OPTIMASI HASIL PENCARIAN HALAMAN WEB MENGGUNAKAN METODE JACOBI

Ishak S. Beno[1,2]

[1]School of Mathematics and Computer Sciences, Heriot-Watt University,
Edinburgh, EH14 4AS, Scotland, United Kingdom
[2]Program Studi Matematika Jurusan Matematika FMIPA Uncen, Jayapura
email: is.beno65@gmail.com, isb2@hw.ac.uk

Abstrak. Prinsip pencarian informasi berbasis mesin pencari (*search engines*) umumnya terjadi secara acak - *random walk problem*. Keluaran yang ditampilkan oleh mesin pencari, seperti *Google*, *Yahoo*! dan *Bing* ini diurutkan berdasarkan dua prosedur utama, yakni indeks dan peringkat situs. Dengan mengimplementasikan metode Jacobi dalam contoh kasus web berskala kecil (*micro-web*), tingkat popularitas web yang ditampilkan oleh mesin pencari, persoalan PageRank dapat dipahami secara jelas. Dari hasil percobaan algoritma yang digunakan (Algoritma PageRank sederhana dan algoritma Jacobi) ditunjukan bahwa meskipun metode Jacobi biasanya dikategorikan sebagai metode iterasi yang lambat dalam menyelesaikan persamaan liner aljabar $Ax = b$, metode ini sudah lebih baik daripada metode sederhana PageRank untuk memberikan optimisasi pencarian web.

***Kata kunci**: Metode Jacobi, PageRank, search engine, analisis link, graf web*

1. PENDAHULUAN

Seiring dengan perkembangan teknologi internet yang begitu pesat, mesin pencari web telah menjadi satu perangkat internet yang sangat penting untuk memperoleh informasi [2]. Namun, ada terdapat banyak mesin pencari yang dapat digunakan untuk melakukan tugas ini, dan semuanya didasarkan pada algoritma pencarian yang berbeda. Lebih lanjut, di antara mesin-mesin pencari ini, Google merupakan yang penyedia pencarian paling populer. Hal ini membuktikan kesuksesan algoritmanya yang sangat sederhana, yang diperkenalkan oleh pendiri Google, Larry Page dan Sergey Brin pada tahun 1998 - PageRank [3, 6, 16].

PageRank merupakan algoritma berbasis struktur link yang memberikan urutan dari pentingnya semua halaman (*page*) di Internet oleh program indeks web (*Google's web crawler*) [2, 5]. Menghitung PageRank sebenarnya sama artinya dengan menghitung distribusi kestabilan dari sebuah matriks transisi, yang disebut matriks Google. Matriks google ini dibangun berdasarkan struktur graf web. Ukuran besarnya matriks google memutihkan banyak algoritma populer dan rumit yang tidak mungkin memiliki kinerja yang sangat baik pada mereka komputasi skala normal.

Menurut Google, algoritma ini bekerja dengan menghitung jumlah dan kualitas dari link ke sebuah halaman web yang bertujuan untuk menhasilkan evaluasi kasar/ awal tentang seberapa penting halaman web tersebut. Asumsi yang dapat diterima secara sederhana yaitu lebih banyak halaman web penting cenderung memiliki lebih banyak link dari situs lainnya. PageRank bukan merupakan satu-satunya algoritma yang dipakai oleh Google untuk mengurutkan hasil mesin pencari, tetapi telah menjadi algoritma pertama dan sangat popular yang duperkenalkan dan digunakan oleh perusahaan raksasa ini [2].

Sebagai pembanding, metode sederhana, *power method*, memberikan hasil yang stabil dan dapat dipercaya terlepas dari faktor kurang efisiensinya. Oleh karena itu metode variansi lainnya, yaitu metode Jacobi diperkenalkan untuk meningkatkan kecepatan konvergensinya.

2. METODE

Dalam proyek ini, metode iterasi Jacobi akan digunakan untuk mengurutkan popularitas dari link sebuah web mikro (micro web). Meskipun demikian, prosedur mendasar yang digunakan didasarkan pada algoritma PageRank.

