Calcolo del PageRank, con una particolare attenzione ai dangling nodes

Ilse C.F. Ipsen, Teresa M. Selee
Corso di Calcolo Scientifico, Vittorio Meini 533644

Algoritmo per il calcolo del PageRank di una matrice di adiacenza

- Consiste nel concentrare (*lumping*) tutti i dangling nodes in uno solo.

Algoritmo per il calcolo del PageRank di una matrice di adiacenza

- Consiste nel concentrare (*lumping*) tutti i dangling nodes in uno solo.
- Applica il metodo delle potenze soltanto alle matrici "concentrate", con la stessa velocità di convergenza dell'intera matrice.

Secondo la tesi delle autrici, l'algoritmo proposto:

- Mantiene i vantaggi dell'algoritmo tradizionale (metodo delle potenze), ma è più veloce.

Secondo la tesi delle autrici, l'algoritmo proposto:

- Mantiene i vantaggi dell'algoritmo tradizionale (metodo delle potenze), ma è più veloce.
- È semplice da implementare.

Secondo la tesi delle autrici, l'algoritmo proposto:

- Mantiene i vantaggi dell'algoritmo tradizionale (metodo delle potenze), ma è più veloce.
- È semplice da implementare.
- Maggiore è la dimensione della rete e il numero dei dangling nodes più l'algoritmo è competitivo.

Secondo la tesi delle autrici, l'algoritmo proposto:

- Mantiene i vantaggi dell'algoritmo tradizionale (metodo delle potenze), ma è più veloce.
- È semplice da implementare.
- Maggiore è la dimensione della rete e il numero dei dangling nodes più l'algoritmo è competitivo.
- C'è la possibilità di differenziare i dangling nodes dal vettore di personalizzazione, sostituendoli con un qualsiasi vettore.

Elementi di partenza e notazioni

- n: Numero di pagine del web

Elementi di partenza e notazioni

- n: Numero di pagine del web
- k: Numero delle pagine non-dangling (quindi $n - k$ sarà il numero dei dangling nodes)

Elementi di partenza e notazioni

- n: Numero di pagine del web
- k: Numero delle pagine non-dangling (quindi $n - k$ sarà il numero dei dangling nodes)
- $H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}$
  dove $H_{11} \in \mathbb{R}^{k \times k}$ rappresenta i link fra le pagine non dangling, $H_{12} \in \mathbb{R}^{k \times (n-k)}$ i link fra pagine non dangling e dangling nodes e gli zeri sono associati ai dangling nodes.

Elementi di partenza e notazioni

- n: Numero di pagine del web
- k: Numero delle pagine non-dangling (quindi $n-k$ sarà il numero dei dangling nodes)
- $H = \begin{bmatrix} H_{11} & H_{12} \\ 0 & 0 \end{bmatrix}$
  dove $H_{11} \in \mathbb{R}^{k \times k}$ rappresenta i link fra le pagine non dangling, $H_{12} \in \mathbb{R}^{k \times (n-k)}$ i link fra pagine non dangling e dangling nodes e gli zeri sono associati ai dangling nodes.
- $H_{11} \geq 0, H_{22} \geq 0, H_{11}e + H_{22}e = e$

Elementi di partenza e notazioni

- $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$, $w \geq 0, ||w|| = w^T e = 1$, con $w_1 \in \mathbb{R}^{k \times 1}$ e $w_2 \in \mathbb{R}^{(n-k) \times 1}$, w è il vettore che prenderà il posto dei dangling nodes.

Elementi di partenza e notazioni

- $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, w \geq 0, ||w|| = w^T e = 1$, con $w_1 \in \mathbb{R}^{k \times 1}$ e $w_2 \in \mathbb{R}^{(n-k) \times 1}$, w è il vettore che prenderà il posto dei dangling nodes.
- La matrice finale è $S \equiv H + dw^T = \begin{bmatrix} H_{11} & H_{12} \\ ew_1^T & ew_2^T \end{bmatrix}$, dove $d \equiv \begin{bmatrix} 0 \\ e \end{bmatrix}$. Vale che $S \geq 0$ e $Se = e$.

