Indeksi centralnosti in algoritmi za njihov izračun

Domen Blenkuš, Matej Filip, Tilen Marc in Aljaž Zalar

19.4.2012

Kaj je indeks centralnosti?

- izmerimo intuitiven občutek, da so nekatere točke **bolj pomembne**
- centralnosti ne moremo enolično definirati
- predstavlja lahko **različne količine** (pomembnost, vplivnost, nadzor,...)

Primer uporabe - študentske volitve

- študentje letnika morajo izvoliti svojega predstavnika v letniku.
- točke v grafu predstavljajo študente
- povezava v grafu med točkama A in B lahko predstavlja:
  1. da je oseba A glasovala za osebo B
  2. da je oseba A prepričala osebo B, da je glasovala za njenega favoritov
  3. da je oseba A prepričala osebo B, da je glasovala za drugega kandidata, kot bi prvotno

Ohlapna definicija

- splošno sprejeta definicija centralnega indeksa zaradi splošne uporame ne obstaja
- bolj **pomembnim točkam** želimo prirediti **večjo vrednost**
- indeksi centralnosti naj bodo odvisni samo od **lege** točke v grafu

Definicija (Centralni indeks)

Naj bo $G = (V, E)$ *obtežen, usmerjen ali neusmerjen multigraf. Funkcija* s *se imenuje centralni indeks natanko takrat, ko velja* $\forall v \in V : G \simeq H \Rightarrow s_G(v) = s_H(\phi(v))$, *kjer* $s_G(v)$ *označuje vrednost* $s(v)$ *v grafu* G.

Sosedska centralnost

- uporablja se pri volilnih sistemih ali, če nas zanima iz katere točke lahko neposredno pridemo v največ ostalih točk

Definicija

Naj bo $G = (V, E)$ digraf. ***Vhodno sosedsko centralnost*** *točke $v \in V$ označimo z $c_{iD}(v)$ in je enaka številu vhodnih povezav točke v ($c_{iD}(v) = d^-(v)$). Podobno* ***izhodno sosedsko centralnost*** *označimo z $c_{oD}(v)$ in je enaka številu izhodnih povezav ($c_{oD}(v) = d^+(v)$).*

- to je primer **lokalne centralnosti**, saj je centralni indeks točke odvisen samo od njenih sosed

Ekscentrična centralnost

- bolnišnico bi radi postavili na tako točko na grafu, da bodo imeli reševalci čimkrajši najdaljši odzivni čas
- za vsako točko pogledamo najkrajše poti do vseh ostalih točk in ji priredimo največjo
- izberemo točko z najmanjšo vrednostjo

Definicija

Naj bo $G = (V, E)$ *graf.* ***Ekscentrična centralnost*** *točke* $v \in V$ *je*

$$c_E(v) = \frac{1}{max\{d(v,u); u \in V\}}$$

Bližinska centralnost

- nakupovalni center bi radi postavili na tako mesto v grafu, da kar se da minimiziramo potne stroške nakupovalcev
- za vsako točko pogledamo najkrajše poti do vseh ostalih točk in ji priredimo vsoto vseh
- izberemo točko z najmanjšo vrednostjo

Definicija

Naj bo $G = (V, E)$ graf. ***Bližinsko centralnost*** *točke $v \in V$ definiramo kot*

$$c_C(u) = \frac{1}{\sum_{v \in V} d(u, v)}.$$

Centralnost posrednika na najkrajši poti

- pove nam kakšnn vpliv ima določena točka na komunikacijo med drugimi točkami

Definicija

Indeks centralnosti posrednika na najkrajši poti *definiramo kot*

$$c_B(v) = \sum_{s \neq v \in V} \sum_{t \neq v \in V} \delta_{st}(v),$$

kjer $\delta_{st}(v)$ *predstavlja odstotek poti med* s *in* t*, ki gredo čez* v*.*

- v nasprotju z bližinsko centralnostjo jo lahko uporabljamo tudi na nepovezanih grafih

Spletna centralnost

- veliko ljudi uporablja svetovni splet za iskanje informacij
- zaradi njegove velike obsežnosti so potrebni zahtevni algoritmi za iskanje
- kako naj spletni iskalnik določi katere strani so najbolj primerne za določeno poizvedbo?
- naprednejši iskalniki poleg navadnega tekstovnega iskanja uporabljajo tudi indekse centralnosti

