math.e

Hrvatski matematički elektronički časopis

Implementacija PageRank algoritma metodom potencija u GNU Octaveu

internet tražilice **page rank** **podatkovno inženjerstvo**

Marko Hajba, Luka Majcenić, Nikola Stjepanović
Veleučilište u Virovitici

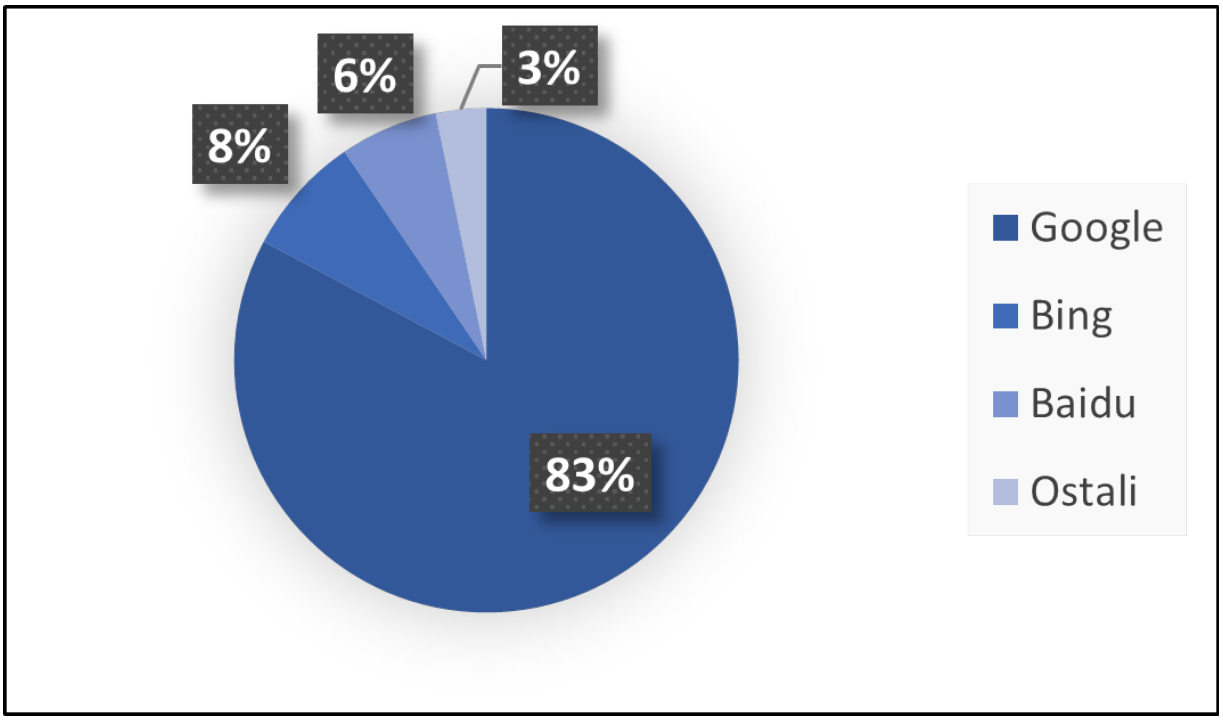
Sažetak

Internetska tražilica ili pretraživač predstavlja sustav koji na temelju traženog pojma rangira stranice po njihovoj relevantnosti za navedeni pojam. S obzirom na kontinuirani brzi porast broja internetskih stranica, odmah je jasno kako je to vrlo zahtjevan problem. Unatoč tome, najpoznatiji internetski pretraživači daju rezultate unutar nekoliko milisekundi, što je motivacija za upoznavanje s PageRank algoritmom za rangiranje internetskih stranica i testiranje njegove implementacije bazirane na metodi potencija u besplatnom programskom paketu GNU Octave.

Ključne riječi: internetski pretraživač, PageRank algoritam, metoda potencija, GNU Octave

1 Uvod

Moderni život gotovo je nezamisliv bez interneta, koji omogućava nikada brži i jednostavniji pristup informacijama. Veliku ulogu u tome imaju internetski pretraživači (tražilice). Internetski pretraživač ili tražilica softverski je sustav koji je osmišljen za provođenje internetskog pretraživanja, tj. sustavnog pretraživanja *World Wide Weba* s ciljem pronalaska određenih podataka navedenih u tekstualnom upitu [3]. Na temelju unesenog pojma pretraživač daje popis relevantnih internetskih stranica, pri čemu, što je stranica relevantnija za uneseni pojam, to se nalazi bliže vrhu rezultata pretraživanja. Prvi pretraživači devedesetih godina prošlog stoljeća su rangirali stranice na temelju indeksiranja – prolazili su kroz svaku stranicu i brojali koliko puta se traženi pojam pojavljuje na određenoj stranici. Takav način u moderno vrijeme je neefikasan, jer je s vremenom značajno porastao broj internetskih stranica te je stoga postalo vremenski neefikasno prolaziti kroz svaku internetsku stranicu. Također, takav pristup nije dobar prikaz relevantnih stranica, jer neka stranica može sadržavati uneseni pojam puno puta, bez da sadrži korisne informacije o njemu. Moderni pretraživači, npr. Googleova tražilica, koriste drukčije pristupe pri rangiranju internetskih stranica. Rangiranje internetskih stranica od strane Googlea obuhvaća brojne kriterije (često se navodi 200 ili više kriterija), kao što je brzina učitavanja stranice, povezanost, posjećenost, sigurnost, lokacija itd. [5]. PageRank algoritam spominje se u nekoliko kriterija, a uzima u obzir posjećenost, važnost i popularnost neke internetske stranice. Google je najpopularniji internetski pretraživač na svijetu. Svakodnevno se izvršava više od 3,5 milijardi pretraga odnosno 40 tisuća svake sekunde [4]. Prema statistici NetMarket Sharea, za razdoblje od svibnja 2019. do travnja 2021. godine, Google zauzima daleko najveći udio na tržištu internetskih pretraživača sa 82,80% korisnika. Kineski Baidu je sljedeći po popularnosti s 7,70%, a odmah iza njega nalazi se Bing (6,28%) [9]. Na Slici 1 prikazana je zastupljenost internetskih pretraživača za navedeno razdoblje.



