & (1-\alpha)\, e v_1^{(2)^T} & G_{12}^{2} \\ e u_1^{(1)^T} & e u_1^{(2)^T} & e u_2^T \end{bmatrix}$$

where

$$\begin{aligned} G_{11}^{11} &= \alpha H_{11}^{11} + (1-\alpha)\, e v_1^{(1)^T}, \\ G_{11}^{12} &= \alpha H_{11}^{12} + (1-\alpha)\, e v_1^{(2)^T}, \\ G_{12}^{1} &= \alpha H_{12}^{1} + (1-\alpha)\, e v_2^T, \\ G_{12}^{2} &= \alpha H_{12}^{2} + (1-\alpha)\, e v_2^T, \end{aligned}$$

$$u = \alpha\omega + (1-\alpha)\, v = \begin{bmatrix} u_1^{(1)} \\ u_1^{(2)} \\ u_2 \end{bmatrix} = \begin{bmatrix} \alpha w_1^{(1)} + (1-\alpha)\, v_1^{(1)} \\ \alpha w_1^{(2)} + (1-\alpha)\, v_1^{(2)} \\ \alpha w_2 + (1-\alpha)\, v_2 \end{bmatrix}.$$

At this moment, $u_1^T = \left[u_1^{(1)^T} \;,\; u_1^{(2)^T} \right]$, and the matrices G_{11} and G_{12} are given by

$$G_{11} = \begin{bmatrix} G_{11}^{11} & G_{11}^{12} \\ (1-\alpha)\, e v_1^{(1)^T} & (1-\alpha)\, e v_1^{(2)^T} \end{bmatrix}, \qquad G_{12} = \begin{bmatrix} G_{12}^{1} \\ G_{12}^{2} \end{bmatrix}.$$

Theorem 3.4.1. *Using the notation above and defining $G^{(2)}$ by*

$$G^{(2)} = \begin{bmatrix} G_{11}^{11} & G_{12}^{1} e & G_{11}^{12} e \\ u_1^{(1)^T} & u_2^T e & u_1^{(2)^T} e \\ (1-\alpha)\, v_1^{(1)^T} & \alpha + (1-\alpha)\, v_2^T e & (1-\alpha)\, v_1^{(2)^T} e \end{bmatrix},$$

then $G^{(2)}$ is a stochastic matrix of order $k_1 + 2$ with the same nonzero eigenvalues as the full Google matrix G.

Furthermore, let $\hat{\sigma}^T = \hat{\sigma}^T G^{(2)}$, $\hat{\sigma} \geq 0$, $\|\hat{\sigma}\| = 1$, partitioned by $\hat{\sigma}^T = \left[\hat{\sigma}_{1:k_1}^T \;\; \hat{\sigma}_{k_1+1} \;\; \hat{\sigma}_{k_1+2} \right]$, where $\hat{\sigma}_{k_1+1}$ and $\hat{\sigma}_{k_1+2}$ are two scalars, then the PageRank vector π is given by

$$\pi^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{pmatrix} G_{12} \\ u_2^T \end{pmatrix} \end{bmatrix} \tag{3.5}$$

where the vector σ *is*

$$\sigma^T = \begin{bmatrix} \hat{\sigma}_{1:k_1}^T & \hat{\sigma}^T \begin{pmatrix} G_{11}^{12} \\ u_1^{(2)^T} \\ (1-\alpha)\, v_1^{(2)^T} \end{pmatrix} & \hat{\sigma}_{k_1+1} \end{bmatrix}. \tag{3.6}$$

Proof. See [102] based on [93] □

Theorem 3.4.1 shows that to compute the PageRank vector π, we can compute the stationary distribution $\hat{\sigma}$ of the stochastic matrix $G^{(2)}$ and then recover the PageRank vector π according to (3.5) and (3.6).

To compute the stationary distribution $\hat{\sigma}$ of the stochastic lumped matrix $G^{(2)}$ with order k_1+2 we can use a simple algorithm which applies the Power method. This algorithm and the algorithm that applies the Power method to the full matrix G have the same convergence rate. This is due to the fact that $G^{(2)}$ and G have the same nonzero eigenvalues. The new algorithm, Algorithm 8 can save a large amount of operations.

Algorithm 8 Lumping 2 method

Given H_{11}^{11}, H_{11}^{12}, H_{12}^{1}, H_{12}^{2}, $v_1^{(1)}$, $v_1^{(2)}$, v_2, $w_1^{(1)}$, $w_1^{(2)}$, w_2, α, tol;
Choose an initial vector $\hat{\sigma}^T := \begin{bmatrix} \hat{\sigma}_{1:k_1}^T & \hat{\sigma}_{k_1+1} & \hat{\sigma}_{k_1+2} \end{bmatrix}$ with $\hat{\sigma} \geq 0$, $\|\hat{\sigma}\| := 1$;
while no convergence **do**
  Compute $w_{1:k_1}^T := \alpha \hat{\sigma}_{1:k_1}^T H_{11}^{11} + (1-\alpha)\, v_1^{(1)^T} + \alpha \hat{\sigma}_{k_1+1} w_1^{(1)^T}$;
  Compute $w_{k_1+1} := \alpha \left[\hat{\sigma}_{1:k_1}^T H_{12}^{1} e + \hat{\sigma}_{k_1+1} w_2^T e + \hat{\sigma}_{k_1+2} \right] + (1-\alpha)\, v_2^T e$;
  Compute $w_{k_1+2} := \alpha \hat{\sigma}_{1:k_1}^T H_{11}^{12} e + (1-\alpha)\, v_1^{(2)^T} e + \alpha \hat{\sigma}_{k_1+1} w_1^{(2)^T} e$;
  Set $\hat{\sigma}^T := \begin{bmatrix} w_{1:k_1}^T & w_{k_1+1} & w_{k_1+2} \end{bmatrix}$;
end while
Compute $x^T := \alpha \hat{\sigma}_{1:k_1}^T H_{11}^{12} + (1-\alpha)\, v_1^{(2)^T} + \alpha \hat{\sigma}_{k_1+1} w_1^{(2)^T}$;
Compute $y^T := \alpha \left[\hat{\sigma}_{1:k_1}^T H_{12}^{1} + x^T H_{12}^{2} + \hat{\sigma}_{k_1+1} w_2^T \right] + (1-\alpha)\, v_2^T$;
Compute $\pi^T := \begin{bmatrix} \hat{\sigma}_{1:k_1}^T & x^T & y^T \end{bmatrix}$;

In Algorithm 8, the remainder of the PageRank vector can be computed directly by means of two matrix-vector multiplications.

The explicit expression for the PageRank vector π can be given as follows.

Theorem 3.4.2. *Partitioning* $\pi^T = \left[\pi_1^T, \pi_2^T\right] = \left[\pi_1^{(1)^T}, \pi_1^{(2)^T}, \pi_2^T\right]$ *with* $\pi_1^{(1)}$ *being* $k_1 \times 1$, $\pi_1^{(2)}$ *being* $k_2 \times 1$ *and* π_2 *being* $(n-k) \times 1$ *and using the notation given above, we have*

$$\pi_1^{(1)^T} = \left[(1-\alpha)\, v_1^{(1)^T} + \rho w_1^{(1)^T}\right] \left(I - \alpha H_{11}^{11}\right)^{-1},$$

$$\pi_1^{(2)^T} = \alpha \pi_1^{(1)^T} H_{11}^{12} + (1-\alpha)\, v_1^{(2)^T} + \rho w_1^{(2)^T},$$
$$\pi_2^T = \alpha \pi_1^T H_{12} + (1-\alpha)\, v_2^T + \alpha\,(1 - \|\pi_1\|)\, w_2^T,$$

where

$$\rho = \alpha \frac{1 - (1-\alpha)\left[v_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right)^{-1} e + \alpha v_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right)^{-1} H_{11}^{12} e + v_1^{(2)^T} e\right]}{1 + \alpha\left[w_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right)^{-1} e + \alpha w_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right)^{-1} H_{11}^{12} e + w_1^{(2)^T} e\right]}.$$

Proof. See [93] □

The Theorem 3.4.2 shows that the PageRank vector π can be obtained by computing $v_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right)^{-1}$ and $w_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right)^{-1}$. Therefore, we need to solve two systems of linear equations with the same coefficient matrix $\left(I - \alpha H_{11}^{11}\right)^{-1}$ and different right-hand sides.

In [86] has been considered computing the PageRank vector, via solving a system of linear equations with the similar coefficient matrix, in the case where the dangling vector equals the personalization vector ($w = v$). In contrast, Theorem 3.4.2 shows that it still holds for only $v_1^{(1)} = w_1^{(1)}$.

3.5 The New LumpingE Methods

In this section we intend to contribute for the acceleration of the PageRank computation by combining reordered techniques with extrapolation. We propose two novel algorithms (LumpingE methods) by considering standard Aitken extrapolation within the lumping method.

To illustrate these new methods, the next section (Section 3.6) provides some numerical experiments that compare the new LumpingE methods with standard Power method, Lumping 1 method and Lumping 2 method. The results obtained show the benefits from our proposal.

3.5.1 The LumpingE 1 method

As mentioned before, the dangling nodes can be lumped into a single node to obtain a stochastic reduced matrix with the same eigenvalues as the full matrix. The PageRank computation for the nondangling nodes is performed separately.

The H matrix can be partitioned into

$$\begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}$$

where the $(n-k)$ zero rows represent the dangling nodes (D), $H_{11} \geq 0,\ k \times k$ links among nondangling nodes (ND) and $H_{12} \geq 0,\ k \times (n-k),$ links from ND to D.

Theorem 3.5.1 (LumpingE 1 method). *Let* $X = \begin{bmatrix} I_k & 0 \\ 0 & L \end{bmatrix}$, *with* $L = I_{n-k} - \frac{1}{n-k}\hat{e}e^T$, $\hat{e} = e - e_1 = [0, 1, 1, \ldots, 1]^T$ *the first canonical basis vector, and* $I_n = [e_1 \cdots e_n]$ *the identity matrix of order* n.

Then, $XGX^{-1} = \begin{bmatrix} G^{(1)} & * \\ 0 & 0 \end{bmatrix}$, *where* $G^{(1)} = \begin{bmatrix} G_{11} & G_{12}e \\ u_1^T & u_2^T e \end{bmatrix}$.

The matrix $G^{(1)}$ *is stochastic of order* $k+1$ *with the same nonzero eigenvalues as* G.

Let $\sigma^T G^{(1)} = \sigma^T,\quad \sigma^T \geq 0,\quad \|\sigma\| = 1.$

Then the PageRank vector π^T *equals* $\pi^T = \left[\sigma_{1:k}^T \quad \sigma^T \begin{pmatrix} G_{12} \\ u_2^T \end{pmatrix}\right]$ *with partition*

$$\sigma^T = \begin{cases} \left[\sigma_{1:k}^T - \frac{\left(\Delta\sigma_{1:k}^T\right)^2}{\Delta^2\sigma_{1:k}^T} \quad \sigma_{k+1} - \frac{(\Delta\sigma_{k+1})^2}{\Delta^2\sigma_{k+1}}\right], & \text{for iterations } s, l\times s, \cdots, l \in \mathbb{N} \\ \left[\sigma_{1:k}^T \quad \sigma_{k+1}\right], & \text{for other iterations} \end{cases}$$

where σ_{k+1} is a scalar and s defines the step by which the extrapolation is applied.

Proof. 1) Similarity transformation:

From

$$X^{-1} = \begin{bmatrix} I_k & 0 \\ 0 & L^{-1} \end{bmatrix},$$

it follows that

$$XGX^{-1} = \begin{bmatrix} I_k & 0 \\ 0 & L \end{bmatrix} \begin{bmatrix} G_{11} & G_{12} \\ eu_1^T & eu_2^T \end{bmatrix} X^{-1} = \begin{bmatrix} I_k G_{11} & I_k G_{12} \\ Leu_1^T & Leu_2^T \end{bmatrix} X^{-1}$$

$$= \begin{bmatrix} G_{11} & G_{12} \\ Leu_1^T & Leu_2^T \end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & L^{-1} \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12}L^{-1} \\ Leu_1^T & Leu_2^T L^{-1} \end{bmatrix}.$$

.

Considering $L^{-1} = I_{n-k} + \hat{e}e^T$, $e^T e = n-k$, $\hat{e}e^T e = (n-k)\,\hat{e}$ and simplifying $I_{n-k} = I$, let

$Le = \left(I_{n-k} - \frac{1}{n-k}\hat{e}e^T\right) e = e - \frac{1}{n-k}\hat{e}e^T e = e - \frac{1}{n-k}\hat{e}\,(n-k) = e - \hat{e} = e_1$

so $Leu_1^T = e_1 u_1^T$

$$\begin{aligned} Leu_2^T L^{-1} &= \left(I_{n-k} - \tfrac{1}{n-k}\hat{e}e^T\right) eu_2^T L^{-1} = I_{n-k} eu_2^T L^{-1} - \tfrac{1}{n-k}\hat{e}e^T eu_2^T L^{-1} \\ &= eu_2^T L^{-1} - \tfrac{1}{n-k}\,(n-k)\,\hat{e}u_2^T L^{-1} = eu_2^T L^{-1} - \hat{e}u_2^T L^{-1} = (e-\hat{e})\,u_2^T L^{-1} \\ &= e_1 u_2^T L^{-1} = e_1 u_2^T \left(I_{n-k} + \hat{e}e^T\right) = e_1 u_2^T \left(I + \hat{e}e^T\right) \end{aligned}$$

and

$$G_{12}L^{-1} = G_{12}\left(I_{n-k} + \hat{e}e^T\right) = G_{12}\left(I + \hat{e}e^T\right).$$

Then

$$XGX^{-1} = \begin{bmatrix} G_{11} & G_{12}L^{-1} \\ Leu_1^T & Leu_2^T L^{-1} \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12}\left(I + \hat{e}e^T\right) \\ e_1 u_1^T & e_1 u_2^T \left(I + \hat{e}e^T\right) \end{bmatrix}$$

has the same eigenvalues as G.

We choose a different partitioning and separate the leading $k+1$ rows and columns, in order to reveal the eigenvalues, and observe that

$$\left(I_{n-k} + \hat{e}e^T\right) e_1 = \left(I + \hat{e}e^T\right) e_1 = e,$$

$$G_{12}\left(I + \hat{e}e^T\right) e_1 = G_{12}e,$$

$$u_2^T \left(I + \hat{e}e^T\right) e_1 = u_2^T e$$

and

$$XGX^{-1} = \begin{bmatrix} G_{11} & G_{12}\left(I + \hat{e}e^T\right) \\ e_1 u_1^T & e_1 u_2^T \left(I + \hat{e}e^T\right) \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12}e & \bullet \\ u_1^T & u_2^T e & \bullet\bullet \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} G^{(1)} & * \\ 0 & 0 \end{bmatrix}.$$

So, we obtain the block triangular matrix

$$XGX^{-1} = \begin{bmatrix} G^{(1)} & * \\ 0 & 0 \end{bmatrix} \quad \text{where} \quad G^{(1)} = \begin{bmatrix} G_{11} & G_{12}e \\ u_1^T & u_2^T e \end{bmatrix},$$

with at least $n-k-1$ zero eigenvalues.

2) Expression for PageRank:

Considering

$$* = G^{(2)} = \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \left(I + \hat{e}e^T\right) [e_2 \ \cdots \ e_{n-k}] \ \text{ then } \ XGX^{-1} = \begin{bmatrix} G^{(1)} & G^{(2)} \\ 0 & 0 \end{bmatrix}.$$

The vector $\begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix}$ is an eigenvector for XGX^{-1} associated with the eigenvalue $\lambda = 1$.

That is $\begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} XGX^{-1} = \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix}$ because

$$\begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} \begin{bmatrix} G^{(1)} & G^{(2)} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} \Leftrightarrow \begin{bmatrix} \sigma^T G^{(1)} & \sigma^T G^{(2)} \end{bmatrix} = \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix}$$

and $\sigma^T G^{(1)} = \sigma^T$ is true.

So, multiplying $\begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} XGX^{-1} = \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix}$ by X on the right

$$\begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} XGX^{-1}X = \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} X \ \Leftrightarrow \ \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} XG = \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} X$$

So $\hat{\pi}^T = \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} X$ is an eigenvector of G associated with $\lambda = 1$ and a multiple of the stationary distribution π of G.

The dominant eigenvalue 1 of G is distinct [38, 63, 76, 78] and $G^{(1)}$ has the same nonzero eigenvalues as G. So, the stationary distribution σ of $G^{(1)}$ is unique.

Using the original partitioning which separates the k leading elements we will express $\hat{\pi}$ in terms of quantities in the matrix G.

Let

$$\hat{\pi}^T = \begin{bmatrix} \sigma_{1:k}^T & \left(\sigma_{k+1} \ \ \sigma^T G^{(2)}\right) \end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & L \end{bmatrix}.$$

Multiplying out

$$\hat{\pi}^T = \begin{bmatrix} \sigma_{1:k}^T & \left(\sigma_{k+1} \ \ \sigma^T G^{(2)}\right) L \end{bmatrix}$$

shows that $\hat{\pi}^T$ has the same leading elements as $\sigma^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma_{k+1} \end{bmatrix}$.

Now, we must examine the trailing $n-k$ components of $\hat{\pi}^T$. To do this we partition the matrix $L = I_{n-k} - \frac{1}{n-k}\hat{e}e^T$ and distinguish the first row and column,

$$L = \begin{bmatrix} 1 & 0 \\ -\frac{1}{n-k}e & I - \frac{1}{n-k}ee^T \end{bmatrix}.$$

Then the eigenvector part associated with the dangling nodes is

$$z^T = \begin{bmatrix}\sigma_{k+1} & \sigma^T G^{(2)}\end{bmatrix} L = \left[\sigma_{k+1} - \frac{1}{n-k}\sigma^T G^{(2)} e \quad \sigma^T G^{(2)}\left(I - \frac{1}{n-k} e e^T\right)\right]$$

To remove the terms containing $G^{(2)}$ in z, we simplify

$$\left(I + \hat{e}e^T\right)[e_2 \ \cdots \ e_{n-k}]\, e = \left(I + \hat{e}e^T\right)\hat{e} = (n-k)\,\hat{e}.$$

Hence

$$G^{(2)} e = \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}\left(I + \hat{e}e^T\right)[e_2 \ \cdots \ e_{n-k}]\, e = \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}(n-k)\,\hat{e}$$

$$\Leftrightarrow G^{(2)} e = (n-k)\begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}\hat{e} \tag{3.7}$$

and

$$\frac{1}{n-k}\sigma^T G^{(2)} e = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}\hat{e} = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e - \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1 = \sigma_{k+1} - \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1,$$

where we used $\hat{e} = e - e_1$, and the fact that σ is the stationary distribution of $G^{(1)}$, so

$$\sigma_{k+1} = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e.$$

Therefore the leading element of z equals

$$z_1 = \sigma_{k+1} - \frac{1}{n-k}\sigma^T G^{(2)} e = \sigma_{k+1} - \left(\sigma_{k+1} - \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1\right) = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1.$$

For the remaining elements of z, we use (3.7) to simplify

$$G^{(2)}\left(I - \frac{1}{n-k} e e^T\right) = G^{(2)} - \frac{1}{n-k} G^{(2)} e e^T = G^{(2)} - \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}\hat{e}e^T.$$

Replacing

$$\left(I + \hat{e}e^T\right)[e_2 \ \cdots \ e_{n-k}] = [e_2 \ \cdots \ e_{n-k}] + \hat{e}e^T$$

in $G^{(2)}$ yields

$$\begin{aligned}
z_{2:n-k}^T &= \sigma^T G^{(2)} \left(I - \tfrac{1}{n-k} e e^T\right) \\
&= \sigma^T \left(G^{(2)} - \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \hat{e} e^T\right) \\
&= \sigma^T \left(\begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \left(I + \hat{e} e^T\right) [e_2 \ \cdots \ e_{n-k}] - \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \hat{e} e^T\right) \\
&= \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \left(\left(I + \hat{e} e^T\right) [e_2 \ \cdots \ e_{n-k}] - \hat{e} e^T\right) \\
&= \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \left([e_2 \ \cdots \ e_{n-k}] + \hat{e} e^T - \hat{e} e^T\right) \\
&= \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} [e_2 \ \cdots \ e_{n-k}] .
\end{aligned}$$

Therefore the eigenvector part associated with the dangling nodes is

$$\begin{aligned}
z^T &= \begin{bmatrix} z_1 & z_{2:n-k}^T \end{bmatrix} \\
&= \left[\sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1 \quad \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} [e_2 \ \cdots \ e_{n-k}]\right] \\
&= \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} [e_1 \ \ e_2 \ \cdots \ e_{n-k}] \\
&= \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}
\end{aligned}$$

and

$$\hat{\pi}^T = \left[\sigma_{1:k}^T \ \ \sigma^T \begin{pmatrix} G_{12} \\ u_2^T \end{pmatrix}\right].$$

Since π is unique, as discussed in section 2.1, we conclude that $\hat{\pi} = \pi$ if $\hat{\pi}^T e = 1$.

This follows, again, from the fact that σ is the stationary distribution of $G^{(1)}$ and

$$\sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e = \sigma_{k+1}.$$

3) Aitken Extrapolation applied every s-step:

For simplicity, we use $A = {G^{(1)}}^T$, so ${\sigma^{(l)}}^T = {\sigma^{(l-1)}}^T G^{(1)} \Leftrightarrow \sigma^{(l)} = A\sigma^{(l-1)}$ and consider that matrix A has $k+1$ distinct eigenvectors u_i: $Au_i = \lambda_i u_i$.

Assuming that the iterate $\sigma^{(l)}$ can be expressed as a linear combination of the first two eigenvectors:

$$\sigma^{(l)} = u_1 + \alpha_2 u_2, \tag{3.8}$$

we can calculate an estimate of the principal eigenvector u_1 in closed form using two subsequent iterates of the Power method $\sigma^{(l+1)}$, $\sigma^{(l+2)}$.

As seen in section 3.2, this approximation becomes increasingly accurate as l becomes larger.

Considering that (λ_1, u_1) and (λ_2, u_2) are eigenpairs of A, the first eigenvalue λ_1 of a Markov matrix is 1 and applying two iterations of the Power method, we obtain

$$\sigma^{(l+1)} = A\sigma^{(l)} = A\left[u_1 + \alpha_2 u_2\right] = Au_1 + \alpha_2 Au_2 = \lambda_1 u_1 + \alpha_2\lambda_2 u_2 = u_1 + \alpha_2\lambda_2 u_2 \tag{3.9}$$

and

$$\sigma^{(l+2)} = A\sigma^{(l+1)} = A\left[u_1 + \alpha_2\lambda_2 u_2\right] = Au_1 + \alpha_2\lambda_2 Au_2 = u_1 + \alpha_2\lambda_2^2 u_2 \tag{3.10}$$

Defining,

$$g_i = \left(\Delta\sigma_i^{(l)}\right)^2 = \left(\sigma_i^{(l+1)} - \sigma_i^{(l)}\right)^2 \tag{3.11}$$

and

$$h_i = \Delta^2\sigma_i^{(l)} = \sigma_i^{(l+2)} - 2\sigma_i^{(l+1)} + \sigma_i^{(l)} \tag{3.12}$$

where σ_i represents the component i of vector σ.

Using (3.8) and (3.9) in (3.11),

$$g_i = \left[(u_1)_i + \alpha_2\lambda_2 (u_2)_i - (u_1)_i - \alpha_2 (u_2)_i\right]^2 = \alpha_2^2 (\lambda_2 - 1)^2 (u_2)_i^2 .$$

Using (3.8), (3.9) and (3.10) in (3.12),

$$h_i = (u_1)_i + \alpha_2\lambda_2^2 (u_2)_i - 2\left[(u_1)_i + \alpha_2\lambda_2 (u_2)_i\right] + \alpha_2 (u_2)_i = \alpha_2 (\lambda_2 - 1)^2 (u_2)_i .$$

Considering

$$f_i = \frac{g_i}{h_i} = \frac{\alpha_2^2 (\lambda_2 - 1)^2 (u_2)_i^2}{\alpha_2 (\lambda_2 - 1)^2 (u_2)_i} = \alpha_2 (u_2)_i .$$

Therefore, $f = \frac{g}{h} = \alpha_2 u_2$.

Hence, from equation (3.8), we have a closed-form solution for u_1:

$$u_1 = \sigma^{(l)} - \alpha_2 u_2 \Leftrightarrow u_1 = \sigma^{(l)} - f \Leftrightarrow u_1 = \sigma^{(l)} - \frac{\left(\Delta\sigma^{(l)}\right)^2}{\Delta^2\sigma^{(l)}}.$$

This u_1 is an approximation of the principal eigenvector of A since it's based on the assumption that $\sigma^{(l)}$ can be expressed as a linear combination of u_1 and u_2.

So, to accelerate convergence of the power method, every s-step, we apply Aitken Extrapolation and instead of having $\sigma^T = \left[\sigma_{1:k}^T \;\; \sigma_{k+1}\right]$ we have $\sigma^T = \left[\sigma_{1:k}^T - \frac{\left(\Delta\sigma_{1:k}^T\right)^2}{\Delta^2\sigma_{1:k}^T} \;\; \sigma_{k+1} - \frac{(\Delta\sigma_{k+1})^2}{\Delta^2\sigma_{k+1}}\right]$. □

Next we present the pseudocode for the LumpingE 1 method.

