Informationsteknologiska sidrankningsalgoritmer

FREDRIK CEDERVALL
och DAVID TÖRNQUIST

KTH Datavetenskap
och kommunikation

Examensarbete
Stockholm, Sverige 2010

Informationsteknologiska sidrankningsalgoritmer

F R E D R I K C E D E R V A L L
o c h D A V I D T Ö R N Q U I S T

Examensarbete i datalogi om 15 högskolepoäng
vid Programmet för datateknik
Kungliga Tekniska Högskolan år 2010
Handledare på CSC var Henrik Eriksson
Examinator var Mads Dam

URL: www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2010/cedervall_fredrik_OCH_tornquist_david_K10035.pdf

Kungliga tekniska högskolan
Skolan för datavetenskap och kommunikation

KTH CSC
100 44 Stockholm

URL: www.kth.se/csc

Sammanfattning

Sidrankningsalgoritmer används i sökmotorerna på Internet för att inbördes väga webbsidorna mot varandra och rangordna sökresultaten efter dessa vikter. Att modellerna dessa baseras på representerar verkligt beteende och vad som intuitivt implicerar relevans i sökresultatet är essentiellt för att finna *rätt* information i den enorma datamängd som Internet utgör.

Google har till stor del sidrankningsalgoritmen PageRank att tacka för sina framgångar. I dess kölvatten har bland annat SALSA och HITS vuxit fram som alternativa algoritmer, vars grund finns i de ifrågasatta svagheterna i PageRank.

Den enorma storleken och snabba expansionen av Internet har medfört att sidrankningsalgoritmer transformerats till ett beräkningsproblem. Denna uppsats förklarar, undersöker och resonerar kring praktiska och prestandamässiga skillnader mellan olika metoder att beräkna PageRank.

Abstract

Site ranking algorithms are used in search engines on the Internet to inter-weigh the web sites toward each other and order the search results according to these weights. That the models these are based on represent real behavior and what is intuitively implicating relevance in the search results is essential to find the *right* information in the enormous data volume the Internet constitutes.

Google has in large part the site ranking algorithm PageRank to thank for its success. In its aftermath, SALSA and HITS have, among others, evolved as alternative algorithms, with their base in the questioned weaknesses of PageRank.

The vast extent and fast expansion of the Internet has caused site ranking algorithms to transform into computational issues. This thesis explains, investigates and reasons around practical and performance-wise differences between the different methods of calculating PageRank.

Förord

Vi vill tacka Henrik Eriksson som tog sig an att ta över handledarskapet av oss när vi befann oss i en frustrerande situation av utebliven kommunikation och riktning. Med Henriks hjälp återfann vi motivationen och fokus på uppgiften. Få saker får en så motiverad som ifrågasättandet av ens val—”Varför har ni valt det här? Borde inte projektbeskrivningen avskräckt er?”. Tack Henrik.

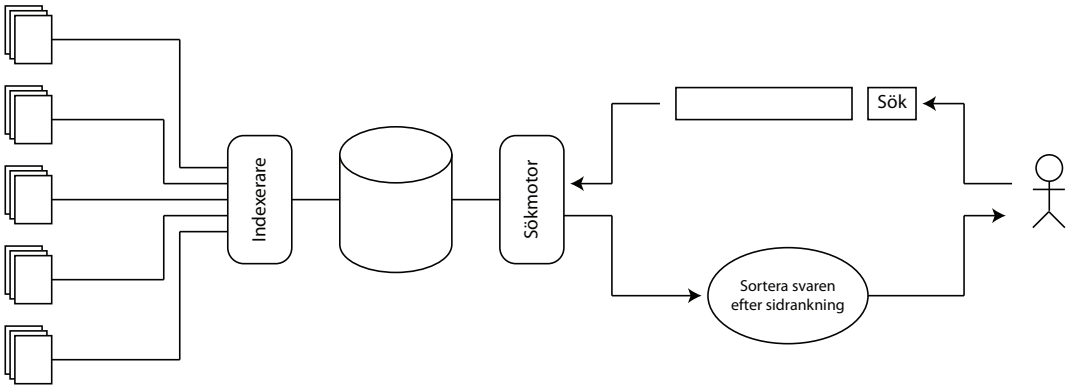
Innehåll

Kapitel 1

Inledning

Rankningsalgoritmer är ett aktuellt ämne inom informationsteknologin. Eftersom mängden information som görs tillgänglig har ökat explosionsartat den senaste tiden har behovet att rangordna informationen och göra den mer lättillgänglig blivit allt mer akut.