Slika 1: Zastupljenost internetskih pretraživača za razdoblje 05.2019. – 04.2021 [9]

.

2 Metoda potencija

Metoda potencija najjednostavnija je metoda za određivanje najveće (po apsolutnoj vrijednosti) svojstvene vrijednosti matrice i pripadnog svojstvenog vektora. Algoritam metode potencija je vrlo jednostavan i iterativan, ali ponekad sporo konvergira. U slučaju rijetko popunjene matrice A (matrica u kojoj većina elemenata ima vrijednost nula), metoda potencija je vrlo brza. Neka je matrica A matrica dijagonalizabilna i neka ima jedinstvenu po apsolutnoj vrijednosti najveću svojstvenu vrijednost. Tada su svojstvene vrijednosti matrice $A\lambda_1, \lambda_2, \ldots, \lambda_n$ i neka su poredane tako da vrijedi

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|.$$

Neka je $x^{(0)}$ početni vektor. Tada se metoda potencija svodi na množenje matrice A vektorom $x^{(k)}$ u svakoj iteraciji, tj.

$$x^{(k+1)} = Ax^{(k)}. \qquad (1)$$

Budući da elementi umnoška $Ax^{(k)}$ mogu postati jako veliki ili jako mali brojevi, potrebno je normirati vektore $x^{(k)}, k = 0, 1, \ldots$. Algoritam metode potencija možemo zapisati kao u Algoritmu 1.

Algoritam 1 Metoda potencija za računanje svojstvenog vektora jedinstvene najveće po apsolutnoj vrijednosti svojstvene vrijednosti

Ulazi: Matrica A sa svojstvenim vrijednostima $\lambda_1, \lambda_2, \ldots, \lambda_n$, pri čemu vrijedi $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|$; početni vektor $x^{(0)}$

Izlaz: : svojstveni vektor koji pripada svojstvenoj vrijednosti λ_1

1: $y^{(0)} = \frac{x^{(0)}}{\|x^{(0)}\|}$
2: $k = 0$
3: **ponavljaj**
4: $\quad x^{(k+1)} = Ay^{(k)}$
5: $\quad y^{(k+1)} = \frac{x^{(k+1)}}{\|x^{(k+1)}\|}$
6: $\quad k = k + 1$
7: **do konvergencije**

Kriterij konvergencije može biti zadani broj iteracija ili tolerancija na razliku između rješenja dvije susjedne iteracije.Nešto više o metodi i njenoj konvergenciji može se pronaći u [2].

3 PageRank algoritam

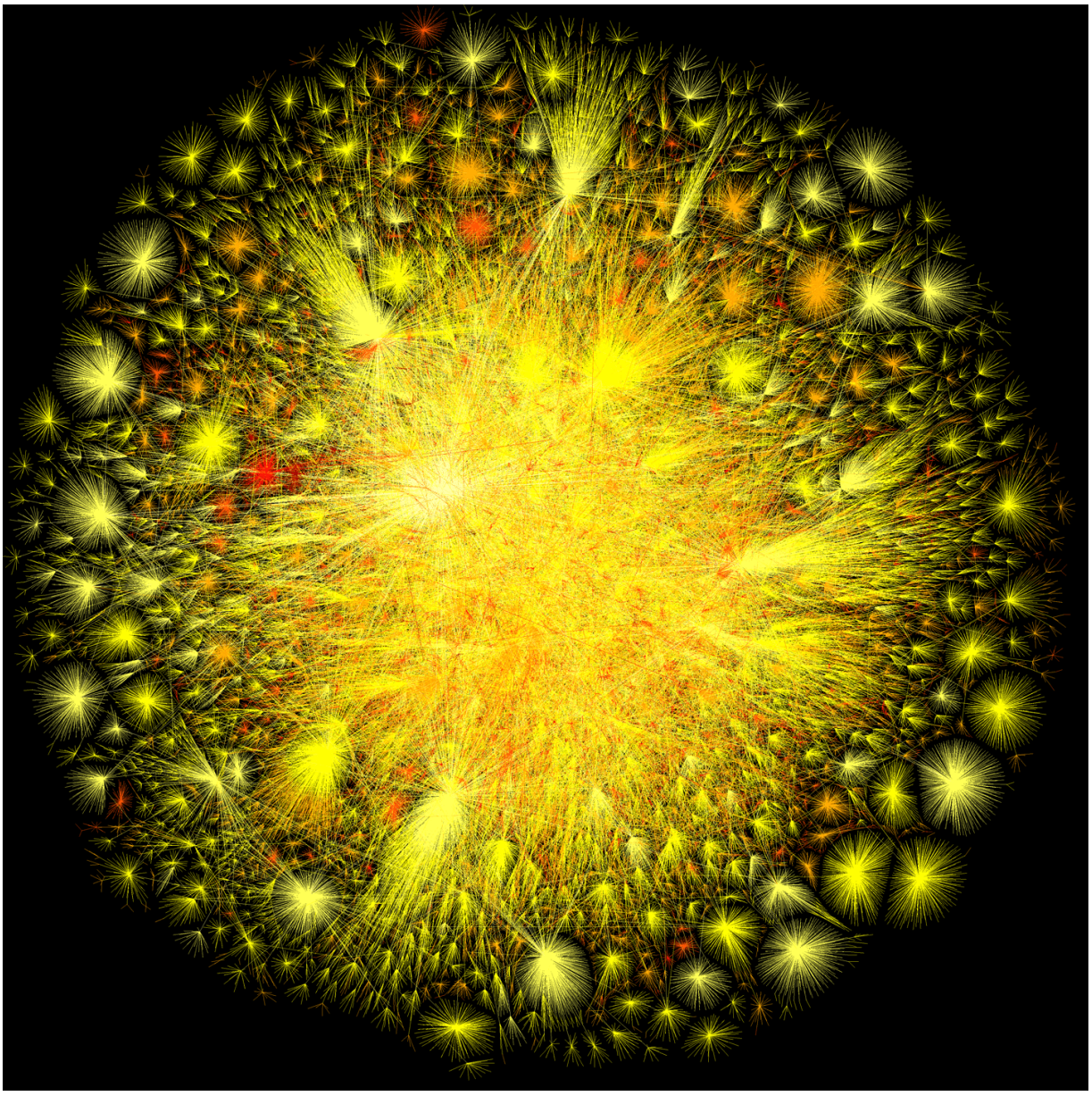
PageRank je algoritam koji koristi tvrtka Google u svom internetskom pretraživaču prilikom rangiranja internetskih stranica. PageRank algoritam dobio je ime po Larryu Pageu, jednom od osnivača Googlea. U ranim devedesetima prošlog stoljeća, kada su internetski pretraživači koristili samo tekstualne sustave rangiranja postojao je niz problema. Potraga za pojmom kao što je "Matematika" bila je problematična. Najviše rangirana stranica koju bi prikazivala jedna od ranijih internetskih pretraživača bila bi napisana na kineskom jeziku s ponavljanjima riječi "Matematika" i ne bi sadržavala nikakav drugi podatak. Također, kada bi se tražile informacije o riječi "matrica" očekivalo se da bi stranica "www.matrica.hr" bila najrelevantnije mjesto za takav upit. Međutim, mogu postojati milijuni stranica koje koriste riječ "matrica" u svojoj domeni, a stranica www.matrica.hr možda nije ona koja se najčešće koristi. Neka internetska stranicu sadrži riječ "matrica" milijun puta i ništa drugo. Zar bi tada imalo smisla da ta internet stranica bude prikazana prva u pretraživaču bez obzira što sadržaj stranice nije relevantan? Međutim, ako je broj ponavljanja tražene riječi na nekoj stranici osnovni kriterij rangiranja, onda se upravo to događalo. Navedeni problemi su motivirali nove i bolje pristupe rangiranju stranica. Jedno od tih novijih rješenja je i PageRank algoritam [6].