Elementi di partenza e notazioni

- $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, w \geq 0, ||w|| = w^T e = 1$, con $w_1 \in \mathbb{R}^{k \times 1}$ e $w_2 \in \mathbb{R}^{(n-k) \times 1}$, w è il vettore che prenderà il posto dei dangling nodes.
- La matrice finale è $S \equiv H + dw^T = \begin{bmatrix} H_{11} & H_{12} \\ ew_1^T & ew_2^T \end{bmatrix}$, dove $d \equiv \begin{bmatrix} 0 \\ e \end{bmatrix}$. Vale che $S \geq 0$ e $Se = e$.
- Il vettore di personalizzazione è $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, v \geq 0, ||v|| = 1$

Elementi di partenza e notazioni

- $w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, w \geq 0, ||w|| = w^T e = 1$, con $w_1 \in \mathbb{R}^{k \times 1}$ e $w_2 \in \mathbb{R}^{(n-k) \times 1}$, w è il vettore che prenderà il posto dei dangling nodes.
- La matrice finale è $S \equiv H + dw^T = \begin{bmatrix} H_{11} & H_{12} \\ ew_1^T & ew_2^T \end{bmatrix}$, dove $d \equiv \begin{bmatrix} 0 \\ e \end{bmatrix}$. Vale che $S \geq 0$ e $Se = e$.
- Il vettore di personalizzazione è $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, v \geq 0, ||v|| = 1$
- La matrice del web è $G \equiv \alpha S + (1 - \alpha) ev^T, 0 \leq \alpha \leq 1$

Elementi di partenza e notazioni, la procedura di *lumping*

- La matrice G è stocastica, quindi G ha un'unica distribuzione stazionaria: π, il vettore di *PageRank*, tale che $\pi^T G = \pi^T, \pi \geq 0, ||\pi|| = 1$
  Scriviamo $\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix}, \pi_1 \in \mathbb{R}^{k \times 1}, \pi_2 \in \mathbb{R}^{(n-k) \times 1}$

Elementi di partenza e notazioni, la procedura di *lumping*

- La matrice G è stocastica, quindi G ha un'unica distribuzione stazionaria: π, il vettore di *PageRank*, tale che $\pi^T G = \pi^T, \pi \geq 0, ||\pi|| = 1$
  Scriviamo $\pi = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix}, \pi_1 \in \mathbb{R}^{k \times 1}, \pi_2 \in \mathbb{R}^{(n-k) \times 1}$
- Sia P una matrice di permutazione, M matrice stocastica e
  $PMP^T = \begin{bmatrix} M_{11} & \dots & M_{1,k+1} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ M_{k+1,1} & \dots & M_{k+1,k+1} \end{bmatrix}$ una partizione
  allora M si dice *concentrabile* (*lumpable*) rispetto a tale partizione se ogni vettore del tipo $M_{ij}e$ è multiplo di e per $i \neq j, 1 \leq i, j \leq k+1$

Elementi di partenza e notazioni, la procedura di *lumping*

- Si può condensare la notazione espressa in precedenza, scrivendo: $G = \begin{bmatrix} G_{11} & G_{12} \\ eu_1^T & eu_2^T \end{bmatrix}$, con $G_{11} \in \mathbb{R}^{k \times k}$ e $G_{12} \in \mathbb{R}^{(n-k) \times k}$, dove $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \equiv \alpha w + (1-\alpha)v$.
  L'elemento (i, j) di G_{11}, corrisponde al blocco M_{ij} per $1 \leq i, j \leq k$
  La riga i di G_{12} corrisponde al blocco $M_{i,k+1}$ per $1 \leq i \leq k$
  La colonna i di eu_1^T corrisponde a $M_{k+1,i}$ per $1 \leq i \leq k$
  eu_2^T corrisponde a $M_{k+1,k+1}$
  Chiaramente con questa notazione G è *concentrabile*.

Formula per il calcolo del PageRank

- Proposizione

  Sia $X \equiv \begin{bmatrix} I_k & 0 \\ 0 & L \end{bmatrix}$, dove $L = I_{n-k} - \frac{1}{n-k}\hat{e}e^T$, con $\hat{e} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

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

  Inoltre $G^{(1)}$ è stocastica di ordine $k+1$ con gli stessi autovalori non-zero di G.

Formula per il calcolo del PageRank

- ## Proposizione

  Sia $X \equiv \begin{bmatrix} I_k & 0 \\ 0 & L \end{bmatrix}$, dove $L = I_{n-k} - \frac{1}{n-k}\hat{e}e^T$, con $\hat{e} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

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

  Inoltre $G^{(1)}$ è stocastica di ordine $k+1$ con gli stessi autovalori non-zero di G.

