Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

CALCOLO DEL PAGERANK: UNA FORMULAZIONE BASATA SULLA RISOLUZIONE DI SISTEMI LINEARI

Relatrice:
Chiar.ma Prof.ssa
Valeria Simoncini

Presentata da:
Elisa Merelli

VI Sessione
Anno Accademico 2019/2020

A mia madre,
a cui ogni sacrificio e ogni traguardo sono dedicati.

Indice

Introduzione

L'algoritmo di PageRank nasce dalla necessità di creare una teoria che permetta di cercare informazioni in una biblioteca che conta miliardi di documenti e, in pochi istanti, trovare la migliore corrispondenza con le richieste dell'utente. Tale necessità trova risposta nel pensiero rivoluzionario dei co-fondatori di Google, Larry Page e Sergey Brin, la cui idea chiave è assegnare a ciascuna pagina un'importanza pari alla somma pesata delle importanze delle pagine che la puntano; in questo modo, l'interesse di ogni utente può essere reso oggettivo e calcolabile.
Tradizionalmente, il problema dell'individuazione del vettore di PageRank è risolto attraverso la ricerca dell'autovettore associato all'autovalore dominante di una certa matrice che rappresenta la struttura Web: in quest'ottica, la prima parte dell'elaborato presenta la modellizzazione matematica del Web e il metodo canonico utilizzato per la ricerca del vettore di PageRank.
In seguito, l'interesse sarà quello di riformulare tale problema in un modo equivalente che chiederà la risoluzione di un sistema lineare. Tale approccio ci permetterà di testare e confrontare diverse strategie risolutive in un caso reale, considerando la rete di links delle Università Stanford-Berkeley. Dunque, con l'ausilio di esperimenti numerici, concluderemo la trattazione evidenziando pregi e criticità dei metodi descritti per risoluzione della classe di problemi legati al PageRank.

Premesse

Introduciamo qui alcune nozioni che saranno utilizzate nel seguito dell'elaborato.

Definizione (Grafo)**.** Un *grafo* è una coppia ordinata $G = (V, E)$, con V insieme dei nodi e E insieme degli archi tale che $E \subseteq V \times V$. Se gli elementi di E sono coppie ordinate, allora si dice che il grafo G è *orientato* (o diretto), altrimenti si dice che è *non orientato* (o indiretto).

Definizione (Matrice di adiacenza)**.** Dato un grafo G con n nodi, una sua *matrice di adiacenza* è una matrice $A \in \mathbb{R}^{n \times n}$ i cui elementi $(a_{i,j})$ sono non nulli solo se il grafo G ha almeno un arco che connette il nodo i al nodo j.

Definizione (Grafo di adiacenza)**.** Data una matrice $A \in \mathbb{R}^{n \times n}$, il suo *grafo di adiacenza* è un grafo G con n nodi, che presenta un arco dal nodo i al nodo j solo se l'elemento $(a_{i,j})$ di A è non nullo.

Definizione (Matrice stocastica per colonna)**.** Una matrice $A = (a_{i,j}) \in \mathbb{R}^{n \times n}$ si dice *stocastica per colonna* se i suoi elementi sono tutti non negativi, e se la somma degli elementi su ogni colonna è 1.

Notazioni

- Sia $A \in \mathbb{R}^{n \times n}$ una matrice, indicheremo con I la matrice identità di ordine n e con $\mathbf{1} = (1, 1, \ldots, 1)$ il vettore unitario di dimensione n;
- siano $v, w \in \mathbb{R}^n$ due vettori, indicheremo con $\langle v, w \rangle$ il prodotto scalare dei due vettori.

Capitolo 1

Calcolo del vettore di PageRank

1.1 Vettore di PageRank

Definizione 1.1. Il *vettore di PageRank* è il vettore le cui componenti indicano le importanze delle n pagine del Web:

$$\boldsymbol{\mathcal{I}} = (\mathcal{I}(P_1), \mathcal{I}(P_2), \ldots, \mathcal{I}(P_n)). \tag{1.1}$$

dove $P_1, P_2, \ldots, P_n$ indicano le pagine.

I creatori dell'algoritmo PageRank di Google, Larry Page e Sergey Brin, lavorarono ad una strategia per assegnare a ciascuna pagina Web un'importanza relativa alla richiesta dell'utente, e nel 1996 svilupparono così l'idea chiave dell'algoritmo:

> "L'importanza di una pagina è data dalla somma pesata delle importanze delle pagine che la puntano"

Osservazione 1.1. E' interessante notare che l'importanza di una pagina, definita come appena visto, non è manipolabile o corruttibile: infatti, se avessimo definito l'importanza di una pagina unicamente in base al numero di pagine che la puntano, sarebbe bastato creare innumerevoli pagine fittizie che puntassero alla nostra pagina per conferirle importanza.

Dunque l'importanza della pagina P_j è:

$$\mathcal{I}(P_j) = \frac{1}{m_1}\mathcal{I}(P_1) + \frac{1}{m_2}\mathcal{I}(P_2) + \cdots + \frac{1}{m_s}\mathcal{I}(P_s) = \sum_{i \in K_j} \frac{1}{m_i}\mathcal{I}(P_i) \tag{1.2}$$

dove $P_1, \ldots, P_s$ sono le pagine che puntano a P_j, e $m_1, \ldots, m_s$ sono i rispettivi numeri di outlinks (links uscenti). Equivalentemente, possiamo definire K_j come l'insieme degli inlinks (links entranti) di P_j, e m_i come il numero di outlinks di P_i.

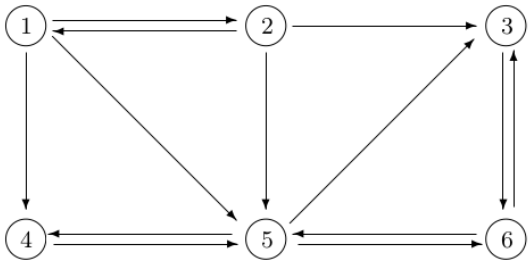
Per poter lavorare efficacemente con questo concetto è utile raffigurare il Web utilizzando un grafo orientato i cui nodi rappresentano le pagine del Web e gli archi orientati descrivono le connessioni di tali pagine. Tale approccio ci permette di considerare la matrice di adiacenza associata al grafo Web:

Definizione 1.2. La *matrice di Hyperlink* $H = (h_{ij}) \in \mathbb{R}^{n \times n}$ i cui elementi sono definiti come segue:

$$h_{i,j} = \begin{cases} \frac{1}{m_j} & \textit{se } P_j \textit{ punta a } P_i \\ 0 & \textit{altrimenti} \end{cases} \tag{1.3}$$

Osservazione 1.2. Si osservi che la matrice H così definita risulta stocastica per colonna ed estremamente sparsa.

Esempio 1.1. Consideriamo una famiglia di sei pagine web connesse tra loro come mostrato dagli archi del grafo:



La matrice di Hyperlink associata a tale grafo è la seguente:

$$H = \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{1}{3} & 0 \end{bmatrix}$$

Osservazione 1.3. Calcolare (1.2) equivale ad eseguire il prodotto scalare tra la riga i-esima di H e il vettore $\boldsymbol{\mathcal{I}} = (\mathcal{I}(P_1), \mathcal{I}(P_2), \ldots, \mathcal{I}(P_n))^T$.
Quindi se indichiamo con $\boldsymbol{\mathcal{I}} = (\mathcal{I}(P_i))_{i=1,\ldots,n}^T$ il vettore colonna le cui componenti sono i PageRanks delle pagine, otteniamo la seguente importante relazione:

$$\boldsymbol{\mathcal{I}} = H\boldsymbol{\mathcal{I}} \tag{1.4}$$

ossia ci riconduciamo ad un problema agli autovalori con autovalore $\lambda = 1$ e autovettore $\boldsymbol{\mathcal{I}}$. Dunque la determinazione del vettore di PageRank si concretizza tramite la risoluzione di tale problema.

1.2 Modifiche alla matrice di Hyperlink

Abbiamo visto che il vettore di PageRank risulta essere l'autovettore associato all'autovalore unitario della matrice di Hyperlink H, ma tale problema potrebbe non essere ben definito in quanto H potrebbe non ammettere autovalore $\lambda = 1$ oppure quest'ultimo potrebbe non essere un autovalore semplice, perciò si rivela necessario apportare alla matrice delle modifiche che garantiscano la buona posizione del problema.