Den här kandidatuppsatsen ingår i kursen DD143X - Examensarbete inom datalogi, grundnivå. Syftet med uppsatsen är att undersöka sidrankningsalgoritmer som används på webben och göra prestandajämförelser mellan ett urval av metoderna för att beräkna dessa. Vad som avses med sidrankningsalgoritmer är metoder för att rangordna sidor baserat på dess *relativa informationsvärde*. Uppsatsen har tre huvuddelar där den första ämnar återge den relevanta teoretiska del som behövs för fortsatt förståelse av uppsatsen. Den andra delen avser den experimentella del som ämnar besvara nedan specifierad frågeställning. I den tredje delen presenteras och utvärderas erhållna resultat.



Figur 1.1: Diagram över hur sökmotorer applicerar sidrankning

1.1 Bakgrund

Problemet när Internet och informationsteknologin slog igenom var att ordna upp den fragmenterade informationen och göra den tillgänglig. En lösning var att göra informationen sökbar via sökmotorer och den har visat sig fungera. Förutom att lokalisera information måste sökmotorerna även relatera till informationsvärdet och vad användaren eftersöker för att rangordna träffarna. Vad som sökmotorerna processar är en textsökning given av användaren.

Just hur de givna träffarna ska rangordnas är något som alltid har varit ett problem för sökmotorerna. En lösning på detta blev sidrankning. Sidrankning används för att rangordna sidor efter hur pass tillgängliga de är och deras relativa informationsvärde. Detta var revolutionerade då tidigare lösningar på sökmotorer inte satte sidor i relation till varandra.

Hur sidrankning appliceras i sökmotorerna illustreras förenklat i figur 1.1.

1.2 Problemformulering

Undersökningen kommer utgå från PageRank, sidrankningsalgoritmen som är implementerad i Googles sökmotor och som är en stor del av anledningen till att sidrankning har blivit ett så omtalat ämne. Ett problem med sidrankning är den ständigt expanderande och föränderliga informationsmängden. Just den enorma informationsmängd som finns tillgänglig på Internet gör det väldigt resurskrävande att utföra sidrankningsberäkningarna. Det faktum att innehållet och strukturen på webben även förändras kontinuerligt gör att behovet av kontinuerligt uppdaterade rankningar vitalt. Detta problem ligger till grund för den här uppsatsen som ämnar undersöka prestandaskillnader mellan olika implementationer av PageRank.

Efter att idéerna kring PageRank publicerades och implementerades i Googles sökmotor så har det kommit fram en del modifikationer som är tänkta att vara optimeringar av PageRank. Två av dessa är IAD-metoden med Tarjans algoritm för försortering av matriser samt en metod som berör extrapolation för att accelerera PageRank. Tarjans försorteringsalgoritm avser optimering av matrisen IAD-metoden opererar på för att snabba upp beräkningarna av PageRank. Extrapolationsmetoden bygger på att man accelererar konvergensen hos potensmetoden som används i PageRank. Både IAD-metoden och extrapolationsmetoden kommer förklaras ingående senare i uppsatsen.

Problemet som ämnas undersökas i den här uppsatsen är hur pass stora prestandaskillnader som råder mellan originalimplementation av PageRank (potensmetoden), PageRank med IAD-metoden (med Tarjans försorteringsalgoritm) samt PageRank med hjälp av extrapolationsmetoden. De experimentella undersökningarna utförs med hjälp av MATLAB. Mätningarna utförs på matriser som är mindre representationer av Internet, då det är orimligt resursmässigt att utföra beräkningarna på hela Internet.

1.3 Disposition

Uppsatsen är uppdelad i tre delar, presenterade nedan.

Del I - Teoretisk ram Den här delen behandlar de teoretiska delarna som läsaren behöver för att förstå området och senare experimentella del där delar av det som berörts inom den teoretiska ramen kommer implementeras.

Del II - Experimentell del I den här delen implementeras potensmetoden och extrapolationsmetoden för PageRank och jämförs prestandamässigt. Den tredje beräkningsmetoden, IAD-metoden, införs i jämförelsen utifrån tillgängliga forskningsresultat, men ingen egen implementation görs.

Del III - Resultat Uppsatsens resultat presenteras och diskuteras i denna del. Teoretiska och praktiska aspekter belyses ur prestandaperspektiv och användbarhet.

1.4 Formelnotation

Huvudsakligen används standardnotation i formlerna, men för att förstärka sambanden i härledningarna har viss notation införts som inte är vedertagen konvention eller entydig. Nedan finnes en tabell med definitionerna som gäller och som åsidosätter standardnotation i de fall krockar uppstår.