- ## Dimostrazione della proposizione

  Si inizia con alcuni calcoli relativi alla proposizione,

  $X^{-1} = \begin{bmatrix} I_k & 0 \\ 0 & L^{-1} \end{bmatrix}$, $L^{-1} = I_{n-k} + \hat{e}e^T$, quindi

  $XGX^{-1} = \begin{bmatrix} G_{11} & G_{12}(I + \hat{e}e^T) \\ e_1 u_1^T & e_1 u_2^T (I + \hat{e}e^T) \end{bmatrix}$ ha gli stessi autovalori di G e si osserva che $G_{12}(I + \hat{e}e^T)e_1 = G_{12}e$ e che $u_2^T(I + \hat{e}e^T)e_1 = u_2^T e$

Formula per il calcolo del PageRank

- Teorema
  Sia $\sigma^T G^{(1)} = \sigma^T, \sigma \geq 0, ||\sigma|| = 1$, data la partizione $\sigma^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma_{k+1} \end{bmatrix}$, il vettore del PageRank è
  $$\pi^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \end{bmatrix}$$

Formula per il calcolo del PageRank

- **Teorema**
  Sia $\sigma^T G^{(1)} = \sigma^T, \sigma \geq 0, ||\sigma|| = 1$, data la partizione $\sigma^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma_{k+1} \end{bmatrix}$, il vettore del PageRank è
  $$\pi^T = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \end{bmatrix}$$

- **Dimostrazione del teorema**
  Riprendendo la dimostrazione precedente possiamo scrivere $XGX^{-1} = \begin{bmatrix} G^{(1)} & G^{(2)} \\ 0 & 0 \end{bmatrix}$, dove
  $G^{(2)} \equiv \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} (I + \hat{e}e^T) \begin{bmatrix} e_2 & \dots & e_{n-k} \end{bmatrix}$, il vettore $\begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix}$ è un autovettore per XGX^{-1} associato a $\lambda = 1$, quindi $\hat{\pi} \equiv \begin{bmatrix} \sigma^T & \sigma^T G^{(2)} \end{bmatrix} X$ è un autovettore per $\lambda = 1$ relativo a G ed è un multiplo della distribuzione stazionaria π di G. Poiché $G^{(1)}$ ha gli stessi autovalori non-zero di G e l'autovalore dominante di G,1, è singolare, la distribuzione stazionaria σ di $G^{(1)}$ è unica.

Formula per il calcolo del PageRank, dimostrazione

Riscriviamo il vettore esplicitando X e tornando alla partizione che evidenzia i primi k elementi, $\hat{\pi}^T = \begin{bmatrix}\sigma_{1:k}^T & \sigma_{k+1} & \sigma^T G^{(2)}\end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & L \end{bmatrix}$, cioè $\hat{\pi}^T = \begin{bmatrix}\sigma_{1:k}^T & (\sigma_{k+1} & \sigma^T G^{(2)})L\end{bmatrix}$. Le prime k componenti sono uguali a quelle di σ, bisogna esaminare le ultime $n-k$ componenti di $\hat{\pi}^T$, si può partizionare $L = \begin{bmatrix} 1 & 0 \\ -\frac{1}{n-k}e & I - \frac{1}{n-k}ee^T \end{bmatrix}$. La parte relativa ai dangling nodes è: $z^T \equiv \begin{bmatrix}\sigma_{k+1} & \sigma^T G^{(2)}\end{bmatrix} L =$

$= \begin{bmatrix}\sigma_{k+1} - \frac{1}{n-k}\sigma^T G^{(2)} e & \sigma^T G^{(2)}(I - \frac{1}{n-k}ee^T)\end{bmatrix}$

Per semplificare si considera:

$(I + \hat{e}e^T)\begin{bmatrix} e_2 & \dots & e_{n-k}\end{bmatrix} e = (I + \hat{e}e^T)\hat{e} = (n-k)\hat{e}$

Da cui: $G^{(2)}e = (n-k)\begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}\hat{e}$, quindi:

$\frac{1}{n-k}\sigma^T G^{(2)} e = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}\hat{e} =$

$\sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e - \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1 \boxed{=} \sigma_{k+1} - \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1$

Formula per il calcolo del PageRank, dimostrazione

$$\sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e - \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1 \boxed{=} \sigma_{k+1} - \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1$$

l'$\boxed{=}$ viene dal fatto che σ è la distribuzione stazionaria di $G^{(1)}$.