Algorithm 9 LumpingE 1 method

Given H_{11}, H_{12}, v_1, v_2, w_1, w_2, α, tol;
Choose an initial vector $\hat{\sigma}^T := \begin{bmatrix}\hat{\sigma}_{1:k}^T & \hat{\sigma}_{k+1}\end{bmatrix}$ with $\hat{\sigma} \geq 0$, $\|\hat{\sigma}\| = 1$;
while no convergence **do**
  Set $\pi_{1:k}^{(n)^T} := \hat{\sigma}_{1:k}^T$; $\quad \pi_{k+1}^{(n)} := \hat{\sigma}_{k+1}$;
  Compute $\hat{\sigma}_{1:k}^T := \alpha\hat{\sigma}_{1:k}^T H_{11} + (1-\alpha)\, v_1^T + \alpha\hat{\sigma}_{k+1} w_1^T$;
  Compute $\hat{\sigma}_{k+1} := 1 - \hat{\sigma}_{1:k}^T e$;
  Set $\pi_{1:k}^{(n+1)^T} := \hat{\sigma}_{1:k}^T$; $\quad \pi_{k+1}^{(n+1)} := \hat{\sigma}_{k+1}$;
  periodically apply Aitken extrapolation
    Compute $\pi_{1:k}^{(n+2)^T} := \alpha\pi_{1:k}^{(n+1)^T} H_{11} + (1-\alpha)\, v_1^T + \alpha\pi_{k+1}^{(n+1)} w_1^T$;
    Compute $\pi_{k+1}^{(n+2)} := 1 - \pi_{1:k}^{(n+2)^T} e$;
    Compute (pointwise) $g := \left(\pi^{(n+1)^T} - \pi^{(n)^T}\right)^2$;
    Compute $h := \pi^{(n+2)^T} - 2 \cdot \pi^{(n+1)^T} + \pi^{(n)^T}$;
    Compute (pointwise) $\pi^{(n+2)^T} := \pi^{(n)^T} - g/h$;
    Set $\hat{\sigma}_{1:k}^T := \pi_{1:k}^{(n+2)^T}$; $\quad \hat{\sigma}_{k+1} := \pi_{k+1}^{(n+2)}$;
end while
Compute $\hat{\pi}^T := \begin{bmatrix}\hat{\sigma}_{1:k}^T & \alpha\hat{\sigma}_{1:k}^T H_{12} + (1-\alpha)\, v_2^T + \alpha\hat{\sigma}_{k+1} w_2^T\end{bmatrix}$;

3.5.2 The LumpingE 2 Method

As mentioned before, an alternative more complex to this approach considers a refined division of the ND nodes in strongly nondangling nodes (SND), pages with links to pages that are not dangling nodes, and weakly nondangling nodes (WND), pages that are not dangling but that point to only dangling nodes.

This division leads to a matrix H of the form

$$\begin{bmatrix} H_{11}^{11} & H_{11}^{12} & H_{12}^{1} \\ 0 & 0 & H_{12}^{2} \\ 0 & 0 & 0 \end{bmatrix}$$

where H_{11}^{11}, $k_1 \times k_1$ links among SND, H_{11}^{12}, $k_1 \times k_2$, links from SND to WND, H_{12}^{1}, $k_1 \times (n-k)$, links from SND to D, and H_{12}^{2}, $k_2 \times (n-k)$, links from WND to D.

Theorem 3.5.2 (LumpingE 2 method)**.** *Using the notation above and defining $G^{(2)}$ by*

$$G^{(2)} = \begin{bmatrix} G_{11}^{11} & G_{12}^{1}e & G_{11}^{12}e \\ {u_1^{(1)}}^T & u_2^T e & {u_1^{(2)}}^T e \\ (1-\alpha)\,{v_1^{(1)}}^T & \alpha + (1-\alpha)\, v_2^T e & (1-\alpha)\,{v_1^{(2)}}^T e \end{bmatrix},$$

then $G^{(2)}$ is a stochastic matrix of order k_1+2 with the same nonzero eigenvalues as the full Google matrix G.

Let $\hat{\sigma}^T = \hat{\sigma}^T G^{(2)}$, $\hat{\sigma} \geq 0$, $\|\hat{\sigma}\| = 1$, partitioned by

$$\hat{\sigma}^T = \begin{cases} \left[\hat{\sigma}_{1:k_1}^T - \frac{\left(\Delta\hat{\sigma}_{1:k_1}^T\right)^2}{\Delta^2\hat{\sigma}_{1:k_1}^T} \quad \hat{\sigma}_{k_1+1} - \frac{\left(\Delta\hat{\sigma}_{k_1+1}\right)^2}{\Delta^2\hat{\sigma}_{k_1+1}} \quad \hat{\sigma}_{k_1+2} - \frac{\left(\Delta\hat{\sigma}_{k_1+2}\right)^2}{\Delta^2\hat{\sigma}_{k_1+2}}\right], & \text{for iterations} \\ & s, l\times s, \cdots, l \in \mathbb{N} \\ \left[\hat{\sigma}_{1:k_1}^T \quad \hat{\sigma}_{k_1+1} \quad \hat{\sigma}_{k_1+2}\right], & \text{for other iterations} \end{cases}$$

where $\hat{\sigma}_{k_1+1}$ and $\hat{\sigma}_{k_1+2}$ are two scalars and s defines the step by which the extrapolation is applied.

Then the PageRank vector π is given by

$$\pi^T = \left[\sigma_{1:k}^T \quad \sigma^T \begin{pmatrix} G_{12} \\ u_2^T \end{pmatrix}\right]$$

where

$$\sigma^T = \left[\hat{\sigma}_{1:k_1}^T \quad \hat{\sigma}^T \begin{pmatrix} G_{11}^{12} \\ {u_1^{(2)}}^T \\ (1-\alpha)\,{v_1^{(2)}}^T \end{pmatrix} \quad \hat{\sigma}_{k_1+1}\right].$$

Proof. Let

$$X^{-1} = \begin{bmatrix} I_{k_1} & 0 & 0 \\ 0 & I_{k_2} & 0 \\ 0 & 0 & L \end{bmatrix},$$

where $L = I_{n-k} - \frac{1}{n-k}\hat{e}e^T$ and $\hat{e} = e - e_1 = [0, 1, 1, \ldots, 1]^T$. It follows from theorem 3.5.1 that

$$XGX^{-1} = \begin{bmatrix} G^{(1)} & * \\ 0 & 0 \end{bmatrix},$$

where

$$\begin{aligned} G^{(1)} &= \begin{bmatrix} G_{11} & G_{12}e \\ u_1^T & u_2^T e \end{bmatrix} = \begin{bmatrix} G_{11}^{11} & G_{11}^{12} & G_{12}^{1}e \\ (1-\alpha)\, e v_1^{(1)^T} & (1-\alpha)\, e v_1^{(2)^T} & G_{12}^{2}e \\ u_1^{(1)^T} & u_1^{(2)^T} & u_2^T e \end{bmatrix} \\ &= \begin{bmatrix} G_{11}^{11} & G_{11}^{12} & G_{12}^{1}e \\ (1-\alpha)\, e v_1^{(1)^T} & (1-\alpha)\, e v_1^{(2)^T} & \left(\alpha + (1-\alpha)\, v_2^T e\right) e \\ u_1^{(1)^T} & u_1^{(2)^T} & u_2^T e \end{bmatrix}, \end{aligned}$$

where we used $H_{12}^2 e = e$.

The matrix $G^{(1)}$ is stochastic of order $k+1$ with the same nonzero eigenvalues as G.

Let

$$\sigma^T G^{(1)} = \sigma^T, \quad \sigma \geq 0, \quad \|\sigma\| = 1.$$

and partition $\sigma = \begin{bmatrix} \sigma_{1:k}^T & \sigma_{k+1} \end{bmatrix}$, where σ_{k+1} is a scalar.

As seen in theorem 3.5.1 that the PageRank vector π satisfies

$$\pi^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{pmatrix} G_{12} \\ u_2^T \end{pmatrix} \end{bmatrix}.$$

Defining the permutation matrix z by

$$Z = \begin{bmatrix} I_{k_1} & 0 & 0 \\ 0 & 0 & 1 \\ 0 & I_{k_2} & 0 \end{bmatrix}.$$

We have

$$\hat{G}^{(1)} = ZG^{(1)}Z^T = \begin{bmatrix} G_{11}^{11} & G_{12}^{1}e & G_{11}^{12} \\ u_1^{(1)^T} & u_2^T e & u_1^{(2)^T} \\ (1-\alpha)\, e v_1^{(1)^T} & \left(\alpha + (1-\alpha)\, v_2^T e\right) e & (1-\alpha)\, e v_1^{(2)^T} \end{bmatrix}.$$

Let

$$\hat{X} = \begin{bmatrix} I_{k_1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \hat{L} \end{bmatrix},$$

where $\hat{L} = I_{k_2} - \frac{1}{k_2}\hat{e}e^T$. From theorem 3.5.1, we obtain

$$\hat{X}\hat{G}^{(1)}\hat{X}^{-1} = \begin{bmatrix} G^{(2)} & * \\ 0 & 0 \end{bmatrix},$$

where

$$G^{(2)} = \begin{bmatrix} G_{11}^{11} & G_{12}^{1}e & G_{11}^{12}e \\ {u_1^{(1)}}^T & u_2^T e & {u_1^{(2)}}^T e \\ (1-\alpha)\,{v_1^{(1)}}^T & \alpha + (1-\alpha)\,v_2^T e & (1-\alpha)\,{v_1^{(2)}}^T e \end{bmatrix}.$$

The matrix $G^{(2)}$ is stochastic of order $k_1 + 2$ and has the same nonzero eigenvalues as $\hat{G}^{(1)}$.

So, $G^{(2)}$ has the same nonzero eigenvalues as G.

Let

$$\hat{\sigma}^T G^{(2)} = \hat{\sigma}^T, \quad \hat{\sigma} \geq 0, \quad \|\hat{\sigma}\| = 1,$$

and partition $\hat{\sigma}^T = \begin{bmatrix} \hat{\sigma}_{1:k_1+1}^T & \sigma_{k_1+2} \end{bmatrix}$ (for all iterations except for every s-step), where σ_{k_1+2} is a scalar.

Following the theorem 3.5.1, to accelerate convergence of power method, every s-step, Aitken Extrapolation is applied to vector $\hat{\sigma}^T$ and matrix $G^{(2)}$.

As seen in theorem 3.5.1, the stationary distribution vector $\hat{\pi}$ of $\hat{G}^{(1)}$ satisfies

$$\hat{\pi}^T = \begin{bmatrix} \hat{\sigma}_{1:k_1+1}^T & \hat{\sigma}^T \begin{pmatrix} G_{11}^{12} \\ {u_1^{(2)}}^T \\ (1-\alpha)\,{v_1^{(2)}}^T \end{pmatrix} \end{bmatrix}.$$

We have $\hat{\pi}^T = \hat{\pi}^T\hat{G}^{(1)} = \hat{\pi}^T Z G^{(1)} Z^T$, i.e., $\left(Z^T\hat{\pi}\right)^T = \left(Z^T\hat{\pi}\right)^T G^{(1)}$.

So, the stationary distribution σ of $G^{(1)}$ satisfies

$$\sigma^T = \hat{\pi}^T Z = \begin{bmatrix} \hat{\sigma}_{1:k_1}^T & \hat{\sigma}^T \begin{pmatrix} G_{11}^{12} \\ {u_1^{(2)}}^T \\ (1-\alpha)\,{v_1^{(2)}}^T \end{pmatrix} & \hat{\sigma}_{k_1+1} \end{bmatrix}.$$

This completes the proof of the theorem.

□

Next we present the pseudocode for the LumpingE 2 method.

Algorithm 10 LumpingE 2 method

Given H_{11}^{11}, H_{11}^{12}, H_{12}^{1}, H_{12}^{2}, $v_1^{(1)}$, $v_1^{(2)}$, v_2, $w_1^{(1)}$, $w_1^{(2)}$, w_2, α, tol;
Choose an initial vector $\hat{\sigma}^T := \begin{bmatrix}\hat{\sigma}_{1:k_1}^T & \hat{\sigma}_{k_1+1} & \hat{\sigma}_{k_1+2}\end{bmatrix}$ with $\hat{\sigma} \geq 0$, $\|\hat{\sigma}\| = 1$;
while no convergence **do**
  Set $\pi_{1:k_1}^{(n)^T} := \hat{\sigma}_{1:k_1}^T$; $\pi_{k_1+1}^{(n)} := \hat{\sigma}_{k_1+1}$; $\pi_{k_1+2}^{(n)} := \hat{\sigma}_{k_1+2}$;
  Compute $w_{1:k_1}^T := \alpha\hat{\sigma}_{1:k_1}^T H_{11}^{11} + (1-\alpha)\, v_1^{(1)^T} + \alpha\hat{\sigma}_{k_1+1} w_1^{(1)^T}$;
  Compute $w_{k_1+1} := \alpha\left[\hat{\sigma}_{1:k_1}^T H_{12}^{1} e + \hat{\sigma}_{k_1+1} w_2^T e + \hat{\sigma}_{k_1+2}\right] + (1-\alpha)\, v_2^T e$;
  Compute $w_{k_1+2} := \alpha\hat{\sigma}_{1:k_1}^T H_{11}^{12} e + (1-\alpha)\, v_1^{(2)^T} e + \alpha\hat{\sigma}_{k_1+1} w_1^{(2)^T} e$;
  Set $\pi_{1:k_1}^{(n+1)^T} := w_{1:k_1}^T$; $\pi_{k_1+1}^{(n+1)} := w_{k_1+1}$; $\pi_{k_1+2}^{(n+1)} := w_{k_1+2}$;
  periodically apply Aitken extrapolation
    Compute $\pi_{1:k_1}^{(n+2)^T} := \alpha\pi_{1:k_1}^{(n+1)^T} H_{11}^{11} + (1-\alpha)\, v_1^{(1)^T} + \alpha\pi_{k_1+1}^{(n+1)} w_1^{(1)^T}$;
    Compute $\pi_{k_1+1}^{(n+2)} := \alpha\left[\pi_{1:k_1}^{(n+1)^T} H_{12}^{1} e + \pi_{k_1+1}^{(n+1)} w_2^T e + \pi_{k_1+2}^{(n+1)}\right] + (1-\alpha)\, v_2^T e$;
    Compute $\pi_{k_1+2}^{(n+2)} := \alpha\pi_{1:k_1}^{(n+1)^T} H_{11}^{12} e + (1-\alpha)\, v_1^{(2)^T} e + \alpha\pi_{k_1+1}^{(n+1)} w_1^{(2)^T} e$;
    Compute (pointwise) $g := \left(\pi^{(n+1)^T} - \pi^{(n)^T}\right)^2$;
    Compute $h := \pi^{(n+2)^T} - 2\cdot\pi^{(n+1)^T} + \pi^{(n)^T}$;
    Compute (pointwise) $\pi^{(n+2)^T} := \pi^{(n)^T} - g/h$;
  Set $\hat{\sigma}_{1:k_1}^T := \pi_{1:k_1}^{(n+2)^T}$; $\hat{\sigma}_{k_1+1} := \pi_{k_1+1}^{(n+2)}$; $\hat{\sigma}_{k_1+2} := \pi_{k_1+2}^{(n+2)}$;
end while
Compute $x^T := \alpha\hat{\sigma}_{1:k_1}^T H_{11}^{12} + (1-\alpha)\, v_1^{(2)^T} + \alpha\hat{\sigma}_{k_1+1} w_1^{(2)^T}$;
Compute $y^T := \alpha\left[\hat{\sigma}_{1:k_1}^T H_{12}^{1} + x^T H_{12}^{2} + \hat{\sigma}_{k_1+1} w_2^T\right] + (1-\alpha)\, v_2^T$;
Compute $\pi^T := \begin{bmatrix}\hat{\sigma}_{1:k_1}^T & x^T & y^T\end{bmatrix}$;

3.6 Numerical Experiments

This section gives an indication of the computing time in typical uses (average of 10 runs). The examples have been run on an Intel Core i7-3770 CPU at 3.40 GHz with 4 cores and it was used `MATLAB` R2015a.

Examples 1 and 2 present small matrices (of 12 and 100 nodes, respectively) to illustrate all the methods involved. Example 4 also uses a build-in matrix to highlight the method's ability to rank well sets of pages with the same PageRank. The matrix of example 3 is available at the SuiteSparse Matrix Collection - a widely used set of sparse matrix benchmarks; it has been used to assess performance of web pages ranking algorithms.

3.6.1 Example 1 - Toy model

We begin this section by illustrating the mechanism of the new LumpingE methods with a toy model.

The toy model has twelve nodes. Five of them are dangling nodes (nodes 2, 4, 7, 8 and 11), two are weakly nondangling nodes (nodes 1 and 6) and the remaining five are strongly nondangling nodes. The model is represented by the directed graph in Figure 3.5.

The twelve nodes in the graph represent twelve web pages. A node labeled with D is a dangling node, WND is a weakly nondangling node and SND is a strongly nondangling node. The directed arcs represent the hyperlinks. Outlinks point out from nodes and inlinks point into nodes. For instance, the weakly nondangling node 6 points to the dangling nodes 7 and 8.

The hyperlink matrix H schematically presented by the associated graph in Figure 3.5 is

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \end{bmatrix}$$

The dangling nodes (pages without outlinks) correspond to zero rows of

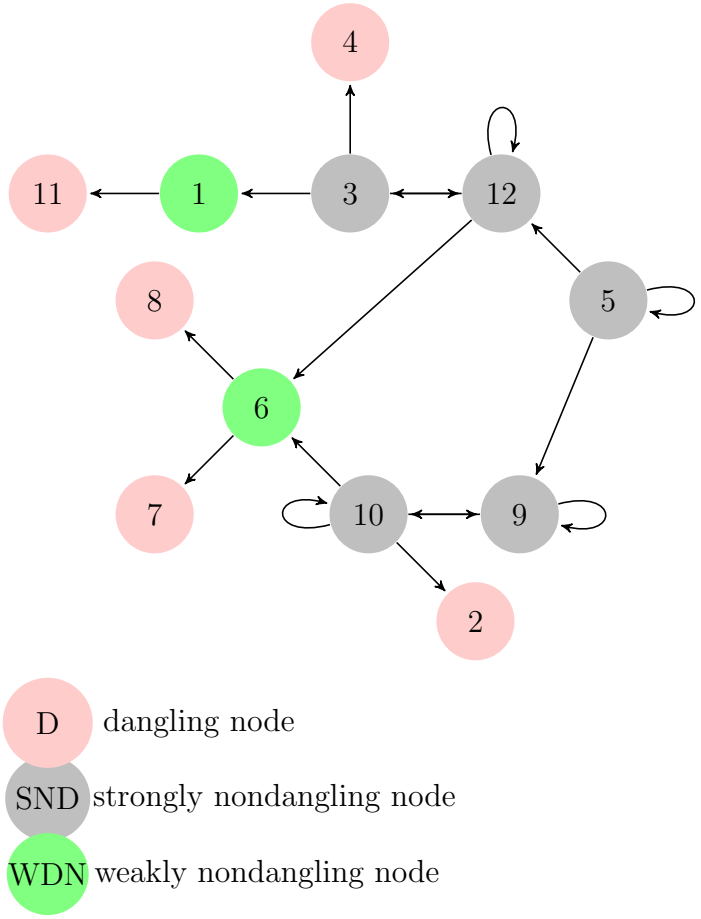


Figure 3.5: Graph of the toy model for the hyperlink matrix H without reordering

the matrix H. The nodes 1 and 6, weakly nondangling nodes, only link to dangling nodes. A strongly nondangling node can link to a dangling node but, it must have, at least, one link to a page that is not a dangling node (a WND or a SND node).

The rows and columns of H can be permuted to allow the application of the Lumping methods.

For the Lumping1 and the LumpingE 1 methods, the rows corresponding to the dangling nodes are at the bottom of the H matrix and the rows corresponding to the nondangling nodes are on top. The H matrix permuted

to apply the Lumping 1 scheme is presented next.

initial		final	1	2	3	4	5	6	7	8	9	10	11	12
1	$\longrightarrow$	1	0	0	0	0	0	0	0	0	0	0	0	1
3	$\longrightarrow$	2	$\frac{1}{3}$	0	0	0	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	0
5	$\longrightarrow$	3	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	0	0	0
6	$\longrightarrow$	4	0	0	0	0	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0
9	$\longrightarrow$	5	0	0	0	0	$\frac{1}{2}$	$\frac{1}{3}$	0	0	0	0	0	0
10	$\longrightarrow$	6	0	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
12	$\longrightarrow$	7	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	$\frac{1}{3}$	0	0	0	0	0
2	$\longrightarrow$	8	0	0	0	0	0	0	0	0	0	0	0	0
4	$\longrightarrow$	9	0	0	0	0	0	0	0	0	0	0	0	0
7	$\longrightarrow$	10	0	0	0	0	0	0	0	0	0	0	0	0
8	$\longrightarrow$	11	0	0	0	0	0	0	0	0	0	0	0	0
11	$\longrightarrow$	12	0	0	0	0	0	0	0	0	0	0	0	0

To apply the Lumping 2 scheme, the H matrix needs to be altered again and its associated graph can be seen in Figure 3.6: H graph matrix resulting after reordering the nondangling, weak and strong, and the dangling nodes. The new position of the nodes is indicated inside the circles.

For the Lumping2 and the LumpingE 2 methods, the first rows of the H matrix correspond to the strongly nondangling nodes, in the middle there are the weakly nondangling nodes and at the bottom the dangling nodes.

This new H matrix permuted to apply the Lumping 2 scheme is

initial		final	1	2	3	4	5	6	7	8	9	10	11	12
3	$\longrightarrow$	1	0	0	0	0	$\frac{1}{3}$	$\frac{1}{3}$	0	0	$\frac{1}{3}$	0	0	0
5	$\longrightarrow$	2	0	$\frac{1}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	0	0	0	0	0
9	$\longrightarrow$	3	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0	0	0	0	0
10	$\longrightarrow$	4	0	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
12	$\longrightarrow$	5	$\frac{1}{3}$	0	0	0	$\frac{1}{3}$	0	$\frac{1}{3}$	0	0	0	0	0
1	$\longrightarrow$	6	0	0	0	0	0	0	0	0	0	0	0	1
6	$\longrightarrow$	7	0	0	0	0	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0
2	$\longrightarrow$	8	0	0	0	0	0	0	0	0	0	0	0	0
4	$\longrightarrow$	9	0	0	0	0	0	0	0	0	0	0	0	0
7	$\longrightarrow$	10	0	0	0	0	0	0	0	0	0	0	0	0
8	$\longrightarrow$	11	0	0	0	0	0	0	0	0	0	0	0	0
11	$\longrightarrow$	12	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.1 reports on the number of iterations for the above example using the standard Power method, the Lumping1 method, the Lumping2 method

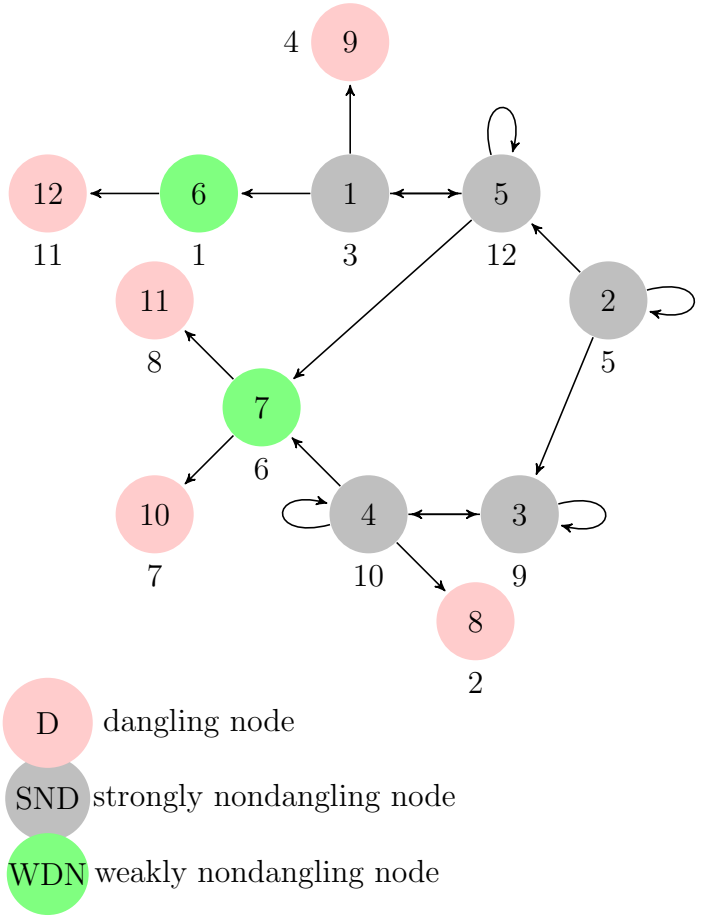


Figure 3.6: Graph for H matrix reordered with D and ND (SND and WND) nodes.

and the new LumpingE methods with Aitken extrapolation (LumpingE 1 and LumpingE 2).

The achieved PageRank is: 9,10,12,6,11,7,8,3,2,1,4,5.

Figure 3.7 presents the convergence history. The one in the upper part illustrates the Power method, the Lumping1 method and the LumpingE 1 method. The other in the lower part depicts the Power, Lumping2 and LumpingE 2 methods. Aitken extrapolation in LumpingE 1 and 2 was taken every 10 iterations.

For this small size example, results show a great reduction in the number of iterations when comparing the basic solution procedure with the Lumping methods and the extrapolated Lumping methods.

method	iterations
Power	30
*Lumping*1	28
*Lumping*2	27
LumpingE 1	21
LumpingE 2	21

Table 3.1: Toy model: Number of iterations for Power, Lumping and LumpingE methods.