Un primo intervento da fare è quello volto ad escludere la presenza di pagine senza outlinks, ossia pagine nelle quali non si riesca a passare ad un'altra pagina: questa situazione corrisponde infatti ad una colonna di elementi nulli

nella matrice di Hyperlink. In tal caso, la matrice non risulterebbe più stocastica per colonna. Per ovviare al problema si decide di sostituire la colonna di zeri corrispondente alla pagina senza outlinks, con una colonna costante. Dunque definiamo una nuova matrice:

$$Q = H + \frac{1}{n}\,\mathbf{1}\,d^T \tag{1.5}$$

dove $\mathbf{1} = (1, \ldots, 1)$, n è il numero di pagine e $d \in \mathbb{R}^n$ è un vettore, detto *dangling indicator*, usato per contrassegnare le pagine senza outlinks:

$$d_j = \begin{cases} 1 & m_j = 0 \\ 0 & altrimenti \end{cases} \qquad \forall j = 1, \ldots, n.$$

Osservazione 1.4. E' doveroso notare che nel grafo Web, per come lo abbiamo definito, l'importanza di ciascun nodo è data dalla probabilità che un camminatore aleatorio si trovi su tale nodo, ad un tempo sufficientemente grande. Determinare il vettore di PageRank equivale quindi a calcolare la distribuzione di probabilità associata alla catena di Markov indotta da un cammino casuale sul grafo Web, in cui la probabilità di passare da un nodo all'altro è quella uniforme secondo i cammini uscenti. In quest'ottica, la modifica appena operata fa sì che il navigatore aleatorio che arriva in una pagina senza outlinks non rimanga bloccato, ma abbia uguale probabilità di passare a qualsiasi altra pagina.

Per come è stata definita, la matrice Q risulta sempre stocastica per colonna, quindi soddisfa la relazione $Q^T\,\mathbf{1} = \mathbf{1}$. Ciò garantisce che Q ammetta sempre autovalore $\lambda = 1$ corrispondente ad un autovettore unitario.
Il prossimo passo è assicurare che l'autovalore $\lambda = 1$ sia semplice, cosicché $\boldsymbol{\mathcal{I}}$ sia l'unico autovettore associato. Questo risultato sarà garantito dal Teorema di Perron-Frobenius, al quale premettiamo due definizioni e una proposizione ausiliaria:

Definizione 1.3. Una matrice A si dice *riducibile* se esiste una matrice di permutazione G tale che:

$$G\,A\,G^T = \begin{pmatrix} X & Y \\ 0 & Z \end{pmatrix}$$

dove le matrici X, Z sono quadrate. Una matrice si dice *irriducibile* se non è riducibile.

Definizione 1.4. Un grafo diretto si dice *fortemente connesso* se per ogni coppia di vertici (P_i, P_j) esiste un percorso nel grafo che li collega.

Proposizione 1.1. *Una matrice è irriducibile se e solo se il grafo di adiacenza ad essa associato è fortemente connesso.*

Dimostrazione. Dimostriamo che una matrice è riducibile se e solo se il suo grafo diretto non è fortemente connesso. Osserviamo intanto che il grafo non varia, se permuto gli elementi della matrice. Supponiamo che la matrice sia riducibile, allora per definizione, è possibile portarla nella forma:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \in \mathbb{R}^{n\times n}$$

con A_{11}, A_{22} quadrate. Sia m la dimensione del blocco A_{11}; partendo dai nodi del grafo con indice compreso tra $m+1$ e n, non è possibile raggiungere i nodi indicizzati da 1 a n, infatti gli elementi della matrice A che corrisponderebbero agli archi, $(a_{i,j})_{m+1\leq i\leq n,\, 1\leq j\leq m}$, sono nulli. Dunque il grafo non è fortemente connesso.
Viceversa, supponiamo che il grafo non sia fortemente connesso, dunque esisterà almeno una coppia (a, b) di nodi non connessi. Sia R l'insieme dei nodi raggiungibili da a e N l'insieme dei nodi non raggiungibili da a; si noti che nessun nodo di N è raggiungibile da un nodo di R (altrimenti avremmo trovato un collegamento tra a e b). Dunque possiamo concludere disponendo gli elementi della matrice associata, in modo che gli indici degli elementi di N precedano quelli di R, ottenendo così una matrice nella forma desiderata. □

Teorema 1.2 (Perron-Frobenius)**.** *Sia $A \in \mathbb{R}^{n\times n}$ una matrice irriducibile e stocastica per colonna. Allora l'autovalore dominante λ_1 è uguale a 1 e, inoltre,*

esiste un unico autovettore associato ad esso con componenti tutte positive (unico autovettore non negativo). Infine se A ha tutti elementi positivi allora $|\lambda_i| < 1,\ i = 2, \ldots, n.$

Il teorema garantisce quindi l'esistenza e l'unicità della soluzione del problema agli autovalori; tuttavia, considerate le grandi dimensioni della matrice Q di Hyperlink con cui stiamo lavorando, essa risulterà quasi certamente riducibile, dunque è necessario apportare un'ultima modifica che assicuri l'irriducibilità. Definiamo quindi una nuova matrice:

$$P = \alpha\, Q + (1 - \alpha)\frac{1}{n}\, \mathbf{1}\, \mathbf{1}^T \tag{1.6}$$

dove $\alpha \in [0, 1]$. Con tale modifica la matrice Q viene scalata di un valore α. Inoltre, aggiungendo $\frac{(1-\alpha)}{n}$ a tutti i termini, viene imposto che il camminatore aleatorio ad ogni passo possa saltare da una pagina all'altra con probabilità $(1 - \alpha)$: per questo motivo, α è detto *coefficiente di teletrasporto*.

Osservazione 1.5. Una modifica alternativa è quella che prevede l'introduzione di un vettore w detto *vettore di personalizzazione* volto ad influenzare l'importanza delle pagine, e dunque ad orientare la ricerca favorendo alcuni risultati. In questo caso la matrice P diventa:

$$\tilde{P} = \alpha\, Q + (1 - \alpha)\, w\, \mathbf{1}^T \tag{1.7}$$

Senza altre informazioni si sceglie di mantenere la distribuzione uniforme dei collegamenti tra le pagine e quindi di definire $w = \frac{1}{n}\mathbf{1}$.

La matrice P ora soddisfa tutte le ipotesi del teorema di Perron-Forbenius, che quindi assicura che il vettore di PageRank desiderato corrisponde all'unico autovettore associato all'autovalore dominante della matrice P.

1.3 Metodo delle potenze

Il metodo delle potenze è il criterio tradizionalmente adottato per risolvere il problema del calcolo del vettore di PageRank. Questa scelta è motivata

da diversi fattori: in prima istanza, vi è certamente l'elevata dimensione del problema, infatti in queste circostanze, i metodi diretti per il calcolo degli autovettori sarebbero eccessivamente costosi e quindi inapplicabili. In secondo luogo, il metodo delle potenze prevede la memorizzazione del solo vettore che convergerà alla soluzione, che è la situazione ottimale, dato che siamo interessati a calcolare unicamente l'autovettore associato all'autovalore dominante, e noto, della matrice di Hyperlink modificata. In questa sede mostriamo brevemente il metodo delle potenze classico, con il suo risultato principale, e successivamente presentiamo la variante che abbiamo utilizzato per il calcolo del vettore di PageRank.
Sia $A \in \mathbb{R}^{n\times n}$, $x^{(0)}$ un vettore iniziale di norma unitaria fissato, l'iterazione del metodo delle potenze è la seguente:

Algorithm 1 Metodo delle potenze

1: **for** $k = 0, 1, 2, \ldots,$ fino a convergenza, **do**
2: $\quad y = A\, x^{(k)}$
3: $\quad x^{(k+1)} = y \,/\, \|y\|$
4: $\quad \lambda^{(k+1)} = (x^{(k+1)})^H A\, x^{(k+1)}$
5: **end for**

dove $(x^{(k+1)})^H$ indica il vettore trasposto coniugato e $\lambda^{(k+1)}$ è calcolato applicando il quoziente di Rayleigh $\frac{x^H Ax}{x^H x}$.

Indicata con (λ_1, x_1) l'autocoppia dominante di A, il metodo delle potenze costruisce due successioni $\{x^{(k)}\}_{k\in\mathbb{N}}$ e $\{\lambda^{(k)}\}_{k\in\mathbb{N}}$ tali che: $x^{(k)} \to x_1$ e $\lambda^{(k)} \to \lambda_1$, per $k \to +\infty$.

Osservazione 1.6. Affinché il metodo converga, si richiede che λ_1 sia un autovalore semplice in modulo.

Il seguente importante teorema fornisce una stima della velocità di convergenza del metodo delle potenze:

Teorema 1.3. *Data una matrice $A \in \mathbb{R}^{n\times n}$ diagonalizzabile, $A = X\Lambda X^{-1}$, con $X = [x_1, \ldots, x_n]$ matrice avente come colonne gli autovettori di A, fissato*

l'iterato iniziale $x^{(0)}$, siano $|\lambda_i|$, per $i = 1, \ldots, n$, gli autovalori della matrice A ordinati in modo decrescente, e sia $|\lambda_1| > |\lambda_2|$, con $|\lambda_1|$ autovalore semplice. Se $X^{-1}x^{(0)} \neq 0$ allora esiste una costante $C > 0$ tale che:

$$\|x^{(i)} - x_1\|_2 \leq C \left|\frac{\lambda_2}{\lambda_1}\right|^i, \qquad i \geq 1. \tag{1.8}$$

Il risultato appena enunciato assicura che la convergenza della successione $\{x^{(k)}\}_{k\in\mathbb{N}}$ sia lineare rispetto al rapporto $|\lambda_1/\lambda_2|$. Di conseguenza, maggiore sarà la distanza dell'autovalore λ_1 dal resto dello spettro di A, maggiore sarà la velocità di convergenza del metodo delle potenze.