Model slučajnega spletnega uporabnika

- najenostavnejši model simulira njegovo obnašanje kot slučajni sprehod po grafu
- sprehod začne na slučajno izbrani spletni strani in nadaljuje po njenih sosedah. To predstavimo z matriko P
- spletni indeks centralnosti predstavlja verjetnost, da se bo uporabnik po velikem številu korakov znašel na določeni spletni strani
- Da bo verjetnost dobro definirana, moramo zahtevati, da:
  1. limita verjetnosti obstaja
  2. je neodvisna od začetne strani
- da bo to res, mora biti matrika P stohastična (matrika mora biti nerazcepna in vsota vsake vrstice mora biti 1)

Model slučajnega spletnega uporabnika

- Ker svetovni splet očitno ni krepko povezan, matrika P ni nujno stohastična
- Stohastičnost dosežemo z naslednjim postopkom:
  1. v vsaki ničelni vrstici vse vrednosti nastavimo na $\frac{1}{n}$ (uporabnik slučajno skoči na neko spletno stran)
  2. matriko E, ki je enake velikosti kot P' in ima vse elemente enake $\frac{1}{n}$ (*matrika naključnega skoka*).
  3. izberemo neko konveksno kombinacijo matrik P' in E. $P' = \alpha P' + (1 - \alpha)E$, kjer je $\alpha \in [0, 1]$.

Indeks PageRank

- naprednejši iskalniki, med njimi tudi Google, ga uporabljajo v svojih algoritmih
- stran ocenimo samo glede na njene topološke značilnosti (samo glede na lego v spletu, neodvisno od vsebine)
- indeks strani je odvisen samo od njenih sosedov in njihove pomembnosti

Indeks PageRank

- določanje indeksov tako postane rekurzivni problem podan z enačbami:

$$c_{PR}(p) = d \sum_{q \in \Gamma_p^-} \frac{c_{PR}(q)}{d^+(q)} + (1 - d),$$

kjer je $c_{PR}(q)$ PageRank strani q, d pa je obtežitveni faktor.

- zapisano v matrični obliki se to glasi:

$$c_{PR} = dPc_{PR} + (1 - d)1_n$$

- Ta sistem ponavadi rešimo s preprosto Jacobijevo iteracijo:

$$c_{PR}^k = dPc_{PR}^{k-1} + (1 - d)1_n$$

Pomembnost povezave v omrežju

- če bi iz grafa radi izključili eno povezavo, moramo to narediti tako, da komunikacije med točkami ne bomo preveč ohromili
- najbolje je, da izključimo povezavo, ki najmanj poveča najkrajšo pot med katerima koli dvema točkama v omrežju

Definicija

Naj bo $G = (V, E)$ graf. ***Indeks popembnosti v omrežju*** *povezave $e \in E$ je*

$$c_P(e) = \max_{u,v \in V} \{d_{G \setminus e}(u, v) - d_G(u, v)\}$$

Osnovni algoritmi za izračun indeksov centralnosti

- uporabnost indeksov centralnosti je odvisna od njihove izračunljivosti
- želimo algoritem, ki vrne rezultat v polinomskem času
- za ogromne grafe (npr. graf svetovnega spleta) je polinomska zahtevnost preveč; uporabimo aproksimacijo, ki jo lahko izračunamo z linearno zahtevnostjo
- grafi se s časom spreminjajo, zato želimo algoritem, ki lahko hitro popravlja indekse

Iskanje najkrajše poti

- večina indeksov centralnosti je povezana z problemom najkrajših poti med vozlišči
- naiven pristop je, da najprej izračunamo najkrajše poti in potem indekse
- pristop je dober, saj vrne rezultat v povprečno nekje med n^2 in n^3 korakih
- za večja omrežja potrebujemo specializirane algoritme

SSSP

- osnovni problem je iskanje najkrajših poti iz izbranega izhodišča do ostalih vozlišč, znan pod imenom Single Source Shortest Path (SSSP)
- prvi algoritem, ki reši problem v polinomskem času, je Dijkstrov algoritem (1959)
- zahtevnost $O(m + n \log(n))$
- primeren za grafe s pozitivnimi utežmi na povezavah

function DIJKSTRA($G = (V, E)$, ω)
    $P = \emptyset$
    $T = V$
    $d(s, v) = \infty$ za vse $v \in V$, $d(s, s) = 0$
    while $P \neq V$ **do**
        $v =$ vozlišče z najmanjšim $d(s, v)$
        $P = P \cup \{v\}$, $T = T \backslash \{v\}$
        for u je sosed v in $u \in T$ **do**
            if $d(s, u) > d(s, v) + \omega(v, w)$ **then**
                $d(s, u) = d(s, v) + \omega(v, w)$
                $pred(u) = v$
            end if
        end for
    end while
end function