PageRank algoritam za računanje važnosti i kvalitete internet stranica, ne uzima u obzir samo broj poveznica do neke stranice, nego i kvalitetu tih poveznica kako bi odredio grubu procjenu važnosti internetske stranice. Temeljna pretpostavka je da će važnije internetske stranice dobiti više poveznica s drugih internetskih stranica. To se najbolje može dočarati ako se internet interpretira kao veliki usmjereni graf, gdje vrhovi predstavljaju internetske stranice, a poveznice među stranicama su bridovi grafa. Ideja i funkcionalnost PageRank algoritma demonstrirat će se kroz nekoliko primjera. Pritom će se koristiti metoda potencija kako bi se izračunao prvi svojstveni vektor, koji predstavlja rang internetskih stranica. Radi jednostavnosti poveznice (bridovi) među stranicama (vrhovima grafa) neće imati težine. U nastavku će poveznice i bridovi te stranice i vrhovi imati jednako značenje.

Postoje internetske stranice koje nude izračunavanje ranga neke domene PageRank algoritmom besplatno, npr. [7] koja koristi skalu $0 - 10$, pri čemu je

10 najveći rang. Tako je moguće provjeriti da stranica Veleučilišta u Virovitici, kao relativno mladog učilišta, ima PageRank 5, dok npr. velika i tradicijom bogata sveučilišta Fakultet elektrotehnike i računarstva te Prirodoslovno-matematički fakultet u Zagrebu imaju PageRank 8, a Googleov internetski pretraživač i Facebook imaju PageRank 9.

Projekt Opte [10] je nastojanje izrade vizualizacijske mape cijelog interneta. Navedena stranica nudi brojne vizualizacije rasta Interneta kroz povijest i načine prikazivanja mape Interneta. Npr. možemo vidjeti zastupljenost po kontinentima i važnost stranica. Jedna ilustracija vizualizacije dana je na Slici 3.



Slika 3: Jedna od ilustracija mape interneta - Opte projekt.

Primjer 1. *Prvi primjer sastoji se od četiri internetske stranice koje su povezane kao na Slici 4. Nije bitan samo broj poveznica na neku internetsku stranicu, već i kvaliteta internetskih stranica od kojih dolaze poveznice. Tako npr. stranica A i stranica D mogu imati različitu važnost bez obzira što obje stranice imaju samo jednu ulaznu poveznicu. Prvi korak je prikazivanje grafa u obliku modificirane matrice susjedstva. Zamislimo da svaka stranica ima jedan „bod" koji raspodjeljuje s drugim stranicama s kojima je povezana. Nadalje, svakoj stranici pripada jedan redak u matrici, stranici A prvi redak, stranici B drugi, stranici C treći i stranici D četvrti redak matrice. Ukoliko neka stranica pokazuje na dvije stranice, taj bod se dijeli između te dvije stranice, odnosno vrha. Npr., stranica B dodjeljuje pola boda stranici A, a drugu polovicu stranici C. Isti princip koristimo za svaki vrh u usmjerenom grafu koji prikazuje danu mrežu. Važno je naglasiti da je usmjerenost grafa bitna odrednica, jer npr. stranica B može pokazivati na stranicu A, ali obrnuto ne mora vrijediti. Element matrice susjedstva na poziciji (i, j) bit će različit od nula, ako postoji poveznica sa i-te stranice na j-tu stranicu u mreži. Matrica susjedstva u ovom primjeru glasi*