Passiamo adesso alla variante del metodo delle potenze pensata per il problema della determinazione del vettore di PageRank; enunciamo subito un risultato che dà informazioni sul legame tra lo spettro della matrice di Hyperlink H iniziale, e lo spettro della sua modifica P definita in (1.6):

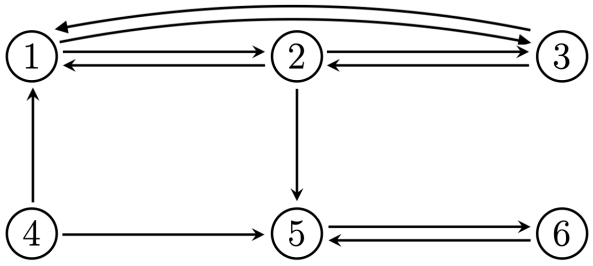
Proposizione 1.4. *Sia $spec(H) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ lo spettro della matrice H, e sia α il coefficiente di teletrasporto fissato, allora lo spettro della matrice modificata P è:*

$$spec(P) = \{1, \alpha\lambda_2, \ldots, \alpha\lambda_n\}. \tag{1.9}$$

In particolare, si ha che se la matrice H ha autovalore 1 multiplo, la matrice P ha autovalore 1 semplice.

Esempio 1.2. Consideriamo il seguente grafo non fortemente connesso e la sua matrice H di Hyperlink associata:

$$H = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



La matrice H risulta stocastica per colonna, ma è riducibile: quindi operiamo le modifiche descritte nella sezione precedente e costruiamo la matrice P. Fissiamo poi il coefficiente $\alpha = 0.85$, e studiamo gli autovalori delle matrici H e P, e gli autovettori di P, con il seguente codice MATLAB:

```
l_H = eig(H)';
e = ones(6,1);
alpha = 0.85;
P = alpha*H + (1-alpha)/6*e*e';
[L,l_P] = eig(P);
```

otteniamo i seguenti risultati:

```
Autovalori di H:
l_H =
   -0.5000    1.0000   -0.5000    1.0000   -1.0000         0
Autovalori di P:
l_P =
    1.0000    0.8500    0.0000   -0.8500   -0.4250   -0.4250


0.4468   -0.3651   -0.3536    0.0000    0.8165    0.1720
0.4297   -0.3651    0.3536    0.0000   -0.4082    0.6052
0.4297   -0.3651    0.3536   -0.0000   -0.4082   -0.7772
0.0572    0.0000   -0.7071    0.0000   -0.0000   -0.0000
0.4690    0.5477    0.0000    0.7071   -0.0000   -0.0000
0.4559    0.5477    0.3536   -0.7071    0.0000    0.0000
```

Si osservi che come previsto dai risultati teorici, anche se la matrice H presentava autovalore 1 multiplo, la matrice P ammette autovalore 1 semplice, e

il secondo autovalore è pari al coefficiente α fissato. Inoltre, come ci aspettavamo, l'autovettore associato all'autovalore 1 di P è l'unico autovettore con tutti elementi non negativi.

Osservazione 1.7. Osserviamo anche che per quanto abbiamo visto nel Teorema 1.3, la scelta del coefficiente α incide fortemente sulla velocità di convergenza del metodo infatti abbiamo che:

$$\frac{|\lambda_2(P)|}{\lambda_1(P)} = |\lambda_2(P)| \leq \alpha.$$

Dunque valori piccoli di α daranno una convergenza più veloce, mentre per $\alpha \approx 1$ si darà maggiore rilevanza alla struttura Web descritta da H.

Concludiamo il capitolo presentando lo pseudo-codice della variante del metodo delle potenze, utilizzato negli esperimenti.
Fissato il vettore iniziale $x^{(0)} \in \mathbb{R}^n$ di norma unitaria, il coefficiente di teletrasporto α e il vettore di personalizzazione w, l'iterazione della variante del metodo delle potenze per il calcolo del PageRank è la seguente:

Algorithm 2 Metodo delle potenze per il calcolo del PageRank

1: **for** $k = 0, 1, 2, \ldots$ fino a convergenza, **do**
2: $\quad y^{(k+1)} = \alpha\, P\, x^{(k)}$,
3: $\quad \delta^{(k+1)} = 1 - \|y^{(k+1)}\|_1$
4: $\quad y^{(k+1)} = y^{(k+1)} + \delta^{(k+1)}\, w$
5: $\quad x^{(k+1)} = \frac{y^{(k+1)}}{\|y^{(k+1)}\|_1}$
6: **end for**

Si osservi che ad ogni passo il vettore $x^{(k+1)}$ viene normalizzato: questo perché in aritmetica finita possono verificarsi numerosi arrotondamenti che incidono sull'accuratezza della soluzione.

Osservazione 1.8. Notiamo che nella versione appena proposta del metodo delle potenze non compare il calcolo di $\lambda^{(k+1)} = (x^{(k+1)})^H A\, x^{(k+1)}$ presente nell'algoritmo 1. Infatti, nel caso della matrice di Hyperlink, l'autovalore dominante $\lambda_1 = 1$ è già noto, dunque non occorre creare la successione $\{\lambda^{(k)}\}_{k\in\mathbb{N}}$.

Il criterio d'arresto si basa sul valore del residuo scalare al passo k, quindi:

$$res^{(k+1)} = \|y^{(k+1)} - x^{(k)}\|_1.$$

questo perché una valutazione del tipo $\left|1 - \frac{x_k^* P x_k}{x_k^* x_k}\right|$ sarebbe risultata troppo costosa.

Osservazione 1.9. Per la valutazione del residuo, la scelta di utilizzare la norma-1, anziché la norma-2, è motivata dal costo computazionale. Infatti, in norma-1 una valutazione del residuo richiede $2n$ flops, mentre in norma-2 costerebbe $3n+1$ flops, e poiché stiamo considerando valori di n molto elevati, la differenza è considerevole.

Capitolo 2

PageRank come sistema lineare

In questo capitolo siamo interessati a riformulare il problema del calcolo del vettore di PageRank in termini della risoluzione sistema lineare, e a presentare alcuni metodi per la soluzione di tale problema.

2.1 Formulazione alternativa

L'idea di riformulare il problema del PageRank fu presentata da Arvind Arasu [5] e successivamente approfondita da David Gleich [1], e prevede di ricondurre la determinazione del vettore di PageRank alla risoluzione di un sistema lineare.
Nel capitolo precedente abbiamo introdotto le seguenti quantità:

1) $Q = H + w\, d^T$ definita in (1.5), con $w = \frac{1}{n}\mathbf{1}$ e d vettore *dangling indicator*;

2) $P = \alpha\, Q + (1-\alpha)\, w\, \mathbf{1}^T$ definita in (1.7);

3) $x^{(i+1)} = Px^{(i)}$: i-esima iterazione del metodo delle potenze vista nel paragrafo 1.3.

Vediamo come combinarle per arrivare alla formulazione desiderata. Partiamo da

$$x^{(i+1)} = Px^{(i)}$$

in cui sostituiamo 2) e otteniamo la seguente:

$$x^{(i+1)} = [\alpha\, Q + (1-\alpha)\, w\, \mathbf{1}^T]\, x^{(i)};$$

scriviamo esplicitamente anche la 1) e arriviamo a:

$$x^{(i+1)} = [\alpha\, (H + w\, d^T) + (1-\alpha)\, w\, \mathbf{1}^T]\, x^{(i)}$$

e dunque alla seguente:

$$x^{(i+1)} = [\alpha H + \alpha\, w\, d^T + (1-\alpha)\, w\, \mathbf{1}^T]\, x^{(i)}. \tag{2.1}$$

Proposizione 2.1. *Valgono le seguenti:*

i) $\mathbf{1}^T x = x^T \mathbf{1} = \|x\|_1$*;*

ii) $d^T x = \|x\| - \|H\, x\|$.

Dimostrazione. La prima relazione segue direttamente dalla definizione di norma-1. Dimostriamo la seconda identità: in (2.1) abbiamo visto che vale:

$$[\alpha H + \alpha\, w\, d^T + (1-\alpha)\, w\, \mathbf{1}^T]\, x = x$$

moltiplichiamo entrambi i membri per $\mathbf{1}^T$ e svolgiamo il prodotto al primo membro, si ottiene:

$$\mathbf{1}^T \alpha\, H\, x + \mathbf{1}^T \alpha\, (w\, d^T)\, x + \mathbf{1}^T (1-\alpha)(w\, \mathbf{1}^T)\, x = \mathbf{1}^T x$$

sfruttiamo la i), ricordando che $\|w\| = 1$:

$$\alpha \|H\, x\| + \alpha\, d^T x + \|x\| - \alpha \|x\| = \|x\|$$

La ii) segue semplificando $\|x\|$ e dividendo per α. □

Ora siamo pronti a concludere la riformulazione: riprendiamo la (2.1) come all'inizio della dimostrazione, e svolgiamo il prodotto per x al primo membro, avremo:

$$\alpha\, H\, x + \alpha\, d^T x\, w + \mathbf{1}^T x\, w - \alpha\, \mathbf{1}^T x\, w = x$$

grazie a i) della Proposizione, otteniamo la seguente:

$$\alpha\, d^T x\, w + \|x\|\, w - \alpha\, \|x\|\, w = x - \alpha\, H\, x$$

da cui segue:

$$\|x\|\, w - \alpha\, (\|x\| - d^T x)\, w = x - \alpha\, H\, x$$

ma per il punto ii) della Proposizione precedente si ha che:

$$\|x\|\, w - \alpha\, \|H\, x\|\, w = x - \alpha\, H\, x$$

da cui:

$$(I - \alpha\, H)\, x = k\, w \tag{2.2}$$

dove $k = \|x\| - \alpha\, \|H\, x\|$.