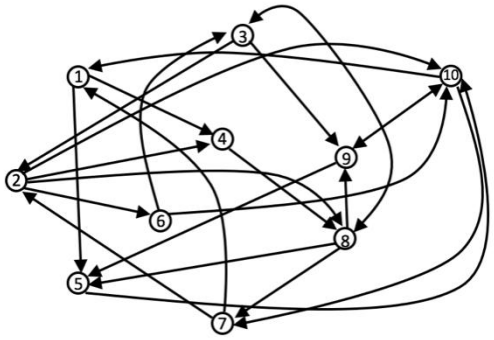
2.1 Algoritma PageRank

Prinsip pengurutan halaman web muncul dalam mesin pencari *online* komersial maupun non-komersial. Ketika sebuah mesin pencari dijalankan, *software* tidak melihat pada setiap halaman di web, melainkan memeriksa *database* raksasa yang disiapkan oleh penyedia mesin pencari, seperti Google, Yahoo!, *DuckDuckGo*, Bing, dan lain-lain [3, 4, 6, 9, 10]. Penyedia mesin pencari menyiapkan database tersebut dengan mencari/ mengungjungi halaman web dari waktu-ke waktu. Pada dasarnya, proses ini melibatkan dua tahapan penting [3, 12, 13, 14]. Tahapan pertama adalah pengindeksan (*indexing*), yaitu tahapan di mana kata-kata kunci dan frasa yang digunakan dalam pencarian diekstrak dan diindeks dengan sebuah metode yang memudahkan pencocokan pencarian di waktu mendatang. Pada tahapan kedua yaitu pengurutan web/ situs (*site ranking*), merupakan tahapan penentuan urutan atau pentingnya sebuah halaman web sebelum pencarian dilakukan. Kemudian, ketika seseorang mencari page menggunakan kata atau frasa tertentu, indeks akan memproduksi suatu daftar page yang memiliki kecocokan, dan ranking akan menghasilkan web yang paling penting sebagai tampilan pertama.

Sebelum membahas lebih jauh mengenan pengurutan halaman web, kita perlu memberikan label atau notasi. Pertama, kita memberi label N halaman web dengan angka-angka unik antara 1 dan N. Selanjutnya, mendefinisikan NxN matriks keterhubungan atau matriks adjasensi $C = (C_{j,k})$ dengan [6],

$$C_{j,k} = \begin{cases} 1, & \text{jika ada } hyperlink \text{ antara page } j \text{ dan } k \\ 0, & \text{lainnya} \end{cases}$$

Perlu diketahui bahwa C tidak simetri karena jika page j memiliki hyperlink ke page k, ini tidak berarti bahwa page k memiliki sebuah link ke page j. Perlu dibedakan antara link masuk dan link keluar. Perlu juga dibuatkan notasi h_j sebagai jumlah hyperlink yang keluar dari page j.



Gambar 1. Web mikro sederhana

2.2 Model Berjalan Acak (Random Walk Model)

Proses *surfing* di Internet secara acak untuk waktu yang sangat lama, berpindah dari page satu ke yang lainnya, secara teori mengikuti dua prosedur/ aturan:

1) 85% dari waktu menelusuri hyperlink dari page di mana kita berada; jika terdapat sejumlah hyperlink h_j dari page j, maka tentukan bobot yang setara untuk setiap link dan pilih satu secara acak dengan probabilitas $1/h_j$.

2) 5% dari waktu untuk melewatkan page lainnya secara acak di web.

Setelah proses yang panjang ini, akan dihitung total berapa kali setiap page telah dikunjungi dan diranking dengan menerapkan probabilitas:

$$x_i = \lim_{N_s \to \infty} \frac{\text{frekuensi mengunjungi page } i}{\text{Jumlah tahapan} N_s}$$

Aturan pertama dapat diinterpretasikan sebagai sebuah hyperlink dari page j ke page i merupakan sebuah pilihan konfidensi untuk page i, dan pentingnya page i dapat dihitung dengan mengasumsikan pilihan-pilihan dari semua hyperlink yang masuk. Pilihan yang dilakukan oleh page *j* tersebar secara merata ke semua halaman yang terhubung, dan diberi bobot berdasarkan pentingnya page *j* itu sendiri. Oleh karena itu, pilihan dari page *j* ke page i dapat diberikan oleh

$$\text{pilihan dari page } j \text{ ke page } i = \frac{c_{j,i}}{h_j} x_j \tag{1}$$

di mana h_j adalah jumlah hyperlink yang keluar dari page j. Pentingnya setiap page ditentukan oleh seberapa penting halaman web tersebut memilihnya.