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Tada u modificiranoj matrici susjedstva Q u i-tom retku, uz uvjet da nije nul redak, stoji

$$Q_{ij} = \frac{A_{ij}}{\sum_{j=1}^{n} A_{ij}},$$

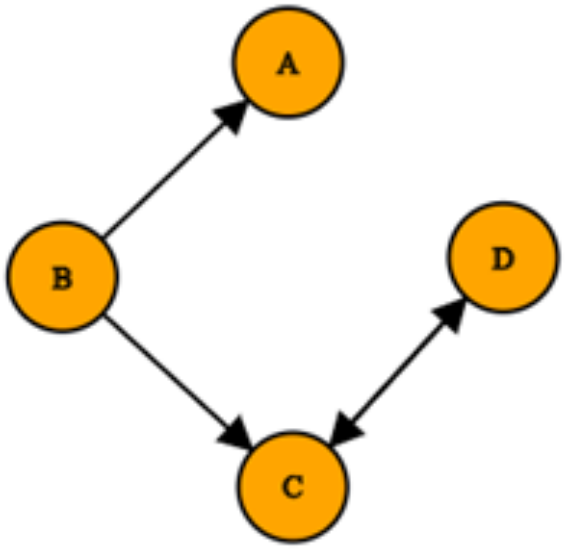
a nul redci se ne mijenjaju. Isti princip vrijedi za svaku stranicu, čime se dobiva matrica:

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

Rang stranica može se dobiti korištenjem metode potencija za matricu koje reprezentira usmjereni graf mreže. Za početnu iteraciju, svaki element vektora r je jednake vrijednosti, jer se na početku uzima da svaka stranica ima jednaki rang, dakle

$$r = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right).$$

Ovakav pristup ima nekoliko problema, a za njihovo razumijevanje i rješavanje uvodimo pojam slučajnog šetača.



Slika 4: Graf mreže koja sadrži četiri internet stranice (vrhovi A, B, C, D). a usmjereni bridovi pokazuju povezanost tih stranica.

3.1 Slučajni šetač

Za dublje razumijevanje PageRank algoritma potrebno je znanje osnova vjerojatnosti. Model slučajnog šetača pretpostavlja da je sljedeća stranica odabrana nasumično. Dakle, rješenje PageRank metode se može promatrati kao asimptotsku vjerojatnost da je slučajni šetač na određenoj stranici. Što je rang stranice veći, veća je vjerojatnost da će slučajni šetač doći do te stranice. Opisana metoda u prošlom odjeljku ima dva problema: „zaglavljen" u stranici i „zaglavljen" u podgrafu. Rješenja za te probleme su opisana u nastavku.

3.1.1 Zaglavljen u stranici

U grafu mogu postojati vrhovi koji nemaju izlaznih bridova (tzv. ponori), tj. postoje stranice koje ne pokazuju niti na jednu drugu stranicu. Tada slučajni šetač ne bi mogao otići niti u jednu drugu internet stranicu. U Primjeru 1 slučajni šetač će „zaglaviti" u slučaju da dođe u vrh A, jer taj vrh ne pokazuje niti na jedan drugi vrh, dok vrhovi B, C i D pokazuju na barem jedan drugi vrh. Taj problem se može riješiti tako da se provjeri zbroj svakog retka modificirane matrice susjedstva Q i ukoliko je on jednak nuli, svaku nulu je potrebno zamijeniti s vrijednosti $1/n$, pri čemu je n ukupni broj vrhova. Stvorene su „nevidljive" poveznice koje zapravo ne postoje između vrhova, ali sprječavaju da šetač zaglavi u nekom vrhu, jer sada postoji vjerojatnost da iz tog vrha skoči u bilo koji vrh. Dodavanjem novih poveznica mijenjaju se vrijednosti matrice Q, čime se generira matrica $\hat{Q}$. Matrica $\hat{Q}$ je tada redak stohastička - zbroj elemenata u svakom retku iznosi jedan. Dobiva se tako da matrici Q dodamo izraz $\frac{1}{n}de^T$, gdje je e jedinični vektor stupac, a vektor d vektor stupac koji na i-tom elementu ima jedinicu ako je i-ti redak matrice Q nul-redak, a inače ima nulu [1].

Tada za Primjer 1 vrijedi:

$$\hat{Q} = Q + \frac{1}{n} d e^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \frac{1}{4} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2)$$

3.1.2 Zaglavljen u podgrafu

Za graf G' kažemo da je podgraf grafa G, ako je skup vrhova grafa G' podskup skupa vrhova grafa G i skup bridova grafa G' podskup skupa bridova grafa G. U grafu mogu postojati podgrafovi koji su izolirani, tj. nemaju bridova prema niti jednom drugom vrhu grafa. To znači da imamo skup stranica koje mogu pokazivati jedna na drugu, ali nisu nikako povezane s drugim stranicama u mreži. U tom slučaju će šetač ostati zarobljen u takvom podgrafu. U Primjeru 1 će do toga doći ako šetač ode na stranicu C, jer nakon toga može završiti samo u stranici C ili D, jer ne postoji poveznica iz vrhova C i D u vrhove A i B.

Problem izoliranih podgrafova povezuje se s problemom reducibilnosti matrice. Matrica $A \in \mathbb{C}^{n \times n}$, $n \geq 2$ je reducibilna ako postoji matrica permutacije P i $p \in \{1, 2, \ldots, n-1\}$ tako da vrijedi

$$P^T A P = \begin{bmatrix} \tilde{A}_{[11]} & \tilde{A}_{[12]} \\ 0 & \tilde{A}_{[22]} \end{bmatrix}, \tilde{A}_{[11]} \in \mathbb{C}^{p \times p}.$$

Ako je $n = 1$, onda je matrica A reducibilna ako je $A = [0]$. Matrica je ireducibilna, ako nije reducibilna.