Osservazione 2.1. Notiamo che $k = k(x)$ è in funzione di x, ma essendo uno scalare non inferisce nella risoluzione del problema del PageRank, infatti k risulta solo un *rescaling* di x che può essere normalizzato per diventare una distribuzione di probabilità.

Con questa precisazione, possiamo concludere che, fissato un valore di $k \in \mathbb{R}$, la (2.2) costituisce un sistema lineare del tipo:

$$A\, x = b \tag{2.3}$$

con $A = I - \alpha\, H$ e $b = k\, w$.

2.2 Metodi iterativi per il PageRank

La matrice A che abbiamo individuato nella sezione precedente risulta essere di dimensioni molto elevate, sparsa e non simmetrica quindi ricercare le soluzioni del sistema (2.3) attraverso metodi diretti potrebbe non essere efficace. Dunque vediamo alcuni metodi iterativi che si prestano al problema.

2.2.1 Metodo di Jacobi

Tra i metodi iterativi stazionari consideriamo il metodo di Jacobi, e vediamo intanto un risultato teorico importante:

Teorema 2.2. *Sia $A \in \mathbb{R}^{n\times n}$ una matrice a dominanza diagonale stretta per colonne, cioè*

$$|A_{ii}| > \sum_{i=1}^{n} |A_{ij}|, \qquad i \neq j,\ j = 1, \ldots, n$$

allora il metodo di Jacobi converge.

Osservazione 2.2. Per come è stata costruita la matrice di Hyperlink H, l'ipotesi del teorema è sempre verificata, perciò questo risultato assicura la convergenza del metodo. Tuttavia quest'ultima sarà fortemente influenzata dal valore che assume il coefficiente di teletrasporto: infatti per valori di α vicini ad 1, la convergenza del metodo risulterà più lenta.

Presentiamo l'iterazione di Jacobi utilizzata per la ricerca del vettore di PageRank: siano $H \in \mathbb{R}^{n\times n}$ la matrice di Hyperlink, $\alpha \in (0, 1)$ il coefficiente di teletrasporto, $x_0 = \frac{1}{n}\mathbf{1}$ il vettore iniziale, $b = (1-\alpha)x_0$ il vettore dei coefficienti, $r_0 = b - x_0 + \alpha H x_0$ il residuo iniziale, e definiamo $D = \mathbf{1} - \alpha * diag(H)$.

Algorithm 3 Metodo di Jacobi

1: **for** $k = 0, 1, 2, \ldots,$ fino a convergenza, **do**
2: $\quad x_{k+1} = x_k + D^{-1} r_k;$
3: $\quad r_{k+1} = b - x_{k+1} + \alpha(H x_{k+1});$
4: **end for**

Osservazione 2.3. Per il calcolo di x_{k+1} non viene calcolata esplicitamente l'inversa di D, poiché risulterebbe costoso. In MATLAB si sceglie di definire un vettore `d = diag(D)` e di effettuare la divisione componente per componente con il vettore del residuo, tramite la seguente implementazione: `x = x + r./d`.

Concludiamo facendo una considerazione circa il costo computazionale del metodo: si richiedono $2n$ flops per il calcolo di x_{k+1}; $2n$ flops, a cui va aggiunto

il costo dell'operazione matrice-per-vettore, per il calcolo di r_{k+1}, e $2n$ flops per calcolare la norma del residuo per il criterio d'arresto, per un totale di $6n$ flops più il costo matrice-per-vettore.

2.2.2 Metodo GMRES

Il metodo GMRES *(Generalized Minimal Residual)* è un metodo iterativo non stazionario per la ricerca della soluzione numerica di un sistema lineare non simmetrico.
Data una matrice $A \in \mathbb{R}^{n\times n}$ invertibile, un vettore $b \in \mathbb{R}^n$ normalizzato e un vettore iniziale $x_0 \neq 0$, definiamo il j-esimo sottospazio di Krylov come:

$$K_j = K_j(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{j-1}r_0\}$$

dove $r_0 = b - Ax_0$ è l'errore iniziale.
Il metodo GMRES approssima la soluzione del problema $Ax = b$ con il vettore $x_j \in K_j$ che minimizza la norma euclidea del residuo $r_j = Ax_j - b$. Tuttavia, la base formata dai vettori $r_0, Ar_0, A^2r_0, \ldots, A^{j-1}r_0$ è numericamente instabile in quanto al crescere di j, i vettori tendono a diventare linearmente dipendenti, perciò per lo spazio K_j si sceglie di usare una base di vettori ortonormali $q_1, q_2, \ldots q_j$, prodotta dal metodo di Arnoldi, che si basa sull'algoritmo di Gram-Schmidt modificato. In questo modo, il vettore $x_j \in K_j$ può essere scritto come $x_j = x_0 + Q_j y_j$ dove Q_j è una matrice $n \times j$ avente come colonne i vettori $q_1, q_2, \ldots q_j$ e $y_j \in \mathbb{R}^j$.
L'algoritmo 4 presenta l'iterazione del metodo GMRES [6].

Per quanto riguarda il costo computazionale, il passaggio più dispendioso è sicuramente quello che richiede l'utilizzo del metodo di Arnoldi. Il resto dell'algoritmo determina y_j mediante la risoluzione di un problema ai minimi quadrati di dimensione $j \ll n$. Inoltre, è importante notare che per calcolare una nuova soluzione, è necessario memorizzare tutti i j vettori della base $q_1, \ldots q_j$. Per limitare questa criticità, è stata proposta una variante del metodo che prendesse in input un numero positivo m corrispondente alla massima

Algorithm 4 Metodo GMRES

1: Siano $r_0 = b - Ax_0$, $\beta = \|r_0\|$, $v_1 = r_0/\beta$;
2: Sia m la dimensione del sottospazio di Krylov e definiamo la matrice $\bar{H}_m = \{h_{ij}\}_{1\leq i\leq m+1, 1\leq j\leq m}$; poniamo $\bar{H}_m = 0$.
3: **for** $j = 1, 2, \ldots, m$, **do**
4: $w_j = Av_j$;
5: **for** $i = 1, 2, \ldots, j$, **do**
6: $h_{ij} = \langle w_j, v_j \rangle$;
7: $w_j = w_j - h_{ij}v_i$;
8: **end for**
9: $h_{y+1,j} = \|w_j\|_2$;
10: **if** $h_{y+1,j} = 0$: **then** poni $m = j$ e vai a 15; **end**;
11: $v_{j+1} = w_j/h_{j+1,j}$;
12: **end for**
13: Trova y_m che minimizzi $\|\beta e_1 - \bar{H}_m y_m\|_2$ e calcola $x_m = x_0 + Q_m y_m$.

dimensione raggiungibile dai sottospazi di Krylov, in questo modo si decide di arrestare il metodo se il residuo è minore della precisione richiesta o se sono state costruite m funzioni di base; a questo punto si calcola x_j e lo si restituisce se la tolleranza è stata raggiunta, oppure si riavvia l'algoritmo usando la soluzione calcolata x_j come dato iniziale. Concludiamo proponendo l'algoritmo di questa versione che prevede anche l'utilizzo della matrice di Hessenberg superiore H_n prodotta dal metodo di Arnoldi.

Algorithm 5 Variante del metodo GMRES

1: Calcola $r_0 = b - Ax_0$, $\beta = \|r_0\|$, $v_1 = r_0/\beta$;
2: genera Q_m e $\bar{H}_m$ con l'algoritmo di Arnoldi;
3: trova y_j che minimizzi $\|\beta e_1 - \bar{H}_m y_m\|_2$;
4: calcola $x_m = x_0 + Q_m y_m$;
5: **if** $\|x_m\| <$ tolleranza, **then** STOP;
6: **else**, poni $x_0 = x_m$ e vai a 1; **end**;

Osservazione 2.4. E' importante notare che questa tecnica, pur aumentando il numero di iterazioni eseguite, riduce considerevolmente il tempo impiegato per trovare la soluzione, il costo computazionale e l'uso di memoria. Tuttavia, mentre la teoria garantisce che il metodo GMRES classico converga in n passi, l'utilizzo di questa variante potrebbe portare a delle stagnazioni della soluzione.