Ada du acara penyelesaian masalah pencarian nilai x_i, yaitu pertama dengan menggunakan sebuah formula sistemeigen (eigensystem) dan metode iterasi sederhana, dan yang akan dikaji secara detail yaitu bagaimana menuliskan persoalan ini ke dalam persamaan sistem linear. Kombinasi aturan 1 dan 2 menjumlahkan semua kontribusi ke page i, memberikan

$$x_i = \frac{1-\delta}{N} + \delta \sum_{j=1}^{N} c_{j,i} \frac{x_j}{h_j}, \quad i = 1, \ldots, N \tag{2}$$

dimana N merupakan jumlah halaman web, $\delta = 0.85$ dan $c_{j,i}$ adalah elemen dari matriks koneksi (0 dan 1) dan C^* merupakan matriks transpos, C. Persamaan (2) di atas dapat dituliskan dalam bentuk matrik seperti

$$x_i = \frac{1-\delta}{N} e + \delta C^* H^{-1} x, \tag{3}$$

dimana$H = diag(h_1, \ldots, h_N)$, $e = (1, 1, \ldots, 1)^*$, dengan mengasumsikan bahwa kita hanya mengkaji halaman yang terhubung, kita dapat mengabaikan persoalan ketakterdefinisian $(1/h_j)$ karena tidak adanya hyperlink yang keluar, $(h_j = 0)$. Persoalan ini dapat dikaji secara terpisah.

Sekarang kita dapat menyelesaikan persoalan (3)dalam kasus umum. Persamaan ini dapat dituliskan dalam bentuk standar $(Ax = b)$ seperti

$$(I - \delta C^* H^{-1}) x = \frac{1-\delta}{N} e \tag{4}$$

2.3 Metode Jacobi

Metode Jacobi terkadang dibahas sebagai metode yang dipakai dalam penyelesaian masalah pengurutan popularitas web page [17]. Iterasi Jacobi secara umum dapat dituliskan dalam persamaan

$$I - \delta C^* H^{-1} = D - L - U \tag{5}$$

di mana D, U, L merupakan matrik diagonal, segitiga atas dan segitiga bawah secara berurutan, sehingga $D = I$, $L + U = \delta C^* H^{-1}$, Sehingga iterasi Jacobi dituliskan

$$x^{k+1} = D^{-1} \left(\frac{1-\delta}{N} e + (L+U) x^k \right) = \frac{1-\delta}{N} e + \delta C^* H^{-1} x^k \tag{6}$$

Oleh karena vektor x merupakan vektor probabilitas (semua elemen ≥ 0, total jumlahan elemen = 1), sebaiknya dimulai dengan tebakan nilai awal. Pilihan $x_i^0 = 1/N$ untuk setiap $i = 1, \ldots, N$ terlihat masuk akal, kecuali ada informasi tambahan lainnya.

$$\sum_{i=1}^{N} x_i^{k+1} = \sum_{i=1}^{N} \left(\frac{1-\delta}{N} + \delta \sum_{j=1}^{N} c_{j,i} \frac{x_j^k}{h_j} \right) = \sum_{i=1}^{N} \frac{1-\delta}{N} + \delta \sum_{i=1}^{N} \sum_{j=1}^{N} c_{j,i} \frac{x_j^k}{h_j} = (1-\delta) +$$

$\delta \sum_{j=1}^{N} x_k^j = 1$.

3. HASIL DAN PEMBAHASAN

Proses pencarian informasi di Internet melalui mesin pencari sebenarnya terjadi secara acak, tanpa disadari pengguna (*user*). Sehingga secara matematika, proses ini tidak lain adalah persoalan berjalan acak (*random walk problem*). Untuk studi kasus web mikro yang dikaji, kami mengimplementasikan metode random walk dan Jacobi untuk menganalisis hasil pencarian informasi tidak hanya berdasarkan tinggkat kepopuleran tetapi juga seberapa penting sebuah

halaman web untuk dikunjungi.

Informasi yang ada ditransfer ke dalam bentuk matriks, kemudian menerapkan metode iterasi Jacobi untuk mencari solusi solusi terhadap persoalan ini yang menghasilkan probabilitas kepopuleran hyperlink di antara vertek pada web mikro.