Per cui il primo elemento di z è

$$z_1 = \sigma_{k+1} - \frac{1}{n-k}\sigma^T G^{(2)} e \boxed{=} \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e_1$$

Per i rimanenti elementi di z si usa la prima realazione in blu

$$G^{(2)}(I - \frac{1}{n-k} ee^T) = G^{(2)} - \frac{1}{n-k} G^{(2)} ee^T = G^{(2)} - \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \hat{e} e^T$$

Rimpiazzando la relazione:
$(I + \hat{e}e^T) \begin{bmatrix} e_2 & \dots & e_{n-k} \end{bmatrix} = \begin{bmatrix} e_2 & \dots & e_{n-k} \end{bmatrix} + \hat{e}e$, nella definizione di $G^{(2)}$, si ottiene:

$$z_{2:n-k}^T = \sigma^T G^{(2)}(I - \frac{1}{n-k} ee^T) = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \begin{bmatrix} e_2 & \dots & e_{n-k} \end{bmatrix}$$ Quindi

la parte di autovettore associata ai dangling nodes è:

$$z = \begin{bmatrix} z_1 & z_{2:n-k}^T \end{bmatrix} = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}, \text{ da cui } \hat{\pi} = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \end{bmatrix}$$

Formula per il calcolo del PageRank, dimostrazione

$z = \begin{bmatrix} z_1 & z_{2:n-k}^T \end{bmatrix} = \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix}$, da cui $\hat{\pi} = \begin{bmatrix} \sigma_{1:k}^T & \sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} \end{bmatrix}$

Per unicità di π, vale che $\hat{\pi}^T e = 1 \Rightarrow \hat{\pi} = \pi$.

Ma $\hat{\pi}^T e = 1$ segue dal fatto che σ è la distribuzione stazionaria di $G^{(1)}$ e che $\sigma^T \begin{bmatrix} G_{12} \\ u_2^T \end{bmatrix} e = \sigma_{k+1}$ (perché $||\hat{\pi}|| = ||\sigma|| = 1$).-

L'algoritmo

Introduzione

L'algoritmo si basa sul teorema per calcolare il PageRank a partire dalla distribuzione stazionaria di σ della matrice "concentrata" $G^{(1)} \equiv \begin{bmatrix} G_{11} & G_{12}e \\ u_1^T & u_2^T e \end{bmatrix}$. L'input consiste nella matrice `H` (nell'implementazione l'algoritmo andrà riadattato alla matrice di adiacenza), il vettore di personalizzazione `v`, il vettore che rimpiazzerà i dangling nodes,`w` e il fattore di amplificazione α. L'output è $\hat{\pi}$, approssimazione del PageRank π, calcolato a partire da $\hat{\sigma}$, che approssima σ

L'algoritmo

Inputs: H,v,w,α Output: pi
Metodo delle potenze applicato a $G^{(1)}$
Si sceglie un vettore di partenza $\hat{\sigma}^T = \begin{bmatrix} \hat{\sigma}_{1:k}^T & \hat{\sigma}_{k+1} \end{bmatrix}$,
con $\hat{\sigma} \geq 0, ||\hat{\sigma}|| = 1$
while il metodo non converge
$\hat{\sigma}_{1:k}^T = \alpha \hat{\sigma}_{1:k}^T H_{11} + (1-\alpha) v_1^T + \alpha \hat{\sigma}_{k+1} w_1^T$
$\hat{\sigma}_{k+1} = 1 - \hat{\sigma}_{1:k} e$
end while
Calcolo del PageRank
$\hat{\pi}^T = \begin{bmatrix} \hat{\sigma}_{1:k} & \alpha \hat{\sigma}_{1:k}^T H_{12} + (1-\alpha) v_2^T + \alpha \hat{\sigma}_{k+1} w_2^T \end{bmatrix}$

Osservazioni

In ogni iterazione si moltiplica il vettore di una matrice sparsa per la matrice H_{11}, di dimensione $k \times k$. Il fattore di convergenza del metodo delle potenze applicato a $G^{(1)}$ è uguale a quello del metodo delle potenze applicato a G, cioè α, perché $G^{(1)}$ ha gli stessi autovalori non-zero di G, tuttavia è più veloce perché fatto su una matrice più piccola.