2.2.3 Metodo BiCGSTAB

Il metodo BiCGSTAB *(Biconjugate Gradient Stabilized)* è un metodo iterativo non stazionario basato sui sottospazi di Krylov; è una variante più avanzata del metodo dei Gradienti Biconiugati (BiCG).
Rispetto a quanto visto nella sezione precedente, si tratta di un metodo che per l'approssimazione della soluzione non richiede l'utilizzo di due cicli annidati, e che non necessita la memorizzazione dei vettori della base dello spazio di Krylov. Invece che sull'ortogonalizzazione di Arnoldi, il metodo BiCGSTAB si basa sull'algoritmo di biortogonalizzazione di Lanczos, il cui principale vantaggio è richiedere la memorizzazione di un numero ridotto di vettori. In breve, la biortogonalizzazione di Lanczos produce una coppia di basi ortogonali per i sottospazi $K_j(A, v) = span\{v, Av, A^2v, \ldots, A^{j-1}v\}$ e $K_j(A^T, w) = span\{w, A^Tw, (A^T)^2w, \ldots, (A^T)^{j-1}w\}$ dove $\langle v, w\rangle = 1$. Il metodo BiCGSTAB è basato sulla scrittura del residuo come prodotto di due polinomi $\psi_j(A)$ e $\phi_j(A)$. L'algoritmo 6 riporta l'iterazione del metodo BiCGSTAB [6].

Concludiamo osservando che l'algoritmo richiede, per ogni iterazione, due prodotti matrice-per-vettore e diverse operazioni tra vettori e scalari.

Osservazione 2.5. Una generalizzazione del metodo BiCGSTAB, è il metodo BiCGSTAB(ℓ). Tale metodo è stato sviluppato per risolvere la classe di equazioni per cui vi è una stagnazione della convergenza di BiCGSTAB [8].

Algorithm 6 Metodo BiCGSTAB

Sia $r_0 = b - Ax_0$, sia r_0^* tale che $\langle r_0, r_0^* \rangle \neq 0$ e poniamo $p_0 = r_0$ e $p_0^* = r_0^*$.
for $j = 0, 1, 2, \ldots$ fino a convergenza, **do**
  $\alpha_j = \langle r_j, r_j^* \rangle / \langle Ap_j, p_j^* \rangle$;
  $x_{j+1} = x_j + \alpha p_j$;
  $r_{j+1} = r_j - \alpha_j A p_j$;
  $r_{j+1}^* = r_j^* - \alpha_j A^T p_j^*$;
  $\beta_j = \langle r_{j+1}, r_{j+1}^* \rangle / \langle r_j, r_j^* \rangle$;
  $p_{j+1} = r_{j+1} + \beta_j p_j$;
  $p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$;
end for

2.2.4 Metodo IDR(s)

La famiglia di metodi IDR(s) è stata presentata da Peter Sonneveld e Martin B. Van Gijzen in [7] per risolvere efficientemente grandi sistemi non simmetrici di equazioni lineari. Il nuovo approccio si basa sul metodo di riduzione della dimensione indotta *(Induced Dimension Reduction)* proposto da Sonneveld nel 1980.

Data una matrice $A \in \mathbb{R}^{n \times n}$ di grandi dimensioni e non simmetrica, il metodo IDR(s) costruisce una approssimazione della soluzione del sistema lineare $Ax = b$, mediante la generazione di una successione di residui appartenenti ad una sequenza di sottospazi annidati. Riportiamo il teorema principale su cui si basa questa famiglia di metodi.

Teorema 2.3. *Sia $A \in \mathbb{R}^{n \times n}$ una matrice e $v_0 \in \mathbb{R}^n$ un vettore non nullo, definiamo $\mathcal{G}_0$ come il sottospazio di Krylov $K_n(A, v_0) = span\{v_0, Av_0, \ldots, A^{n-1}v_0\}$. Sia $\mathcal{S}$ un opportuno sottospazio di $\mathbb{R}^n$ tale che $\mathcal{S}$ e $\mathcal{G}_0$ non condividano un sottospazio invariante non banale di A. Definiamo la successione $\{\mathcal{G}_j\}_{j \in \mathbb{N}}$ come*

$$\mathcal{G}_j = (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{S})$$

dove ω_j è uno scalare non nullo. Allora valgono le seguenti affermazioni:

i) $\mathcal{G}_j \subset \mathcal{G}_{j-1}$, $\forall j > 0$;

ii) $\mathcal{G}_j = \{0\}$, *per qualche* $j \leq n$.

Osservazione 2.6. Osserviamo che l'ipotesi che $\mathcal{S}$ e $\mathcal{G}_0$ non condividano nessun sottospazio invariante non banale di A non è restrittiva: infatti, poiché $\mathcal{G}_0$ è un sottospazio di Krylov, tutti gli autospazi di A in $\mathcal{G}_0$ sono unidimensionali, quindi se $\mathcal{S}$ viene scelto arbitrariamente, l'evento che uno di questi autospazi si trovi in $\mathcal{S}$ ha probabilità zero.

L'algoritmo 7 riporta l'iterazione del metodo IDR(s).
Osserviamo che per eseguire un ciclo di $s+1$ passi, l'algoritmo IDR(s) richiede $s+1$ prodotti matrice-per-vettore, e s^2+s+2 prodotti interni.

Osservazione 2.7. Si può provare che, in aritmetica esatta, il metodo IDR(s) converge alla soluzione usando al più $n + \frac{n}{s}$ prodotti matrice-per-vettore.

Algorithm 7 Metodo IDR(s)

Siano $A \in \mathbb{R}^{n\times n}$, $x_0, b \in \mathbb{R}^n$, $s \in \mathbb{N}$, siano $p_1, p_2, \ldots, p_s \in \mathbb{R}^n$ linearmente indipendenti, e poniamo $P = [p_1, p_2, \ldots, p_s] \in \mathbb{R}^{n\times s}$. Sia $r_0 = b - Ax_0$.
for $n = 0, 1, \ldots, s-1$, **do**
  $v = Ar_n$; $\omega = (v^T r_n)/(v^T v)$;
  $\Delta x_n = \omega r_n$; $\Delta r_n = -\omega v$;
  $r_{n+1} = r_n + \Delta r_n$; $x_{n+1} = x_n + \Delta x_n$;
end for
$\Delta R_{n+1} = (\Delta r_n \cdots \Delta r_0)$; $\Delta X_{n+1} = (\Delta x_n \cdots \Delta x_0)$;
$n = s$;
while $\|r_n\| >$ tolleranza e $n <$ del massimo di iterazioni consentite, **do**
  for $k = 0, 1, \ldots, s$, **do**
    risolvi $P^T \Delta R_n c = P^T r_n$ per c;
    $v = r_n - \Delta R_n c$;
    if $k = 0$, **then**
      $t = Av$;
      $\omega = (t^T v)/(t^T t)$;
      $\Delta r_n = -\Delta R_n c - \omega t$;
      $\Delta x_n = -\Delta X_n c + \omega v$;
    else
      $\Delta x_n = -\Delta X_n c + \omega v$;
      $\Delta r_n = -A\Delta x_n$;
    end if
    $r_{n+1} = r_n + \Delta r_n$;
    $x_{n+1} = x_n + \Delta x_n$;
    $n = n + 1$;
    $\Delta R_n = (\Delta r_{n-1} \cdots \Delta r_{n-s})$;
    $\Delta X_n = (\Delta x_{n-1} \cdots \Delta x_{n-s})$;
  end for
end while

Capitolo 3

Esperimenti numerici e confronti

In questo ultimo capitolo riportiamo i risultati di un'analisi sperimentale volta ad applicare ad un caso reale le nozioni descritte precedentemente, al fine di confrontare i metodi risolutivi, evidenziandone i punti di forza e le criticità.

3.1 Matrice di Stanford-Berkeley

Per produrre gli esperimenti, abbiamo scelto di usare il pacchetto di dati messo a disposizione dalle Università Stanford e Berkeley.
La matrice dei links[1] è una matrice a coefficienti reali di dimensione 8006115×3 in cui nelle prime due colonne vi sono gli indici delle pagine Web, mentre la terza colonna contrassegna con un 1 la presenza di un link tra le pagine della riga corrispondente, e con uno 0 l'assenza. Successivamente, la funzione `spconvert(stanford-berkeley-bool-sorted)` trasforma la matrice dei links in una matrice quadrata sparsa $H = \{h_{i,j}\}$ così definita:

$$h_{i,j} = \begin{cases} m & \textit{se } P_i \textit{ punta a } P_j \\ 0 & \textit{altrimenti} \end{cases} \tag{3.1}$$

[1] `stanford-berkeley-bool-sorted.dat`

dove m è il numero totale di links da P_i a P_j. Infine, viene chiamata la funzione `normalize.m` che provvede a normalizzare in norma-1 le righe di H, restituendo la matrice di Hyperlink desiderata. Precisiamo che le dimensioni della matrice H creata in questa fase, risultano essere 685230×685230, dunque non tutti i links presenti nel database vengono inseriti. Il risultato è una matrice sparsa con 7600595 elementi non zero, che equivalgono ad una percentuale del 1.6×10^{-3} %. Vediamo una rappresentazione della matrice nella Figura 3.1.