APSP

- problem iskanja najkrajših poti med vsemi vozlišči, znan pod imenom All-Pairs Shortest Paths problem (APSP)
- prva možnost: za vsako vozlišče izvedemo Dijkstrov algoritem (preveč potratno)
- druga možnost: Floyd-Warshallov algoritem
- zahtevnost $O(n^3)$

function FLOYD-WARSHALL($G = (V, E)$, ω)
    $d(v, v) = 0$
    $d(v, u) = \omega(v, u)$, če $\{v, u\} \in E$, sicer $d(v, u) = \infty$
    $pred(u, v) = u$, če $\{v, u\} \in E$, sicer $pred(u, v) = 0$
    for $v \in V$ **do**
        for $u \in V$ **do**
            for $w \in V$ **do**
                if $d(u, w) > d(u, v) + d(v, w)$ **then**
                    $d(u, w) = d(u, v) + d(v, w)$
                    $pred(u, w) = pred(v, w)$
                end if
            end for
        end for
    end for
end function

Dinamični APSP

- graf se lahko hitro spreminja
- želimo algoritem, ki hitro izračuna nove indekse iz starih podatkov
- to je veliko težji problem in je še aktualen

Iskanje lastnih vrednosti matrik

- v splošnem težak problem
- za grafe z malo povezavami dobimo t. i. sparse matrike
- veliko različnih algoritmov, primer preprostega algoritma za izračun največje lastne vrednosti je potenčna metoda

Indeks centralnosti posrednika na najkrajši poti

Definicija

Indeks centralnosti posrednika na najkrajši poti definiramo kot

$$c_B(v) = \sum_{s \neq v} \sum_{t \neq v} \delta_{st}(v).$$

Prilagojeni Dijkstrov algoritem

function $\mathrm{SSSP}(G = (V, E), \omega)$
$P = \emptyset$
$T = V$
$d(s, v) = \infty$ for all $v \in V$, d(s,s)=0
while $P \neq V$ **do**
$v =$ vozlišče z najmanjšim $d(s, v)$
$P = P \cup \{v\}$, $T = T \backslash \{v\}$
for u je sosed v in $u \in T$ **do**
if $d(s, u) > d(s, v) + \omega(v, u)$ **then**
$d(s, u) = d(s, v) + \omega(v, u)$
$\mathrm{pred}(s, u) = v$
$\sigma_{su} = \sigma_{sv}$
end if
if $d(s, u) = d(s, v) + \omega(v, u)$ **then**
$\mathrm{pred}(s, u).append(v)$
$\sigma_{su} += \sigma_{sv}$
end if
end for

Odvisnost vozlišča

Definicija

Odvisnost vozlišča $s \in V$ od vozlišča $v \in V$ označimo z δ_{s*} in jo definiramo kot

$$\delta_{s*}(v) = \sum_{t \in V} \delta_{st}(v).$$

Izrek

Za odvisnost $\delta_{s*}(v)$ vozlišča s od poljubnega vozlišča v velja formula

$$\delta_{s*}(v) = \sum_{w : v \in pred(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s*}(w)).$$

Odvisnost para vozlišč

Definicija

Odvisnost para vozlišč $s, t \in V$ od vozlišča $v \in V$ označimo z $\delta_{st}(v)$ in jo definiramo kot

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Izbojšana časovna zahtevnost

Izrek

Indeks centralnosti vseh vozlišč lahko izračunamo s časovno zahtevnostjo $O(mn + n^2 \log n)$.

Hitri približki algoritmov

- Do sedaj smo spoznali nekaj eksaktnih algoritmov za izračun nekatarih indeksov centralnosti. Izračunljivi so v polinomskem času, kar je za velika omrežja problem.
- Zato je naš cilj samo približen izračun vrednosti opazovanega indeksa.
- Pri tem si želimo doseči dober kompromis med časom izvajanja in natančnostjo izračuna.
- V nadaljevanju bomo spoznali nekaj tehnik za doseganje tega kompromisa.

Eppstein-Wangova aproksimacija bližinske centralnosti 1

- Bližinska centralnost: $c_C(v) = \frac{\sum_{x \in V} d(v,x)}{n-1}$.
- Nočemo seštevati po vseh vozliščih $x \in V$.
- Izboljšava je naslednji algoritem:

Eppstein-Wangov algoritem za izračun bližinske centralnosti

1. Slučajno izberi K vzorčnih vozlišč $v_1, v_2, \ldots, v_K$.
2. Za vsako vzorčno vozlišče v_i reši problem vseh parov najkrajših poti.
3. Za vsako vozlišče v izračunaj $\hat{c}_C(v) = \frac{n}{K} \frac{\sum_{i=1}^{K} d(v,v_i)}{n-1}$.