	Kännetecken	**Beskrivning**	**Exempel**
x	Tecken i kursiv stil	Godtycklig variabel	$x = 4$
$\mathbf{x}$	Gement tecken i fetstil	Vektor	$\mathbf{x} = \begin{bmatrix} a & b & c & d \end{bmatrix}$
$\mathbf{x}^T$	Vektor med superscript T	Transponat av vektor	$\mathbf{x}^T = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$
$\overline{\mathbf{x}}$	Överstruken vektor	Vektor härledd ur motsvarande vektor utan streck	$\overline{\mathbf{x}} = \begin{bmatrix} d & c & b & a \end{bmatrix}$
$\mathbf{x}^{(i)}$	Vektor med superscript (i)	Iterativt beräknad vektor av ordning i	$\mathbf{x}^{(3)} = \begin{bmatrix} b & c & d & a \end{bmatrix}$
$\mathbf{X}$	Versalt tecken i fetstil	Matris	$\mathbf{X} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$
$\mathbf{X}^T$	Matris med superscript T	Transponat av matris	$\mathbf{X}^T = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix}$
$\overline{\mathbf{X}}$	Överstruken matris	Matris härledd ur motsvarande matris utan streck	$\overline{\mathbf{X}} = \begin{pmatrix} d & e & f \\ a & b & c \\ g & h & i \end{pmatrix}$

Del I

Teoretisk ram

Kapitel 2

Matematisk grund

I det här kapitlet beskrivs den grundläggande teoretiska ramen kortfattat. Syftet med kapitlet är att säkerställa den matematiska nivån som krävs för senare delar av uppsatsen. De matematiska delarna som berörs i uppsatsen förutsätts att läsaren vara bekant med, men de vitala delarna som är särskilt relevanta för förståelsen kommer att beskrivas. Kapitlet är uppdelat i tre delar där första delen ger en explicit förklaring av den matematiska grund som förutsätts (2.1). Efter det kommer två beskrivande delar som berör linjär algebra och grafteori.

2.1 Förkunskapskrav

För vidare förståelse förutsätts läsaren ha en matematisk färdighet som motsvarar tredje året på ett civilingenjörsprogram med tyngdpunkt i linjär algebra, diskret matematisk och matematisk statistik. De vitala delarna för uppsatsen kommer nedan att beskrivas, men läsaren förutsätts redan vara bekant med dessa. De beskrivande delarnas syfte blir därmed repetition.

Särskilt relevant för vidare läsning är området linjär algebra som berör matriser, vektorer och operationer av dessa. Läsaren förutsätts besitta kunskap i hur man genomför vektor- och matrismultiplikationer.