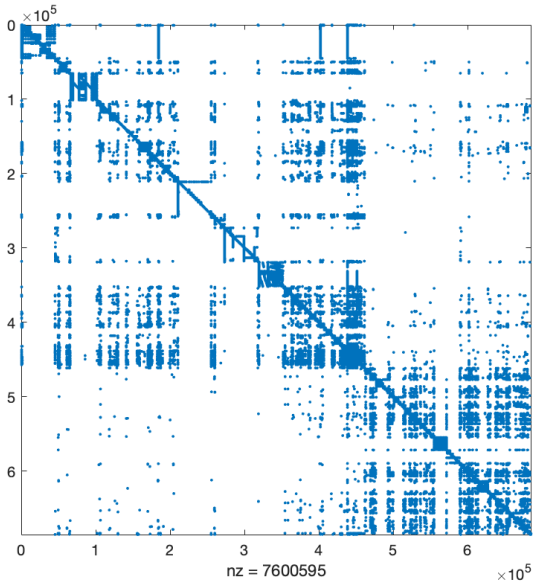


Figura 3.1: Matrice di Hyperlink di Stanford-Berkeley

L'ultima operazione da fare è assicurare che la matrice H sia stocastica per colonna, infatti abbiamo visto nel capitolo 1 che questa condizione è necessaria affinché il problema del calcolo del PageRank sia ben posto. La seguente funzione prende in input la matrice H e la restituisce nella forma desiderata.

```
function H = stoc_col(H)
[n,~] = size(H);
c = sum(H,1);
idx = find(c==0);
```

```
c(idx) = 1;
H = H/spdiags(c', 0 : 0, n, n);
return
```

3.2 Risultati sperimentali

Presentiamo ora gli esperimenti numerici prodotti per testare i metodi descritti nel secondo capitolo, e confrontarli con il metodo tradizionalmente scelto per la determinazione del vettore di PageRank.
Premettiamo che il criterio di arresto di tutti gli esperimenti realizzati, è basato su un numero massimo di iterazioni eseguibili fissato a 1000, e su una tolleranza uguale a 10^{-7}.

Incidenza del coefficiente α

Per prima cosa, mostriamo come vengono influenzati i metodi descritti dalla scelta di dieci diversi valori dal coefficiente di teletrasporto α.

Metodo delle potenze. Il metodo è stato eseguito scegliendo come vettore iniziale il vettore $\frac{1}{n}\mathbf{1}$ e come vettore di personalizzazione $\frac{1-\alpha}{n}\mathbf{1}$, dove n è la dimensione della matrice di Hyperlink H. La Figura 3.2 rappresenta l'andamento della convergenza del metodo, al variare del coefficiente di teletrasporto, e mostra come solo per $\alpha = 0.9999$, la tolleranza richiesta non sia raggiunta. Si nota comunque un peggioramento della convergenza per $\alpha \to 1$, ciò è in linea con quanto osservato in 1.7.

Metodo di Jacobi. Per l'utilizzo di questo metodo abbiamo definito il vettore iniziale $x_0 = \frac{1}{n}\mathbf{1}$, e il vettore dei coefficienti $b = \frac{1-\alpha}{n}\mathbf{1}$. Come nel caso del metodo delle potenze, la Figura 3.3 mostra che il metodo non raggiunge la convergenza solo per l'ultimo valore assunto dal coefficiente α. Anche in questo caso la convergenza peggiora sensibilmente per i valori di α vicini a 1, in accordo con quanto previsto dalla teoria.

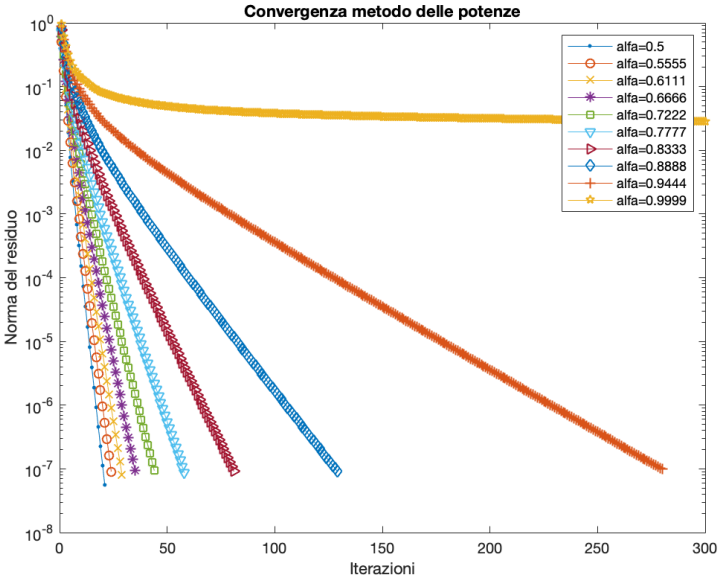


Figura 3.2: Metodo delle potenze al variare di α

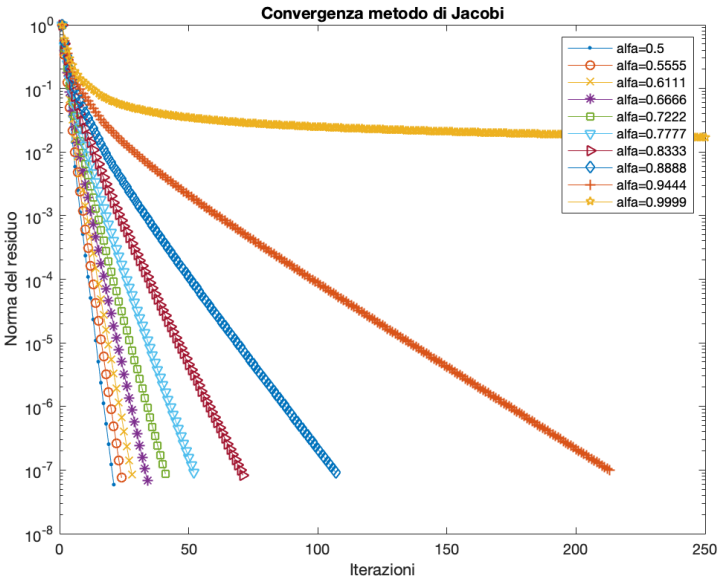


Figura 3.3: Metodo di Jacobi al variare di α

Metodo GMRES. Il metodo GMRES per l'approssimazione della soluzione del sistema lineare $Ax = b$, è stato eseguito con la matrice $A = I - \alpha H$ e il

vettore dei coefficienti $b = \frac{1-\alpha}{n}\mathbf{1}$ normalizzato. Il metodo è utilizzato senza *restart*.

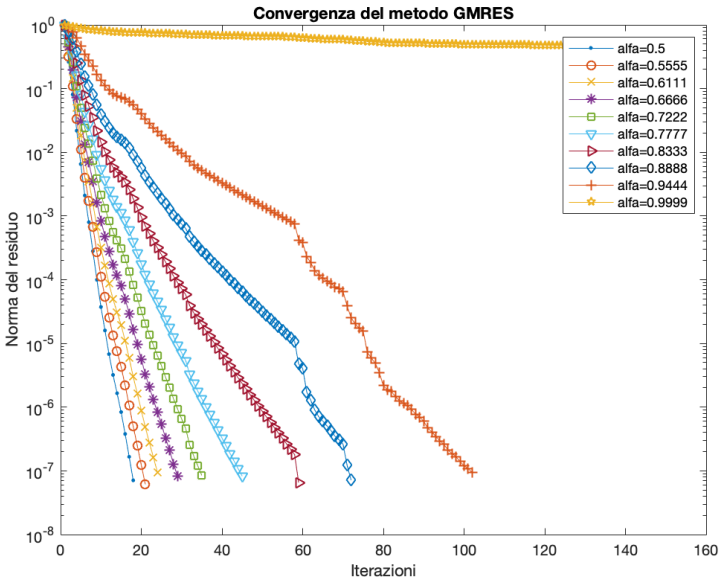


Figura 3.4: Metodo GMRES al variare di α

La Figura 3.4 evidenzia che anche in questo caso la convergenza è raggiunta per tutti sistemi lineari, tranne quello che assume l'ultimo valore di α. La forte influenza di α è riconducibile al fatto che il metodo viene utilizzato con la matrice $A = I - \alpha H$ e il vettore dei coefficienti $b = \frac{1-\alpha}{n}\mathbf{1}$, quindi i valori di α vicini ad 1 tendono a peggiorare notevolmente la convergenza.