Potrebno je definirati još nekoliko važnih pojmova vezanih uz grafove. Put u grafu od vrha v_k do vrha v_l je niz usmjerenih bridova $v_k v_{i1}, v_{i1} v_{i2}, \ldots, v_{ij-1} v_l$, pri čemu je j duljina puta. Nadalje, jako povezani graf je graf u kojemu su svaka dva njegova vrha povezana putem. Može se pokazati sljedeća tvrdnja [2]: *kvadratna matrica*Q je ireducibilna ako i samo ako je njen graf jako povezan.}

Neka matrica Q reprezentira mrežu na dosada opisani način. Ako graf koji reprezentira mrežu ima izolirani podgraf, tada matrica Q ne može biti ireducibilna, jer pripadni graf nije jako povezan. Problem zaglavljenosti u podgrafu rješava se omogućavanjem skoka iz bilo kojeg vrha u bilo koji vrh. Vjerojatnost tog skoka treba biti mala, ali različita od nule. Tome služi faktor prigušenja $\alpha \in (0, 1)$, koji iznosi 0.85 u Googleovom pretraživaču. Kombinacijom rješenja problema zaglavljenosti u vrhu i podgrafu uz $\alpha = 0.85$ dobivamo novu matricu $\tilde{Q}$:

$$\begin{aligned} \tilde{Q} &= \alpha \hat{Q} + (1 - \alpha) \frac{1}{n} e e^T \\ &= 0.85 \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} + 0.15 \frac{1}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{37}{80} & \frac{3}{80} & \frac{37}{80} & \frac{3}{80} \\ \frac{3}{80} & \frac{3}{80} & \frac{3}{80} & \frac{71}{80} \\ \frac{3}{80} & \frac{3}{80} & \frac{71}{80} & \frac{3}{80} \end{bmatrix}, \end{aligned}$$

Odabirom faktora α se smanjuju svojstvene vrijednosti matrice. To znači da će za mali α doći do brže konvergencije ka konačnom rezultatu. Međutim, mali α znači i da će konačni rezultat algoritma loše prezentirati stvarne vrijednosti, jer se dopušta da nasumični skokovi kojih inače nema uvelike utječu na rezultat. Kod

velikog α situacija je obrnuta – algoritam će sporije konvergirati, ali će rezultat biti točniji. Iz navedenih razloga kao dobar odabir za faktor prigušenja, uzima se vrijednost 0.85 [1].

3.2 PageRank algoritam i metoda potencija

Može se pokazati da matrica koja je redak-stohastička i ireducibilna ima najveću svojstvenu vrijednost 1 i da joj je pripadni svojstveni vektor stupac r jedinstven i nenegativan. Matrica susjedstva je modificirana kako bi riješila neke potencijalne probleme pa je potrebno metodu potencije primijeniti na matricu $\tilde{Q}^T$, tj. iteracije su

$$r^{(k+1)} = \tilde{Q}^T r^{(k)}. \tag{3}$$

Gornja formula se koristi iterativno do konvergencije kako je opisano u Poglavlju 2. Međutim, u praksi matrice koje se koriste su jako velikog reda (milijuni ili milijarde). Zato se umjesto matrice $\tilde{Q}$ može koristiti izraz:

$$\begin{aligned} r^{(k+1)} &= \tilde{Q} r^{(k)} \\ &= \alpha Q^T r^{(k)} + \alpha \frac{1}{n} e d^T r^{(k)} + (1-\alpha)\frac{1}{n} e e^T r^{(k)}. \end{aligned}$$

Iterativni postupak koristeći rijetko popunjenu matricu Q uvelike smanjuje vrijeme i memorijski prostor koji je potreban. Važan argument za korištenje metode potencija kako bi se riješio problem PageRanka je činjenica da je to najjednostavnija i vrlo efikasna metoda upravo za rijetko popunjene matrice, a kao rezultat se dobije svojstveni vektor koji pripada najvećoj svojstvenoj vrijednosti. Potrebno je naglasiti da metoda potencije ne mijenja početnu matricu tijekom izvršavanja. Nadalje, može se pokazati da je dovoljno koristiti sljedeći oblik iteracija za metodu potencija [1]:

$$r^{(k+1)} = \alpha Q^T r^{(k)} + \frac{1}{n} e - \frac{1}{n} \| \alpha Q^T r^{(k)} \|_1. \tag{4}$$

Bitna prednost navedene formule proizlazi iz činjenice da nije potreban vektor d, tj. nije potrebno znati koje stranice nemaju poveznicu ni na jednu drugu stranicu.

4 Implementacija u programskom paketu GNU Octave i numerički eksperimenti

Kombinacijom opisane metode potencija i (4) možemo napisati skriptu u programskom paketu GNU Octave koja implementira računanje ranga internet stranica koristeći PageRank algoritam i metodu potencija kao u skripti 1.

```
function [r, iter, rel_err] =
            PageRankMP(Q, a = 0.85, eps = 1e-8)

n = size(Q, 1);
r = zeros(n, 1)+1/n;
e = ones(n, 1);
iter = 0;

do
    ++iter;
    r_start = r;
    r = a*Q'*r + (1/n)*e - (1/n)*norm(a*Q'*r, 1);
    rel_err(iter) = norm(abs(r_start-r), 2)/norm(r);
until(rel_err(iter) < eps)
```

4.1 Primjer 1 - numerički eksperimenti

Tablica 1: Pregled iteracija izračunavanja ranga stranicama Primjera 1 uz $eps = 10^{-2}$. Vrijednosti ranga su navedene u potpunosti ili zaokružene na 11 decimalnih mjesta.