Eppstein-Wangova aproksimacija bližinske centralnosti 2

- Ključno vprašanje je koliko vzorčnih vozlišč potrebujemo, da ocenimo indekse c_C z željeno natančnostjo.
- Na to vprašanje se da odgovoriti s Hoeffdingovo oceno iz teorije verjetnosti:

Lema (Hoeffdingova ocena)

Naj bodo spremenljivke $x_1, x_2, \ldots, x_K$ neodvisne in naj velja $a_i \leq x_i \leq b_i$, kjer so $a_i, b_i \in \mathbb{R}$. Če z μ označimo povprečje $E\left(\frac{\sum x_i}{K}\right)$, potem za vsak $\epsilon > 0$ velja

$$P\left(\left|\frac{\sum x_i}{K} - \mu\right| \geq \epsilon\right) \leq 2 \cdot e^{\frac{-2K^2\epsilon^2}{\sum (b_i - a_i)^2}}.$$

Eppstein-Wangova aproksimacija bližinske centralnosti 3

- Če v zadnjo lemo vstavimo $x_i = \frac{nd(v_i,u)}{n-1}$, $\mu = c_C(v)$, $a_i = 0$ in $b_i = \frac{n\Delta}{n-1}$, dobimo

$$\begin{aligned} P\left(\left|\frac{\sum x_i}{K} - \mu\right| \geq \epsilon\right) &= P\left(|\hat{c}_C(v) - c_C(v)| \geq \epsilon\right) \leq \\ &\leq 2 \cdot e^{\frac{-2K^2\epsilon^2}{\sum (b_i - a_i)^2}} \\ &= 2 \cdot e^{\frac{-2K^2\epsilon^2}{K\sum (\frac{n\Delta}{n-1})^2}} = \\ &= 2 \cdot e^{-2\left(\frac{n-1}{n}\right)^2\left(\frac{K\epsilon^2}{\Delta^2}\right)} \end{aligned}$$

- Naprej za poljuben $\hat{\epsilon} > 0$ postavimo še $\epsilon = \hat{\epsilon}\Delta$ in uporabimo $K = \frac{n^2}{(n-1)^2}\frac{\log n}{2\hat{\epsilon}^2}$ vzorcev, za nekaj algebre v zgornjem izrazu hitro ugotovimo, da je verjetnost za napako večjo od $\hat{\epsilon}$ največ $\frac{2}{n}$.

Eppstein-Wangova aproksimacija bližinske centralnosti 4

Kaj smo s tem algoritmom **pridobili na časovni zahtevnosti**?

- Časovna zahtevnost APSP algoritma je $O(n(n+m))$ za neutežene grafe in $O(n(m+n\log n))$ za utežene grafe.
- Če pa naredimo samo K SSSP algoritmov, dobimo časovni zahtevnosti $O(K(\hat{\epsilon})(n+m))$ in $O(K(\hat{\epsilon})(m+n\log n))$.
- Torej je faktor izboljšanja $\frac{K(\hat{\epsilon})}{n}$.
- V algoritmu je od uporabnika odvisno, kakšno ceno - v smislu velikosti $K(\hat{\epsilon})$ - je pripravljen plačati za veliko verjetnost majhne napake.

Aproksimacija spletnih centralnosti

Najpomembnejši indeks centralnosti v grafu spleta večine spletnih iskalnikov je PageRank. Zato se bomo posvetili predvsem pospešitvenim tehnikam za izračun tega. Obstaja več metod za pospešitev:

1. aproksimacija s cenejšimi izračuni (običajno se izognemu množenju z matriko),
2. pospešitev konvergence,
3. reševanje linearnega sistema enačb namesto reševanja problema lastnih vrednosti,
4. uporaba dekompoziciji grafa spleta,
5. nadgradnja namesto ponovnega računanja.