La sperimentazione. Implementazione dell'algoritmo

```
function y=lumpingPR(H, v, w, a, itmax)
n=length(H);
e=ones(n,1);
d=H*e;
d=d';
dang= d==0;
[y,p]=sort(dang,'ascend');
H=H(p,p);
d=d(p);
k=n-sum(dang);
dh=[1./d(1:k), ones(1,n-k)];
x=rand(1,k+1);
x=x/norm(x);
```

La sperimentazione. Implementazione dell'algoritmo

```
function y=lumpingPR(H, v, w, a, itmax)
n=length(H);
e=ones(n,1);
d=H*e;
d=d';
dang= d==0;

[y,p]=sort(dang,'ascend');
H=H(p,p);
d=d(p);

k=n-sum(dang);
dh=[1./d(1:k), ones(1,n-k)];
x=rand(1,k+1);
x=x/norm(x);
```

La sperimentazione. Implementazione dell'algoritmo

```
for i=1:itmax
s=x.*dh(1:k+1);
s(1:k)=a*s(1:k)*H(1:k,1:k)+(1-a)*v(1:k)+a*s(k+1)*w(1:k);
s(k+1)=1-(s(1:k)*ones(k,1));
        err=max(abs(s-x));
        x=s;
        disp([i,err])
            if err<1.e-13*max(x)
                break
            end
end
y=[s(1:k)
a*(s(1:k).*dh(1:k))*H(1:k,k+1:n)+
(1-a)*v(k+1:n)+a*s(k+1)*w(k+1:n)];
end
```

La sperimentazione. Matrici sparse generate randomicamente

```
function [PR,PRL]=Confronto(n,d)
H = sprand(n,n,d);
H= H~=0;
v = ones(n,1)'/n;
itmax=1000;
a=0.85;
w=ones(1,n);
w=w/sum(w);
dang=rn(H)
if (dang==0)
H(n,:)=0;
end
```

La sperimentazione. Matrici sparse generate randomicamente

```
function [PR,PRL]=Confronto(n,d)
H = sprand(n,n,d);
H= H~=0;
v = ones(n,1)'/n;
itmax=1000;
a=0.85;

w=ones(1,n);
w=w/sum(w);

dang=rn(H)
if (dang==0)
H(n,:)=0;
end
```

La sperimentazione. Matrici sparse generate randomicamente

```
function [PR,PRL]=Confronto(n,d)
H = sprand(n,n,d);
H= H~=0;
v = ones(n,1)'/n;
itmax=1000;
a=0.85;
w=ones(1,n);
w=w/sum(w);

dang=rn(H)

if (dang==0)
H(n,:)=0;
end
```

La sperimentazione. Matrici sparse generate randomicamente

```
function a=rn(H)
[m,n]=size(H);
e=ones(m,1);
d=H*e;
d=d';
dang= d==0;
a=sum(dang);
end
```

La sperimentazione. Matrici sparse generate randomicamente

```
function [PR,PRL]=Confronto(n,d)
H = sprand(n,n,d);
H= H~=0;
v = ones(n,1)'/n;
itmax=1000;
a=0.85;
w=ones(1,n);
w=w/sum(w);
dang=rn(H)

if (dang==0)
H(n,:)=0;
end
```

La sperimentazione. Matrici sparse generate randomicamente

```
tic
u = PageRank(H, v, a, itmax);
PR=toc
tic
v = lumpingPR(H, v, w, a, itmax);
PRL=toc
end
```

La sperimentazione. Matrici sparse generate randomicamente

n	nz	$dang$	t_{PR}	it_{PR}	t_L	it_L
1.000	100	905	0.005	13	0.007	13
1.000	1.000	368	0.05	170	0.08	163
1.000	10.000	0	0.017	25	0.031	29
10.000	1.000	9043	0.021	16	0.013	15
10.000	10.000	3662	0.23	172	0.24	159
10.000	100.000	0	0.14	26	0.25	30
100.000	10.000	90497	0.42	20	0.06	18
100.000	100.000	36737	1.5	54	1.13	52
100.000	1.000.000	9	2.06	26	3.32	31
1.000.000	100.000	904835	4.5	17	0.86	15
1.000.000	1.000.000	368127	60	163	35.6	129
1.000.000	10.000.000	44	45.8	26	62.4	31