# iter	r_A	r_B	r_C	r_D	$rel(i)$
0	0.25	0.25	0.25	0.25	-
1	0.196875	0.090625	0.409375	0.303125	0.4292
2	0.1178515625	0.0793359375	0.3755078125	0.4273046875	0.2583
3	0.09626123047	0.06254345703	0.4594702148	0.3817250977	0.1634
...	...	...			
19	0.07664726482	0.05378754993	0.4432887926	0.4262763926	0.0115
20	0.0766472525	0.05378754377	0.4389821862	0.4305830175	0.0098

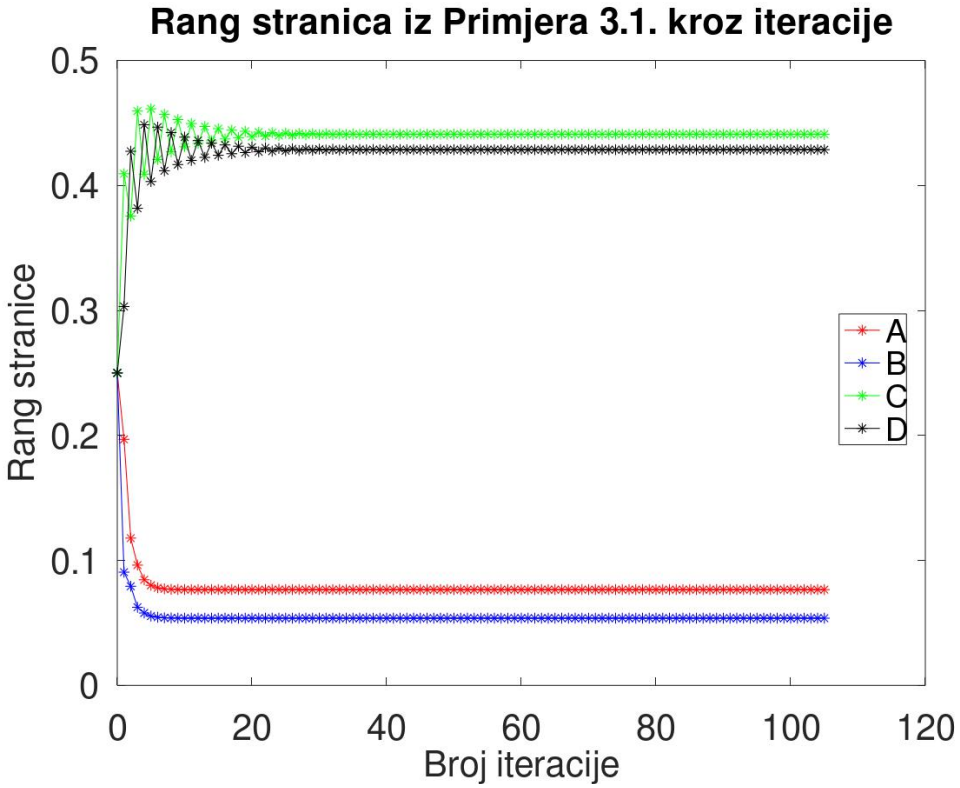
Tablica 1 prikazuje vrijednost ranga stranica i relativnu pogrešku kroz iteracije za odabir $eps = 10^{-2}$, dok su u tablici 2 rezultati dobiveni za odabir $eps = 10^{-8}$. Konvergencija se postiže u prvom slučaju nakon 20 iteracija, a u drugom nakon 105 iteracija. Provjerom je utvrđeno da je l_1 norma vektora r jednaka jedan kroz sve iteracije, što se i očekuje. Nadalje, pripadna svojstvena vrijednost svojstvenog vektora rješenja r bi trebala biti jedan. Provjeru možemo izvršiti pomoću izraza

$$\lambda_1^{(i)} = (\tilde{Q}^T r)_i / r_i.$$

Uz odabira $eps = 10^{-2}$ dobivamo vektor $(1.00000, 1.00000, 1.00834, 0.99150)^T$, a za $eps = 10^{-8}$ vektor $(1, 1, 1, 1)^T$, zaokružimo li rezultate na 5 decimalnih mjesta, što je u skladu s dopuštenom pogreškom i pokazuje da je rješenje s manjom dopuštenom pogreškom točnije i u ovom segmentu.

Tablica 2: Pregled iteracija izračunavanja ranga stranicama Primjera 1 uz $eps = 10^{-8}$. Prvih 20 iteracija je jednako kao u tablici 1, ali zbog uvjeta veće točnosti očekujemo više iteracija. Vrijednosti rangova zaokružujemo na 11 decimalnih mjesta.

# iter	r_A	r_B	r_C	r_D	$rel(i)$
...	...	...	...	...	...
21	0.07664724726	0.05378754116	0.4426428121	0.4269223995	0.00832214
22	0.07664724503	0.05378754004	0.4395312846	0.4300339303	0.0070745
23	0.07664724409	0.05378753957	0.4421760849	0.4273891315	0.0060128
...	...	...	...	...	...
103	0.07664724339	0.05378753922	0.4409609099	0.4286043075	$1.35710 \cdot 10^{-8}$
104	0.07664724339	0.05378753922	0.4409609048	0.4286043126	$1.1535 \cdot 10^{-8}$
105	0.07664724339	0.05378753922	0.4409609091	0.4286043083	$\{9.8051 \cdot 10^{-9}\}$



Slika 5: Graf koji prikazuje vrijednosti ranga pojedinih stranica iz primjera 1 kroz iteracije metode potencija PageRank algoritma za toleranciju 10^{-8}. Vrijednosti ranga se u početku brzo mijenjaju, a onda su promjene u vektoru ranga sve manje dok ne postignemo visoku relativnu točnost.

4.2 Matrice velike dimenzije

Postoje brojne baze matrica susjedstva za stvarne mreže, a koje su dostupne na internetu. Biblioteka *Stanford Network Analysis Project* (SNAP) razvija se od 2004. godine i nudi bazu podataka i alate za analizu velikih mreža. Na internetskoj stranici [12] dostupni su podaci o projektima i člancima koji su proizašli radom na ovoj biblioteci.