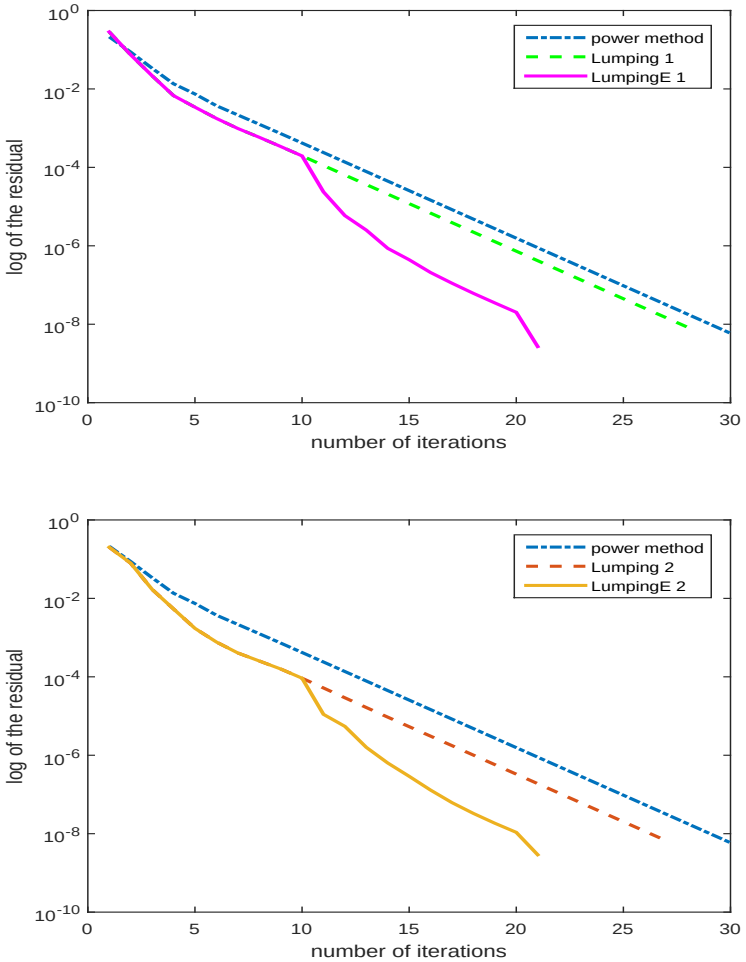
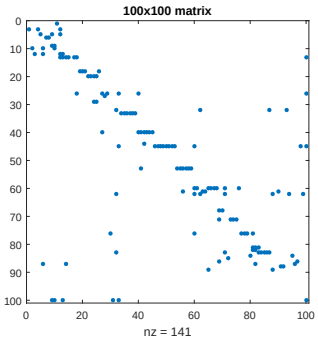


Figure 3.7: Toy model: Convergence history for Power, Lumping and LumpingE methods.

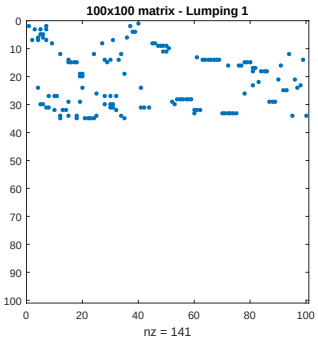
3.6.2 Example 2 - test1 matrix

In the second example we consider a web with 100 pages. In this case a Google matrix of dimension 100×100 (test1 matrix) with 65% of dangling nodes (D), 13% of weakly nondangling nodes (WND) and 22% of strongly nondangling nodes (SND) is considered.

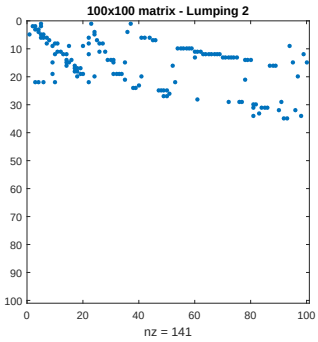
Figure 3.8 depicts the structures of the original as well as the reordered adjacency matrices. The 100×100 matrix has 141 nonzero elements that are represented as *dots*.



(a) Original matrix



(b) Lumping1 matrix



(c) Lumping2 matrix

Figure 3.8: 100×100 matrix: structures of the original and the two reordered matrices, where the *dots* represent nonzero elements and *white* stands for zero elements.

Table 3.2 reports the number of iterations and time obtained with the five methods: Power, Lumping1, LumpingE 1, Lumping2 and LumpingE 2 methods considering $\alpha = 0.85$. The elapsed time, reported in milliseconds (ms), is measured as an average of the time required to execute 10 times the solution procedure.

method	iterations	time (ms)
$Power$	44	2.012
$Lumping1$	39	1.150
$Lumping2$	41	2.523
$LumpingE$ 1	26	1.044
$LumpingE$ 2	25	1.615

Table 3.2: 100×100 matrix: Timings (ms) and number of iterations for Power, Lumping and LumpingE methods.

In terms of number of iterations the lumping approach allows for a convergence in a lower number of steps, representing the new proposal a reduction of about 40%. The elapsed time is reduced with LumpingE methods representing a reduction of about 2 times with respect to the classical Power method. The non-accelerated version, for this matrix, is not always favorable when compared with the Power method. Althought the Lumping1 method is able to achieve a reduction in time, the costs of nodes classification, reordering the matrix and recover the PageRank vector, in case of the Lumping2 method, supersedes the benefits of reducing the size of the matrix. Furthermore, the proposed extrapolated versions produce better results with respect to the non-extrapolated ones, both in number of iterations and in computing time.

The convergence history for these five methods is depicted in Figure 3.9 (Aitken extrapolation was taken every 10 iterations).

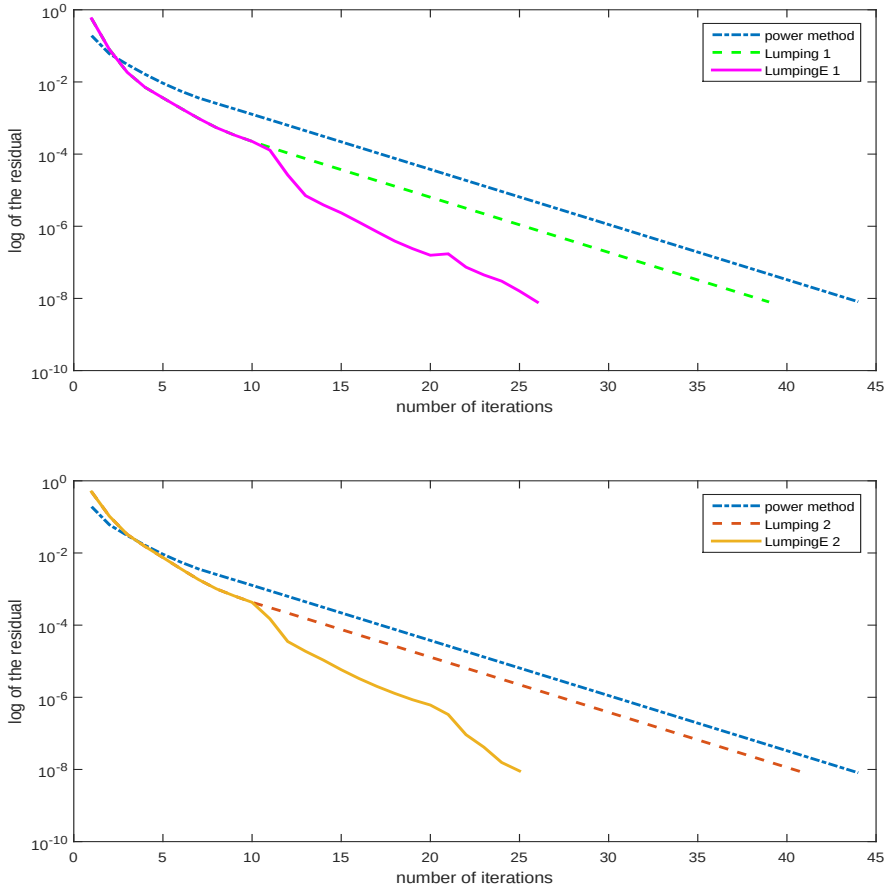


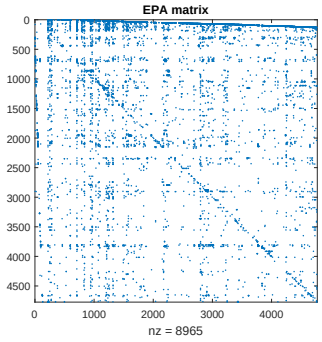
Figure 3.9: 100 × 100 matrix: Convergence history for Power, Lumping and LumpingE methods.

3.6.3 Example 3 - EPA matrix

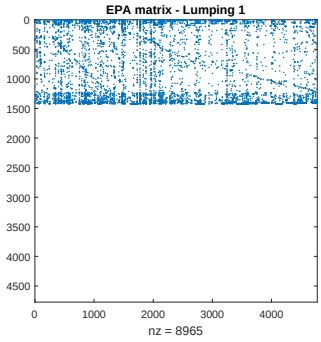
In this example we consider the EPA matrix – Kleinberg: Pajek network, pages linking to www.epa.gov. It is a 4772×4772 matrix with 8965 nonzeros, 4711 strongly connected components, 70.18% dangling nodes (D), 19.72% strongly nondangling nodes (SND) and 10.10% weakly nondangling nodes (WND). That is, a linear system of size 4772×4772 for the original MAAOR iteration, of size 3626×3626 for Lumping1-MAAOR, and 1363×1363 for Lumping2-MAAOR.

Figure 3.10 depicts the structures of the original EPA matrix (Figure 3.10(a)), the EPA matrix reordered considering two type of nodes (dangling and nondangling, Figure 3.10(b)) and considering three type of nodes (D,

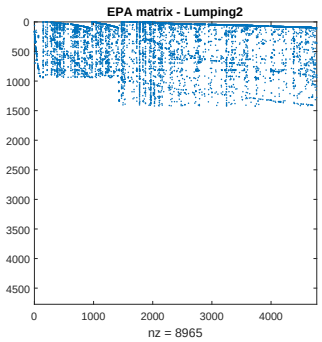
SND and WND, Figure 3.10(c)). The EPA matrix has 8965 nonzero elements that are represented as *dots*.



(a) Original matrix



(b) Lumping1 matrix



(c) Lumping2 matrix

Figure 3.10: EPA matrix: structures of the original and the two reordered matrices, where the *dots* represent nonzero elements and *white* stands for zero elements.

Table 3.3 reports the number of iterations and time obtained with the Power method, Lumping methods and LumpingE methods for $\alpha = 0.85$. The elapsed time, in seconds (s), is measured as an average of the time required to execute 10 times the solution procedure.

In terms of the number of iterations the partition on dangling and nondangling nodes has no advantage compared to the Power method, and, at least for the EPA matrix, extrapolation with lumping1 worsens the results. On the contrary, the partition of the nondangling nodes in strongly and weakly with extrapolation (LumpingE 2 method) causes a significant reduction in the number of iterations necessary to obtain convergence.

method	iterations	time (s)
$Power$	86	0.1344
$Lumping1$	94	0.1304
$Lumping2$	96	0.1396
$LumpingE$ 1	130	0.1508
$LumpingE$ 2	71	0.1211

Table 3.3: EPA matrix: Timings (s) and number of iterations for Power, Lumping and LumpingE methods.

The convergence history for the referred methods in case of the EPA matrix is depicted in Figure 3.11 (Aitken extrapolation was taken every 10 iterations).

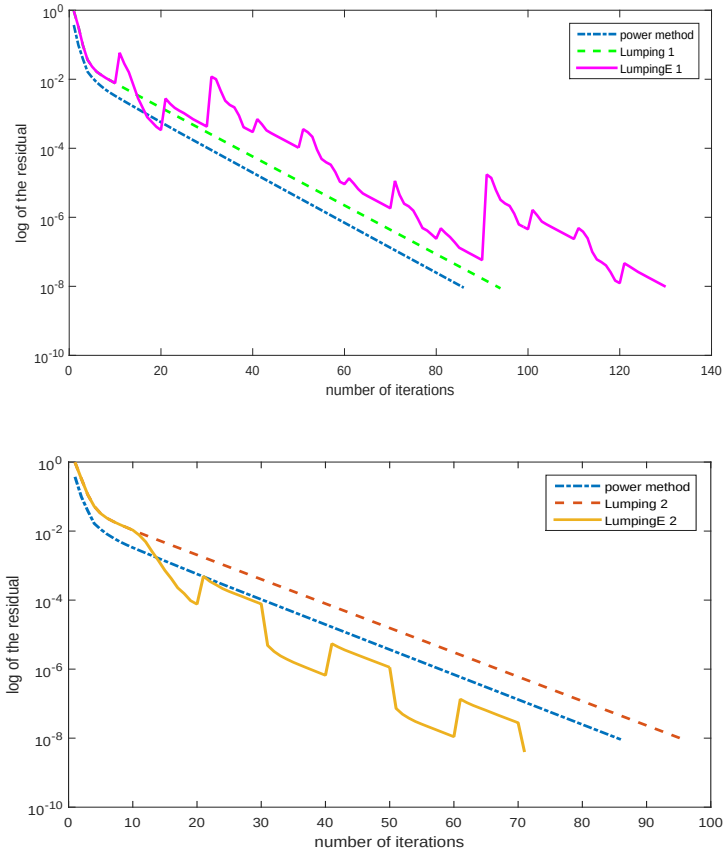


Figure 3.11: EPA matrix: Convergence history for Power, Lumping and LumpingE methods.

In order to understand the poor results obtained with LumpingE1, we investigated the eigenvalues of the original EPA matrix and also of the reduced matrices, resulting from the application of the lumpings, on which the Aitken extrapolation method was performed. As stated in Section 3.3 the reduced matrix $G^{(1)}$ used in the Lumping1 method has the same nonzero eigenvalues as the full Google matrix G. From Section 3.4, we conclude that the further reduced matrix $G^{(2)}$ used for Lumping2 has also the same nonzero eigenvalues as G.

Figure 3.12 depicts the eigenvalues of the original EPA matrix. They are the same as the ones of the two reduced matrices for Lumping1 and Lumping2.

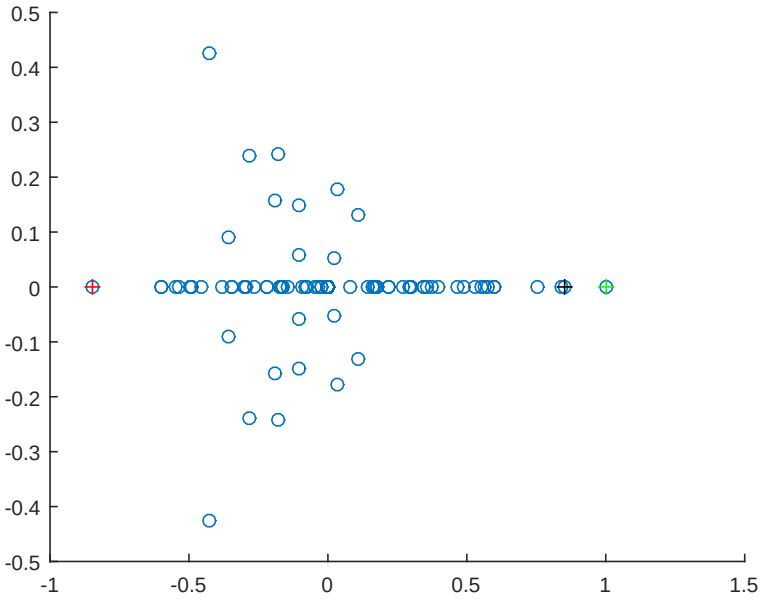


Figure 3.12: Eigenvalues of the EPA matrix, where the blue circles represent the eigenvalues. The dominant eigenvalue (λ_1) is represented with a green plus, the subdominant eigenvalue (λ_2) is represented with a red plus and λ_3 is a black plus.

The value of the first five eigenvalues of the EPA matrix is reported by Table 3.4.

The EPA matrix (matrix G) is a Google matrix ($G = \alpha S + (1-\alpha)E$ with $E = ev^T$ and $S = H + dw^T$), that is primitive (stochastic, irreducible and aperiodic). Then, G is a Markov matrix and, as such, for any starting vector, the Power method converges to a unique positive vector (π). G is a stochastic

Eigenvalues of EPA matrix
0.7531
0.8427
0.8500
-0.8500
1.0000

Table 3.4: First five eigenvalues of the EPA matrix.

matrix so $\lambda_1 = 1$ and $|\lambda_1| = 1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Also, it was proved that $|\lambda_2| \approx \alpha$ (see Section 2.2.1).

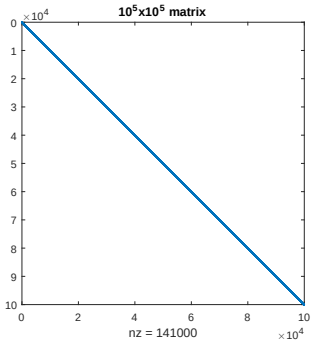
As stated in Section 3.2 Aitken extrapolation is able to accelerate the convergence of the Power method if $|\lambda_2| > |\lambda_3|$. However, when $|\lambda_2| = |\lambda_3|$ Aitken extrapolation performs poorly.

As we can observe in Figure 3.12 and Table 3.4 the dominant eigenvalue is $\lambda_1 = 1$ and the absolute value of second and the third eigenvalues are equal, $|\lambda_2| = |\lambda_3| = \alpha = 0.85$ which explains the behavior of the LumpingE1 method. Although, for the LumpingE2 method the extrapolation was able to achieved good results.

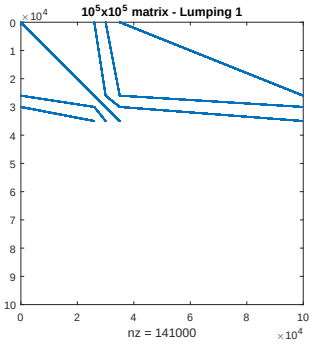
3.6.4 Example 4 - test2 matrix

In this section we use a test matrix build-in by the authors. This matrix is proposed to deliver large sets of pages with equal PageRank. The matrix is a $10^5 \times 10^5$ matrix with 141000 nonzeros, 65% dangling nodes, 22% strong nondangling nodes and 13% weakly nondangling nodes.

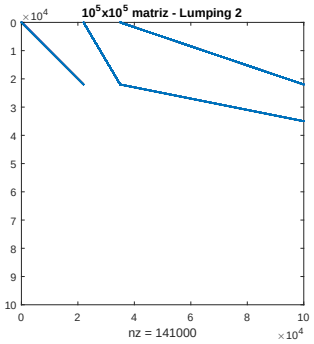
The spy of test2 matrix is presented in Figure 3.13 along with the two reordered matrices for Lumping1 and Lumping2. This matrix is the very sparse.



(a) Original matrix



(b) Lumping1 matrix



(c) Lumping2 matrix

Figure 3.13: test2 matrix: structures of the original and the two reordered matrices, where the *dots* represent nonzero elements and *white* stands for zero elements.

The number of iterations and time, in seconds (s), obtained with the Power method, Lumping methods and LumpingE methods for $\alpha = 0.85$ are reported in Table 3.5. Times reported at columns time1 and time2 are the average of 10 runs.

In Table 3.5 time2 refers to the time needed to compute the methods and recover the PageRank vector in case of the lumpings, while time1 also includes the time necessary to classify the nodes and reorder the matrices ($0.126s$ for Lumping1 and LumpingE 1, $0.088s$ for Lumping2 and LumpingE 2).

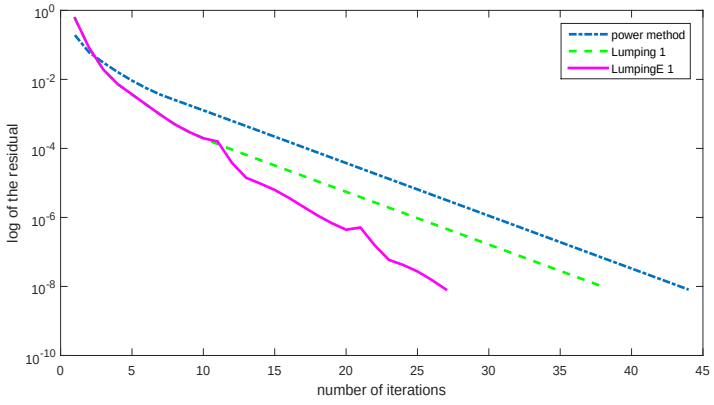
method	iterations	time1 (s)	time2 (s)
Power	44	0.175	0.175
*Lumping*1	38	0.213	0.087
*Lumping*2	42	0.152	0.064
LumpingE 1	27	0.172	0.046
LumpingE 2	25	0.131	0.043

Table 3.5: test2 matrix: Timings and number of iterations for Power, Lumping and LumpingE methods with $\alpha = 0.85$.

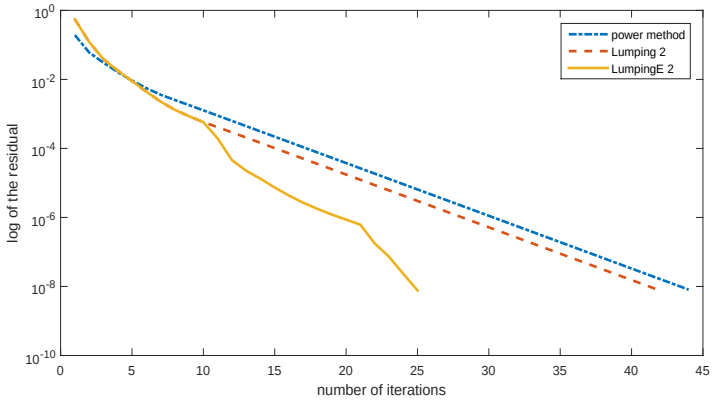
The convergence history is depicted in figures 3.14(a) and 3.14(b). For a clear understanding, one should mention that Aitken extrapolation in LumpingE 1 and 2 was taken every 10 iterations.

The number of iterations for the Lumping approaches is smaller than the original Power method. The reduction is particularly impressive for the new extrapolated versions. The reduction in the number of iterations with the LumpingE 2 method compared with the original Lumping2 is significant. This is relevant for real problems, usually of high dimension, since one might be interested in obtaining an approximate solution after a few number of iterations. A computation with just a few iterations can be already enough to provide useful information.

Generally, the elapsed time is reduced with the use of Lumping techniques even considering time1. The benefits are greater if the overall time is taken into account (time2). Once the matrix nodes are sorted and reordered, we may consider that it is not necessary to repeat these operations whenever the PageRank vector is calculated. Thus, it may make sense not to consider these calculations when comparing the methods. With this perspective, extrapolation presents good results when compared to the Power method. Indeed, a partition on dangling and nondangling nodes gives rise to a reduction of $2\times$ without extrapolation and a reduction of almost $4\times$ with extrapolation with respect to the time required by the classical Power method. The partition of the nondangling nodes in strongly and weakly reduces the computation time in approximately $3\times$ and $4\times$ considering, respectively, without and with extrapolation. Our combined versions, LumpingE, perform always better than



(a)



(b)

Figure 3.14: test2 matrix: Convergence history for Power, Lumping and LumpingE methods.

the original Lumping versions. It should be mentioned that, if we consider time2, the cost to prepare data is not compensated by the gains in aggregating the nodes in dangling and nondangling, at least for the test matrix used.

3.7 Conclusions

In this chapter we proposed new acceleration methods for the PageRank computations, LumpingE methods, based on a combination of Lumping

methods with Extrapolation. Numerical results illustrated the dynamics of the iterative process as well as provided some insight on the achieved success based on a significant reduction in the number of iterations and elapsed time required for convergence.

4

Acceleration of PageRank as a linear system

Contents

4.1 The MAAOR framework

The well-known Accelerated Overrelaxation (AOR) method for the solution of linear systems of algebraic equations has been around for about four decades and several variations of this method have been proposed [32, 61, 113, 133, 143].

We begin this section describing a very recent method, the Matrix Analogue of the AOR (MAAOR) iterative method, developed by Hadjidimos in [58]. It follows the description of the Generalized AOR (GAOR) method and the AOR method. The MAAOR method generalizes both the AOR and the GAOR methods.

In this chapter the MAAOR family of methods is explored for the first time in the context of PageRank computations. Several methods within the MAAOR family are compared.

Additionally, the Lumping methods that have been applied to the eigenproblem formulation can also be used in the linear system formulation. Therefore, we propose a novel approach combining the Lumping and MAAOR methods for the solution of the linear system.

Numerical experiments illustrating the MAAOR method and the MAAOR method combined with Lumping techniques applied to PageRank computations show the merits of our new proposal.

4.1.1 The Matrix analogue of the AOR method (MAAOR method)

Consider the system $Ax = b$ and the splitting of matrix A in $A = D-L-U$ (2.15), where matrix $A \in \mathbb{C}^{n\times n}$ (A is a nonsingular matrix with nonvanishing diagonal elements, $det(A) \neq 0$), vectors $b, x \in \mathbb{C}^n$, $x^{(0)} \in \mathbb{C}^n$ arbitrary, and where $D = diag(A)$ is the diagonal part of A, $-L$ is the strictly lower triangular part of A and $-U$ is the strictly upper triangular part of A.

To simplify the notation we set

$$\tilde{L} = D^{-1}L, \quad \tilde{U} = D^{-1}U, \quad \tilde{b} = D^{-1}b, \quad \tilde{A} = D^{-1}A = I - \tilde{L} - \tilde{U} \tag{4.1}$$

The MAAOR iterative method can be written as

$$x^{(k+1)} = H_{R,W}x^{(k)} + d_{R,W} \qquad k = 0, 1, 2, \ldots \tag{4.2}$$

with

$$H_{R,W} = \left(I - R\tilde{L}\right)^{-1}\left[(I - W) + (W - R)\,\tilde{L} + W\tilde{U}\right] \tag{4.3}$$

and

$$d_{R,W} = \left(I - R\tilde{L}\right)^{-1} W\tilde{b} \tag{4.4}$$

where $R \in \mathbb{R}^{n\times n}$ is any diagonal matrix and $W \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $det\,(diag(W)) \neq 0$.

The MAAOR method solves the linear system $\tilde{A}x = \tilde{b}$ and, in view of (4.1), it also solves the system $Ax = b$.

However, the iteration matrix $H_{R,W}$ of the MAAOR method does not possess the "extrapolation" property that is shared by both iteration matrices $H_{r,\omega}$ of AOR method and $H_{r\Omega,r\Omega}$ of the GAOR method.