Metodo BiCGSTAB(ℓ). Per l'esecuzione di questo metodo sono stati utilizzati gli stessi dati iniziali descritti nel paragrafo precedente.
Inoltre, abbiamo ritenuto opportuno mostrare la convergenza del metodo BiCGSTAB(ℓ) al variare del coefficiente di teletrasporto α, per diversi valori di ℓ: infatti sappiamo dalla teoria che questa quantità incide molto sulla velocità di convergenza del metodo; dunque in Figura 3.5 possiamo osservare come varia il metodo per cinque diversi valori di ℓ, con $\alpha = 0.9$ fissato. Infine le Figure

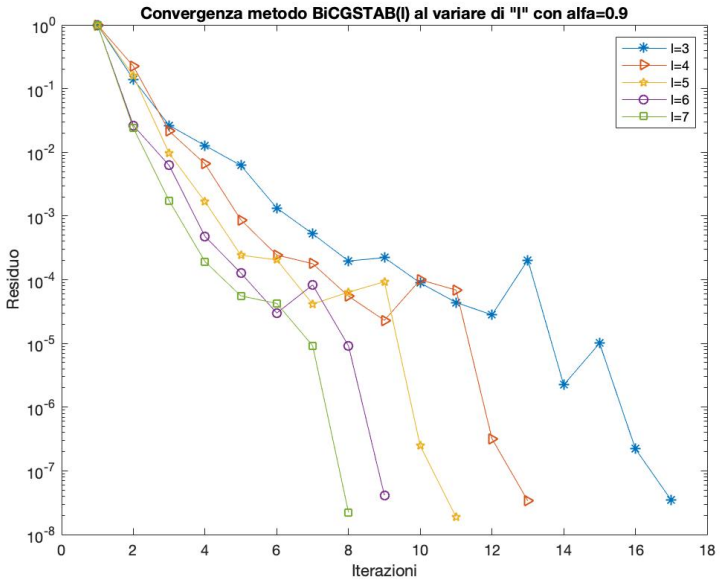


Figura 3.5: Metodo BiCGSTAB(ℓ) al variare di ℓ, con $\alpha = 0.9$

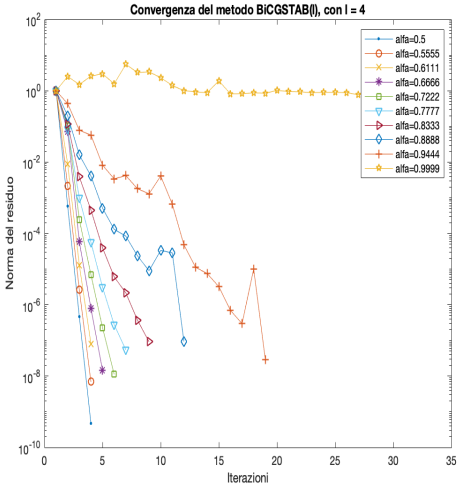


Figura 3.6: $\ell = 4$

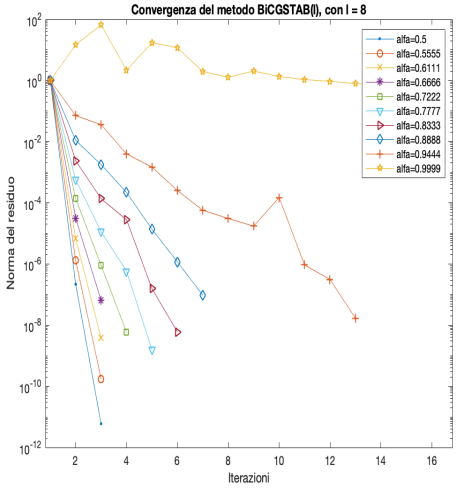


Figura 3.7: $\ell = 8$

Figura 3.8: Metodo BiCGSTAB(ℓ) al variare di α

3.6 e 3.7, delineano la convergenza del metodo al variare di α, rispettivamente con i valori $\ell = 4$ e $\ell = 8$.

Metodo IDR(s). Il metodo IDR(s) è stato eseguito considerando la matrice $A = I - \alpha H$, il vettore dei coefficienti $b = \frac{1-\alpha}{n}\mathbf{1}$ normalizzato e il vettore iniziale $x_0 = \frac{1}{n}\mathbf{1}$. Nel paragrafo 2.2.4, abbiamo osservato che il metodo IDR(s) dipende dalla scelta di un parametro s; tuttavia, al contrario di quanto visto per BiCGSTAB(ℓ), la Figura 3.9 ci mostra che nel caso in esame, la scelta del parametro s non incide particolarmente sulla velocità di convergenza del metodo. D'altra parte, sappiamo che il metodo IDR(s), opera $s+1$ prodotti matrice-per-vettore e $\mathcal{O}(s^2)$ prodotti interni, per iterazione. Per queste ragioni scegliamo di utilizzare il metodo con $s = 6$. La Figura 3.10 ci mostra i risultati di convergenza al variare del coefficiente di teletrasporto α. Notiamo che, come per i metodi precedenti, la convergenza non è raggiunta solo per $\alpha = 0.9999$.

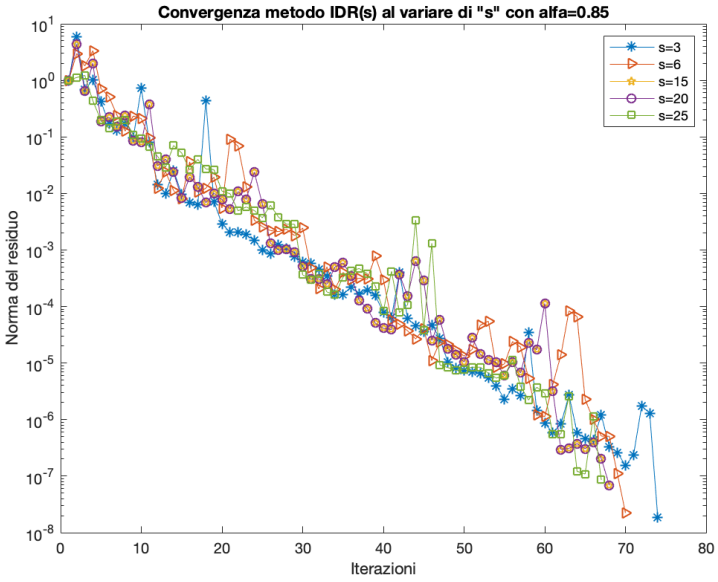


Figura 3.9: Metodo IDR(s) al variare di s, con $\alpha = 0.85$

Osservazione 3.1. Si noti che i metodi BiCGSTAB(ℓ) e IDR(s) non presentano una convergenza monotona. Questo è dovuto al fatto che, contrariamente a GMRES, ad ogni iterazione non minimizzano la norma del residuo.

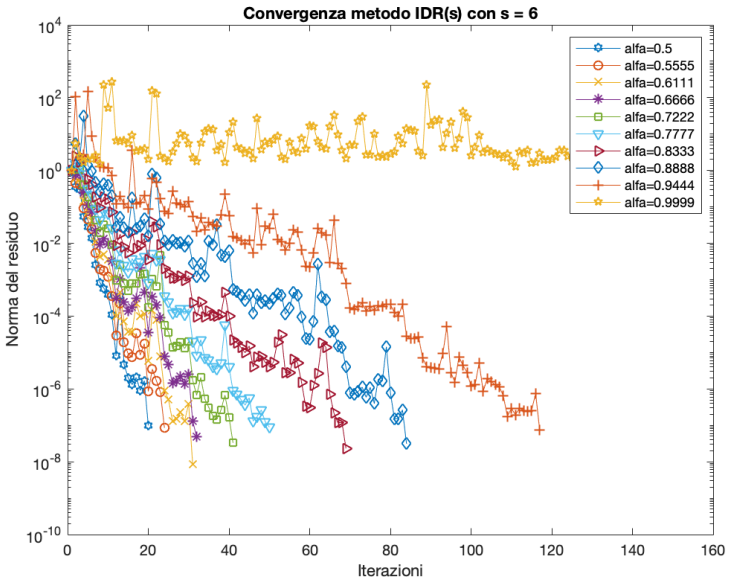


Figura 3.10: Metodo IDR(6) al variare di α

Confronto tra metodi

Grafici. E' importante prendere coscienza del fatto che i metodi descritti nel secondo capitolo, hanno costi computazionali molto diversi tra loro, dunque un confronto diretto basato sul numero di iterazioni in cui si raggiunge la convergenza potrebbe non essere indicativo della situazione. Per questa ragione presentiamo alcuni esperimenti numerici che confrontano i metodi rispetto al tempo di esecuzione richiesto. Specifichiamo che per produrre tali risultati si è usato il metodo BiCGSTAB(8) e il metodo IDR(6).
Osserviamo che le Figure 3.11, 3.12 e 3.13 evidenziano come il metodo delle potenze risulti il più veloce nei primi tre scenari, mentre nella Figura 3.14, notiamo che non raggiunge nemmeno la convergenza. Al contrario, BiCGSTAB(8) raggiunge sempre la convergenza, pur richiedendo elevati tempi di esecuzione. Notiamo inoltre che GMRES è il metodo più costoso in termini di tempo di CPU: questo è dovuto al fatto che ad ogni iterazione, l'algoritmo esegue un procedimento di ortogonalizzazione di Gram-Schmidt modificato.