Kao primjer velike matrice koristit će se Googleova matrica, koja je javno objavljena i korištena na natjecanju iz programiranja 2002. godine. Kao pobjednik tog natjecanja izašao je Daniel Egnor koji je za prvo mjesto osvojio nagradu od 10 000 američkih dolara s projektom „Geografsko pretraživanje". Više o tom natjecanju možete pročitati u [8]. Matrica je dostupna na stranici SNAP projekta, kao i [13] u raznim formatima. Matrica je reda 916 428, što znači da ima približno 840 milijardi elemenata od čega je različito od nule samo 5 105 039 ($\approx 6.1 \cdot 10^{-4}\%$). Spremanje svih elemenata matrice u memoriju, uključujući nule, nije moguće. Međutim, matricu je moguće spremiti u formatu rijetko popunjene matrice (engl. *sparse matrix*). Tada se spremaju samo indeksi elemenata koji nisu nula i njihove vrijednosti u obliku (# retka, # stupca, vrijednost elementa). Ovaj format omogućuje da se prethodna matrica spremi na svega nekoliko desetaka MB memorije, ovisno o formatu. S obzirom da je to matrica susjedstva, potrebno ju je modificirati. Matrica je modificirana u traženi oblik uz pomoć funkcija za rad s rijetko popunjenim matricama. Zatim je metodom potencija izračunat svojstveni vektor r. U Tablici 3. prikazano je deset najbolje rangiranih stranica. Rezultat je uspoređen s rezultatom Matlabove ugrađene funkcije *centrality*() s istim parametrom a i tolerancijom te je utvrđena relativna pogreška u odnosu na Matlabovo rješenje reda 10^{-12}, što pokazuje da je implementacija uspješna. Matlab bi mogao koristiti drugačiji kriterij zaustavljanja, odakle može doći dominantna pogreška.

Tablica 3: Deset najbolje rangiranih internetskih stranica u Googleovoj matrici iz 2002. godine.

$R.br.$	Indeks u matrici	Rang r	Broj izlaznih poveznica	Broj ulaznih poveznica
1.	597622	0.00091458	22	5354
2.	41910	0.000912013	10	4408
3.	163076	0.000895056	36	4731
4.	537040	0.000889934	27	6326
5.	384667	0.000779103	20	4010
6.	504141	0.000757542	19	5271
7.	486981	0.000717764	6	3908
8.	605857	0.000710848	22	4550
9.	32164	0.000705518	32	5418
10.	558792	0.000702166	22	4206

Kako bismo ubrzali izvršavanje skripti, možemo koristiti transponiranu matricu Q' kao ulaz u funkciju *PageRankMP*(), jer tada ne trebamo transponirati matricu Q u svakom koraku metode potencija, već koristimo ulaznu matricu direktno. Algoritam 3 i skripta 3 demonstriraju računanje ranga stranica koristeći PageRank algoritam i metodu potencija. Implementacija je dostupna na GitHub poveznici https://github.com/markohajba/PageRank_PowerMethod.

Algoritam 2 Računanje ranga stranica koristeći RageRank algoritam i metodu potencija

1: Učitaj matricu susjedstva Q i transponiraj ju
2: Modificiraj matricu susjedstva tako da sadrži vjerojatnosti skoka sa stranice i na stranicu j
3: Pozovi metodu potencija nad dobivenom matricom i proslijedi željene parametre za faktor prigušenja i toleranciju
4: Dodatno: sortiraj vektor ranga i ispiši deset najbolje rangiranih stranica s njihovim indeksima u originalnom vektoru

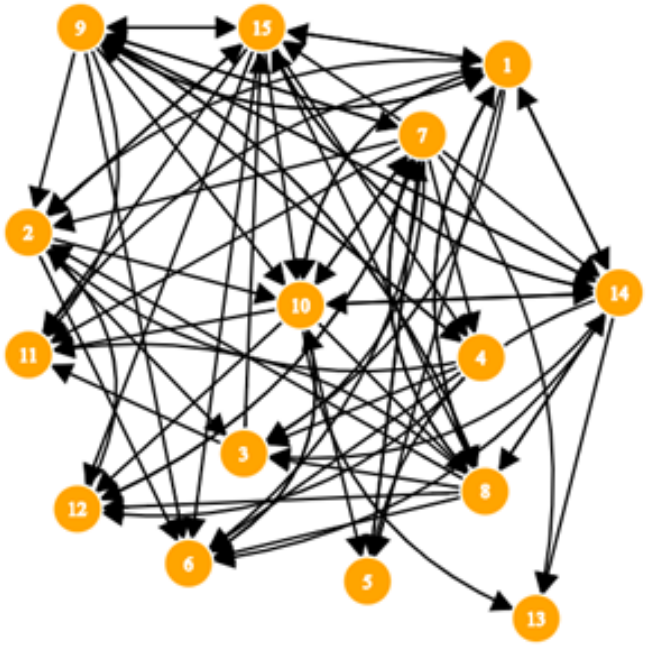
Skripta 2. *Skripta koja čita matricu susjedstva Q mreže stranica iz datoteke, modificira ju, poziva metodu potencija i prikazuje deset najbolje rangiranih stranica s indeksom kao identifikacijskim brojem stranice. Ukupno vrijeme izvršavanja skripte na računalu AIME Workstation A.I. T502 na jednoj niti procesora Ryzen 9 5900X, pod sustavom Ubuntu 20.04. i GNU Octave verzije 5.2.0, je približno jedna minuta za traženu relativnu točnost 10^{-15}. Funkcija find() je ovdje iznimno bitna, jer kada bismo koristili ugnježđene for petlje za modifikaciju matrice susjedstva, izvođenje programa trajalo bi neusporedivo duže. S obzirom da je matrica susjedstva transponirana, gledamo sumu po stupcima, a ne retcima. Nadalje, za sortiranje vektora ranga koristimo funkciju sort(), ali u izlazu tražimo uz sortirani vektor i originalne indekse.*

```
format none

function [r,iter,rel_err] = PageRankMP(Q, a=0.85, epsilon=1e-8)

n = size(Q, 1);
r = zeros(n, 1)+1/n;
e = ones(n, 1);
iter = 0;

do
    ++iter;
    r_start = r;
    r = a*Q*r + (1/n)*e - (1/n)*norm(a*Q*r, 1);
    rel_err(iter) = norm(abs(r_start-r), 2)/norm(r);
until(rel_err(iter) < epsilon)

endfunction

# ucitamo datoteku s matricom susjedstva
load('web-Google.mat');
Q = Problem.A';      # transponiramo matricu susjedstva
s = size(Q, 1);
n = s;
n1 = zeros(s,1);     # pomocni vektor
[ii,jj] = find(Q);   # spremamo indekse ne-nul elemenata u Q'

disp('Pokrecemo konstrukciju modificirane matrice susjedstva:')
tic()
for i = 1:s
    n1(i)=sum(Q(:,i));
end
for k = 1:length(ii)
    Q(ii(k), jj(k)) = Q(ii(k), jj(k)) / n1(jj(k));
end
disp('Vrijeme za pretvorbu:')
toc()

tic()
[r, iter, rel_err] = PageRankMP(Q, a = 0.85, epsilon = 1e-15);
disp('Vrijeme potrebno za metodu potencija:')
toc()

## sortiranje vektora koji sadrzi rangove stranica i ispis top 10
disp("10 najbolje rangiranih stranica s indeksom kao ID")
disp("rang          indeks")
[r_sorted, indices] = sort(r, 'descend');
[r_sorted(1:10), indices(1:10)]
```