If it did then either

$$H_{R,W} = I - R^{-1}W + R^{-1}WH_{R,R} \quad \text{or} \quad H_{R,W} = I - R^{-1}W + H_{R,R}R^{-1}W \tag{4.5}$$

would hold true.

Note that for $R = rW$ both equality of (4.5) are valid but for $R \neq rW$ neither of them holds [58].

By using the M-matrix theory, the regular splitting theory and the Perron-Frobenius theory for nonnegative matrices, it is possible to conclude that the elements of the two diagonal matrix-parameters W and R cannot be found to belong to intervals that are determined via the spectral radius of the modulus of the Jacobi iteration matrix of A except in the case of the AOR method and in special cases of the GAOR method.

Hadjidimos found a relation between the moduli of the two successive vectors of the MAAOR method and determined sufficient convergence conditions for the MAAOR method when A is a nonsingular H-matrix.

Also, if A is an H-matrix and $O \leq R \leq W \leq I$, $det(W) \neq 0$, the best MAAOR method is the one that corresponds to the Generalized Gauss-Seidel iterative method, that is, when $R = W = I$ [58].

Considering that $R = r\Omega_1$ (*acceleration matrix parameter*), $W = \omega\Omega_2$ (*overrelaxation matrix parameter*), r and ω are two scalars ($\omega \neq 0$), $\Omega_1, \Omega_2 \in \mathbb{R}^{n\times n}$ are diagonal matrices with $det\,(diag(\Omega_2)) \neq 0$, I is the identity matrix, O is the null matrix and choosing special parameters in the MAAOR method we obtain other well-known iterative methods [60], see Table 4.1.

The pseudocode for the MAAOR method is given in Algorithm 11.

r	ω	Ω_1	Ω_2	$R = r\Omega_1$	$W = \omega\Omega_2$	Method
0	1	I	I	O	I	Jacobi
1	1	I	I	I	I	Gauss-Seidel
ω	ω	I	I	ωI	ωI	SOR
0	ω	I	I	O	ωI	EJ or JOR (Extrapolated Jacobi or Jacobi Overrelaxation)
1	ω	I	I	I	ωI	EGS (Extrapolated Gauss-Seidel)
r	ω	I	I	rI	ωI	AOR
1	1	Ω_2	Ω_2	Ω_2	Ω_2	GSOR (Generalized SOR)
r	1	Ω_2	Ω_2	$r\Omega_2$	Ω_2	GAOR
1	1	Ω_1	Ω_2	Ω_1	Ω_2	MAAOR
$R = diag(r_1 I_{n_1}, r_2 I_{n_2}), W = diag(\omega_1 I_{n_1}, \omega_2 I_{n_2})$ for block two-cycle matrices with $r_1, r_2 \neq 0$ and $n_1 + n_2 = n$, where n_1, n_2 are the orders of the two diagonal blocks						MAOR (Modified AOR)
$R = W = diag(\omega_1 I_{n_1}, \omega_2 I_{n_2})$, for block two-cycle matrices with $n_1 + n_2 = n$, where n_1, n_2 are the orders of the two diagonal blocks						MSOR (Modified SOR)

Table 4.1: Iterative methods for specific values of the parameters R and W.

Algorithm 11 The MAAOR Method

Given matrix A, vector b and diagonal matrices R, W;
Choose an initial guess $x^{(0)}$;
Compute D, the diagonal of A;
Compute L, the strict lower triangular part of A;
Compute U, the strict upper triangular part of A;
Compute $\tilde{L} := D^{-1}L$;
Compute $\tilde{U} := D^{-1}U$;
Compute $\tilde{b} := D^{-1}b$;
Compute $e := W\tilde{b}$;
Compute $M := I + R\tilde{L}$;
Compute $N := (I - W) - (W - R)\tilde{L} - W\tilde{U}$;
Let $k := 0$;
while no convergence **do**
    Solve $Mx^{(k+1)} := Nx^{(k)} + e$;
    Update $k := k + 1$;
end while
Normalize $x^{(k)}$;

4.1.2 The Generalized accelerated overrelaxation method (GAOR method)

For $R = r\Omega$ (a scalar multiple of Ω), $W = \Omega \in \mathbb{R}^{n\times n}$ a diagonal matrix with $det\,(diag(\Omega)) \neq 0$ and r a scalar then the MAAOR method reduces to the GAOR method.

The GAOR method (Generalized AOR method) is a generalization of the AOR method (Accelerated overrelaxation method), i.e., it is a scalar-matrix-parameter analog of the scalar-parameter AOR method, with $R = r\Omega$ being the *acceleration* and $W = \Omega$ the *overrelaxation matrix-parameters*.

The GAOR iterative method in [69] is a matrix-scalar analogue of the AOR method in [54].

The first articles on GAOR are due to James [69] and Song [125].

In [125], Song considered the GAOR method presented by James and gave some convergence theorems for various kinds of matrices, namely when the matrix A is positive definite, an H-, L-, or M-matrix, or strictly or irreducibly diagonally dominant, in case of nonsingular systems.

As mentioned in Section 2.3 our matrix is an M-matrix and it is known that an M-matrix is also a H-matrix. So, the convergence theorems presented in [125] apply to our study.

Later, in [126], Song studied the convergence of the extrapolated iterative methods for solving singular linear systems.

Also, in [55], Hadjidimos presented some first results about the GAOR method when matrix A has certain properties and later, in [59], proposed new theoretical results concerning the convergence theory of the method.

Recently, in [60], the same author determined intervals of convergence for the various parameters involved in the GAOR method for the solution of the linear complementary problem (LCP).

In [28] Darvishi and Hessari studied the convergence of the GAOR method for diagonally dominant coefficient matrices and gave regions of convergence. These results were later improved by several authors: Tian et al. [129] that worked with strictly diagonally dominant coefficient matrices; Whang et al. [132] assumed that the matrices were strictly α diagonally dominant; Gao and Li [42] that used strictly doubly diagonally dominant matrices and Zhang et al. [148] that proposed three new types of preconditioners to improve the convergence rate of the GAOR method.

Also, Nasabzadeh and Toutounian [109] presented a new GAOR method, based on a block splitting of the coefficient matrix, that is well-defined even when some elements of the diagonal of A are zero. The method converges when A is an H-matrix.

Considering the assumptions in the previous section, the GAOR iterative method can be written as

$$x^{(k+1)} = H_{r\Omega,\Omega}x^{(k)} + d_{r\Omega,\Omega} \qquad k = 0, 1, 2, \ldots \tag{4.6}$$

$$H_{r\Omega,\Omega} = \left(I - r\Omega\tilde{L}\right)^{-1}\left[(I-\Omega) + (1-r)\,\Omega\tilde{L} + \Omega\tilde{U}\right] \tag{4.7}$$

$$d_{r\Omega,\Omega} = \left(I - r\Omega\tilde{L}\right)^{-1}\Omega\tilde{b} \tag{4.8}$$

where r is a real scalar and $\Omega \in \mathbb{R}^{n\times n}$ a diagonal matrix with $det\,(diag(\Omega)) \neq 0$.

For $r = 0$ the GAOR method is a Modified Extrapolated Jacobi iterative method with *extrapolation matrix-parameter* Ω [57]. That is, $d_{O,\Omega} = \Omega\tilde{b}$ and $H_{O,\Omega} = (I-\Omega) + \Omega\left(\tilde{L} + \tilde{U}\right)$.

For $r \neq 0$, from (4.7), and with some algebra, we can write

$$\begin{aligned}
& H_{r\Omega,\Omega} = \left(I - r\Omega\tilde{L}\right)^{-1}\left[(I-\Omega) + (1-r)\,\Omega\tilde{L} + \Omega\tilde{U}\right] \\
\Leftrightarrow\ & H_{r\Omega,\Omega} = \left(I - r\Omega\tilde{L}\right)^{-1}\left[\left(I - r\Omega\tilde{L}\right) - \Omega + \Omega\tilde{L} + \Omega\tilde{U}\right] \\
\Leftrightarrow\ & H_{r\Omega,\Omega} = \left(I - r\Omega\tilde{L}\right)^{-1}\left[\left(1 - \tfrac{1}{r} + \tfrac{1}{r}\right)\left(I - r\Omega\tilde{L}\right) - \Omega + \Omega\tilde{L} + \Omega\tilde{U}\right] \\
\Leftrightarrow\ & H_{r\Omega,\Omega} = \left(I - r\Omega\tilde{L}\right)^{-1}\left[\left(1 - \tfrac{1}{r}\right)\left(I - r\Omega\tilde{L}\right) + \tfrac{1}{r}\left(I - r\Omega\tilde{L}\right) - \Omega + \Omega\tilde{L} + \Omega\tilde{U}\right] \\
\Leftrightarrow\ & H_{r\Omega,\Omega} = \left(1 - \tfrac{1}{r}\right)I + \left(I - r\Omega\tilde{L}\right)^{-1}\left[\tfrac{1}{r}I - \Omega\tilde{L} - \Omega + \Omega\tilde{L} + \Omega\tilde{U}\right] \\
\Leftrightarrow\ & H_{r\Omega,\Omega} = \left(1 - \tfrac{1}{r}\right)I + \left(I - r\Omega\tilde{L}\right)^{-1}\left[\tfrac{1}{r}I - \tfrac{1}{r}r\Omega + \tfrac{1}{r}r\Omega\tilde{U}\right]
\end{aligned}$$

$$\Leftrightarrow\ H_{r\Omega,\Omega} = \left(1 - \tfrac{1}{r}\right)I + \tfrac{1}{r}\left(I - r\Omega\tilde{L}\right)^{-1}\left[(I - r\Omega) + r\Omega\tilde{U}\right] \tag{4.9}$$

Forming $H_{r\Omega,r\Omega}$ using (4.3) with acceleration matrix-parameter $R = r\Omega$ and overrelaxation matrix-parameter $W = r\Omega$ we obtain

$$\begin{aligned}
H_{r\Omega,r\Omega} &= \left(I - r\Omega\tilde{L}\right)^{-1}\left[(1 - r\Omega)\,I + (r\Omega - r\Omega)\,\tilde{L} + r\Omega\tilde{U}\right] \\
&= \left(I - r\Omega\tilde{L}\right)^{-1}\left[(I - r\Omega) + r\Omega\tilde{U}\right]
\end{aligned}$$

and (4.9) can be written as

$$H_{r\Omega,\Omega} = \left(1 - \frac{1}{r}\right)I + \frac{1}{r}H_{r\Omega,r\Omega} \tag{4.10}$$

So, for $r \neq 0$ the GAOR method is an Extrapolated Modified SOR iterative method with extrapolation scalar-parameter $\frac{1}{r}$ of a Modified SOR with *overrelaxation matrix-parameter* $r\Omega$.

From (4.7) the GAOR method is a scalar-matrix-parameter analog of the scalar-parameter AOR method [54], where $R = r\Omega$ is the *acceleration* and $W = \Omega$ the *overrelaxation matrix-parameters*, respectively.

From (4.10) the GAOR method is an *extrapolation*, with *extrapolation parameter* $\frac{1}{r}$ of a Modified SOR, with *overrelaxation matrix* $R = r\Omega$.

The pseudocode for the GAOR method is given in Algorithm 12.

Algorithm 12 The GAOR Method

Given matrix A, vector b, parameter r and diagonal matrices W, $R = rW$;
Choose an initial guess $x^{(0)}$;
Compute D, the diagonal of A;
Compute L, the strict lower triangular part of A;
Compute U, the strict upper triangular part of A;
Compute $\tilde{L} := D^{-1}L$;
Compute $\tilde{U} := D^{-1}U$;
Compute $\tilde{b} := D^{-1}b$;
Compute $e := W\tilde{b}$;
Compute $M := I + R\tilde{L}$;
Compute $N := (I - W) - (W - R)\tilde{L} - W\tilde{U}$;
Let $k := 0$;
while no convergence **do**
    Solve $Mx^{(k+1)} := Nx^{(k)} + e$;
    Update $k := k + 1$;
end while
Normalize $x^{(k)}$;

4.1.3 The Accelerated overrelaxation method (AOR method)

The Accelerated Overrelaxation Method (AOR method) was first purposed by Hadjidimos in [54] and it is a two-parameter generalization of the SOR method, such that when the two parameters involved are equal it coincides with the SOR method.

Also, the well-known methods of Jacobi, of Gauss-Seidel, of Simultaneous Overrelaxation are special cases of the AOR method.

Let the system $Ax = b$ and the splitting of matrix A in $A = D - L - U$ (2.15), be given satisfying all the assumptions made in Section 4.1.1 and $det\,(D) \neq 0$.

The AOR method is the result of splitting $A = M - N$ where

$$M = \omega^{-1}\,(D - rL) \quad \text{and} \quad N = \omega^{-1}\left[(1-\omega)\,D + (\omega - r)\,L + \omega U\right]. \tag{4.11}$$

The AOR method consists of the following scheme:

$$x^{(k+1)} = H_{r,\omega}x^{(k)} + d_{r,\omega} \qquad k = 0, 1, 2, \ldots \tag{4.12}$$

with the iteration matrix $H_{r,\omega}$ being given by

$$H_{r,\omega} = M^{-1}N \Leftrightarrow H_{r,\omega} = (D - rL)^{-1}\left[(1-\omega)\,D + (\omega - r)\,L + \omega U\right] \tag{4.13}$$

and

$$d_{r,\omega} = M^{-1}b \Leftrightarrow d_{r,\omega} = \omega\,(D - rL)^{-1}\,b, \tag{4.14}$$

where r and $\omega \in \mathbb{C}\backslash\{0\}$ are two constants, r is the *acceleration parameter* and ω is the *overrelaxation parameter* of the scheme.

With some algebra,

$$\begin{aligned}
& H_{r,\omega} = (D - rL)^{-1}\left[(1-\omega)\,D + (\omega - r)\,L + \omega U\right] \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1}\left[\tfrac{r(1-\omega)}{r}D + (\omega - r)\,L + \omega U\right] \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1}\left[\tfrac{r-\omega+\omega-r\omega}{r}D + \tfrac{\omega-r}{r}rL + \omega U\right] \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1}\left[\tfrac{r-\omega}{r}D + \tfrac{\omega(1-r)}{r}D - \tfrac{r-\omega}{r}rL + \tfrac{\omega}{r}rU\right] \\
\Leftrightarrow\ & H_{r,\omega} = (D - rL)^{-1}\left[\left(1 - \tfrac{\omega}{r}\right)(D - rL) + \tfrac{\omega}{r}\left((1-r)\,D + rU\right)\right] \\
\Leftrightarrow\ & H_{r,\omega} = \left(1 - \tfrac{\omega}{r}\right)I + \tfrac{\omega}{r}\,(D - rL)^{-1}\left[(1-r)\,D + rU\right]
\end{aligned} \tag{4.15}$$

and using (4.13) to form $H_{r,r}$:

$$H_{r,r} = (D - rL)^{-1}\left[(1-r)\,D + rU\right]$$

we conclude that the iteration matrix (4.13) can be written as:

$$H_{r,\omega} = \left(1 - \frac{\omega}{r}\right) I + \frac{\omega}{r} H_{r,r} \tag{4.16}$$

and the vector (4.14) is given by

$$d_{r,\omega} = \frac{\omega}{r} r \left(D - rL\right)^{-1} b. \tag{4.17}$$

As in (4.1), to simplify the notation we set

$$\tilde{L} = D^{-1}L,\ \tilde{U} = D^{-1}U,\ \tilde{b} = D^{-1}b,\ \tilde{A} = D^{-1}A.$$

Hence, the original system $Ax = b$ is equivalent to $D^{-1}Ax = D^{-1}b \Leftrightarrow \tilde{A}x = \tilde{b}$ and the splitting of matrix A in $A = D - L - U$ is equivalent to $D^{-1}A = D^{-1}D - D^{-1}L - D^{-1}U \Leftrightarrow \tilde{A} = I - \tilde{L} - \tilde{U}$.

Also, $A = M - N \Leftrightarrow D^{-1}A = D^{-1}M - D^{-1}N \Leftrightarrow \tilde{A} = \tilde{M} - \tilde{N}$ where $\tilde{M} = D^{-1}M$ and $\tilde{N} = D^{-1}N$.

Then, from (4.11), we have $\tilde{M} = \omega^{-1}\left(I - r\tilde{L}\right)$ and

$$\tilde{N} = \omega^{-1}\left[(1-\omega) I + (\omega - r)\tilde{L} + \omega\tilde{U}\right].$$

With this notation, the iteration matrix (4.16) will become

$$H_{r,\omega} = \tilde{M}^{-1}\tilde{N} \Leftrightarrow H_{r,\omega} = \left(I - r\tilde{L}\right)^{-1}\left[(1-\omega) I + (\omega - r)\tilde{L} + \omega\tilde{U}\right] \tag{4.18}$$

and the vector (4.17)

$$d_{r,\omega} = \tilde{M}^{-1}\tilde{b} \Leftrightarrow d_{r,\omega} = \omega\left(I - r\tilde{L}\right)^{-1}\tilde{b}. \tag{4.19}$$

Therefore, the AOR iterative method is given by

$$x^{(k+1)} = H_{r,\omega}x^{(k)} + d_{r,\omega} \qquad k = 0, 1, 2, \ldots$$

$$\Leftrightarrow \quad x^{(k+1)} = \left(I - r\tilde{L}\right)^{-1}\left[(1-\omega) I + (\omega - r)\tilde{L} + \omega\tilde{U}\right]x^{(k)} + \omega\left(I - r\tilde{L}\right)^{-1}\tilde{b} \tag{4.20}$$

It is possible to observe that for specific values of the parameters r and ω, the AOR method reduces to other well-known methods.

For $r = 0$ the AOR method is an Extrapolated Jacobi method with *extrapolation parameter* ω.

In this case, by (4.13) and (4.14), the iteration matrix is $H_{O,\omega} = D^{-1}\left[(1-\omega)\,D + \omega\,(L+U))\right]$ and the vector $d_{O,\omega} = \omega D^{-1}\tilde{b}$.

Also, by (4.18) and (4.19), $H_{O,\omega} = (1-\omega)\,I + \omega\left(\tilde{L}+\tilde{U}\right)$ and $d_{O,\omega} = \omega\tilde{b}$.

For $r \neq 0$ the AOR method is an Extrapolated SOR method with *extrapolation parameter* $\frac{\omega}{r}$ and *overrelaxation parameter* r (ESOR method [107]).

Considering the splitting $A = D-L-U \Leftrightarrow A = M-N$ and the iteration matrix $H_{r,\omega}$ given by (4.13) we can observe that, for different values of r and ω, the AOR method reduces to the methods referred in Table 4.2.

r	ω	$H_{r,\omega} = M^{-1}N$		Method
0	1	$M = D$	$N = L+U$	Jacobi
0	ω	$M = \omega^{-1}D$	$N = \omega^{-1}\left[(1-\omega)D + \omega(L+U)\right]$	EJ or JOR
1	1	$M = D-L$	$N = U$	Gauss-Seidel
1	ω	$M = \omega^{-1}(D-L)$	$N = \omega^{-1}\left[(1-\omega)D + (\omega-1)L + \omega U\right]$	EGS
ω	ω	$M = \omega^{-1}(D-\omega L)$	$N = \omega^{-1}\left[(1-\omega)D + \omega U\right]$	SOR
r	ω	$M = \omega^{-1}(D-rL)$	$N = \omega^{-1}\left[(1-\omega)D + (\omega-r)L + \omega U\right]$	AOR

Table 4.2: Iterative methods for specific values of the parameters r and ω considering $A = D-L-U$.

On the other hand, considering the splitting $\tilde{A} = I-\tilde{L}-\tilde{U} \Leftrightarrow \tilde{A} = \tilde{M}-\tilde{N}$ and the iteration matrix $H_{r,\omega}$ given by (4.18) the *AOR method* reduces to the methods referred in Table 4.3 [56].

r	ω	$H_{r,\omega} = \tilde{M}^{-1}\tilde{N}$		Method
0	1	$\tilde{M} = I$	$\tilde{N} = \tilde{L}+\tilde{U}$	Jacobi
0	ω	$\tilde{M} = \omega^{-1}I$	$\tilde{N} = \omega^{-1}\left[(1-\omega)I + \omega(\tilde{L}+\tilde{U})\right]$	EJ or JOR
1	1	$\tilde{M} = I-\tilde{L}$	$\tilde{N} = \tilde{U}$	Gauss-Seidel
1	ω	$\tilde{M} = \omega^{-1}(I-\tilde{L})$	$\tilde{N} = \omega^{-1}\left[(1-\omega)I + (\omega-1)\tilde{L} + \omega\tilde{U}\right]$	EGS
ω	ω	$\tilde{M} = \omega^{-1}(I-\omega\tilde{L})$	$\tilde{N} = \omega^{-1}\left[(1-\omega)I + \omega\tilde{U}\right]$	SOR
r	ω	$\tilde{M} = \omega^{-1}(I-r\tilde{L})$	$\tilde{N} = \omega^{-1}\left[(1-\omega)I + (\omega-r)\tilde{L} + \omega\tilde{U}\right]$	AOR

Table 4.3: Iterative methods for specific values of the parameters r and ω considering $\tilde{A} = I-\tilde{L}-\tilde{U}$.

The Jacobi Overrelaxation (JOR) method is also called Extrapolated Ja-

cobi (EJ) method or Simultaneous Overrelaxation method and the AOR method is called Extrapolated SOR (ESOR) method (for $r \neq 0$) [107]. Furthermore, for $r = 1$ the AOR method is called the Extrapolated Gauss-Seidel (EGS) method [126].

Next, we relate the AOR method with the GAOR method.

Proposition 4.1.1. *The GAOR method results from the AOR method considering $r = r\Omega$, $\omega = \Omega$ and Ω a diagonal matrix.*

Proof. Considering equations (4.18) and (4.19), then substituting $r = r\Omega$ and $\omega = \Omega = \omega I$ we obtain

$$H_{r\Omega,\Omega} = \left(I - r\Omega\tilde{L}\right)^{-1}\left[(I-\Omega) + (1-r)\,\Omega\tilde{L} + \Omega\tilde{U}\right] \quad \Leftrightarrow (4.7)$$

and

$$d_{r\Omega,\Omega} = \Omega\left(I - r\Omega\tilde{L}\right)^{-1}\tilde{b} = \left(I - r\Omega\tilde{L}\right)^{-1}\Omega\tilde{b} \quad \Leftrightarrow (4.8)$$

□

For more information about the AOR method, its properties and numerical results see [54, 58, 61].

The pseudocode for the AOR method is given in Algorithm 13.

Algorithm 13 The AOR Method

Given matrix A, vector b and parameters r, ω;
Choose an initial guess $x^{(0)}$;
Compute D, the diagonal of A;
Compute L, the strict lower triangular part of A;
Compute U, the strict upper triangular part of A;
Compute $e := \omega b$;
Compute $M := rL + D$;
Compute $N := (1-\omega)D - (\omega - r)L - \omega U$;
Let $k := 0$;
while no convergence **do**
    Solve $Mx^{(k+1)} := Nx^{(k)} + e$;
    Update $k := k + 1$;
end while
Normalize $x^{(k)}$;

4.2 Lumping on a Linear System

In Chapter 3 the Lumping method was applied to calculate the PageRank vector as an eigenvector problem, that is, Lumping with Power method. Two algorithms were presented in sections 3.3 and 3.4 and also in paper [102] based on the work developed by Ipsen and Selee [67] and Lin et al. [93].

Later Yu et al. [146] developed two algorithms analogous to the ones by Ipsen and Lin but they used a linear system formulation for the PageRank calculations, that is, lumping with Jacobi iterations.

Next we propose a novel approach where the Lumping method will also be used from a sparse linear system point of view but, in this case, a combination of Lumping methods with MAAOR method.

4.2.1 Lumping 1 with MAAOR method

In Section 3.3 the Lumping 1 method, developed by Ipsen and Selee [67], was used considering that the PageRank calculation is an eigenvector problem. Next we will combine the MAAOR method with the Lumping 1 method.

As seen before, the eigenvector formulation of the PageRank problem is $\pi^T G = \pi^T$ and $\pi^T e = 1$ which is equivalent to the sparse linear system formulation of the problem $\pi^T (I - \alpha H) = v^T$.

That is, the classical system $Ax = b \Leftrightarrow x^T A^T = b^T$ can be solved by the MAAOR method considering that $x^T = \pi^T$, $A^T = (I - \alpha H)$ and $b^T = v^T$.

We know how to exclude the dangling nodes (D) with their artificial links from the computations, by lumping all the dangling nodes into a single node to obtain a smaller matrix [146]. The PageRank of the nondangling nodes (ND) can be computed separately from that of the dangling nodes using the MAAOR method.

Let us permute the rows and columns of matrix H so that the rows corresponding to dangling nodes are at the bottom of the hyperlink matrix, i.e.,

$$P = XHX^T = \begin{array}{c} \\ ND \\ D \end{array} \begin{array}{c} \begin{array}{cc} ND & D \end{array} \\ \begin{bmatrix} H_{11} & H_{12} \\ O & O \end{bmatrix} \end{array} \tag{4.21}$$

where X is a permutation matrix and each row and column has exactly one 1 and all other entries are 0.

The H_{11} submatrix represents the links among nondangling nodes (ND), the H_{12} submatrix represents the links from nondangling nodes (ND) to

dangling nodes (D) and the zero rows of P are associated with the dangling nodes.

It follows from $\pi^T (I - \alpha H) = v^T$ and (4.21) that

$$\begin{array}{ll} & \pi^T (I - \alpha H) = v^T \\ \Leftrightarrow & \pi^T (I - \alpha H) X^T = v^T X^T \\ \Leftrightarrow & \pi^T X^T X (I - \alpha H) X^T = v^T X^T \\ \Leftrightarrow & \pi^T X^T \left(X I X^T - \alpha X H X^T \right) = v^T X^T \end{array}$$

and so,

$$\hat{\pi}^T (I - \alpha P) = \hat{v}^T \tag{4.22}$$

where $\hat{\pi}^T = \pi^T X^T$ and $\hat{v}^T = v^T X^T$.