Dari persoalan web mikro (gambar 1), dapat dituliskan matrik konektivitas webnya, C, seperti berikut,

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Dengan elemen diagonal matrik H ($h_1 = 2, h_2 = 4, h_3 = 2, h_4 = 1, h_5 = 1, h_6 = 2, h_7 = 2, h_8 = 3, h_9 = 1, h_{10} = 2$). Untuk mengimplementasikan iterasi Jacobi kita memperoleh matriks terlibat dalam proses iterasi.

$$C^*H^{-1} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^* \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}^{-1}$$

$$C^*H^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1 & 0 \\ 1/2 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 1/2 \\ 0 & 1/4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 1/4 & 0 & 0 & 1 & 1/2 & 0 & 0 & 0 & 0 \end{bmatrix} = W$$

Matriks W adalah matriks non-negatif (semua elemen $w_{i,j} \geq 0$) dan total jumlahan kolumnya adalah 1 dan elemen diagonalnya adalah 0. Dalam kasus yang umum, hal ini benar: matriks C^*H^{-1} harus positif jumlahan kolumnya 1 dan diaginonalnya 0. Hal ini secara otomatis menjamin bahwa norm matriks $\|C^*H^{-1}\|_1 = 1$.

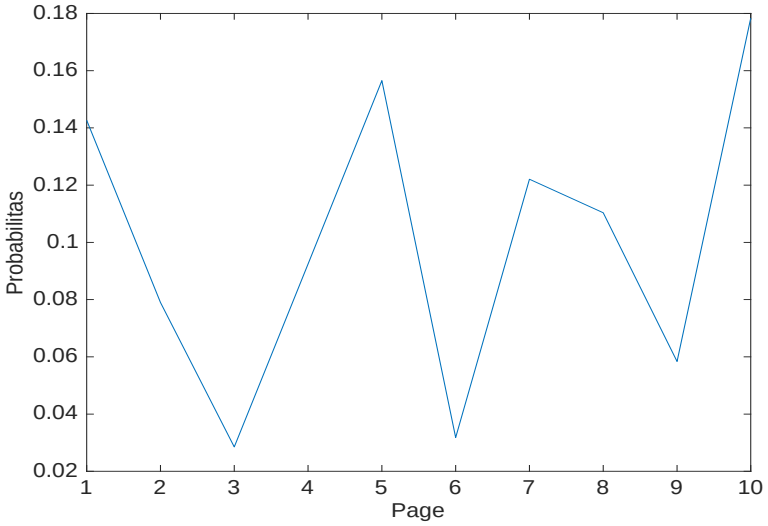
Tabel 1. Iterasi Jacobi

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
0.1	0.12107	0.14922	0.14868	0.14321	0.14152	0.14223	0.14226	0.14283
0.1	0.075917	0.078476	0.082282	0.078892	0.078172	0.079327	0.079042	0.078762
0.0575	0.030406	0.030406	0.028231	0.028462	0.028806	0.0285	0.028435	0.028539
0.07875	0.07875	0.082588	0.095095	0.095672	0.092628	0.091758	0.092305	0.092257
0.17083	0.16481	0.15842	0.15479	0.15441	0.15545	0.15751	0.1571	0.15621
0.03625	0.03625	0.031132	0.031676	0.032485	0.031765	0.031612	0.031857	0.031796
0.085833	0.11895	0.1279	0.1221	0.12018	0.12255	0.12219	0.12159	0.12227
0.12125	0.10319	0.09807	0.10188	0.11332	0.11309	0.11035	0.10985	0.11026
0.085833	0.073792	0.057159	0.055709	0.055863	0.059203	0.059284	0.058377	0.058209
0.16375	0.19686	0.18663	0.17956	0.17752	0.17682	0.17725	0.17918	0.17887

x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
0.14299	0.14262	0.14255	0.14269	0.14268	0.14265	0.14268	0.14267	0.14266
0.079096	0.079079	0.078903	0.078984	0.079029	0.078975	0.078981	0.079002	0.078988
0.028513	0.028488	0.028518	0.028517	0.028501	0.028508	0.028512	0.028507	0.028508
0.092438	0.092578	0.092418	0.092352	0.092429	0.092432	0.092408	0.092421	0.092424
0.15642	0.15659	0.15647	0.15652	0.15657	0.15651	0.1565	0.15653	0.15652
0.031737	0.031808	0.031804	0.031767	0.031784	0.031794	0.031782	0.031783	0.031788
0.12226	0.12187	0.12203	0.12214	0.12203	0.12203	0.12208	0.12205	0.12204
0.11016	0.11038	0.1105	0.11032	0.11028	0.11036	0.11035	0.11033	0.11035
0.058368	0.058329	0.058382	0.058427	0.058378	0.05836	0.058384	0.058383	0.058376
0.17802	0.17825	0.17843	0.17828	0.17832	0.17838	0.17832	0.17832	0.17835