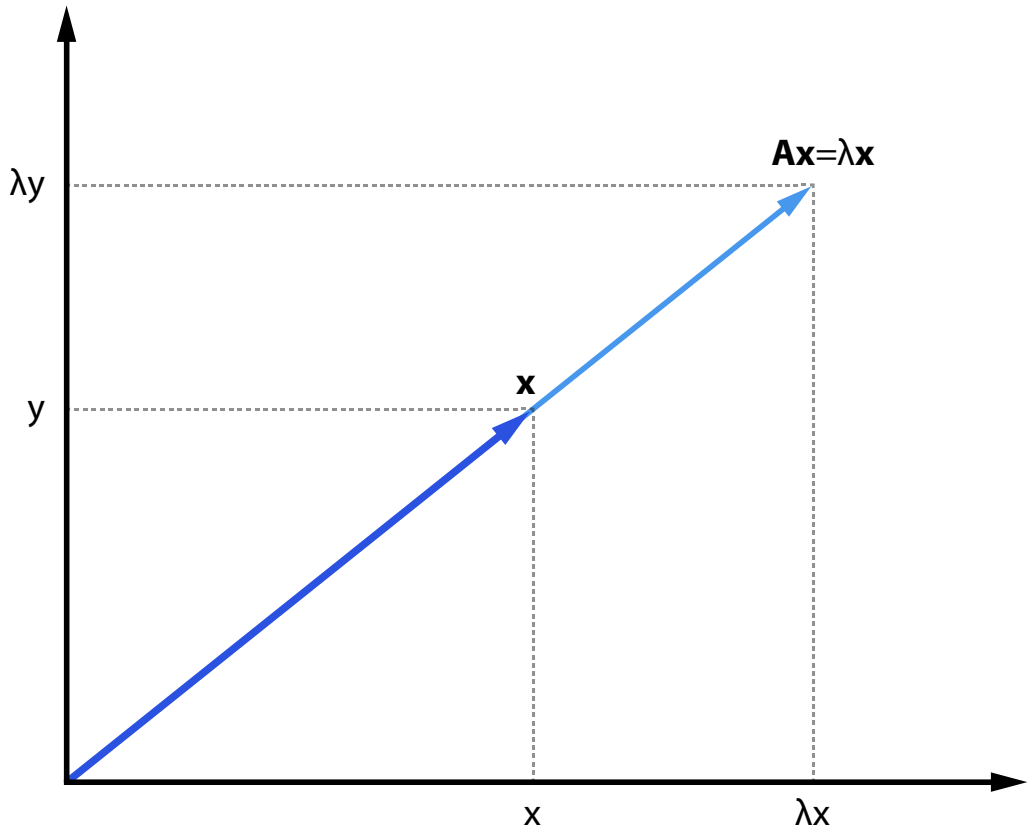
2.2 Linjär algebra

2.2.1 Egenvärden och egenvektorer

Definitionen av en egenvektor $\mathbf{x}$ är en kolumnvektor som via multiplikation med en matris $\mathbf{A}$ avbildas på samma linje och genom origo som $\mathbf{x}$. Se formel (2.1). [1]

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \tag{2.1}$$

$\mathbf{A}$ är en godtycklig $n \times n$ matris och $\mathbf{x}$ är en icke-noll-vektor i $\mathbb{R}^n$. $\mathbf{A}\mathbf{x}$ motsvarar en skalärmultiplikation av $\mathbf{x}$ där λ är skalfaktorn och kallas för egenvärdet av $\mathbf{A}$. Egenvektorn $\mathbf{x}$ är en egenvektor av $\mathbf{A}$ med avseende på egenvärdet λ. [1]



Figur 2.1: Visualisering av egenvektor och multiplikation med **A**

En vektor har som känt både en magnitud och en riktning. Karaktäristiskt för egenvektorn är att den genomgått en linjär transformation där riktningen på vektorn är oförändrad. Även magnituden kan vara oförändrad, vilket händer i de fall då egenvärdet λ är lika med 1. Ett tvådimensionellt exempel visualiseras i figur 2.1.

2.3 Grafteori

En graf inom den diskreta matematiken är ett objekt bestående av en mängd kanter (eng. edges) och en mängd av noder (eng. nodes), även kallade hörn (eng. vertices) [2]. Grafer används i den här uppsatsen för att representera Internet där en webbsida representeras av en nod och där en en hyperlänk representeras av en riktad kant där riktningen symboliserar vilken sida som länkar till vilken.

$$\mathbf{P}_{m,n} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \cdots & p_{m,n} \end{pmatrix}$$

Figur 2.2: Stokastisk matris

2.3.1 Markovkedjor

En markovkedja är en stokastisk process med markovegenskaper som sker i diskret tid. Med stokastisk process avses en slumpmässig vandring över olika tillstånd där de olika tillstånden representeras av olika stokastiska variabler. En stokastisk variabel kallas även för slumpvariabel och är diskret då den kan anta uppräkneligt många olika värden baserad på dess sannolikhetsfunktion [3]. En stokastisk process är med andra ord en samling stokastiska variabler där för varje tillstånd ges sannolikheter för nästa tillstånd. Sannolikheterna beror helt på respektive tillstånds stokastiska variabels sannolikhetsfördelning och därmed kan vissa tillstånd vara mer sannolika än andra. [5]

Att processen sker i diskret tid innebär att det finns uppräkneliga (alt. oändligt uppräkneliga) tidssteg [3]. Vad som menas med markovegenskapen är att den stokastiska processen endast är beroende av nuvarande tillstånd och är därmed minneslös. [3]

För att beteckna sannolikheterna vid övergångar mellan olika tillstånd i en markovkedja kan man använda sig av en radstokastisk matris (kallas även för övergångsmatris, se figur 2.2). I figuren betecknar $p_{i,j}$ sannolikheten att man går från tillstånd i till tillstånd j i ett tidssteg. I en radstokastisk matris gäller alltid att varje radsumma är lika med 1 enligt formel (2.2), då man med 100 % sannolikhet befinner sig i något tillstånd i nästkommande tidssteg. [5]

$$\sum_j p_{i,j} = 1 \tag{2.2}$$

Låter man den stokastiska processen fortsätta mot oändligheten erhålls den stationära fördelningen. Den stationära fördelningen betecknar sannolikheterna för att befinna sig i ett visst tillstånd efter oändlig tid oberoende av starttillstånd. Den stationära fördelningen betecknas av radvektorn $\pi = (\pi_1, \ldots, \pi_k)$. För att en radvektor π skall vara den stationära fördelningen måste formel (2.3) uppfyllas där $\mathbf{P}$ är den stokastiska matrisen (övergångsmatrisen). Vad som även måste uppfyllas av π är att alla element är större än eller lika med 0 samt att summan av alla element är 1. [5]