Finally, multiplying $\hat{\pi}^T = \pi^T X^T$ by X on the right we obtain the PageRank vector

$$\pi^T = \hat{\pi}^T X. \tag{4.23}$$

Partition consistently with (4.21) that is $\hat{\pi} = \left[\hat{\pi}_1^T, \hat{\pi}_2^T\right]^T$ and $\hat{v} = \left[\hat{v}_1^T, \hat{v}_2^T\right]^T$, we have

$$\begin{array}{ll} & \hat{\pi}^T (I - \alpha P) = \hat{v}^T \\ \Leftrightarrow & \left[\hat{\pi}_1^T, \hat{\pi}_2^T\right] \left[\begin{bmatrix} I & O \\ O & I \end{bmatrix} - \alpha \begin{bmatrix} H_{11} & H_{12} \\ O & O \end{bmatrix} \right] = \left[\hat{v}_1^T, \hat{v}_2^T\right] \\ \Leftrightarrow & \left[\hat{\pi}_1^T (I - \alpha H_{11}), -\alpha \hat{\pi}_1^T H_{12} + \hat{\pi}_2^T I\right] = \left[\hat{v}_1^T, \hat{v}_2^T\right]. \end{array}$$

Then,

$$\hat{\pi}_1^T (I - \alpha H_{11}) = \hat{v}_1^T \Leftrightarrow \hat{\pi}_1^T = \hat{v}_1^T (I - \alpha H_{11})^{-1} \tag{4.24}$$

and

$$-\alpha \hat{\pi}_1^T H_{12} + \hat{\pi}_2^T I = \hat{v}_2^T \Leftrightarrow \hat{\pi}_2^T = \alpha \hat{\pi}_1^T H_{12} + \hat{v}_2^T. \tag{4.25}$$

Therefore, with the division of the nodes into nondangling nodes and dangling nodes, it is sufficient to apply the MAAOR algorithm on a smaller matrix H_{11} (of the nondangling nodes) to compute the PageRank vector.

A new algorithm that combines Lumping 1 with the MAAOR method, Lumping1-MAAOR method, is presented in Algorithm 14.

Algorithm 14 Lumping1-MAAOR Method

Reorder the hyperlink matrix and vector v to get (4.21);
Given H_{11}, H_{12}, $\hat{v}_1$, $\hat{v}_2$, α, tol;
Solve $\hat{\pi}_1^T (I - \alpha H_{11}) = \hat{v}_1^T$, with the MAAOR method;
Compute $\hat{\pi}_2^T = \alpha \hat{\pi}_1^T H_{12} + \hat{v}_2^T$;
Set $\hat{\pi} = \left[\hat{\pi}_1^T, \hat{\pi}_2^T\right]^T$;
Compute the PageRank vector $\pi^T = \hat{\pi}^T X$;
Normalize $\pi^T = \frac{\pi^T}{\|\pi^T\|_1}$;

4.2.2 Lumping 2 with MAAOR method

Next we combine the MAAOR method with the Lumping 2 method. As mentioned in Section 3.4 dividing the nondangling nodes (ND) in two types, weakly nondangling nodes (WND, nodes that are not dangling but point only to dangling nodes) and strongly nondangling nodes (SND), we obtain three types of nodes.

All the dangling nodes (D) are lumped into a single node and the same is done with the weakly nondangling nodes. The PageRank of the strongly nondangling nodes is computed separately [93].

For the Lumping 2 method the rows and columns of H are permuted, so that the rows corresponding to dangling nodes are at the bottom of the hyperlink matrix and the rows corresponding to strongly nondangling nodes are at the top of the matrix. So, we obtain a new matrix $P = XHX^T$,

$$P = XHX^T = \begin{array}{c} \\ SND \\ WND \\ D \end{array} \begin{array}{c} \begin{array}{ccc} SND & WND & D \end{array} \\ \begin{bmatrix} H_{11}^{11} & H_{11}^{12} & H_{12}^{1} \\ O & O & H_{12}^{2} \\ O & O & O \end{bmatrix} \end{array} \tag{4.26}$$

where submatrix H_{11}^{11} represents the links among SND, H_{11}^{12} the links from SND to WND, H_{12}^{1} the links from SND to D, and H_{12}^{2} the links from WND to D and X is a permutation matrix.

Partition consistently with (4.26) we have $\hat{\pi} = \left[\hat{\pi}_1^T, \hat{\pi}_2^T\right]^T = \left[\hat{\pi}_1^{(1)^T}, \hat{\pi}_1^{(2)^T}, \hat{\pi}_2^T\right]^T$ and $\hat{v} = \left[\hat{v}_1^T, \hat{v}_2^T\right]^T = \left[\hat{v}_1^{(1)^T}, \hat{v}_1^{(2)^T}, \hat{v}_2^T\right]^T$.

From (4.22) and (4.26) we obtain

$$\begin{aligned}
&\hat{\pi}^T (I - \alpha P) = \hat{v}^T \\
\Leftrightarrow\ &\left[\hat{\pi}_1^{(1)^T}, \hat{\pi}_1^{(2)^T}, \hat{\pi}_2^T\right] \left[\begin{bmatrix} I & O & O \\ O & I & O \\ O & O & I \end{bmatrix} - \alpha \begin{bmatrix} H_{11}^{11} & H_{11}^{12} & H_{12}^{1} \\ O & O & H_{12}^{2} \\ O & O & O \end{bmatrix}\right] = \left[\hat{v}_1^{(1)^T}, \hat{v}_1^{(2)^T}, \hat{v}_2^T\right] \\
\Leftrightarrow\ &\left[\hat{\pi}_1^{(1)^T}, \hat{\pi}_1^{(2)^T}, \hat{\pi}_2^T\right] \begin{bmatrix} I - \alpha H_{11}^{11} & -\alpha H_{11}^{12} & -\alpha H_{12}^{1} \\ O & I & -\alpha H_{12}^{2} \\ O & O & I \end{bmatrix} = \left[\hat{v}_1^{(1)^T}, \hat{v}_1^{(2)^T}, \hat{v}_2^T\right] \\
\Leftrightarrow\ &\left[\hat{\pi}_1^{(1)^T} (I - \alpha H_{11}^{11}), -\alpha \hat{\pi}_1^{(1)^T} H_{11}^{12} + \hat{\pi}_1^{(2)^T}, -\alpha \hat{\pi}_1^{(1)^T} H_{12}^{1} - \alpha \hat{\pi}_1^{(2)^T} H_{12}^{2} + \hat{\pi}_2^T\right] = \\
&= \left[\hat{v}_1^{(1)^T}, \hat{v}_1^{(2)^T}, \hat{v}_2^T\right].
\end{aligned}$$

Then,

$$\hat{\pi}_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right) = \hat{v}_1^{(1)^T} \Leftrightarrow \hat{\pi}_1^{(1)^T} = \hat{v}_1^{(1)^T} \left(I - \alpha H_{11}^{11}\right)^{-1}, \tag{4.27}$$

$$\hat{\pi}_1^{(2)^T} = \alpha \hat{\pi}_1^{(1)^T} H_{11}^{12} + \hat{v}_1^{(2)^T}, \tag{4.28}$$

$$\hat{\pi}_2^T = \alpha \hat{\pi}_1^{(1)^T} H_{12}^{1} + \alpha \hat{\pi}_1^{(2)^T} H_{12}^{2} + \hat{v}_2^T. \tag{4.29}$$

With this in mind we will present the following Lumping algorithm, Lumping2-MAAOR method (Algorithm 15), with respect to three type of nodes for computing PageRank. The MAAOR method will be applied on a even smaller matrix H_{11}^{11} (of the strongly nondangling nodes) to compute the first part, $\pi_1^{(1)^T}$, of the PageRank vector.

In Algorithm 15, the remainder of the PageRank vector can be computed directly by means of matrix-vector multiplications.

Algorithm 15 Lumping2-MAAOR Method

Reorder the hyperlink matrix and vector v to get (4.26);
Given H_{11}^{11}, H_{11}^{12}, H_{12}^{1}, H_{12}^{2}, $\hat{v}_1^{(1)}$, $\hat{v}_1^{(2)}$, $\hat{v}_2$, α, tol;
Solve $\hat{\pi}_1^{(1)^T} (I - \alpha H_{11}^{11}) = \hat{v}_1^{(1)^T}$, with the MAAOR method;
Compute $\hat{\pi}_1^{(2)^T} = \alpha \hat{\pi}_1^{(1)^T} H_{11}^{12} + \hat{v}_1^{(2)^T}$ and $\hat{\pi}_2^T = \alpha \hat{\pi}_1^{(1)^T} H_{12}^{1} + \alpha \hat{\pi}_1^{(2)^T} H_{12}^{2} + \hat{v}_2^T$;
Set $\hat{\pi} = \left[\hat{\pi}_1^{(1)^T}, \hat{\pi}_1^{(2)^T}, \hat{\pi}_2^T\right]^T$;
Compute the PageRank vector $\pi^T = \hat{\pi}^T X$;
Normalize $\pi^T = \frac{\pi^T}{\|\pi^T\|_1}$;

4.3 Numerical Experiments

To complement the description, this section gives an indication of the computing time (seconds) in typical uses (average of 3 runs). The first three examples and the fifth example have been run on an Intel Core i7-3770 CPU at 3.40 GHz with 4 cores and it was used `MATLAB` R2015a. The fourth example have been run on an Intel(R) Xeon(R) CPU E3-1535M v6 @ 3.10 GHz and 32 GB RAM of memory with `MATLAB` R2015a.

Examples 1 and 2 present small matrices (of 12 and 100 nodes, respectively) to illustrate all the methods involved. Example 5 also uses a build-in matrix to highlight the method's ability to rank well sets of pages with the same PageRank. The matrices of examples 3 and 4 are available at the SuiteSparse Matrix Collection - a widely used set of sparse matrix benchmarks; both have been used to assess performance of web pages ranking algorithms.

4.3.1 Example 1 - Toy model

The toy example of twelve nodes described in Section 3.6.1 was used to illustrate the following iterative methods: Jacobi, Gauss-Seidel, SOR (with $\omega = 0.5$ and $\omega = 1.5$) and AOR (with $\omega = 0.5, r = 2$ and $\omega = 1.5, r = 0.5$) applied to the full matrix A. The model has five dangling nodes (nodes 2, 4, 7, 8 and 11), two weakly nondangling nodes (nodes 1 and 6) and the other five are strongly nondangling nodes.

Table 4.4 reports the number of iterations obtained with the methods above.

method	iterations
$Jacobi$	23
$Gauss - Seidel$	12
$SOR\ with\ \omega = 0.5$	48
$SOR\ with\ \omega = 1.5$	34
$AOR\ with\ \omega = 0.5.\ r = 2$	39
$AOR\ with\ \omega = 1.5.\ r = 0.5$	152

Table 4.4: Toy model: Number of iterations for Jacobi, Gauss-Seidel, SOR and AOR methods.

All methods achieved the same PageRank vector:

$$\pi = (9, 10, 12, 6, 11, 7, 8, 3, 2, 1, 4, 5).$$

Comparing with the results obtained in Section 3.6.1 we observed that the PageRank scores obtained by the linear system approach and by the eigenvector approach were equal.

The best method, in terms of number of iterations, is the Gauss-Seidel method.

Table 4.5 reports the number of iterations obtained with the MAAOR method for different values of matrices R and W, considering $\alpha = 0.85$ and $||r|| < 10^{-8}||b||$, r the residual and b the right hand side of the linear system. These different values leads to other iterative methods as the GAOR method, the GSOR method, the AOR method and the other methods described in Table 4.1 of Section 4.1.1. The number of iterations is listed for the MAAOR method applied to the full 12×12 matrix and also for the reduced matrix obtained with Lumping1-MAAOR method (as described in Section 4.2.1) and the further reduced matrix resulting from Lumping2-MAAOR method (as described in Section 4.2.2).

In Table 4.5 Ω is the diagonal of A, O the null matrix and I the identity matrix.

Method	R	W	MAAOR (it)	Lumping1-MAAOR (it)	Lumping2-MAAOR (it)
$Jacobi$	O	I	23	22	23
$Gauss - Seidel$	I	I	12	12	11
$SOR\ (\omega = 0.5)$	$0.5I$	$0.5I$	48	47	44
$SOR\ (\omega = 1.5)$	$1.5I$	$1.5I$	34	28	26
$Extrapolated\ Jacobi\ (\omega = 0.5)$	O	$0.5I$	56	54	51
$Extrapolated\ Gauss - Seidel\ (\omega = 1.5)$	I	$1.5I$	32	31	27
$Extrapolated\ Gauss - Seidel\ (\omega = 0.5)$	I	$0.5I$	37	36	34
$AOR\ (\omega = 1.5,\ r = 0.5)$	$0.5I$	$1.5I$	152	168	174
$AOR\ (\omega = 0.5,\ r = 1.5)$	$1.5I$	$0.5I$	35	35	34
$AOR\ (\omega = 0.5,\ r = 2)$	$2I$	$0.5I$	39	39	39
$AOR\ (\omega = 0.5,\ r = 5)$	$5I$	$0.5I$	102	103	99
$GSOR$	Ω	Ω	32	31	30
$GAOR\ (r = 0.5)$	0.5Ω	Ω	36	35	34
$GAOR\ (r = 1.5)$	1.5Ω	Ω	27	26	26
$GAOR\ (r = 0)$	O	Ω	40	39	37
$GAOR\ (r = 1)$	Ω	Ω	32	31	30
$GAOR\ (r = 3)$	3Ω	Ω	29	29	29
$MAAOR\ (\omega = 1.5,\ r = 0.5)$	0.5Ω	1.5Ω	42	42	41
$MAAOR\ (\omega = 0.5,\ r = 1.5)$	1.5Ω	0.5Ω	63	63	62
$MAAOR\ (\omega = 0.8,\ r = 3)$	3Ω	0.8Ω	36	36	36

Table 4.5: Toy model: Number of iterations for MAAOR method applied to the full matrix A, Lumping1-MAAOR method and Lumping2-MAAOR method for $tol < 10^{-8}$ and $\alpha = 0.85$. It includes GAOR, GSOR, AOR and other methods.

All methods achieved the same PageRank vector referred above.

As reported in Table 4.4, the best MAAOR method, applied to the full matrix A, in terms of number of iterations (12), is the Gauss-Seidel method. That is, the best parameters for the MAAOR method are $R = W = I$. The

Lumping1-MAAOR method mimics the results obtained with the MAAOR method applied to the full matrix, the best result is obtained with the Gauss-Seidel method (12 iterations). For the Lumping2-MAAOR method the best case is also the Gauss-Seidel method but it only needs 11 iterations to achieve convergence.

The table above does not include time needed to converge because, as the example is very small, convergence is very fast. Consequently, convergence time is similar for several methods and the best method in terms of time varies.
Also, in terms of convergence time, the lumping approaches are not good choices for this toy example. This is easily explained by the small size of the matrix. The time required to construct the reduced matrices and to classify their nodes is very significant when compared to the execution time of the methods. In addition, lumping also requires extra time to recover the Pagerank vector. This time (although small) depends on each method and is added to the convergence time. Consequently, for a very small matrix, the lumping strategy does not pay off in terms of convergence time. So, for this toy model, we only focus on the number of iterations.
We will verify if lumping is a faster strategy observing the next examples where the matrices are bigger.

Compared to the results achieved in Section 3.6.1, where the Lumping methods are used in an eigenvector point of view (Lumping with Power method and Aitken extrapolation), the linear system approach is better.
The best results with the eigenvector approach are obtained using LumpingE1 and LumpingE2 methods. These two methods need 21 iterations to converge. Therefore, the linear system approach using Lumping2-MAAOR method (with $R = W = I$ corresponding to the Gauss-Seidel method) is the best with only 11 iterations.

Next we will investigate the effect of using different damping factors in the PageRank problem. Table 4.6 reports the number of iterations obtained with the full MAAOR method for the same values of matrices R and W presented in Table 4.5 and several α. In addition to $\alpha = 0.85$, the cases $\alpha = 0.80$, $\alpha = 0.90$, $\alpha = 0.95$ and $\alpha = 0.99$ were analyzed.

Different the damping factors α leads to different PageRank vectors.
Increasing α slows down the convergence rate. Observing the Table 4.6 we conclude that a higher α leads to an increase in the number of iterations. The order of magnitude of this increase varies, it may be a small increase

Method	$\alpha = 0.80$ it.	$\alpha = 0.85$ it.	$\alpha = 0.90$ it.	$\alpha = 0.95$ it.	$\alpha = 0.99$ it.
Jacobi	20	23	26	29	32
$Gauss - Seidel$	11	12	13	15	17
SOR $(\omega = 0.5)$	44	48	52	57	62
SOR $(\omega = 1.5)$	33	34	34	34	34
$Extrapolated\ Jacobi$ $(\omega = 0.5)$	52	56	61	68	75
$Extrapolated\ Gauss - Seidel$ $(\omega = 1.5)$	31	32	32	32	32
$Extrapolated\ Gauss - Seidel$ $(\omega = 0.5)$	35	37	39	42	45
AOR $(\omega = 1.5;\ r = 0.5)$	114	152	229	460	2308
AOR $(\omega = 0.5;\ r = 1.5)$	34	35	36	39	42
AOR $(\omega = 0.5;\ r = 2)$	37	39	44	47	53
AOR $(\omega = 0.5;\ r = 5)$	80	102	152	295	1461
$GSOR$	28	32	36	42	47
$GAOR$ $(r = 0.5)$	32	36	41	47	54
$GAOR$ $(r = 1.5)$	23	27	30	35	40
$GAOR$ $(r = 0)$	35	40	45	53	60
$GAOR$ $(r = 1)$	28	32	36	42	47
$GAOR$ $(r = 3)$	26	29	31	34	37
$MAAOR$ $(\omega = 1.5;\ r = 0.5)$	41	42	43	45	46
$MAAOR$ $(\omega = 0.5;\ r = 1.5)$	57	63	71	80	90
$MAAOR$ $(\omega = 0.8;\ r = 3)$	33	36	39	44	48

Table 4.6: Toy model: Number of iterations for MAAOR considering several α ($tol < 10^{-8}$).

like the one verified in the MAAOR method with $\omega = 1.5$ and $r = 0.5$ or a substantial increase, like that of the AOR method with $\omega = 0.5$ and $r = 5$. In fact, the latter method required 14× more to converge with $\alpha = 0.99$ than with the conventional $\alpha = 0.85$, and the greater α is the greater the effort ($\alpha = 0.99$ required 5× to converge than $\alpha = 0.95$). These results are in accordance with what was stated in Section 3.1.

4.3.2 Example 2 - test1 matrix

For this second example it was used the matrix with 100 nodes described in Section 3.6.2. Following the same procedure as in the previous example, we began to analyze the Jacobi, Gauss-Seidel, SOR (with $\omega = 0.5$ and $\omega = 1.5$) and AOR (with $\omega = 0.5, r = 2$ and $\omega = 1.5, r = 0.5$) iterative methods applied to the full matrix A.

Table 4.7 reports the number of iterations and time (in milliseconds, ms) obtained with the methods above.

The best method, applied to the full matrix A, in terms of number of

method	iterations	time (ms)
$Jacobi$	31	3.284
$Gauss - Seidel$	17	1.290
$SOR\ with\ \omega = 0.5$	59	2.846
$SOR\ with\ \omega = 1.5$	37	2.507
$AOR\ with\ \omega = 0.5.\ r = 2$	48	2.942
$AOR\ with\ \omega = 1.5.\ r = 0.5$	608	21.68

Table 4.7: Test1 matrix (with 100 nodes): Number of iterations and convergence time for Jacobi, Gauss-Seidel, SOR and AOR methods.

iterations and time is, by far, the Gauss-Seidel method with 17 iterations and 1.290 ms.

All methods achieved the same PageRank vector and they all agree with the solution provided by the eigenvalue approach (see Section 3.6.2). The node with higher PageRank is node 82 which is a strongly dangling node.

The first 10 nodes with higher PageRank and their type are given in Table 4.8:

Type:	SND	SND	SND	WND	SND	SND	SND	D	D	D
Node:	82	81	10	6	9	12	83	69	7	8

Table 4.8: 100×100 matrix: first 10 nodes and their type.

Table 4.9 reports the number of iterations and time until convergence obtained with the MAAOR method for different values of matrices R and W. It includes other iterative methods as the GAOR, GSOR, AOR applied to the test1 matrix (100×100 matrix). The choice of parameters for the matrices R and W is the same as in Table 4.5.
The number of iterations and time until convergence ($||r|| < 10^{-8}||b||$, r the residual and b the right hand side of the linear system) are listed for the MAAOR method applied to the full 100×100 matrix, for the reduced matrix obtained with Lumping1-MAAOR method and for the further reduced matrix resulting from Lumping2-MAAOR method.

All MAAOR methods achieved the same PageRank vector. The first 10 nodes with higher PageRank are in Table 4.8.

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
Jacobi	31	3.284	28	1.188	31	1.768
Gauss-Seidel	17	1.290	16	0.883	17	1.513
SOR ($\omega = 0.5$)	59	2.846	53	1.366	59	1.872
SOR ($\omega = 1.5$)	37	2.507	48	1.189	38	1.675
Extrapolated Jacobi ($\omega = 0.5$)	70	3.775	64	1.052	71	1.925
Extrapolated Gauss-Seidel ($\omega = 1.5$)	33	1.997	36	0.798	33	1.969
Extrapolated Gauss-Seidel ($\omega = 0.5$)	44	2.483	41	0.838	45	1.751
AOR ($\omega = 1.5;\ r = 0.5$)	608	21.68	–	–	687	7.471
AOR ($\omega = 0.5;\ r = 1.5$)	38	2.239	37	0.833	40	1.669
AOR ($\omega = 0.5;\ r = 2$)	48	2.942	46	0.882	46	1.692
AOR ($\omega = 0.5;\ r = 5$)	1400	49.60	357	4.618	1405	14.08
GSOR	40	2.138	38	0.814	38	1.653
GAOR ($r = 0.5$)	46	2.122	44	0.869	45	1.776
GAOR ($r = 1.5$)	32	1.532	31	0.753	31	1.613
GAOR ($r = 0$)	51	2.376	49	0.902	50	1.721
GAOR ($r = 1$)	40	1.788	38	0.803	38	1.629
GAOR ($r = 3$)	36	1.659	33	0.769	33	1.592
MAAOR ($\omega = 1.5;\ r = 0.5$)	41	1.760	46	0.882	46	1.688
MAAOR ($\omega = 0.5;\ r = 1.5$)	73	2.585	71	1.492	71	1.868
MAAOR ($\omega = 0.8;\ r = 3$)	44	1.747	43	1.231	42	1.657

Table 4.9: Test1 matrix (with 100 nodes): Number of iterations and convergence time in milliseconds (ms) for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods for $tol < 10^{-8}$ and $\alpha = 0.85$. Symbol – appears whenever the maximum number of iterations (2000) was exceeded.

As in case of Table 4.7, the best of all MAAOR methods applied to the full matrix A, both in number of iterations and time, is the Gauss-Seidel method.
For the MAAOR method applied to the reduced matrix obtained with Lumping1 method we conclude that the best result, in terms of number of iterations, is also obtained by the Gauss-Seidel method. Comparing with the full MAAOR the Lumping1-MAAOR method took 1 less iteration to converge and it was faster. However, in terms of time it needs to converge, the best results does not necessarily correspond to the Gauss-Seidel method. From Table 4.9, regarding time, we observe that, for the lumping1 approach, the best choice of parameters is the one corresponding to the GAOR method with $r = 1.5$ followed by the GAOR method with $r = 3$.
Considering the MAAOR method applied to the further reduced matrix resulting from Lumping2 method, the best MAAOR method, in terms of number of iterations, is, again, the Gauss-Seidel method with the same number if iterations than the full MAAOR.

Analyzing Table 4.9 we conclude that, in general, Lumping1 presents the best times of the three approaches. Also, in most cases, Lumping2 is faster

than full MAAOR but slower than Lumping1.
However, although the matrix used in this example is larger than that of the previous example, it is still very small, so the time required to achieve convergence is also very small and varies from simulation to simulation.

Compared with the results reported in Section 3.6.2, where the lumping methods are used in an eigenvector point of view (lumping with power method and Aitken extrapolation), the linear system approach achieves better results. In terms of number of iterations, the best result with the eigenvector approach is obtained using the LumpingE2 method with 25 iterations. The linear system approach using Lumping1-MAAOR method (with $R = W = I$ corresponding to the Gauss-Seidel method) only took 16 iterations and the Lumping2-MAAOR method (Gauss-Seidel method) converged in 17 iterations.

Overall, and in line with the literature, Gauss-Seidel method seems to be the best approach to solve the PageRank problem (for this test case). Indeed, the performance of Gauss-Seidel applied to the PageRank linear system is considered excellent, given its modest memory requirements. It often converges in roughly half the number of Power method iterations [46].

In Tables 4.7 and 4.9 $\omega = 0.5$ and $\omega = 1.5$ were the parameters selected for the SOR method. However, they may not be the best parameters for the SOR method. So, next we will investigate which are the most convenient parameters for this method and this matrix.
Figure 4.1 shows the results obtained applying the SOR method for different values of the parameter ω (from 0.1 to 1.9, step 0.1) to the full 100×100 matrix. It considers $\alpha = 0.85$ and tolerance $tol < 10^{-8}$.

Observing Figure 4.1 we conclude that if we chose $\omega \in [0.8, 1.4]$ we will have an adequate choice of parameters. The best result for the SOR method in terms of number of iterations (11) and time is obtained using $\omega = 1.1$. The second best is $\omega = 1.2$ with 14 iterations and $\omega = 1$ (corresponding to the Gauss-Seidel method) only appears in third place with 17 iterations. So, contrary to what was stated above, the SOR method is not the best method if we operate on the full 100×100 matrix.