x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}
0.14267	0.14267	0.14266	0.14267	0.14267	0.14267	0.14267	0.14267	0.14267
0.078985	0.078993	0.078991	0.078988	0.07899	0.078991	0.078989	0.07899	0.07899
0.02851	0.028509	0.028508	0.028509	0.028509	0.028509	0.028509	0.028509	0.028509
0.092414	0.092417	0.092421	0.092418	0.092418	0.092419	0.092419	0.092418	0.092419
0.15651	0.15652	0.15652	0.15652	0.15652	0.15652	0.15652	0.15652	0.15652
0.031785	0.031784	0.031786	0.031786	0.031785	0.031785	0.031786	0.031785	0.031785
0.12206	0.12206	0.12205	0.12206	0.12206	0.12205	0.12206	0.12206	0.12206
0.11035	0.11034	0.11034	0.11034	0.11034	0.11034	0.11034	0.11034	0.11034
0.058381	0.058329	0.058382	0.058427	0.058378	0.05836	0.058384	0.058383	0.058376
0.17834	0.17825	0.17843	0.17828	0.17832	0.17838	0.17832	0.17832	0.17835

x_{28}	x_{29}	x_{30}	x_{31}
0.14267	0.14267	0.14267	0.14267
0.07899	0.07899	0.07899	0.07899
0.028509	0.028509	0.028509	0.028509
0.092419	0.092418	0.092419	0.092419
0.15652	0.15652	0.15652	0.15652
0.031785	0.031785	0.031785	0.031785
0.12206	0.12206	0.12206	0.12206
0.11034	0.11034	0.11034	0.11034
0.058379	0.05838	0.05838	0.05838
0.17834	0.17834	0.17834	0.17834



Gambar 2. Distribusi Probabilitas web page model berjalan acak

Dari tabel 1 dihasilkan probabilitas distribusi kunjungan terhadap web mikro dengan 10 halaman, terlihat bahwa pada iterasi ke-30, distribusinya telah stabil dan hasil pencarian yang lebih akurat dapat ditampilkan oleh mesin pencari. Secara jelas, terlihat juga (dari tabel dan gambar 2) bahwa terdapat 5 page hasil pencarian yang lebih akurat akurat (populer) yang memenuhi kriteria pencarian, yakni halaman 10, 5, 1, 7 dan 8 secara berurutan. Hal ini menjelaskan bahwa optimasi pencarian (kepopuleran suatu page) tidak didasarkan pada berapa banyak hyperlink (masuk dan keluar) pada suatu page (verteks).

Juga, jika total distribusi probabilitasnya dijumlahkan maka akan menghasilkan 1 (memenuhi prinsip dasar probabilitas). Hal ini menunjukan bahwa radius spectral dari iterasi matriks konektivitasnya $\rho(\delta C^* H^{-1}) \leq \delta$ [17]. Oleh karena itu, berdasarkan teorema Richardson [17] untuk 1-norm, kekonvergenan metode Jacobi terjamin jika $\delta \leq 1$. Kami memakai $\delta = 0.85$, tetapi bisa juga menggunakan nilai lainnya. Semakin kecil δ yang digunakan maka semakin cepat pula konvergensi dicapai. Tetapi jika mengacu pada eror dari aprosimasi *page rank* vektor x^k, $\left\|x^k - x\right\|_1 \leq \delta^k \|x^0 - x\|_1$, semakin kecil δ yang digunakan maka semakin identik pula page rank vektor x, sehingga akan memberikan informasi yang lebih kurang.

4. KESIMPULAN

Satu dari banyak algoritma terkenal untuk memberikan solusi terhadap persoalan pencarian halaman web adalah algoritma PageRank milik Google [1, 5, 15, 16]. Proses komputasi sebuah persoalan Optimasi pencarian web (PageRank) sebenarnya adalah mengkalkulasi kestabilan distribusi dari sebuah matiks transisi, atau yang disebut Matriks Google [2], yaitu yang didasarkan pada struktur graf web.