Pospešitev konvergence 1

- Osnova za to pospešitveno metodo je potenčna metoda za določitev dominantnega lastnega vektorja.
- Ker vsaka iteracija potenčne metode zahteva množenje z matriko A, ki je za graf spleta zelo drago, je naš cilj zmanjšati število iteracij.
- Aitkenova metoda, temelji na predpostavki, da lahko približek $x^{(k-2)}$ zapišemo kot linearno kombinacijo prvih dveh lastnih vektorjev, to sta u in v. S to predpostavko sta tudi naslednja dva približka $x^{(k-1)}$ in $x^{(k)}$ linearni kombinaciji prvih dveh lastnih vektorjev:

$$x^{(k-2)} = u + \alpha v,$$

$$x^{(k-1)} = u + \alpha \lambda_2 v,$$

$$x^{(k)} = u + \alpha \lambda_2^2 v.$$

Pospešitev konvergence 2

- Če definiramo

$$y_i = \frac{\left(x_i^{(k-1)} - x_i^{(k-2)}\right)^2}{x_i^{(k)} - 2x_i^{(k-1)} + x_i^{(k-2)}},$$

z nekaj algebre pridemo do $y = \alpha v$ in tako

$$u = x^{(k-2)} - y.$$

- Sedaj na tem izvajamo potenčno metodo.
- Obstajajo tudi posplošitve Aitkenove metode, ko predpostavimo, da je približek $x^{(k-2)}$ linearna kombinacija prvih k lastnih vektorjev in izpeljemo podobne algoritme pravkar predstavljenemu.

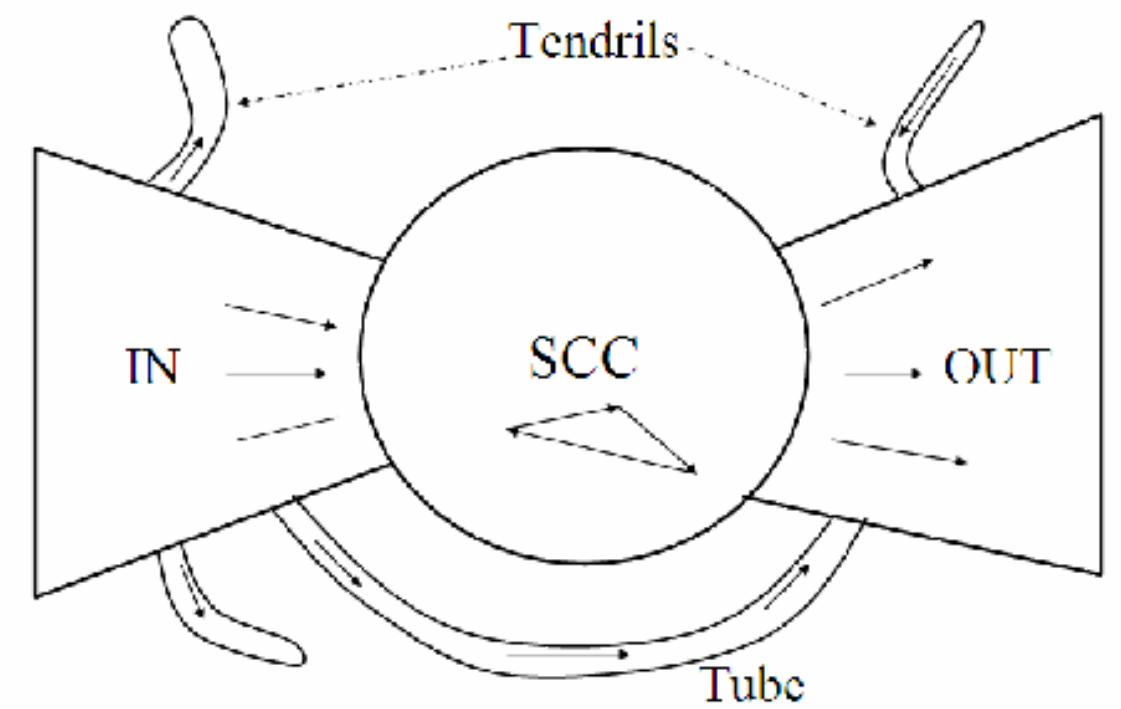
Dekompozicijske tehnike

- Graf spleta je zelo velik in vsakodnevno raste.
- Zato so raziskovalci predlagali, da se ga razbije na več manjših komponent.
- Ideja je izračun centralnosti znotraj posamezne komponente, nato pa pametno utežiti komponente in izračunat centralnosti v celem grafu.
- Obstaja veliko bločnih razbitij. V nadaljevanju si bomo ogledali dva primera:
  1. graf spleta kot “metuljček”,
  2. Kamvarjevo bločno razbitje.