In Table 4.9 $\omega = 0.5$ and $\omega = 1.5$ were also the parameters selected for the Extrapolated Gauss-Seidel method, that is, the AOR method with $r = 1$ fixed and varying ω (from 0.1 to 1.9, step 0.1). To investigate the best ω parameters for this method the following figure was created (Figure 4.2).

The smaller number of iterations for the Extrapolated Gauss-Seidel method

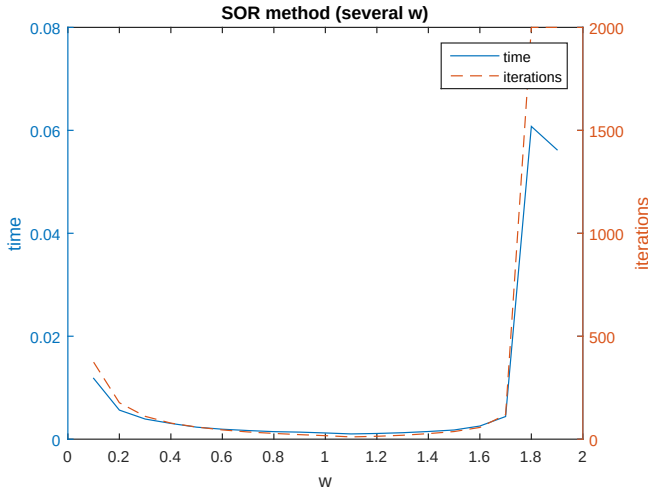


Figure 4.1: 100×100 matrix: Number of iterations and time of the SOR method for different values of ω.

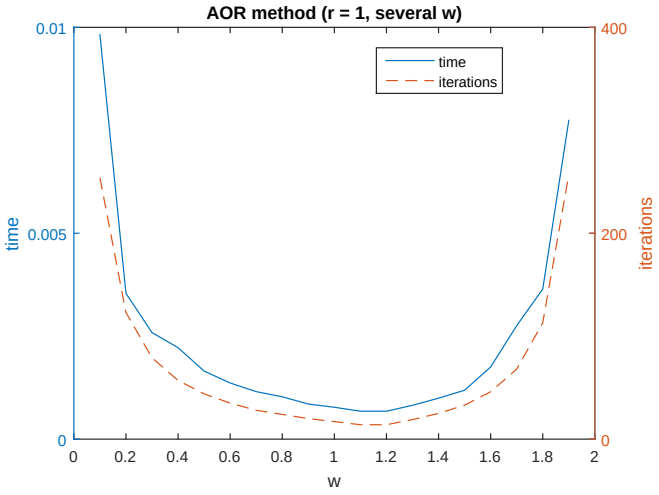
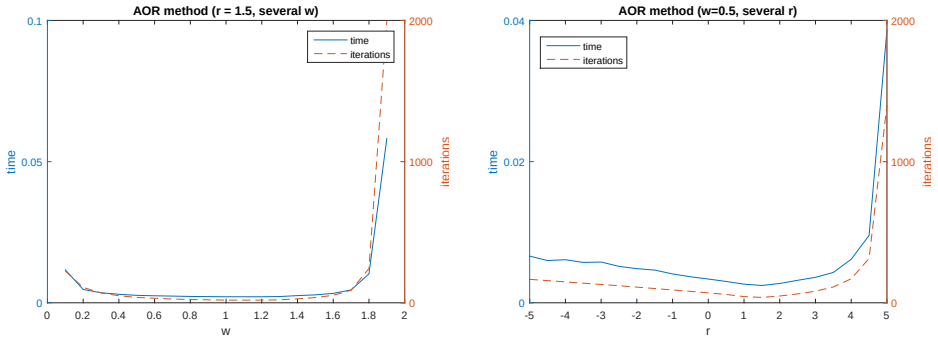


Figure 4.2: 100×100 matrix: Number of iterations and time of the AOR method with $r = 1$ for different values of ω (Extrapolated Gauss-Seidel method).

is obtained within the interval $\omega \in [0.9, 1.3]$. The best cases are $\omega = 1.1$ and $\omega = 1.2$ with 14 iterations (which is also an improvement over the Gauss-Seidel method).

Finally, in Figure 4.3 we present two simulations for the AOR method applied to the full 100×100 matrix (test1 matrix). In Figure 4.3(a) we have a fixed $r = 1.5$ and ω varies from 0.1 to 1.9 with step 0.1. In Figure 4.3(b)

we have a fixed $\omega = 0.5$ and r varies from -5 to 5 with step 0.5.



(a) $r = 1.5$, several ω (b) $\omega = 0.5$, several r

Figure 4.3: 100×100 matrix: AOR method fixing ω or r.

The best results for the AOR method with $r = 1.5$ (Figure 4.3(a)) both in iterations (19) and time are obtained using $\omega = 1.0$, $\omega = 1.1$ and $\omega = 1.2$. The best result for the AOR method with $\omega = 0.5$ (Figure 4.3(b)) in terms of number of iterations (38) and time is obtained using $r = 1.5$.

4.3.3 Example 3 - EPA matrix

We consider now the EPA matrix – Kleinberg: Pajek network, pages linking to www.epa.gov. It is a 4772×4772 matrix with 8965 nonzeros, 4711 strongly connected components, 70.18% dangling nodes, 19.72% strongly nondangling nodes and 10.10% weakly nondangling nodes.

That is, a linear system of size 4772×4772 for the original MAAOR iteration, of size 3626×3626 for Lumping1-MAAOR, and 1363×1363 for Lumping2-MAAOR.

As a first approach we begin to study the EPA matrix obtained with MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods, for some MAAOR parameters, considering the standard $\alpha = 0.85$. These results are presented in Table 4.10, which reports the number of iterations and time (in seconds, s) until convergence ($||r|| < 10^{-8}||b||$, r the residual and b the right hand side of the linear system) for the EPA matrix.

The best result, for the three methods, is obtained with GAOR with $r = 1.5$. The overall speed of convergence is high due to the tuned sparse

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
Jacobi	110	0.030	119	0.013	123	0.010
Gauss-Seidel	60	0.019	61	0.011	60	0.009
SOR ($\omega = 0.5$)	176	0.046	176	0.017	170	0.012
SOR ($\omega = 1.5$)	2767	0.686	2761	0.223	2760	0.152
Extrapolated Jacobi ($\omega = 0.5$)	227	0.044	228	0.015	221	0.014
Extrapolated Gauss-Seidel ($\omega = 1.5$)	–	–	–	–	–	–
Extrapolated Gauss-Seidel ($\omega = 0.5$)	128	0.036	130	0.016	129	0.010
AOR ($\omega = 1.5;\ r = 0.5$)	4553	1.214	4541	0.362	4538	0.241
AOR ($\omega = 0.5;\ r = 1.5$)	73	0.023	78	0.009	79	0.007
AOR ($\omega = 0.5;\ r = 2$)	157	0.045	156	0.014	165	0.013
AOR ($\omega = 0.5;\ r = 5$)	1332	0.361	1331	0.100	1331	0.078
GSOR	60	0.015	61	0.009	60	0.007
GAOR ($r = 0.5$)	84	0.025	84	0.011	81	0.010
GAOR ($r = 1.5$)	35	0.013	38	0.007	37	0.006
GAOR ($r = 0$)	110	0.025	119	0.013	123	0.010
GAOR ($r = 1$)	60	0.017	61	0.010	60	0.007
GAOR ($r = 3$)	3873	1.030	3873	0.331	3872	0.225
MAAOR ($\omega = 1.5;\ r = 0.5$)	4553	1.185	4541	0.347	4538	0.236
MAAOR ($\omega = 0.5;\ r = 1.5$)	73	0.019	78	0.011	79	0.007
MAAOR ($\omega = 0.8;\ r = 3$)	–	–	–	–	–	–

Table 4.10: EPA matrix: Number of iterations and convergence time in seconds for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods for $tol < 10^{-8}$ and $\alpha = 0.85$. Symbol – appears whenever the maximum number of iterations (5000) was exceeded.

matrix operations used; compressed sparse row (CSR) format is used and all operations are done on this sparse data structure scheme.

The hybrid Lumping2-MAAOR is the one that requires less computation time while all three provide similar number of iterations to reach convergence. These findings can also be seen in [103].

In a second approach, we tried to seek the best combination of parameters for each method within the MAAOR family of methods. So, several experiments were carried out.

Beginning with the simple SOR method, the choice of parameter ω was made within the interval stated in Theorem 2.3.9. Table 4.11 shows the results obtained applying the SOR method for different values of ω (from 0.1 to 1.9, step 0.1) to the full EPA matrix. Figure 4.4 illustrates these results. Both, in Figure 4.4 and in Table 4.11, we present the number of iterations and time (in seconds) to obtain convergence for the SOR method with a tolerance $tol < 10^{-8}$ and $\alpha = 0.85$. The symbol $-$ appears whenever the maximum number of iterations (2000) is exceeded.

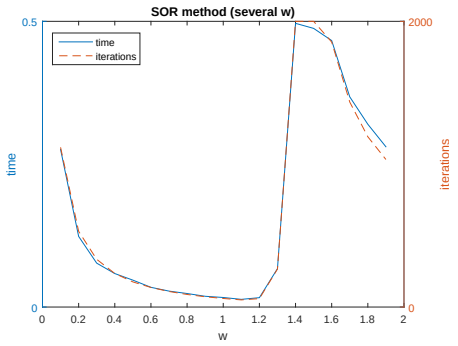


Figure 4.4: EPA matrix: SOR method (several ω).

method	iterations	time (s)
SOR ($\omega = 0.1$)	1118	0.276
SOR ($\omega = 0.2$)	530	0.124
SOR ($\omega = 0.3$)	334	0.077
SOR ($\omega = 0.4$)	235	0.059
SOR ($\omega = 0.5$)	176	0.047
SOR ($\omega = 0.6$)	137	0.035
SOR ($\omega = 0.7$)	109	0.028
SOR ($\omega = 0.8$)	88	0.024
SOR ($\omega = 0.9$)	72	0.019
SOR ($\omega = 1.0$)	60	0.017
SOR ($\omega = 1.1$)	50	0.013
SOR ($\omega = 1.2$)	59	0.016
SOR ($\omega = 1.3$)	266	0.067
SOR ($\omega = 1.4$)	–	–
SOR ($\omega = 1.5$)	–	–
SOR ($\omega = 1.6$)	1855	0.466
SOR ($\omega = 1.7$)	1435	0.368
SOR ($\omega = 1.8$)	1192	0.320
SOR ($\omega = 1.9$)	1033	0.281

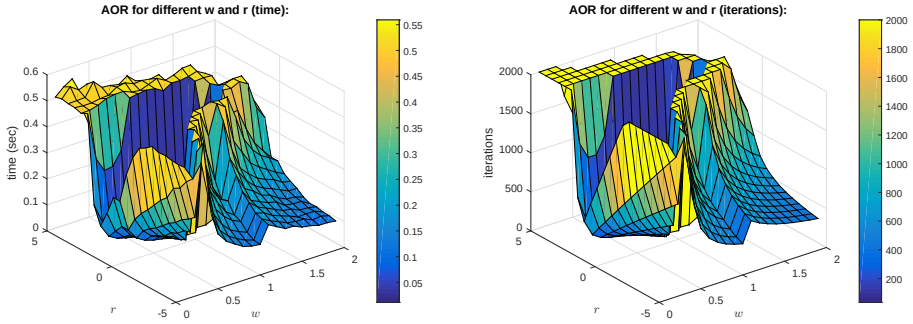
Table 4.11: EPA matrix: SOR method (several ω).

Looking at Figure 4.4 we conclude that, both in time and iterations, if we choose ω within the interval $[0.8, 1.2]$ we will have good results. From Table 4.11 we observe that the best result for the SOR method in terms of number of iterations (50) and time (0.013 s) was obtained using $\omega = 1.1$. The second best was the SOR method for $\omega = 1.2$ with 59 iterations and 0.016 s. Consequently, these were the two new parameters chosen for the SOR method.

Next, searching for the best parameters ω and r for the AOR method, we developed a `MATLAB` code in which the parameter ω varies from 0.1 to 1.9 (step 0.1) and the parameter r varies from -5 to 5 (step 0.5). Three dimensional graphs were created to illustrate the results obtained. Figure 4.5(a) shows the convergence time (in seconds) with the different values of ω and r for the full EPA matrix, and Figure 4.5(b) presents the number of iterations necessary to converge.

The best result for the AOR method applied to the full EPA matrix was obtained with $\omega = 1$ and $r = 1.5$ both in terms of number of iterations (35) and time (0.010 s).

Taking into account the EPA matrix as well as the other the matrices studied, we may concluded that the best parameters for the AOR method sum up to $\omega = 1$, $\omega = 1.1$, $\omega = 1.2$ and $r = 1.5$, $r = 1$. With this results in mind we made two simulations fixing r ($r = 1.5$, $r = 1$) and varying ω and also four simulations fixing ω ($\omega = 1$, $\omega = 1.1$, $\omega = 1.2$, $\omega = 0.5$) and varying r. The last case, AOR method with $\omega = 0.5$ and several r, was selected for

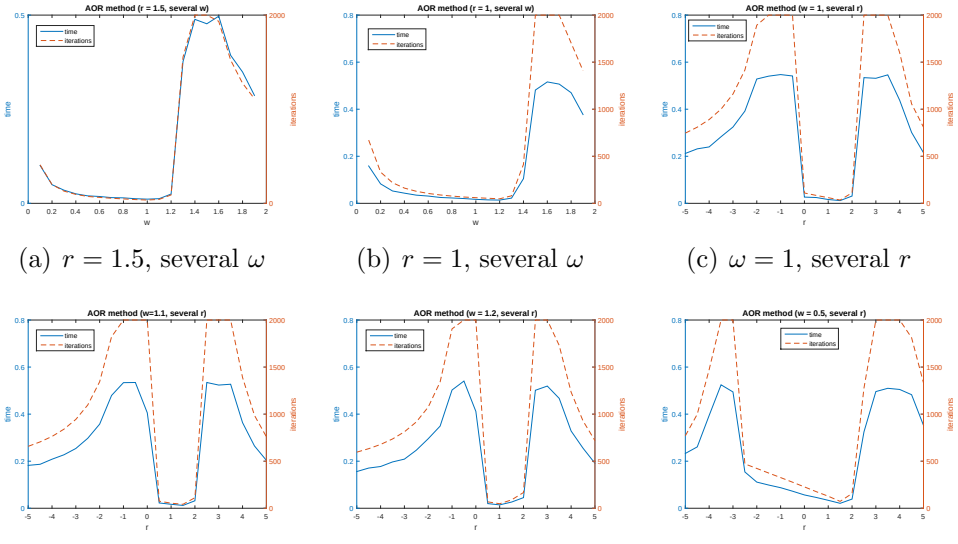


(a) Time (b) Iterations

Figure 4.5: EPA matrix: AOR method for different ω and r (time and iterations).

comparison because it was widely used in the first approach (Table 4.10).

The graphs that illustrate these six simulations are given in Figure 4.6. In all cases it was used the full EPA matrix.



(a) $r = 1.5$, several ω (b) $r = 1$, several ω (c) $\omega = 1$, several r

(d) $\omega = 1.1$, several r (e) $\omega = 1.2$, several r (f) $\omega = 0.5$, several r

Figure 4.6: EPA matrix: AOR method fixing ω or r.

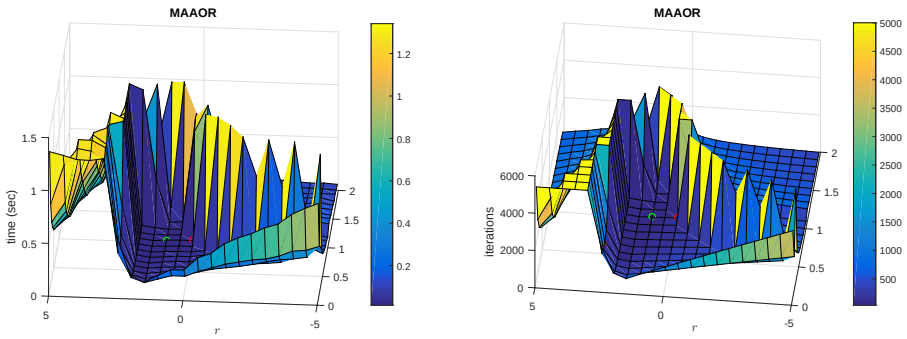
On the one hand, from Figure 4.6(a) and 4.6(b) were r was fixed, we can see that if ω remains within $[0.8, 1.2]$ we have an adequate choice of parameters. The best result for AOR method with fixed $r = 1.5$ was obtained with $\omega = 1$ (35 iterations and 0.011 s). Also, for the AOR method with fixed $r = 1$ the best parameter was $\omega = 1.2$ (48 iterations and 0.014 s). This case

(AOR with $r = 1$) corresponds to the Extrapolated Gauss-Seidel method.

On the other hand, Figures 4.6(c) to 4.6(f) illustrates the results obtained fixing ω and varying $r \in [-5, 5]$. For all cases it is safe to say that with $r \in [0.5, 2]$ we have acceptable results. The best result for AOR method with fixed $\omega = 1$ was obtained with $r = 1.5$ (35 iterations and 0.012 s), for AOR method with fixed $\omega = 1.1$ was $r = 1.5$ (42 iterations and 0.012 s), for AOR method with fixed $\omega = 1.2$ was $r = 1$ (48 iterations and 0.014 s) and, finally, for AOR method with fixed $\omega = 0.5$ the best result was achieved with $r = 1.5$ (73 iterations and 0.021 s). This last case was not selected to incorporate the final selection of parameters.

Now, we analyze the choice of parameters for the MAAOR method varying both ω and r considering the full MAAOR method, the Lumping1-MAAOR method and the Lumping2-MAAOR method.

New `MATLAB` codes were developed to create three dimensional graphs that allow a more immediate recognition of the appropriate choice of parameters. Starting with the MAAOR method for several ω and r applied to the full EPA matrix, it is possible to observe the convergence time (in seconds) in Figure 4.7(a) and the number of iterations necessary to converge in Figure 4.7(b).



(a) Time (b) Iterations

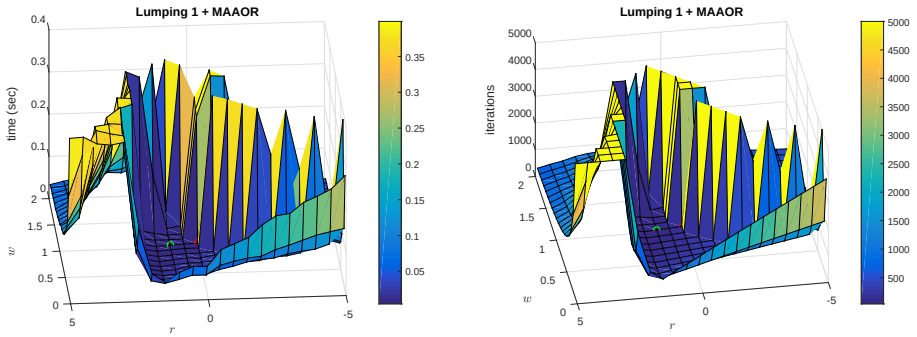
Figure 4.7: EPA matrix: full MAAOR, several ω and r (time and iterations).

The red circle in Figure 4.7(a) and 4.7(b) corresponds to the results obtained by the Jacobi method and the green circle corresponds to the Gauss-Seidel method. The deep blue in the graphs illustrates the best results, it implies, usually, a lower number of iterations and a lower convergence time. In yellow we have the worst results.

From Figure 4.7(a) and 4.7(b) we can observe that there are a wide range of combinations of values for ω and r that presents good results.

For the full MAAOR method applied to the EPA matrix the best results were obtained considering MAAOR with $\omega = 1$ and $r = 1.5$, which corresponds to the GAOR method with $r = 1.5$. In this case the GAOR method needed 35 iterations and 0.010 s to converge. In some simulations the MAAOR method with $\omega = 0.9$ and $r = 1.5$ was faster, despite needing more iterations (40).

Figure 4.8(a) shows the convergence time (in seconds) with the different values of ω and r for the reduced EPA matrix obtained with the Lumping1-MAAOR method, Figure 4.8(b) illustrates the number of iterations necessary to converge considering the same method.



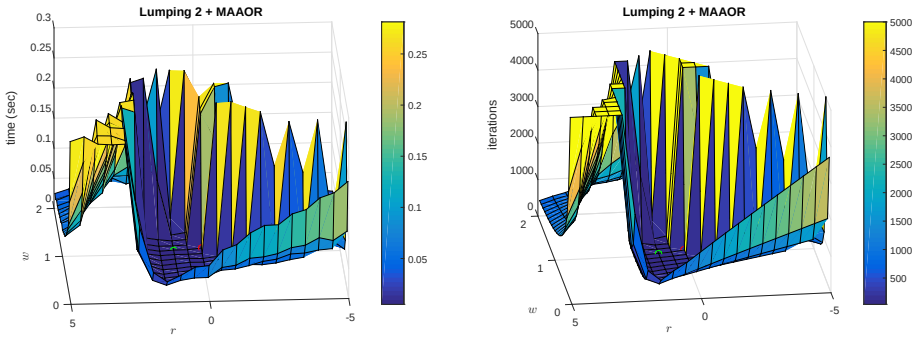
(a) Time (b) Iterations

Figure 4.8: EPA matrix: Lumping1-MAAOR, several ω and r (time and iterations).

In Figure 4.8(a) and 4.8(b), the red circle corresponds to the results of the Jacobi method and the green circle to the Gauss-Seidel method. As in the case of the full MAAOR method, the best choice of parameters corresponds to MAAOR with $\omega = 1$ and $r = 1.5$, that is, the GAOR method with $r = 1.5$. Comparing with the full MAAOR, although the Lumping1-MAAOR method took more iterations (38), the time needed to converge was significantly smaller (0.006 s). In other simulations it was possible to obtain another combination of parameters that was faster, even with more iterations, for instance, MAAOR with $\omega = 0.8$ and $r = 1.5$.

Taking into account the further reduced EPA matrix obtained with the Lumping2-MAAOR method we can observe the correspondent results in time and iterations in Figures 4.9(a) and 4.9(b), respectively.

Mimicking the full MAAOR and the Lumping1-MAAOR methods, the best result obtained with the Lumping2-MAAOR method corresponds to



(a) Time (b) Iterations

Figure 4.9: EPA matrix: Lumping2-MAAOR, several ω and r (time and iterations).

the GAOR method with $r = 1.5$ (i.e., MAAOR with $\omega = 1$ and $r = 1.5$). It was the fastest (0.004 s) of the three approaches and it took 37 iterations to converge. There are other combination of parameters that, sometimes, were faster, such as MAAOR with $\omega = 0.9$ and $r = 1.5$.

Finally, analyzing all the situations described above, it was possible to conclude about the most promising parameters for all the methods. So, the best parameters known for each method, within the MAAOR family of methods, were used in a new simulation with results illustrated in Table 4.12.

Analyzing Table 4.12, as was expected considering the previous results, we observe that the best combination of parameters, in terms of number of iterations, for the three approaches, is the MAAOR with $\omega = 1$ and $r = 1.5$ (i.e., GAOR method with $r = 1.5$). In this case, full MAAOR took 35 iterations and 0.011 s to converge, Lumping1-MAAOR needed more iterations (38) but it was faster (0.007 s) and, finally, Lumping2-MAAOR took 37 iterations but was the fastest of the three strategies (0.006 s).

For the full MAAOR, the fastest choice of parameters was MAAOR with $\omega = 1$ and $r = 1.5$ with 0.011 s. However, for the Lumping1-MAAOR approach, we had several combination of parameters that took around 0.007 s and the same happened with the Lumping2-MAAOR approach where 0.006 s was the minimum time achieved. In reality, for the Lumping1 option the fastest method (0.006543 s) was MAAOR with $\omega = 0.9$ and $r = 1.5$, the second best (0.006640 s) was MAAOR with $\omega = 0.8$ and $r = 1.5$, the third (0.006741 s) was GAOR with $r = 1.5$, and the AOR method with $\omega = 1.1$ and $r = 1.5$ took the fourth position with 0.006770 s. Due the proximity of these results, different simulations change this ranking.

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
Jacobi	110	0.030	119	0.013	123	0.011
Gauss-Seidel	60	0.018	61	0.010	60	0.009
SOR ($\omega = 1.2$)	59	0.020	66	0.011	68	0.008
SOR ($\omega = 1.1$)	50	0.013	51	0.010	51	0.006
Extrapolated Jacobi ($\omega = 1.2$)	–	–	–	–	–	–
Extrapolated Gauss-Seidel ($\omega = 1.2$)	48	0.012	50	0.007	50	0.006
Extrapolated Gauss-Seidel ($\omega = 1.1$)	54	0.013	54	0.009	54	0.006
AOR ($\omega = 1.2;\ r = 1.5$)	89	0.021	101	0.011	104	0.009
AOR ($\omega = 1.1;\ r = 1.5$)	42	0.012	47	0.007	49	0.006
AOR ($\omega = 1;\ r = 1.5$)	35	0.014	38	0.008	37	0.006
AOR ($\omega = 0.9;\ r = 1.5$)	40	0.014	42	0.007	43	0.006
AOR ($\omega = 1.2;\ r = 0.5$)	68	0.018	68	0.009	66	0.007
GSOR	60	0.016	61	0.011	60	0.007
GAOR ($r = 0.5$)	84	0.023	84	0.011	81	0.010
GAOR ($r = 3$)	3873	1.046	3873	0.351	3872	0.259
GAOR ($r = 0$)	110	0.027	119	0.011	123	0.009
GAOR ($r = 1.5$)	35	0.011	38	0.007	37	0.006
GAOR ($r = 2$)	112	0.034	116	0.013	116	0.010
MAAOR ($\omega = 1.6;\ r = 1.5$)	1934	0.535	1930	0.143	1929	0.106
MAAOR ($\omega = 1.5;\ r = 1.5$)	2767	0.699	2761	0.234	2760	0.170
MAAOR ($\omega = 1.4;\ r = 1.5$)	–	–	–	–	–	–
MAAOR ($\omega = 0.9;\ r = 1.5$)	40	0.011	42	0.007	43	0.006
MAAOR ($\omega = 0.8;\ r = 1.5$)	47	0.012	45	0.007	49	0.006
MAAOR ($\omega = 1.5;\ r = 2$)	2204	0.562	2200	0.169	2198	0.117
MAAOR ($\omega = 1.4;\ r = 2$)	3526	0.921	3518	0.266	3516	0.179
MAAOR ($\omega = 1.3;\ r = 2$)	–	–	–	–	–	–
MAAOR ($\omega = 1.2;\ r = 2$)	167	0.041	189	0.018	195	0.013
MAAOR ($\omega = 1.1;\ r = 2$)	112	0.036	115	0.013	115	0.012

Table 4.12: EPA matrix: Number of iterations and convergence time in seconds for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods for $tol < 10^{-8}$ and $\alpha = 0.85$. Symbol – appears whenever the maximum number of iterations (5000) was exceeded.