Algoritma ini digunakan untuk menjawab persoalan optimasi hasil pencarian dengan mengkalkulasi vectoreigen dari matriks konektivitas, yang secara sederhana menggunakan *power method* untuk mendeskripsikan hyperlink pada halaman web [5].

Dengan mempertimbangkan kenyataan bahwa ada terdapat lebih dari 3 miliar halam web [5], proses komputasi ini dapat berlangsung beberapa hari, untuknya itu perlu ditingkatkan kecepatan komputasinya. Iterasi Jacobi merupakan salah satu metode ampuh yang mampu menjawab persoalan ini. Dua alasan penting mengapa optimasi persoalan hasil pencarian web sangat penting. Pertama, untuk mengurangi lamanya waktu pencarian. Kedua, untuk mengkomputasikan vektor PageRank dengan tujuan untuk mengelompokan (*indexing*) topik-topik sensitif dari halaman-halaman web tertentu [8, 7, 11].

DAFTAR PUSTAKA

[1] Kamvar, S., Haveliwala, T., & Golub, G. (2004). Adaptive methods for the computation of PageRank. Linear Algebra and its Applications, **386**, 51-65.

[2] Sun, H., & Wei, Y. (2006). A note on the PageRank algorithm. *Applied Mathematics and computation*, **179(2)**, 799-806.

[3] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: bringing order to the web, disadur dari http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf, September 2016

[4] Altman, A., & Tennenholtz, M. (2005, June). Ranking systems: the PageRank axioms. In *Proceedings of the 6th ACM conference on Electronic commerce*, 1-8, ACM.

[5] Avrachenkov, K., & Litvak, N. (2006). The effect of new links on Google PageRank. *Stochastic Models*, **22(2)**, 319-331.

[6] Berkhin, P. (2005). A survey on pagerank computing. *Internet Mathematics*, **2(1)**, 73-120.

[7] Kamvar, S., Haveliwala, T., Manning, C., & Golub, G. (2003). Exploiting the block structure of the web for computing pagerank. *Stanford University Technical Report*, disadur dari http://ilpubs.stanford.edu:8090/579/1/2003-17.pdf, September 2016.

[8] Kamvar, S. D., Haveliwala, T. H., Manning, C. D., & Golub, G. H. (2003). Extrapolation methods for accelerating PageRank computations. In *Proceedings of the 12th international conference on World Wide Web*, 261-270, ACM.

[9] Zhu, Y., Ye, S., & Li, X. (2005). Distributed PageRank computation based on iterative aggregation-disaggregation methods. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, 578-585, ACM.

[10] Arasu, A., Novak, J., Tomkins, A., & Tomlin, J. (2002). PageRank computation and the structure of the web: Experiments and algorithms. In *Proceedings of the Eleventh International World Wide Web Conference, Poster Track*, 107-117.

[11] Langville, A. N., & Meyer, C. D. (2006). A reordering for the PageRank problem. *SIAM Journal on Scientific Computing*, **27(6)**, 2112-2120.

[12] Langville, A. N., & Meyer, C. D. (2004). Deeper inside pagerank. *Internet Mathematics*, **1(3)**, 335-380.

[13] Del Corso, G. M., Gulli, A., & Romani, F. (2005). Fast PageRank computation via a sparse linear system. *Internet Mathematics*, **2(3)**, 251-273.

[14] Gleich, D., Zhukov, L., & Berkhin, P. (2004). Fast parallel PageRank: A linear system approach. *Yahoo! Research Technical Report YRL-2004-038, available via http://research. yahoo. com/publication/YRL-2004-038. pdf*, *13*, 22.

[15] Zhu, Y., Ye, S., & Li, X. (2005). Distributed PageRank computation based on iterative aggregation-disaggregation methods. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, 578-585, ACM.

[16] Bianchini, M., Gori, M., & Scarselli, F. (2005). Inside pagerank. *ACM Transactions on Internet Technology (TOIT)*, **5(1)**, 92-128.

[17] Loisel, S., Boulton, L., (2015). *Module of Numerical Analysis B: Applied Linear Algebra*. Heriot-Watt University, Edinburgh.