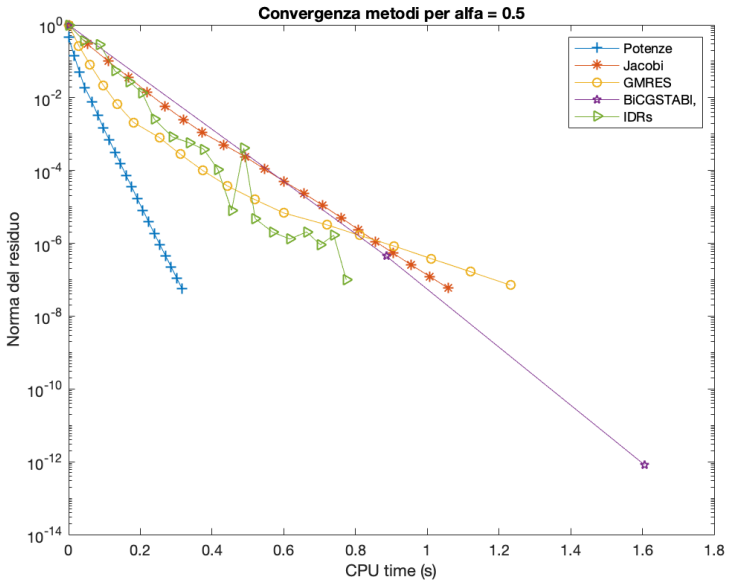


Figura 3.11: Confronto metodi per $\alpha = 0.5$

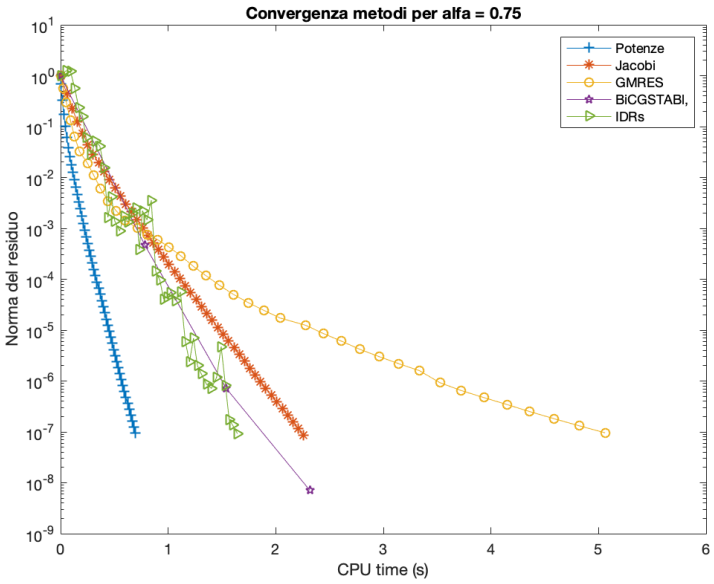


Figura 3.12: Confronto metodi per $\alpha = 0.75$

Tabelle. Concludiamo presentando quattro tabelle riassuntive del numero di iterazioni richiesto da ciascun metodo descritto, con relativo tempo di esecuzio-

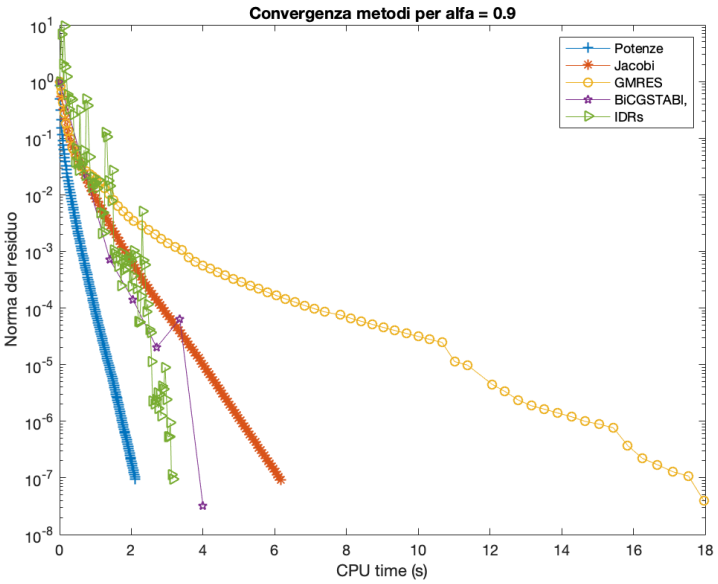


Figura 3.13: Confronto metodi per $\alpha = 0.9$

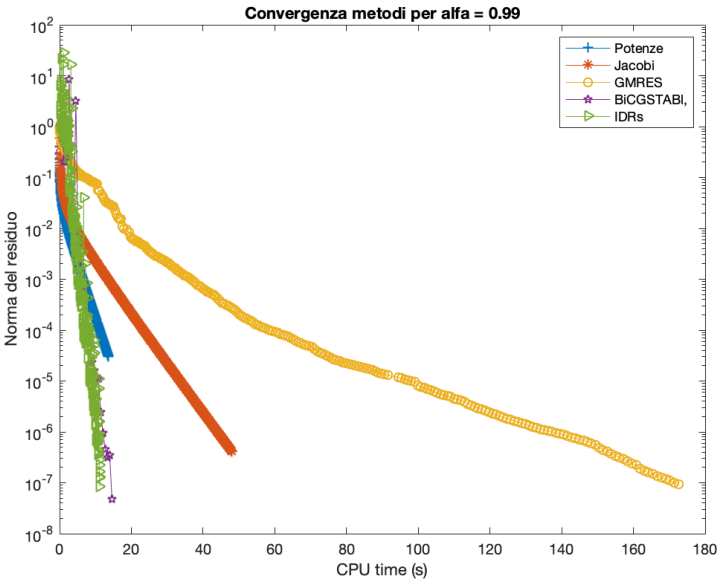


Figura 3.14: Confronto metodi per $\alpha = 0.99$

ne e memoria necessaria (intesa come numero di vettori utilizzati dal metodo), per quattro diversi valori di α.

In accordo con quanto mostrato nei grafici precedenti, possiamo notare che per valori di α sufficientemente lontani da 1, il metodo delle potenze si rivela il più efficiente in termini di tempo di CPU. Tuttavia per $\alpha = 0.99$, abbiamo visto che tale metodo non raggiunge la convergenza, al contrario di IDR(s) che risulta il più veloce.

Metodo	N. iterazioni	CPU time(s)	Memoria(vett)
Jacobi	21	1.359	3
Potenze	21	0.3362	3
GMRES	18	1.225	21
BiCGSTAB(ℓ)	3	1.6214	21
IDR(s)	20	0.7873	23

Tabella 3.1: Risultati per $\alpha = 0.5$

Metodo	N. iterazioni	CPU time(s)	Memoria(vett)
Jacobi	46	2.4287	3
Potenze	50	0.7425	3
GMRES	39	5.1837	42
BiCGSTAB(ℓ)	4	2.4632	21
IDR(s)	43	1.793	23

Tabella 3.2: Risultati per $\alpha = 0.75$

Metodo	N. iterazioni	CPU time(s)	Memoria(vett)
Jacobi	119	6.1369	3
Potenze	145	2.1277	3
GMRES	76	17.8947	79
BiCGSTAB(ℓ)	7	4.0518	21
IDR(s)	85	3.2594	23

Tabella 3.3: Risultati per $\alpha = 0.9$

Metodo	N. iterazioni	CPU time(s)	Memoria(vett)
Jacobi	1000	48.5628	3
Potenze	1000	16.3871	3
GMRES	768	173.482	771
BiCGSTAB(ℓ)	83	16.7245	21
IDR(s)	947	12.5897	23

Tabella 3.4: Risultati per $\alpha = 0.999$

Bibliografia

[1] D. Gleich, L. Zhukov, P. Berkhin. *Fast Parallel PageRank: A Linear System Approach*, Technical Report YRL-2004-038; Yahoo! Research Labs: Sunnyvale, CA, USA, 2004. 15

[2] Valeria Simoncini. *Lucidi del corso di Matematica Computazionale*, (2019).

[3] Davide Palitta, Valeria Simoncini. *Dispense del corso di Calcolo Numerico: Modulo di Algebra Lineare Numerica*, (2020).

[4] D. Bini, M. Capovani, O. Menchi. *Metodi Numerici per l'Algebra Lineare*, Zanichelli Bologna, (1988).

[5] A. Arasu, J. Novak, A. Tomkins, J. Tomlin. *PageRank Computation and the Structure of the Web: Experiments and Algorithms.* In *Proceedings of the Eleventh International World Wide Web Conference*, (2002). 15

[6] Y. Saad. *Iterative Methods for Sparse Linear Systems*, SIAM J. SCI. COMPUT, https://doi.org/10.1137/1.9780898718003, (2000). https://web.stanford.edu/class/cme324/saad.pdf . 19, 21

[7] P. Sonneveld, M.B. Van Gijzen. *IDR(s): A Family of Simple and Fast Algorithms for Solving Large Nonsymmetric Linear Systems.* SIAM J. SCI. COMPUT, Vol. 31, No. 2, pp. 1035–1062 (2008). doi:10.1137/070685804. 22

[8] G.L.G. Sleijpen, D.R. Fokkema. *BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum.* Electronic Transactions on Numerical Analysis, Volume 1, pp. 11-32, (1993) 21

Ringraziamenti

I primi ringraziamenti sono accademici, perché solo grazie all'ambiente accogliente e stimolante del Dipartimento di Matematica di Bologna, sono riuscita a crescere personalmente e professionalmente, e a raggiungere questo traguardo.
Voglio sinceramente ringraziare la mia relatrice, Prof.ssa Valeria Simoncini, per la passione coinvolgente con cui insegna la sua materia, e la disponibilità con cui mi ha aiutato a redigere questo elaborato.