For the Lumping2-MAAOR option the situation was analogous, the fastest combination of parameters was MAAOR with $\omega = 0.9$ and $r = 1.5$, the second best was GAOR with $r = 1.5$, the third was Extrapolated Gauss-Seidel with $\omega = 1.2$, and the fourth was AOR method with $\omega = 1.1$ and $r = 1.5$.

Therefore, for most choice of parameters, the lumping strategy took more iterations to converge than the full MAAOR, however, in all cases, the lumping strategy took less time to converge. Also, a further reduction on the matrix is worth the effort to move to Lumping2-MAAOR.

In some situations the advantage can be impressive. For instance, in case of MAAOR with $\omega = 1.4$ and $r = 2$, the no lumping approach required $3.5\times$ to converge than the Lumping1 and more than $5\times$ than the Lumping2.

For this example, and contrary to most results in the literature, the Gauss-Seidel is superseded by recent MAAOR family of methods in the PageRank problem. Also, both lumping strategies bring better results to the PageRank

problem.

This EPA model is relatively small, so larger models were tested.

4.3.4 Example 4 - wikipedia-20070206 matrix

Next we explore the wikipedia-20070206 matrix – Gleich: Wikipedia pages at Feb 6, 2007. A 3566907×3566907 matrix with 45030389 nonzeros, 1203340 strongly conneted components, 2.84% dangling nodes, 88.07% strong nondangling nodes and 9.09% weakly nondangling nodes. In contrast with the previous example, this one concerns a dense matrix.

In order to understand if the computation has acceptable costs, we present in Table 4.13 results with the much larger matrix wikipedia-20070206. Following the strategy used in Section 4.3.3, as a first approach we used the same choice of parameters as in Table 4.10. This procedure was set out in [103].

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
Jacobi	138	86.02	138	89.05	143	60.83
Gauss-Seidel	3	16.71	5	20.70	4	13.88
SOR ($\omega = 0.5$)	6	18.43	8	22.36	7	15.00
SOR ($\omega = 1.5$)	2	16.28	4	20.20	3	13.35
Extrapolated Jacobi ($\omega = 0.5$)	159	98.09	159	101.1	165	68.75
Extrapolated Gauss-Seidel ($\omega = 1.5$)	3	18.96	5	24.73	4	15.36
Extrapolated Gauss-Seidel ($\omega = 0.5$)	3	18.98	5	24.86	4	15.30
AOR ($\omega = 1.5;\ r = 0.5$)	6	20.87	8	32.27	7	16.52
AOR ($\omega = 0.5;\ r = 1.5$)	2	18.37	4	24.03	3	14.84
AOR ($\omega = 0.5;\ r = 2$)	1	17.75	3	23.49	2	14.40
AOR ($\omega = 0.5;\ r = 5$)	1	17.71	2	22.82	1	13.99
GSOR	3	16.70	6	21.09	5	14.14
GAOR ($r = 0.5$)	8	22.03	11	28.58	11	18.13
GAOR ($r = 1.5$)	2	18.21	4	23.94	3	14.72
GAOR ($r = 0$)	140	86.92	140	89.96	146	62.34
GAOR ($r = 1$)	3	16.70	6	21.09	5	14.43
GAOR ($r = 3$)	1	17.61	3	23.32	2	14.61
MAAOR ($\omega = 1.5;\ r = 0.5$)	8	22.08	11	28.75	11	18.83
MAAOR ($\omega = 0.5;\ r = 1.5$)	2	18.31	4	24.00	3	15.34
MAAOR ($\omega = 0.8;\ r = 3$)	1	17.65	3	23.34	2	14.64

Table 4.13: wikipedia matrix: convergence time (in seconds, s) for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods for $tol < 10^{-8}$ and $\alpha = 0.85$.

This problem is particularly interesting since the percentage of dangling nodes is extremely reduced. The effort in solving the reduced system is thus comparable to the original. Even though, lumping showed that it is a crucial

strategy to gain performance, in particular Lumping2-MAAOR outperforms all others. The correct choice of the linear solver is however dependent of the approach followed. The choice SOR with $\omega = 1.5$, Gauss-Seidel and AOR with $\omega = 0.5$ and $r = 5$ are the best for Lumping2-MAAOR.

In a second approach, a new table (Table 4.14) with the same choice of parameters used in the EPA matrix example (Table 4.12) was then presented so that the results could be compared.

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
Jacobi	138	86.37	138	87.57	143	73.72
Gauss-Seidel	3	16.78	5	20.25	4	16.19
SOR ($\omega = 1.2$)	2	16.23	4	19.79	3	15.62
SOR ($\omega = 1.1$)	2	16.28	5	20.32	3	15.59
Extrapolated Jacobi ($\omega = 1.2$)	133	84.66	133	84.87	138	72.04
Extrapolated Gauss-Seidel ($\omega = 1.2$)	3	19.07	5	24.05	4	18.52
Extrapolated Gauss-Seidel ($\omega = 1.1$)	3	19.08	5	23.98	4	18.33
AOR ($\omega = 1.2;\ r = 1.5$)	2	18.38	4	23.57	3	17.97
AOR ($\omega = 1.1;\ r = 1.5$)	2	18.34	4	23.47	3	18.12
AOR ($\omega = 1;\ r = 1.5$)	2	18.22	4	23.99	3	17.51
AOR ($\omega = 0.9;\ r = 1.5$)	2	18.26	4	24.30	3	17.74
AOR ($\omega = 1.2;\ r = 0.5$)	6	20.86	8	27.85	7	19.48
GSOR	3	16.76	6	21.87	5	16.81
GAOR ($r = 0.5$)	8	21.95	11	29.09	11	22.04
GAOR ($r = 3$)	1	17.61	3	23.77	2	17.33
GAOR ($r = 0$)	140	88.48	140	94.43	146	76.70
GAOR ($r = 1.5$)	2	20.98	4	25.77	3	18.03
GAOR ($r = 2$)	1	20.32	3	24.98	2	17.14
MAAOR ($\omega = 1.6;\ r = 1.5$)	2	20.67	4	25.85	3	17.22
MAAOR ($\omega = 1.5;\ r = 1.5$)	2	19.63	4	21.17	3	16.04
MAAOR ($\omega = 1.4;\ r = 1.5$)	2	21.37	4	25.50	3	17.70
MAAOR ($\omega = 0.9;\ r = 1.5$)	2	21.67	4	25.61	3	17.69
MAAOR ($\omega = 0.8;\ r = 1.5$)	2	20.91	4	26.00	3	17.86
MAAOR ($\omega = 1.5;\ r = 2$)	1	19.38	3	26.17	2	17.11
MAAOR ($\omega = 1.4;\ r = 2$)	1	19.63	3	25.96	2	16.76
MAAOR ($\omega = 1.3;\ r = 2$)	1	19.71	3	24.75	2	16.07
MAAOR ($\omega = 1.2;\ r = 2$)	1	19.83	3	23.73	2	18.87
MAAOR ($\omega = 1.1;\ r = 2$)	1	19.86	3	22.88	2	18.10

Table 4.14: wikipedia matrix: Number of iterations and convergence time (in seconds, s) for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods for $tol < 10^{-8}$ and $\alpha = 0.85$.

The best results are obtained with SOR ($\omega = 1.1$) and SOR ($\omega = 1.2$), followed by MAAOR ($\omega = 1.5,\ r = 1.5$) and MAAOR ($\omega = 1.3,\ r = 2$) for the Lumping2-MAAOR approach.
As in Table 4.13 we conclude that the most efficient approach is the Lumping2-MAAOR method. Whatever the choice of parameters the Lumping2-MAAOR method is always the fastest and, generally, the Lumping1-MAAOR method is worse than the full MAAOR method.
These results are in line with the ones obtained with the EPA matrix (Ex-

ample 3).

Next, we will investigate if the best choices for the parameter ω in case of the SOR method applied to the wikipedia matrix are the same as for the EPA matrix.

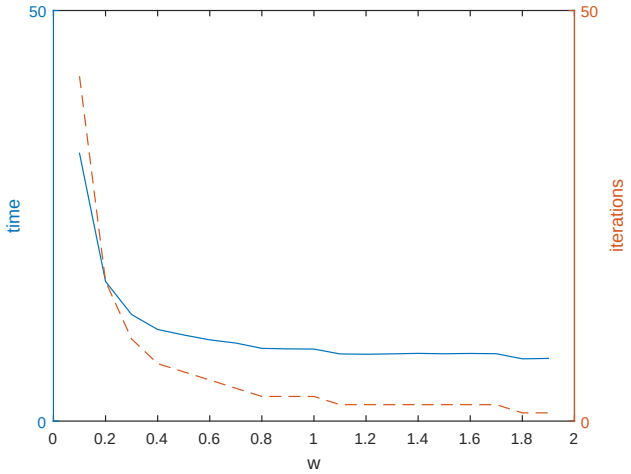


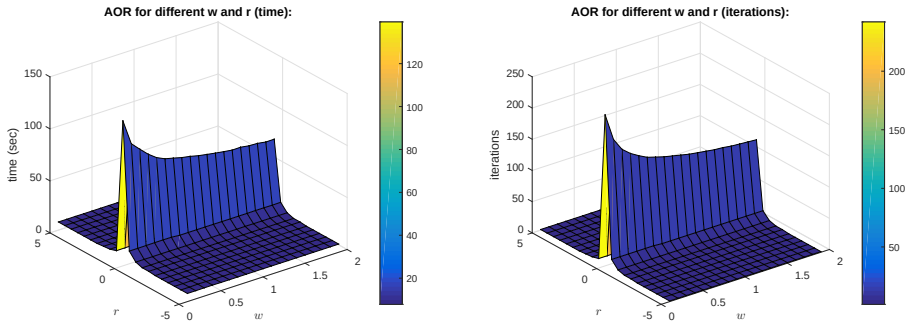
Figure 4.10: wikipedia matrix: Number of iterations and time of the SOR method for different values of ω.

In fact, the appropriate parameters for the SOR method depends on the matrix used. For the EPA matrix $\omega \in [0.8, 1.2]$ is a good choice, however, for the wikipedia matrix, higher values of ω are preferable, that is $\omega \in [1.1, 1.9]$. The fastest SOR methods are the ones with $\omega = 1.8$ and $\omega = 1.9$, that are able to converge in approximately 7.6 s.

Now, searching for the best parameters ω and r for the AOR method applied to the full wikipedia matrix we present Figure 4.11.

From Figure 4.11 we conclude that there are a wide range of values for the parameters that present good performance. However, the fastest AOR method is the one with $\omega = 0.6$ and $r = -2.5$ that converged in 7.798 s (1 iteration). Similar convergence time is obtained with several other combination of parameters.

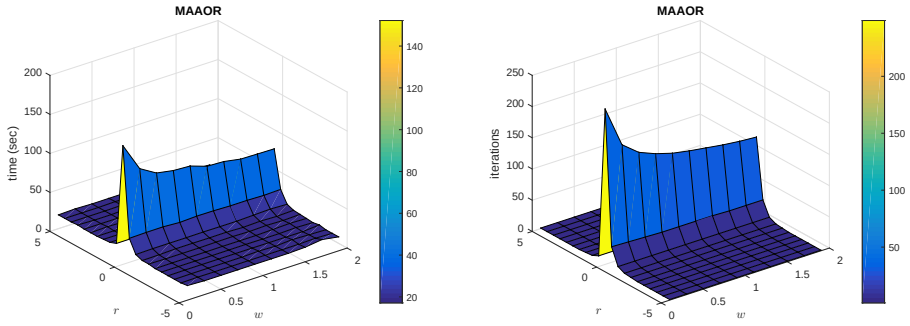
Finally, in an attempt to select the most appropriate MAAOR parameter combinations for the three strategies in the case of wikipedia matrix, we construct the following three-dimensional plots. Figure 4.12(a) illustrates the time (in seconds) required to achieve convergence for different values of ω (from 0.1 to 1.9, step 0.2) and for r (from -5 to 5, step 0.5) and Figure



(a) Time (b) Iterations

Figure 4.11: wikipedia matrix: Number of iterations and time of the AOR method for different values of ω and r.

4.12(b) the number of iterations required in the case of the full MAAOR method.

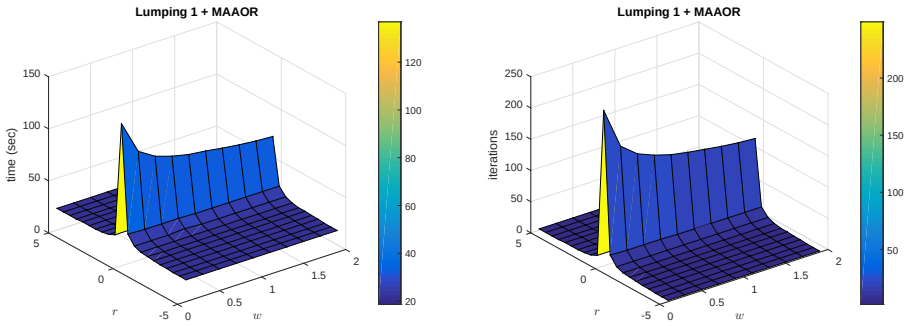


(a) Time (b) Iterations

Figure 4.12: wikipedia matrix: full MAAOR, several ω and r (time and iterations).

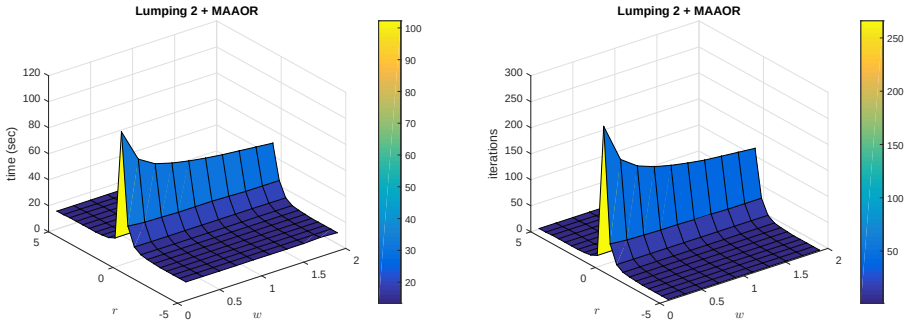
In Figure 4.13 we did the same but using the Lumping1-MAAOR method. Again, in Figure 4.14 the procedure was analogous for the Lumping2-MAAOR method.

Observing the three MAAOR figures above we conclude that all the plots have, approximately, the same shape. Also, the shape of the plots for the AOR method (Figure 4.11) is analogous. That is, the combination of the adequate ω and r parameters is similar for all methods applied to the wikipedia matrix and is different from the one appropriate for the EPA matrix.



(a) Time (b) Iterations

Figure 4.13: wikipedia matrix: Lumping1-MAAOR, several ω and r (time and iterations).



(a) Time (b) Iterations

Figure 4.14: wikipedia matrix: Lumping2-MAAOR, several ω and r (time and iterations).

Based on the information obtained with the plots above we sum up some of the best combination of MAAOR parameters in Table 4.15.

The best choice of MAAOR parameters for the three approaches is $\omega = 1.5$ and $r = 1.5$. Lumping2-MAAOR method is always faster than Lumping1-MAAOR and full MAAOR. Lumping1-MAAOR is slower than full MAAOR and takes more iterations. So, in case of the wikipedia matrix, the Lumping1-MAAOR approach is not a good alternative to full MAAOR. However, the Lumping2-MAAOR is, even though it needs more iterations than the no lumping version.

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
MAAOR ($\omega = 0.1;\ r = -4.5$)	1	17.49	2	21.12	2	14.25
MAAOR ($\omega = 0.1;\ r = -4.0$)	1	17.52	2	20.94	2	14.22
MAAOR ($\omega = 0.1;\ r = -3.5$)	1	17.54	3	21.55	2	14.26
MAAOR ($\omega = 0.1;\ r = -2.5$)	1	17.50	4	22.17	3	14.64
MAAOR ($\omega = 0.1;\ r = 5.0$)	1	17.69	2	20.95	1	13.79
MAAOR ($\omega = 0.5;\ r = -4.5$)	1	17.74	2	20.89	2	14.20
MAAOR ($\omega = 0.5;\ r = 2.5$)	1	17.68	3	21.51	2	14.23
MAAOR ($\omega = 0.5;\ r = 4.5$)	1	17.70	2	20.88	1	13.81
MAAOR ($\omega = 0.9;\ r = -4.0$)	1	17.84	2	20.89	2	14.25
MAAOR ($\omega = 1.3;\ r = 3.5$)	1	18.43	2	20.90	2	14.20
MAAOR ($\omega = 1.3;\ r = 5.0$)	1	17.81	2	20.89	1	13.82
MAAOR ($\omega = 1.5;\ r = 1.5$)	2	17.14	4	18.63	3	13.35
MAAOR ($\omega = 1.7;\ r = 3.5$)	1	17.81	2	20.90	2	14.21
MAAOR ($\omega = 1.7;\ r = 4.5$)	1	19.24	2	20.88	1	13.78
MAAOR ($\omega = 1.9;\ r = -5.0$)	1	17.84	2	20.89	2	14.17
MAAOR ($\omega = 1.9;\ r = -4.5$)	1	17.81	2	20.90	2	14.22
MAAOR ($\omega = 1.9;\ r = 4.0$)	1	18.47	2	20.88	2	14.22
MAAOR ($\omega = 1.9;\ r = 4.5$)	1	17.80	2	20.90	1	13.78

Table 4.15: wikipedia matrix: some of the best choice of parameters for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods ($tol < 10^{-8}$ and $\alpha = 0.85$).

4.3.5 Example 5 - test2 matrix

In this section we use a test matrix build-in by the authors. This matrix is proposed to deliver large sets of pages with equal PageRank. The matrix is a $10^5 \times 10^5$ matrix with 141000 nonzeros, 65% dangling nodes, 22% strong nondangling nodes and 13% weakly nondangling nodes.

It is also important to understand if these methods deal well with a web with large sets of pages with equal PageRank. Also, it is known that for increasing values of the damping factor α the computation of the PageRank computations brings additional difficulties to convergence. For that purpose we use a large generated matrix to mimic this behavior. Table 4.16 shows the number of iterations and time in seconds for the three approaches for $\alpha = 0.85$ and Table 4.17 for $\alpha = 0.95$.

Comparing tables 4.16 and 4.17 it can be seen that higher damping factor makes the problem more difficult to solve, but nonetheless, only two of the several variants tested for the three methods failed to converge (for the maximum number of iterations fixed). For this matrix, since it is sparser than the others, the number of floating-point operations is greatly reduced, in spite of the large dimension. The gains achieved with lumping versions are outstanding. Lumping2-MAAOR continues to offer interesting reduction in computation time with respect to Lumping1-MAAOR specially for large

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
Jacobi	31	0.127	28	0.059	31	0.059
Gauss-Seidel	17	0.092	16	0.041	17	0.050
SOR ($\omega = 0.5$)	59	0.196	53	0.079	59	0.075
SOR ($\omega = 1.5$)	37	0.144	48	0.074	38	0.062
Extrapolated Jacobi ($\omega = 0.5$)	70	0.243	64	0.091	71	0.082
Extrapolated Gauss-Seidel ($\omega = 1.5$)	33	0.143	36	0.057	33	0.064
Extrapolated Gauss-Seidel ($\omega = 0.5$)	44	0.174	41	0.067	45	0.070
AOR ($\omega = 1.5;\ r = 0.5$)	608	1.765	–	–	687	0.513
AOR ($\omega = 0.5;\ r = 1.5$)	38	0.156	37	0.066	40	0.063
AOR ($\omega = 0.5;\ r = 2$)	48	0.190	46	0.072	46	0.071
AOR ($\omega = 0.5;\ r = 5$)	1400	3.973	357	0.384	1405	0.990
GSOR	40	0.152	38	0.069	38	0.063
GAOR ($r = 0.5$)	46	0.176	44	0.066	45	0.066
GAOR ($r = 1.5$)	32	0.136	31	0.058	31	0.060
GAOR ($r = 0$)	51	0.181	49	0.077	50	0.071
GAOR ($r = 1$)	40	0.150	38	0.056	38	0.061
GAOR ($r = 3$)	36	0.152	33	0.054	33	0.060
MAAOR ($\omega = 1.5;\ r = 0.5$)	41	0.171	46	0.072	46	0.071
MAAOR ($\omega = 0.5;\ r = 1.5$)	73	0.263	71	0.100	71	0.087
MAAOR ($\omega = 0.8;\ r = 3$)	44	0.171	43	0.070	42	0.069

Table 4.16: test2 matrix: Number of iterations and convergence time for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods for $tol < 10^{-8}$ and $\alpha = 0.85$. Symbol – appears whenever the maximum number of iterations (5000) was exceeded.

values of the damping factor.

method	MAAOR		Lumping1-MAAOR		Lumping2-MAAOR	
	it.	time	it.	time	it.	time
Jacobi	45	0.169	44	0.070	39	0.063
Gauss-Seidel	24	0.111	25	0.050	22	0.051
SOR ($\omega = 0.5$)	80	0.277	79	0.106	65	0.081
SOR ($\omega = 1.5$)	39	0.154	54	0.080	39	0.063
Extrapolated Jacobi ($\omega = 0.5$)	98	0.328	97	0.120	77	0.089
Extrapolated Gauss-Seidel ($\omega = 1.5$)	34	0.156	39	0.070	33	0.060
Extrapolated Gauss-Seidel ($\omega = 0.5$)	57	0.222	59	0.089	52	0.075
AOR ($\omega = 1.5;\ r = 0.5$)	–	–	–	–	–	–
AOR ($\omega = 0.5;\ r = 1.5$)	49	0.206	46	0.076	48	0.074
AOR ($\omega = 0.5;\ r = 2$)	67	0.258	61	0.095	67	0.086
AOR ($\omega = 0.5;\ r = 5$)	–	–	–	–	–	–
GSOR	60	0.211	60	0.090	57	0.078
GAOR ($r = 0.5$)	69	0.267	69	0.101	65	0.087
GAOR ($r = 1.5$)	49	0.198	50	0.087	47	0.072
GAOR ($r = 0$)	78	0.253	78	0.104	73	0.088
GAOR ($r = 1$)	60	0.202	60	0.092	57	0.077
GAOR ($r = 3$)	47	0.179	47	0.078	47	0.071
MAAOR ($\omega = 1.5;\ r = 0.5$)	45	0.182	50	0.081	50	0.079
MAAOR ($\omega = 0.5;\ r = 1.5$)	107	0.368	109	0.151	103	0.120
MAAOR ($\omega = 0.8;\ r = 3$)	59	0.250	57	0.087	60	0.088

Table 4.17: test2 matrix: Number of iterations and convergence time for MAAOR, Lumping1-MAAOR and Lumping2-MAAOR methods for $tol < 10^{-8}$ and $\alpha = 0.95$. Symbol – appears whenever the maximum number of iterations (5000) was exceeded.

4.4 Conclusions

We have reviewed the MAAOR algorithm, which encloses the most well-known stationary iterative methods, as well as the Lumping methods to segregate dangling and nondangling nodes within the PageRank computations. Hybrid methods combining both were proposed: Lumping1-MAAOR and Lumping2-MAAOR. Numerical experiments illustrate the effectiveness of the two proposed methods. While the Lumping part allows for a clever computation on a smaller portion of the coefficients matrix, MAAOR provides a plethora of non-expensive numerical iterative linear solvers. This combination has reveled to be well adapted to the problem at hands as well as requiring low computational costs. Quite large problems were solved fast.

Before summing up, it is worth to mention that for large values of the damping factor α nonstationary iterative method, such as GMRES and BiCGstab, together with appropriate preconditioners, are a viable approach, since the greater computational cost to incur may be compensated by their effectiveness. Yet, these methods may converge slowly (or stagnate) for low dimensional search subspaces [138]. On the contrary, stationary iterative

methods are inexpensive but may show convergence problems for values of α closer to 1, which are not the case for PageRank. Nevertheless, results on Table 4.17 illustrate that these hybrid methods can cope with large values of the damping factor.

Moreover, other nonstationary methods can perform better than the Gauss-Seidel, assumed in the literature to be the best approach. With the results in this thesis, we explored for the first time the MAAOR family of methods, which proved to be superior both in number of iterations and computing time for more demanding problems.

5

Conclusions

The work of this thesis was focused on the study of numerical approaches to improve the convergence of the PageRank algorithm in its academic formulation. As PageRank is a numerical method for ordering web pages and it is the basis of the most used search engine success, it has been the target of scientific study over the past decade and a half.

In Chapter 2 we introduced the PageRank model and the mathematical issues regarding the PageRank problem. It was also presented the state of the art on the various acceleration methods of the Pagerank problem available in the literature. Additionally, a review of linear stationary iterative methods, on nonstationary methods used for PageRank computations, on multilinear PageRank and on parallel PageRank computations was included in this chapter. Nowadays, the problem of PageRank is more complex than its initial formulation, some of the new applications of the PageRank problem have been addressed.