4.3 Proizvoljne matrice susjedstva

Rad s rijetko popunjenim matricama pokazao se ključan u radu s matricama velikog reda. To nas je motiviralo da dublje istražimo rad s takvim tipom matrica. Jedan od način je generiranje slučajnih matrica susjedstva za željeni broj stranica u mreži. Implementirane su tri metode. Prva metoda, metoda A, koristi ugnježdene petlje i ne koristi funkcije za rad s rijetko popunjenim matricama. Druga metoda, metoda B, koristi funkciju *sparse*() za koju je potrebno pripremiti argumente ulaza. Treća metoda, metoda C, koristi funkcije *sprand*() i *spones*(). Metode koje koriste funkcije za rad s rijetko popunjenim matricama, osim što omogućavaju njihov zapis u slučaju jako velikih dimenzija, daju višestruko ubrzanje njihovog generiranja, a zatim i rada s njima.

Na računalu AIME Workstation A.I. T502 s procesorom Ryzen 9 5900X vrijeme potrebno za generiranje slučajne matrice reda 1000 na jednoj niti procesora metodom A je 63 -64 s, metodom B 0.3-0.4s, a metodom C 0.009 - 0.03 s. Slučajnu matricu istog reda i popunjenosti kao Googleova matrica u 4.2 možemo generirati metodom C za približno 1.6 s. Implementacije metoda dostupne su na GitHub poveznici https://github.com/markohajba/PageRank_PowerMethod.



Slika 7: Prikaz međusobnih poveznica prvih petnaest stranica u slučajno generiranoj matrici susjedstva reda 1000.

5 Zaključak

PageRank algoritam daje bitan kriterij za rangiranje internetskih stranica u Googleovom internetskom pretraživaču. U radu su dane osnove teorije algoritma koje su potkrijepljene primjerima. U programskom paketu GNU Octave pojašnjena je implementacija algoritma PageRank pomoću metode potencija. Detalji su dani na primjeru male mreže stranica, a zatim je implementacija testirana na matrici visokog reda. Korištenjem funkcija za rad sa rijetko popunjenim matricama, dobiva se značajno ubrzanje izvršavanja algoritma. Točnost dobivenog rješenja za matricu velikog reda provjerena je pomoću Matlabove ugrađene funkcije *centrality*() i utvrđena je visoka relativna točnost rješenja. Demonstrirano je generiranje slučajnih matrica susjedstva mreže pomoću 3 metode. Dodatno su navedeni izvori gdje se mogu pronaći matrice velikog reda iz stvarnog svijeta. {20}

Bibliografija

[1] Andersson, E., Ekstrom, P. (2004): *Investigating Google's PageRank algorithm*. Uppsala: Uppsala University.

[2] Drmač, Z., *Numerička analiza 1*, PMF-MO, Zagreb, https://web.math.pmf.unizg.hr/ drmac/na001.pdf (pristupljeno 15.9.2020.)

[3] Thanuskodi, S. (2020): *Use of Digital Resources Among Social Scien-tists: An Evaluative Study.* Karaikudi: Alagappa University.

[4] Arbona, https://www.arbona.hr/blog/seo/najpopularnije-trazilice-i-navike-korisni... (pristupljeno 2.9.2020)

[5] Backlinko, https://backlinko.com/google-ranking-factors (pristupljeno 6.9.2020.)

[6] Cornell University, http://pi.math.cornell.edu/ mec/Winter2009 /RalucaRemus/Lecture3/lecture... (pristupljeno 9.9.2020)

[7] Google PageRank Checker https://smallseotools.com/google-pagerank-checker/

[8] Mooglemb, http://mooglemb.com/threads/googlecache/64.233.187.104 /programming-conte... (pristupljeno 12.9.2020.)

[9] NetMartketShare: https://netmarketshare.com/ (pristupljeno 10.6.2021.)

[10] Opte projekt: https://www.opte.org/the-internet (pristupljeno 7.7.2021.)

[11] Ryte Wiki, https://en.ryte.com/wiki/Random_Surfer_Model (pristupljeno 2.9.2020.)

[12] SNAP http://snap.stanford.edu/about.html (pristupljeno 7.7.2021.)

[13] University of Florida: https://www.cise.ufl.edu/research/sparse/matrices /SNAP/web-Google.html (pristupljeno 6.7.2021.)

ISSN 1334-6083