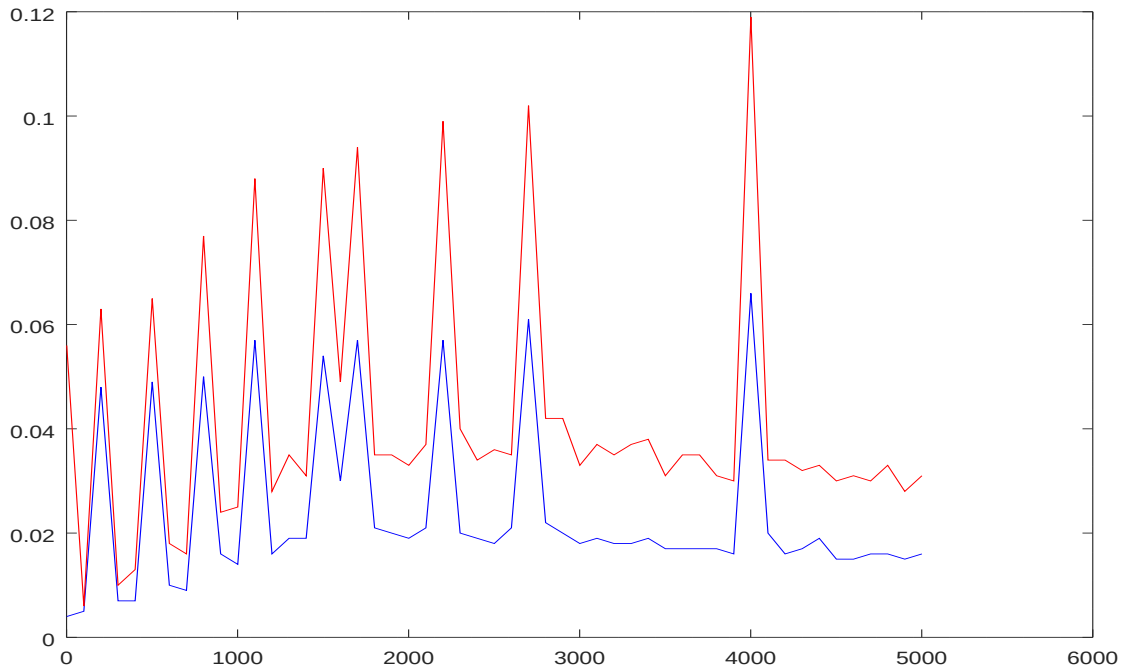
La sperimentazione. Matrici sparse generate randomicamente

n	nz	$dang$	t_{PR}	it_{PR}	t_L	it_L
1.000	100	905	0.005	13	0.007	13
1.000	1.000	368	0.05	170	0.08	163
1.000	10.000	0	0.017	25	0.031	29
10.000	1.000	9043	0.021	16	0.013	15
10.000	10.000	3662	0.23	172	0.24	159
10.000	100.000	0	0.14	26	0.25	30
100.000	10.000	90497	0.42	20	0.06	18
100.000	100.000	36737	1.5	54	1.13	52
100.000	1.000.000	9	2.06	26	3.32	31
1.000.000	100.000	904835	4.5	17	0.86	15
1.000.000	1.000.000	368127	60	163	35.6	129
1.000.000	10.000.000	44	45.8	26	62.4	31