The work was divided into two parts: the spectral path, which consists in the calculation of the PageRank vector by solving a problem of eigenvalues and eigenvectors, and the calculation of the PageRank vector through the solution of sparse linear systems of equations of large dimension.

In Chapter 3 we proposed a hybrid method that results from the combination of a classical acceleration numerical technique with a recent aggregation algorithm of page without outlinks. That is, a family of LumpingE methods, combining partitioning, matrix reduction and extrapolation, was proposed to accelerate PageRank computations. Numerical results illustrating the dynamics of the iterative process, number of iterations and CPU time were provided. We concluded that, despite the additional costs caused by the

preparation of the matrix, final retrieval of the PageRank vector and extrapolation, the new proposed family allows for a significant reduction both in number of iterations and CPU time required for convergence.

In Chapter 4 it was implemented a family of stationary methods that was recently developed and is quite comprehensive (Matrix Analogue of the Accelerated Overrelaxation (MAAOR) family of methods) with the aim of monitoring specifications of this family able to accelerate the calculation of PageRank. The MAAOR algorithm encloses the most well-known stationary iterative methods. In addition, a new method was developed combining the previously mentioned page aggregation algorithm (the Lumping methods to segregate dangling and nondangling nodes) with this new family of methods. Effectively, hybrid methods combining both were proposed: Lumping1-MAAOR and Lumping2-MAAOR. Numerical experiments illustrate the effectiveness of the two proposed methods. While the Lumping part allows for a clever computation on a smaller portion of the coefficients matrix, MAAOR provides a plethora of non-expensive numerical iterative linear solvers. This combination has reveled to be well adapted to the problem at hands as well as requiring low computational costs. Quite large problems were solved fast. Also we explored a huge number of possible combinations of the MAAOR parameters, delivering different iterative approaches, in order to select the (possibly) best MAAOR parametrization. We concluded that, contrary to most results in the literature, the Gauss-Seidel method can be superseded by the recent MAAOR family of methods in the PageRank problem.

The results indicate that there are advantages in the use of both proposed hybrid formulations.

Together with the presented developments, the contents of this thesis have been published in scientific journals and presented in international conferences.

Publications In particular, Chapters 3 and 4 gave rise to the following articles and proceedings:

- Mendes, I. R. and Vasconcelos, P. B., PageRank computations with MAAOR and Lumping Methods, Mathematics in Computer Science, June 2018, Volume 12, Issue 2, pp 129-141.
- Mendes, Isabel R. and Vasconcelos, Paulo B., PageRank computations using Lumping and extrapolation techniques, proceedings in 2nd International Conference on Numerical and Symbolic Computation: Developments and Applications, pp 225-244, 2015, ISBN 978-989-96264-7-8.

- Mendes, I. and Vasconcelos, P., Lumping method with acceleration for the PageRank computation, 14th International Conference on Computational Science and Its Applications (ICCSA 2014), pp 221-224, IEEE 2014.

Comunications

- International Conference on Numerical and Symbolic Computation Developments and Applications (SYMCOMP 2017), Guimarães, Universidade do Minho, $6-7$ April 2017, "PageRank computations with MAAOR and Lumping methods".

- 2nd International Conference on Numerical and Symbolic Computation Developments and Applications (SYMCOMP 2015), Faro, $26-27$ March 2015, "PageRank computations using Lumping and extrapolation techniques".

- 14th International Conference on Computational Science and Its Applications (ICCSA 2014), Guimarães, Portugal, 30 June to 3 July 2014, "Lumping with acceleration for PageRank computation".

- Congreso de Metodos Numericos en Ingenieria, Bilbao, $25-28$ June 2013, "Extrapolation for the PR computation on reordered matrices".

Bibliography

[1] P. Albrecht and M. Klein. Extrapolated Iterative Methods for Linear Systems Approximation. *SIAM journal on numerical analysis*, 21(1):192–201, 1984.

[2] J. Alpert and N. Hajaj. Official Google Blog: We knew the web was big, 2008.

[3] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[4] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. PageRank computation and the structure of the web: Experiments and algorithms. In *Proceedings of the Eleventh International World Wide Web Conference, Poster Track*, pages 107–117, 2002.

[5] G. Avdelas and A. Hadjidimos. Optimum accelerated overrelaxation method in a special case. *Mathematics of Computation*, 36(153):183–187, 1981.

[6] G. Avdelas, A. Hadjidimos, and A. Yeyios. Some theoretical and computational results concerning the accelerated overrelaxation (AOR) method. *L'Analyse Numérique et la Théorie de L'approximation*, 9(1):5–10, 1980.

[7] K. Avrachenkov, D. Nemirovsky, and K. S. Pham. A survey on distributed approaches to graph based reputation measures. In *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, page 82. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[8] O. Axelsson. *Iterative solution methods*. Cambridge university press, 1996.

[9] A. Bakushinskii and B. Polyak. Solution of variational inequalities. *Doklady Akademii Nauk SSSR*, 219(5):1038–1041, 1974.

[10] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*, volume 43. Siam, 1994.

[11] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics*, 182(2):418–477, 2002.

[12] P. Berkhin. A survey on pagerank computing. *Internet Mathematics*, 2(1):73–120, 2005.

[13] M. Bianchini, M. Gori, and F. Scarselli. Inside pagerank. *ACM Transactions on Internet Technology (TOIT)*, 5(1):92–128, 2005.

[14] P. Boldi, M. Santini, and S. Vigna. PageRank as a function of the damping factor. In *Proceedings of the 14th international conference on World Wide Web*, pages 557–566. ACM, 2005.

[15] S. Brain. Google annual search statistics, 2015.

[16] C. Brezinski and M. Redivo-Zaglia. The PageRank vector: Properties, computation, approximation, and acceleration. *SIAM Journal on Matrix Analysis and Applications*, 28(2):551–575, 2006.

[17] C. Brezinski and M. Redivo Zaglia. Generalizations of Aitken's process for accelerating the convergence of sequences. *Computational & Applied Mathematics*, 26(2):171–189, 2007.

[18] C. Brezinski and M. Redivo-Zaglia. Rational extrapolation for the PageRank vector. *Mathematics of Computation*, 77(263):1585–1598, 2008.

[19] C. Brezinski, M. Redivo-Zaglia, and S. Serra-Capizzano. Extrapolation methods for PageRank computations. *Comptes Rendus Mathematique*, 340(5):393–397, 2005.

[20] S. Brin and L. Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[21] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient PageRank approximation via graph aggregation. *Information Retrieval*, 9(2):123–138, 2006.

[22] Y.-M. Bu and T.-Z. Huang. An Adaptive Reordered Method for Computing PageRank. *Journal of Applied Mathematics*, 2013, 2013.

[23] G. Buehrer, S. Parthasarathy, and M. Goyder. Data mining on the cell broadband engine. In *Proceedings of the 22nd annual international conference on Supercomputing*, pages 26–35. ACM, 2008.

[24] S. Chien, C. Dwork, R. Kumar, and D. Sivakumar. Towards exploiting link evolution. 2001.

[25] P. Choudhari, E. Baikampadi, P. Patil, and S. Gadekar. Parallel and Improved PageRank Algorithm for GPU-CPU Collaborative Environment.

[26] M. Cleve. Numerical computing with MATLAB. *SIAM Philadelphia*, 2004.

[27] L. Cvetković and D. Herceg. Convergence theory for AOR method. *Journal of Computational Mathematics*, pages 128–134, 1990.

[28] M. Darvishi and P. Hessari. On convergence of the generalized AOR method for linear systems with diagonally dominant coefficient matrices. *Applied Mathematics and Computation*, 176(1):128–133, 2006.

[29] D. De Jager. *PageRank: Three distributed algorithms.* PhD thesis, Imperial College, 2004.

[30] M. de Kunder. WorldWideWebSize. com - The size of the World Wide Web (The Internet), 2015.

[31] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[32] M. Dehghan and M. Hajarian. Modied AOR iterative methods to solve linear systems. *Journal of Vibration and Control*, 20(5):661–669, 2014.

[33] G. Del Corso, A. Gulli, and F. Romani. Exploiting Web matrix permutations to speedup PageRank computation. *Informe técnico*, 2004.

[34] G. M. Del Corso, A. Gulli, and F. Romani. Fast PageRank computation via a sparse linear system. *Internet Mathematics*, 2(3):251–273, 2005.

[35] G. M. Del Corso, A. Gullí, and F. Romani. Comparison of Krylov subspace methods on the PageRank problem. *Journal of Computational and Applied Mathematics*, 210(1):159–166, 2007.

[36] N. T. Duong, Q. A. P. Nguyen, A. T. Nguyen, and H.-D. Nguyen. Parallel pagerank computation using gpus. In *Proceedings of the Third Symposium on Information and Communication Technology*, pages 223–230. ACM, 2012.

[37] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceedings of the 13th international conference on World Wide Web*, pages 309–318. ACM, 2004.

[38] L. Eldén. A note on the eigenvalues of the Google matrix. *arXiv preprint math/0401177*, 2004.

[39] D. Fiala, F. Rousselot, and K. Ježek. PageRank for bibliographic networks. *Scientometrics*, 76(1):135–158, 2008.

[40] S. P. Frankel. Convergence rates of iterative treatments of partial differential equations. *Mathematical Tables and Other Aids to Computation*, 4(30):65–75, 1950.

[41] V. Freschi. Protein function prediction from interaction networks using a random walk ranking algorithm. In *Bioinformatics and Bioengineering, 2007. BIBE 2007. Proceedings of the 7th IEEE International Conference on*, pages 42–48. IEEE, 2007.

[42] Z.-x. Gao and H.-b. Li. A note on the convergence of the generalized AOR iterative method for linear systems. *World Acad. Sci. Eng. Tech.*, 68:711–713, 2012.

[43] D. Gleich, L. Zhukov, and P. Berkhin. Fast parallel PageRank: A linear system approach. *Yahoo! Research Technical Report YRL-2004-038, available via http://research. yahoo. com/publication/YRL-2004-038. pdf*, 13:22, 2004.

[44] D. Gleich, L. Zhukov, and P. Berkhin. Scalable computing for power law graphs: Experience with parallel pagerank. In *Proc. SuperComputing*. Citeseer, 2005.

[45] D. F. Gleich. PageRank beyond the Web. *SIAM Review*, 57(3):321–363, 2015.

[46] D. F. Gleich, A. P. Gray, C. Greif, and T. Lau. An inner-outer iteration for computing pagerank. *SIAM Journal on Scientific Computing*, 32(1):349–371, 2010.

[47] D. F. Gleich, L.-H. Lim, and Y. Yu. Multilinear PageRank. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1507–1541, 2015.

[48] G. H. Golub and C. Greif. Arnoldi-type algorithms for computing stationary distribution vectors, with application to PageRank. Technical report, Technical Report SCCM-04-15, Stanford University Technical Report, 2004.

[49] G. H. Golub and C. Greif. An Arnoldi-type algorithm for computing page rank. *BIT Numerical Mathematics*, 46(4):759–771, 2006.

[50] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[51] M. Gori, A. Pucci, V. Roma, and I. Siena. ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In *IJCAI*, volume 7, pages 2766–2771, 2007.

[52] A. Y. Govan, C. D. Meyer, and R. Albright. Generalizing Google's PageRank to rank national football league teams. In *Proceedings of the SAS Global Forum*, volume 2008, 2008.

[53] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment, 2004.

[54] A. Hadjidimos. Accelerated overrelaxation method. *Mathematics of Computation*, 32(141):149–157, 1978.

[55] A. Hadjidimos. On the generalisation of the basic iterative methods for the solution of linear systems. *International journal of computer mathematics*, 14(3-4):355–369, 1983.

[56] A. Hadjidimos. A survey of the iterative methods for the solution of linear systems by extrapolation, relaxation and other techniques. *Journal of Computational and Applied Mathematics*, 20:37–51, 1987.

[57] A. Hadjidimos. Successive overrelaxation (SOR) and related methods. *Journal of Computational and Applied Mathematics*, 123(1):177–199, 2000.

[58] A. Hadjidimos. The matrix analogue of the scalar AOR iterative method. *Journal of Computational and Applied Mathematics*, 288:366–378, 2015.

[59] A. Hadjidimos, A. Psimarni, and A. Yeyios. On the convergence of some generalized iterative methods. *Linear algebra and its applications*, 75:117–132, 1986.

[60] A. Hadjidimos and M. Tzoumas. On the solution of the Linear Complementarity Problem by the Generalized Accelerated Overrelaxation iterative method. *Journal of Optimization Theory and Applications*, 165(2):545–562, 2015.

[61] A. Hadjidimos and A. Yeyios. The principle of extrapolation in connection with the accelerated overrelaxation method. *Linear Algebra and Its Applications*, 30:115–128, 1980.

[62] A. Hadjidimos and A. Yeyios. On some extensions of the accelerated overrelaxation (AOR) theory. *International Journal of Mathematics and Mathematical Sciences*, 5(1):49–60, 1982.

[63] T. Haveliwala and S. Kamvar. The second eigenvalue of the Google matrix. *Stanford University Technical Report*, 2003.

[64] T. Haveliwala, S. Kamvar, D. Klein, C. Manning, and G. Golub. Computing PageRank using power extrapolation. *Stanford University Technical Report*, 2003.

[65] I. C. Ipsen and S. Kirkland. Convergence analysis of an improved pagerank algorithm. *Center for Research in Scientific Computation*, 7(6):8, 2003.

[66] I. C. Ipsen and S. Kirkland. Convergence analysis of a PageRank updating algorithm by Langville and Meyer. *SIAM journal on matrix analysis and applications*, 27(4):952–967, 2006.

[67] I. C. Ipsen and T. M. Selee. PageRank computation, with special attention to dangling nodes. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1281–1296, 2007.

[68] I. C. Ipsen and R. S. Wills. Mathematical properties and analysis of Google's PageRank. *Bol. Soc. Esp. Mat. Apl*, 34:191–196, 2006.

[69] K. R. James. Convergence of matrix iterations subject to diagonal dominance. *SIAM Journal on Numerical Analysis*, 10(3):478–484, 1973.

[70] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM, 2003.

[71] K. Jezek, D. Fiala, and J. Steinberger. Exploration and evaluation of citation networks. In *ELPUB2008*, 2008.

[72] Z. Jia. Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems. *Linear algebra and its applications*, 259:1–23, 1997.

[73] Z. Jia and G. Stewart. An analysis of the Rayleigh–Ritz method for approximating eigenspaces. *Mathematics of Computation*, 70(234):637–647, 2001.

[74] B. Jiang, S. Zhao, and J. Yin. Self-organized natural roads for predicting traffic flow: a sensitivity study. *Journal of statistical mechanics: Theory and experiment*, 2008(07):P07008, 2008.

[75] S. Kamvar and T. Haveliwala. The condition number of the PageRank problem. 2003.

[76] S. Kamvar, T. Haveliwala, and G. Golub. Adaptive methods for the computation of PageRank. *Linear Algebra and its Applications*, 386:51–65, 2004.

[77] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. *Stanford University Technical Report*, 2003.

[78] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Extrapolation methods for accelerating PageRank computations. In *Proceedings of the 12th international conference on World Wide Web*, pages 261–270. ACM, 2003.

[79] C. Kohlschütter, P.-A. Chirita, and W. Nejdl. Efficient parallel computation of pagerank. In *Advances in information retrieval*, pages 241–252. Springer, 2006.

[80] G. Kollias, E. Gallopoulos, and D. B. Szyld. Asynchronous iterative computations with Web information retrieval structures: The PageRank case. *arXiv preprint cs/0606047*, 2006.

[81] T. Kumar, P. Sondhi, and A. Mittal. Parallelization of PageRank on multicore processors. In *Distributed Computing and Internet Technology*, pages 129–140. Springer, 2012.

[82] A. N. Langville and C. D. Meyer. Updating PageRank using the group inverse and stochastic complementation. *Informe técnico crsc02-tr32*, 2002.

[83] A. N. Langville and C. D. Meyer. Updating the stationary vector of an irreducible Markov chain. Technical report, Technical Report crsc02-tr33, NC State, Mathematics Dept., CRSC, 2002.

[84] A. N. Langville and C. D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

[85] A. N. Langville and C. D. Meyer. Updating pagerank with iterative aggregation. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 392–393. ACM, 2004.

[86] A. N. Langville and C. D. Meyer. A reordering for the PageRank problem. *SIAM Journal on Scientific Computing*, 27(6):2112–2120, 2006.

[87] A. N. Langville and C. D. Meyer. Updating Markov chains with an eye on Google's PageRank. *SIAM journal on matrix analysis and applications*, 27(4):968–987, 2006.

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

[89] A. N. Langville and C. D. Meyer. *Who's# 1?: the science of rating and ranking*. Princeton University Press, 2012.

[90] C. P. Lee, G. H. Golub, and S. A. Zenios. A two-stage algorithm for computing pagerank and multistage generalizations. *Internet Mathematics*, 4(4):299–327, 2007.

[91] C. P.-C. Lee, G. H. Golub, and S. A. Zenios. A fast two-stage algorithm for computing PageRank and its extensions. *Scientific Computation and Computational Mathematics*, 1(1):1–9, 2003.

[92] S. Levy. How Google's algorithm rules the web. *Wired Magazine*, 2010.

[93] Y. Lin, X. Shi, and Y. Wei. On computing PageRank via lumping the Google matrix. *Journal of Computational and Applied Mathematics*, 224(2):702–708, 2009.

[94] C. Liu and Y. Li. A Parallel PageRank Algorithm with Power Iteration Acceleration. *International Journal of Grid and Distributed Computing*, 8(2):273–284, 2015.

[95] X. Liu, J. Bollen, M. L. Nelson, and H. Van de Sompel. Co-authorship networks in the digital library research community. *Information processing & management*, 41(6):1462–1480, 2005.

[96] B. Manaskasemsak and A. Rungsawang. Parallel PageRank computation on a gigabit PC cluster. In *Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on*, volume 1, pages 273–277. IEEE, 2004.

[97] B. Manaskasemsak and A. Rungsawang. An efficient partition-based parallel PageRank algorithm. In *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, volume 1, pages 257–263. IEEE, 2005.

[98] M. Martins. An improvement for the area of convergence of the accelerated overrelaxation iterative method. *Anal. Numer. Theor. Approx*, 12(1):65–76, 1983.

[99] M. M. Martins. On an accelerated overrelaxation iterative method for linear systems with strictly diagonally dominant matrix. *Mathematics of Computation*, 35(152):1269–1273, 1980.

[100] M. M. Martins. Note on irreducible diagonally dominant matrices and the convergence of the AOR iterative method. *mathematics of computation*, 37(155):101–103, 1981.

[101] M. M. Martins. Generalized diagonal dominance in connection with the accelerated overrelaxation (AOR) method. *BIT Numerical Mathematics*, 22(1):73–78, 1982.

[102] I. Mendes and P. Vasconcelos. Lumping method with acceleration for the pagerank computation. In *Computational Science and Its Applications (ICCSA), 2014 14th International Conference on*, pages 221–224. IEEE, 2014.

[103] I. Mendes and P. Vasconcelos. Pagerank computation with maaor and lumping methods. *Mathematics in Computer Science*, 12(2):129–141, 2018.

[104] X. Meng. Computing BookRank via social cataloging. In *Web slides for CADS 2010 conference., February*, volume 22, 2009.

[105] C. D. Meyer. The character of a finite markov chain. In *Linear Algebra, Markov Chains, and Queueing Models*, pages 47–58. Springer, 1993.

[106] C. D. Meyer. *Matrix analysis and applied linear algebra*, volume 2. Siam, 2000.

[107] N. M. Missirlis and D. J. Evans. On the convergence of some generalized preconditioned iterative methods. *SIAM Journal on Numerical Analysis*, 18(4):591–596, 1981.

[108] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert. GeneRank: using search engine technology for the analysis of microarray experiments. *BMC bioinformatics*, 6(1):233, 2005.

[109] H. Nasabzadeh and F. Toutounian Mashhad. A new generalized AOR iterative method for solving linear systems. *International Journal of Applied Mathematics and Statistics*, 2, 2013.

[110] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 903–910. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

[111] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the web. 1999.

[112] F. Panjeh Ali Beik and N. Nasseri Shams. On the modified iterative methods for m-matrix linear systems. *Bulletin of the Iranian Mathematical Society*, 41(6):1519–1535, 2015.

[113] T. S. Papatheodorou. Block AOR iteration for nonsymmetric matrices. *mathematics of computation*, 41(164):511–525, 1983.

[114] B. T. Polyak and A. Timonina. PageRank: new regularizations and simulation models. In *World Congress*, volume 18, pages 11202–11207, 2011.

[115] B. T. Polyak and A. A. Tremba. Regularization-based solution of the PageRank problem for large matrices. *Automation and Remote Control*, 73(11):1877–1894, 2012.

[116] F. Radicchi. Who is the best player ever? A complex network analysis of the history of professional tennis. *PloS one*, 6(2):e17249, 2011.

[117] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210:307–357, 1911.

[118] A. Rungsawang and B. Manaskasemsak. Pagerank computation using PC cluster. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 152–159. Springer, 2003.

[119] Y. Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[120] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[121] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed pagerank for p2p systems. In *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*, pages 58–68. IEEE, 2003.

[122] P. Sargolzaei and F. Soleymani. PageRank problem, survey and future research directions. In *International Mathematical Forum*, volume 5, pages 937–956. Citeseer, 2010.

[123] S. Shi, J. Yu, G. Yang, and D. Wang. Distributed page ranking in structured p2p networks. In *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, pages 179–186. IEEE, 2003.

[124] A. Sidi. Vector extrapolation methods with applications to solution of large systems of equations and to PageRank computations. *Computers & Mathematics with Applications*, 56(1):1–24, 2008.

[125] Y. Song. On the convergence of the generalized AOR method. *Linear algebra and its applications*, 256:199–218, 1997.

[126] Y. Song. Semiconvergence of extrapolated iterative methods for singular linear systems. *Journal of computational and applied mathematics*, 106(1):117–129, 1999.

[127] W. J. Stewart. *Introduction to the numerical solutions of Markov chains*. Princeton Univ. Press, 1994.

[128] Y. Takahashi. A lumping method for numerical calculations of stationary distributions of Markov chains. *Research Reports on Information Sciences, Series B: Operations Research*, 1975.

[129] G.-X. Tian, T.-Z. Huang, and S.-Y. Cui. Convergence of generalized AOR iterative method for linear systems with strictly diagonally dominant matrices. *Journal of Computational and Applied Mathematics*, 213(1):240–247, 2008.

[130] R. S. Varga. *Matrix iterative analysis*, volume 27. Springer Science & Business Media, 2009.

[131] D. Walker, H. Xie, K.-K. Yan, and S. Maslov. Ranking scientific publications using a model of network traffic. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(06):P06010, 2007.

[132] G. Wang, H. Wen, and T. Wang. Convergence of GAOR iterative method with strictly α diagonally dominant matrices. *Journal of Applied Mathematics*, 2011, 2011.

[133] L. Wang and Y. Song. Preconditioned AOR iterative methods for m-matrices. *Journal of Computational and Applied Mathematics*, 226(1):114–124, 2009.

[134] Y. Wang and D. J. DeWitt. Computing pagerank in a distributed internet search system. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 420–431. VLDB Endowment, 2004.

[135] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.

[136] J. J. Whang, A. Lenharth, I. S. Dhillon, and K. Pingali. Scalable Data-Driven PageRank: Algorithms, System Issues, and Lessons Learned. In *Euro-Par 2015: Parallel Processing*, pages 438–450. Springer, 2015.

[137] R. S. Wills and I. C. Ipsen. Ordinal ranking for Google's PageRank. *SIAM Journal on Matrix Analysis and Applications*, 30(4):1677–1696, 2009.

[138] G. Wu, Y.-C. Wang, and X.-Q. Jin. A preconditioned and shifted gmres algorithm for the pagerank problem with multiple damping factors. *SIAM Journal on Scientific Computing*, 34(5):A2558–A2575, 2012.

[139] G. Wu and Y. Wei. A Power-Arnoldi algorithm for computing PageRank. *Numerical Linear Algebra with Applications*, 14(7):521, 2007.

[140] G. Wu and Y. Wei. An Arnoldi-extrapolation algorithm for computing PageRank. *Journal of Computational and Applied Mathematics*, 234(11):3196–3212, 2010.

[141] G. Wu and Y. Wei. Arnoldi versus GMRES for computing pageRank: A theoretical contribution to google's pageRank problem. *ACM Transactions on Information Systems (TOIS)*, 28(3):11, 2010.

[142] J. Wu and K. Aberer. Using siterank for p2p web retrieval. Technical report, 2004.

[143] M. Wu, L. Wang, and Y. Song. Preconditioned AOR iterative method for linear systems. *Applied Numerical Mathematics*, 57(5):672–685, 2007.

[144] D. Young. Iterative methods for solving partial difference equations of elliptic type. *Transactions of the American Mathematical Society*, 76(1):92–111, 1954.

[145] D. M. Young. *Iterative solution of large linear systems*. Elsevier, 2014.

[146] Q. Yu, Z. Miao, G. Wu, and Y. Wei. Lumping algorithms for computing Google's PageRank and its derivative, with attention to unreferenced nodes. *Information retrieval*, 15(6):503–526, 2012.

[147] H.-F. Zhang, T.-Z. Huang, C. Wen, and Z.-L. Shen. FOM accelerated by an extrapolation method for solving PageRank problems. *Journal of Computational and Applied Mathematics*, 296:397–409, 2016.

[148] X.-F. Zhang, Q.-F. Cui, and S.-L. Wu. Modified preconditioned GAOR methods for systems of linear equations. *Journal of Applied Mathematics*, 2013, 2013.

[149] Y. Zhu, S. Ye, and X. Li. Distributed PageRank computation based on iterative aggregation-disaggregation methods. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 578–585. ACM, 2005.