Graf spleta kot “metuljček” 1

- Do razbitja grafa spleta na obliko “metuljček” so leta 1999 po preiskovanju velikega števila spletnih strani in njihovih povezav prišli Broder z ekipo leta 1999.
- Graf spleta ima obliko metuljčka.
- Obstajajo tri velike komponente
  - velika strogo povezana komponenta,
  - strani, ki kažejo na to komponento, jih pa ne moremo doseči iz nje,
  - strani, ki jih lahko dosežemo iz te velike komponente, vendar se nanjo ne moremo vrniti.
- Opazimo še nekaj tipov manjših komponent, to so lovke, povezave med dvema komponentama in izolirani kosi.

Graf spleta kot “metuljček” 2



Slika : Graf spleta po Broderju

Graf spleta kot "metuljček" 3

- Kako izrabiti to strukturo za pospešitev izračuna PageRanka?
- Ideja je, da ima sedaj prehodna matrika veliko preprostejšo obliko, tj. bločno zgornje trikotno:

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & \dots & P_{1K} \\ 0 & P_{22} & P_{23} & \dots & P_{2K} \\ \vdots & \ddots & P_{33} & \dots & P_{3K} \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & P_{KK} \end{bmatrix} \bigoplus Q_1 \bigoplus Q_2 \bigoplus \dots \bigoplus Q_L.$$

Graf spleta kot "metuljček" 4

Enako delitev naredimo tudi za vektor c_{PR} in velik problem razdelimo na več manjših ter rešujemo s pomočjo obratne substitucije:

$$(I - dQ_L)c_{PR,K+L} = (1-d)1_{m_L},$$

$$\vdots$$

$$(I - dQ_1)c_{PR,K+1} = (1-d)1_{m_1},$$

$$(I - dP_{KK})c_{PR,K} = (1-d)1_{n_K},$$

$$\vdots$$

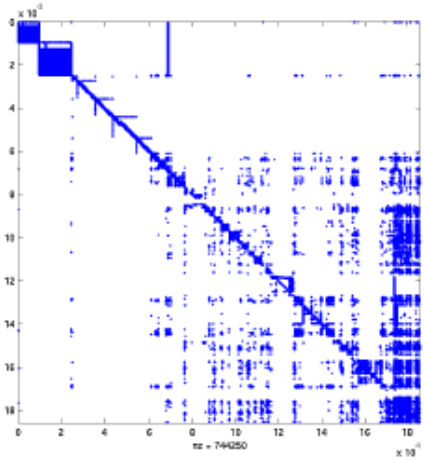
$$(I - dP_{ii})c_{PR,i} = (1-d)1_{n_i} + d\sum_{j=i+1}^{K} c_{PR,i}.$$

Kamvarjeva gnezdeno bločna oblika grafa spleta 1

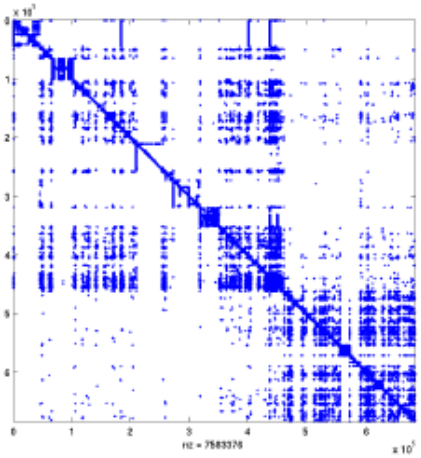
Še bolj razčlenjeno bločno oblika grafa spleta pa so leta 2001 odkrili in izkoristili *Kamvar* s svojo ekipo. Ugotovili so naslednjo strukturo grafa spleta:

1. Graf spleta ima *bločno strukturo*.
2. Bloki so *veliko manjši* kot je celoten graf.
3. Znotraj blokov obstajajo še *gneznedi bloki*, ki ustrezajo domenam, gostiteljem in poddirektorijem v imenu strani.

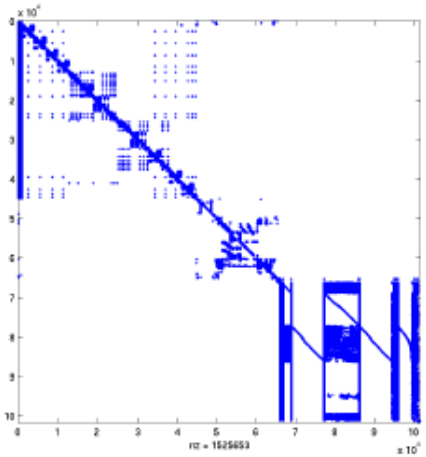
Kamvarjeva gnezdeno bločna oblika grafa spleta 2