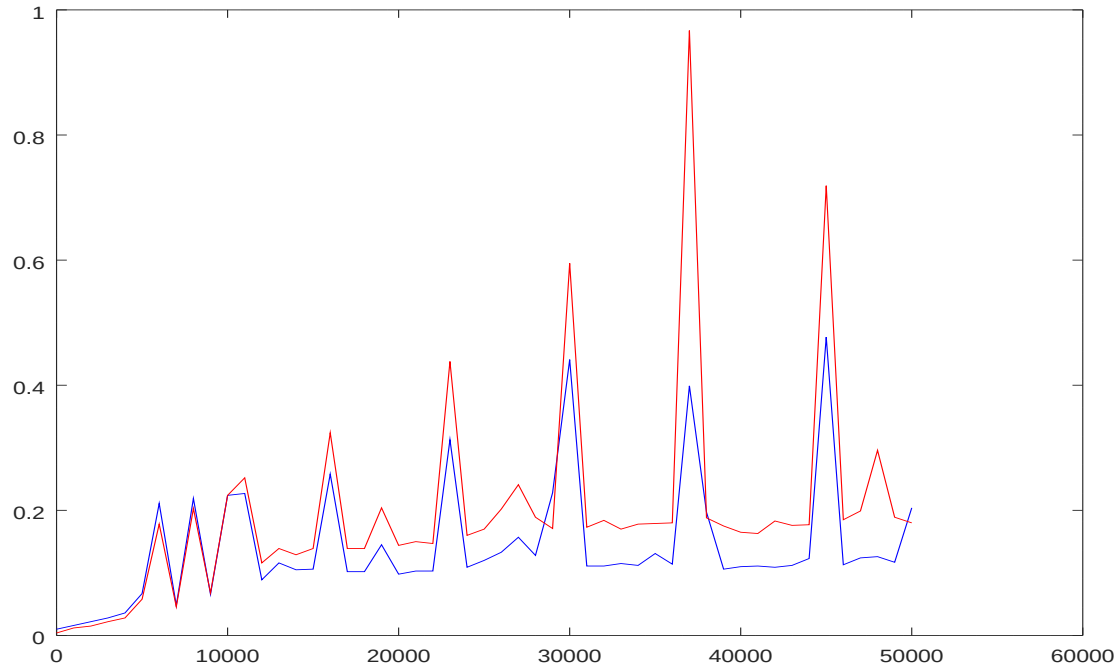
La sperimentazione. Efficienza degli algoritmi al variare del numero di dangling nodes

```
function dangtester(n)
nnz=[1:n/10:n*5+1];
for i=1:51
d(i)=nnz(i)/(n^2);
[PR,PRL]=Confronto(n,d(i));
pr(i)=PR;
l(i)=PRL;
end
plot(nnz,pr,'b')
hold on
plot(nnz,l,'r')
end
```

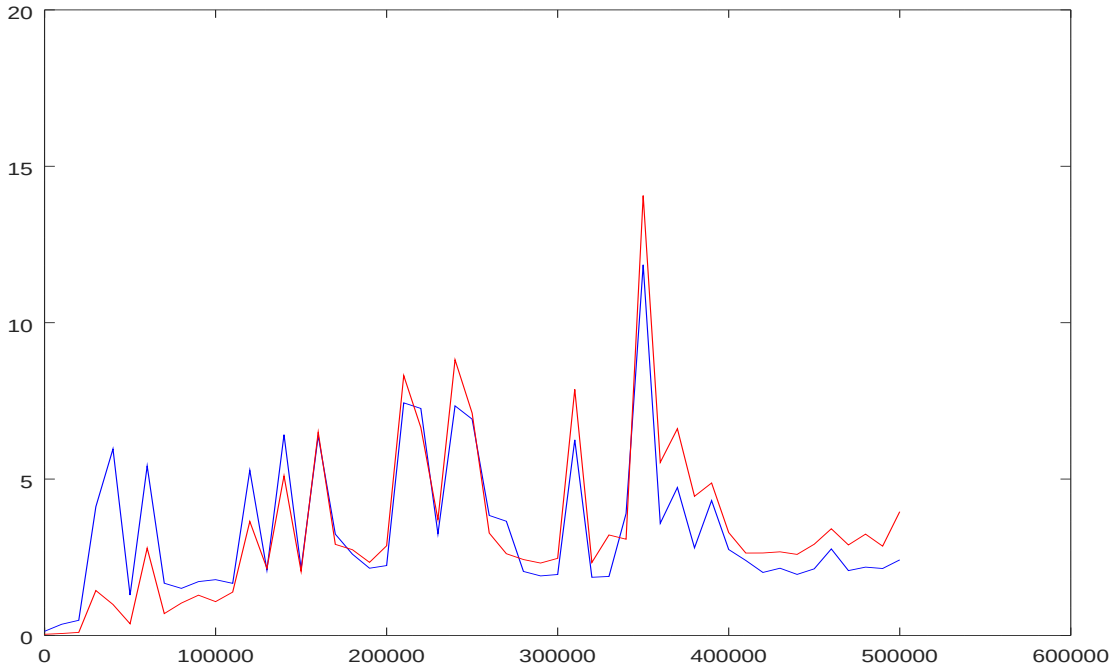
La sperimentazione. Efficienza degli algoritmi al variare del numero di dangling nodes n=1000