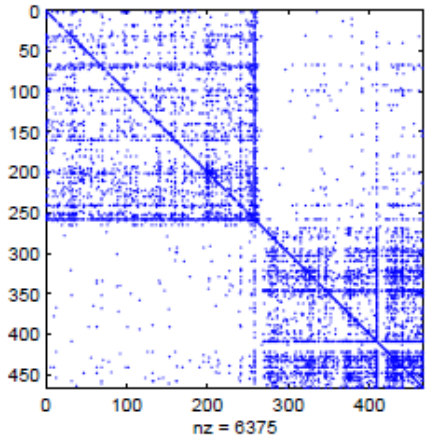
(a) IBM



(b) Stanford/Berkeley



(c) Stanford-50



(d) Stanford/Berkeley Host Graph

Slika : Nekaj grafov s Kamvarjevo analizo

Kamvarjeva gnezdeno bločna oblika grafa spleta 3

Na podlagi tega so predlagali naslednjo izboljšavo PageRank algoritma, ki so jo imenovali BlockRank algoritem:

1. Razbij graf spleta na bloke $I \in \Gamma$, kjer je Γ indeksna množica.
2. Za vsak blok I izračunaj lokalne PageRank indekse $c_{PR(I)}(i)$ za vsak $i \in I$.
3. Naredi utežitev lokalnih indeksov, tako da upoštevaš pomembnost posameznega bloka.
4. Uporabi standardni PageRank algoritem z začetnim vektorjem iz prejšnjih dveh korakov.

Kamvarjeva gnezdeno bločna oblika grafa spleta 4

- V točki 1. in 3. zgornjega algoritma lahko uporabimo običajen PageRank algoritem.
- Glavni problem je, kako formalizirati točko 2.
- To se naredi tako, da se grafu spleta priredi bločni graf. Vsakemu bloku $I \in \Gamma$ ustreza vozlišče v tem grafu. Med vozliščema I in J obstaja usmerjena povezava $\vec{IJ}$ natanko tedaj, ko v prvotnem grafu obstaja povezava $\vec{ij}$ za neki vozlišči $i \in I$ in $j \in J$. Pri tem dopuščamo tudi zanke $\vec{II}$. Teže povezav ω_{IJ} pa določimo kot:

$$\omega_{IJ} = \sum_{i \in I, j \in J} a_{ij} c_{PR(I)}(i),$$

kjer je a_{ij} teža povezave $\vec{ij}$ v prvotnem grafu, $c_{PR(I)}(i)$ pa lokalni PageRank indeks vozlišča i v bloku I.

Kamvarjeva gnezdeno bločna oblika grafa spleta 5

- Sedaj lahko uporabimo običajen PageRank algoritem za bločni graf B, da dobimo PageRank indekse povezav b_I.
- Iz korakov 2 in 3 dobimo začetni vektor za korak 3, ki je

$$c_{PR}^{(0)}(i) = c_{PR(I)}(i) b_I.$$

- Naredimo še en PageRank in smo končali.

Kamvarjeva gnezdeno bločna oblika grafa spleta 6

Kaj je prednost bločnega PageRank algorima v primerjavi s PageRankom?

1. Glavna prednost je pohitritev algoritma, ki izvira iz dejstva, da ni potrebno hraniti celotnega PageRank vektorja v spominu računalnika, saj ta sestoji iz nekaj bilijonov vhodov.
2. V BlockRank algoritmu veliko lokalnih PageRank vektorjev *hitro skonvergira*. Zato več časa lahko posvetimo slabo-konvergirajočim vektorjem. To so taki vektorji, ki pripadajo dobro gnezdenim blokom.
3. Lokalne PageRank vektorje lahko *računamo vzporedno* na več procesorjih, ki potem rezultate pošljejo glavnemu procesorju. Hkrati se izkaže tudi, da korak 1 v algoritmu lahko naredimo vzporedno s korakom 0, tj. določanjem prehodne matrike $A = (a_{ij})$.
4. V veliko primerih lahko lokalne PageRank indekse uporabimo za *dinamično računanje*. Torej če se zgodijo neke spremembe v grafu spleto samo za blok I, potem moramo ponoviti lokalni

Kamvarjeva gnezdeno bločna oblika grafa spleta 7

Kaj pa pravijo eksperimentalni rezultati?

Web Page	Local	Global
http://aa.stanford.edu	0.2196	0.4137
http://aa.stanford.edu/aeroastro/AAfolks.html	0.0910	0.0730
http://aa.stanford.edu/aeroastro/AssistantsAero.html	0.0105	0.0048
http://aa.stanford.edu/aeroastro/EngineerAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/Faculty.html	0.0459	0.0491
http://aa.stanford.edu/aeroastro/FellowsAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/GraduateGuide.html	0.1244	0.0875
http://aa.stanford.edu/aeroastro/Labs.html	0.0387	0.0454
http://aa.stanford.edu/aeroastro/Links.html	0.0926	0.0749
http://aa.stanford.edu/aeroastro/MSAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/News.html	0.0939	0.0744
http://aa.stanford.edu/aeroastro/PhdAero.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/aacourseinfo.html	0.0111	0.0039
http://aa.stanford.edu/aeroastro/aafaculty.html	0.0524	0.0275
http://aa.stanford.edu/aeroastro/aalabs.html	0.0524	0.0278
http://aa.stanford.edu/aeroastro/admitinfo.html	0.0110	0.0057
http://aa.stanford.edu/aeroastro/courseinfo.html	0.0812	0.0713
http://aa.stanford.edu/aeroastro/draftcourses.html	0.0012	0.0003
http://aa.stanford.edu/aeroastro/labs.html	0.0081	0.0044
http://aa.stanford.edu/aeroastro/prospective.html	0.0100	0.0063
http://aa.stanford.edu/aeroastro/resources.html	0.0112	0.0058
http://aa.stanford.edu/aeroastro/visitday.html	0.0123	0.0068

Slika : Primerjava lokalnih in globalnih PageRank indeksov bloka

Kamvarjeva gnezdeno bločna oblika grafa spleta 8

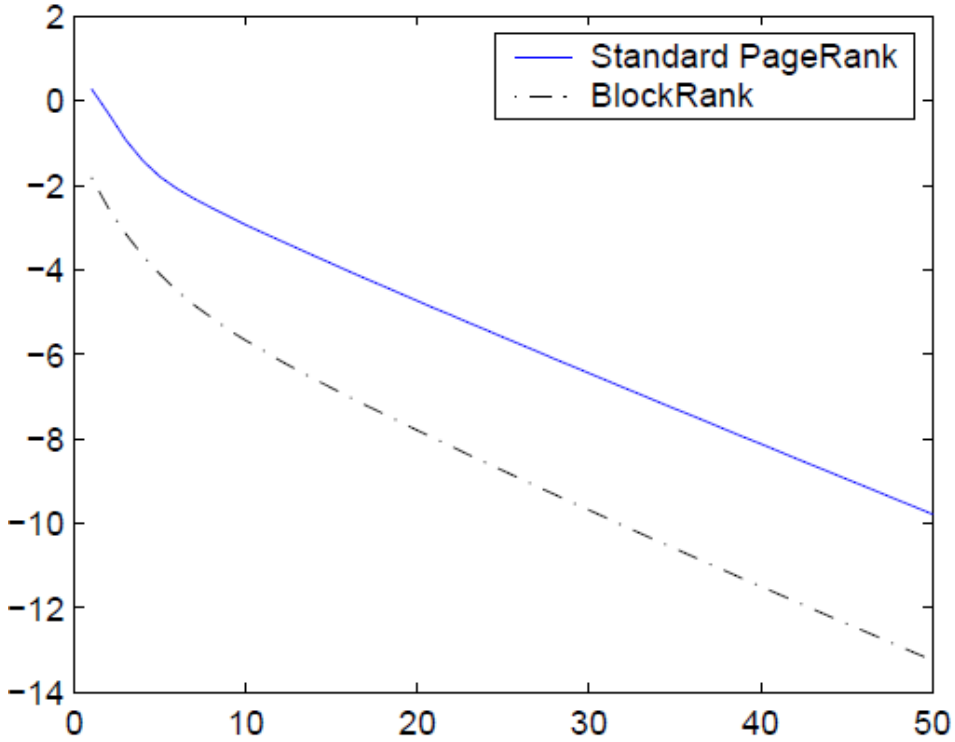
Step	Wallclock time
1	17m 11s
2	7m 40s
3	0m 4s
4	56m 24s
Total	**81m 19s**

Slika : Časovna zahtevnost posameznega koraka za LARGEWEB graf

Algorithm	Wallclock time
Standard	180m 36s
Standard (using url-sorted links)	87m 44s
BlockRank (no pipelining)	81m 19s
BlockRank (w/ pipelining)	57m 06s

Slika : Primerjava metod za LARGEWEB graf

Kamvarjeva gnezdeno bločna oblika grafa spleta 9



Slika : Primerjava metod po številu iteracij za LARGEWEB graf