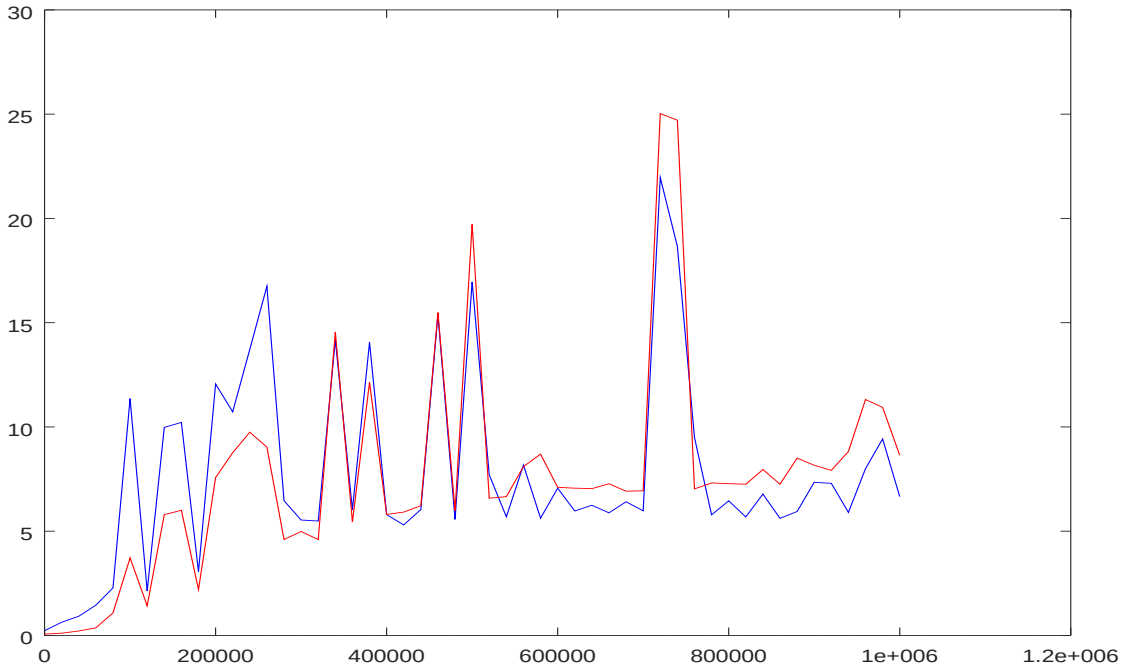
La sperimentazione. Efficienza degli algoritmi al variare del numero di dangling nodes n=10000



La sperimentazione. Efficienza degli algoritmi al variare del numero di dangling nodes n=100000



La sperimentazione. Efficienza degli algoritmi al variare del numero di dangling nodes n=200000



La sperimentazione. Due casi reali

```
function tester(A,n)
i=A(:,1);
j=A(:,2);
k=size(i)(1);
H=sparse(i,j,ones(k,1),n,n);
dang=rn(H)
v = ones(n,1)'/n;
itmax=1000;
w=ones(1,n);
w=w/sum(w);
a = 0.85;
tic
u = lumpingPR(H, v, w, a, itmax);
tempo_lumping=toc
tic
u = PageRank(H, v, a, itmax);
tempo_PageRank=toc end
```

La sperimentazione. Due casi reali

\	n	$dang$	t_{PR}	it_{PR}	t_L	it_L
Berkeley Stanford	685230	4744	36	168	84	191
Notre Dame	325729	187788	13.09	163	13.6	180

Bibliografia


Dario A.Bini, *Il problema del Page Rank*, Appunti del corso di Calcolo Scientifico, 2015


Ilse C.F. Ipsen and Teresa M. Selee, *PageRank computation, with special attention to dangling nodes*, Society for Industrial and Applied Mathematics, 2007


Jure Leskovec and Andrej Krevl, *SNAP Datasets: Stanford Large Network Dataset Collection*, `http://snap.stanford.edu/data`, 2014

Appendice

La function seguente permette di verificare che l'algoritmo di lumping fornisca effettivamente il vettore di PageRank

```
function verificaPR(n,nz)
%Definizione matrice
dens=nz/(n^2);
H=sprand(n,n,dens);
H= H~=0;
v = ones(n,1)'/n;
%Definizione parametri
itmax=1000;
a=0.85;
w=ones(1,n);
w=w/sum(w);
dang=rn(H)
```

Appendice

```
%Calcolo del vettore di permutazione
e=ones(n,1);
d=H*e;
d=d';
dang= d==0;
[y,p]=sort(dang,'ascend');
%Calcolo dei due vettori di PageRank
u = PageRank(H, v, a, itmax);
v = lumpingPR(H, v, w, a, itmax);
[a,l]=sort(u(p),'descend');
[b,g]=sort(v,'descend');
l-g
end
```