$$\pi\mathbf{P} = \pi \tag{2.3}$$

Formel (2.3) visar att π är en egenvektor till $\mathbf{P}$ med egenvärde 1. Vad som karak-

täriserar den stationära fördelningen π är att den inte förändras av den stokastiska matrisen, den påverkas med andra ord inte av att man går ett steg till i den stokastiska processen. Den stationära fördelningen kan även representeras av formel (2.4) för alla tillstånd i då den stationära fördelningen är oberoende av starttillståndet. π_j representerar då sannolikheten att efter oändligt många steg befinna sig i tillstånd j.

$$\lim_{k\to\infty} \left(\mathbf{P}^k\right)_{i,j} = \boldsymbol{\pi}_j \tag{2.4}$$

Den stationära fördelningen π kallas även den dominanta egenvektorn för $\mathbf{P}$ och är unik då grafen är irreducibel (se 2.3.2).

2.3.2 Irreducibilitet

Inom grafteorin säger man att en graf är irreducibel då man givet två noder alltid kan finna en väg mellan dessa. Detta skall gälla för alla par av noder i grafen för att irreducibilitet skall råda. Detta gäller även för markovkedjor då det skall råda en sannolikhet större än noll (inte nödvändigtvis i ett steg) att övergå från ett tillstånd till ett annat för alla par av tillstånd. [5]

Kapitel 3

PageRank

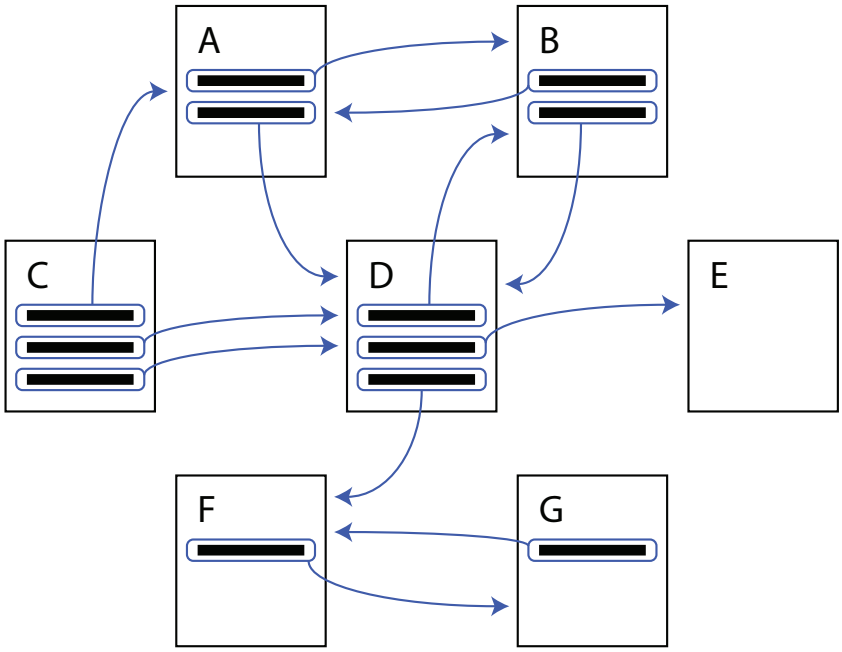
Det ökade behovet av att kunna rangordna information på Internet utmynnade 1998 i att Lawrence Page och Sergei Brin m.fl. formulerade ett koncept som de kallade för PageRank [11]. PageRank bygger på att varje sida på Internet får en rankning. Denna rankning innefattar hur pass viktig sidan är i form av länkar från andra sidor [8]. Senare delar av kapitlet beskriver mer ingående om faktorer i en sidas betydelse och hur de beräknas.

Som *proof of concept* av PageRank skapades sökmotorn Google som idag är den erkänt överlägsna sökmotorn på Internet [8]. Syftet med det här kapitlet är att förklara konceptet PageRank och dess funktionalitet. Kapitlet är uppdelat i fyra delar där den första delen beskriver hur konceptet PageRank representerar Internet. Den andra delen handlar om implementationen av PageRank vilket exemplifieras i den tredje delen. Sista delen behandlar olika metoder för att beräkna PageRank och ligger till grund för senare experimentella del i uppsatsen.

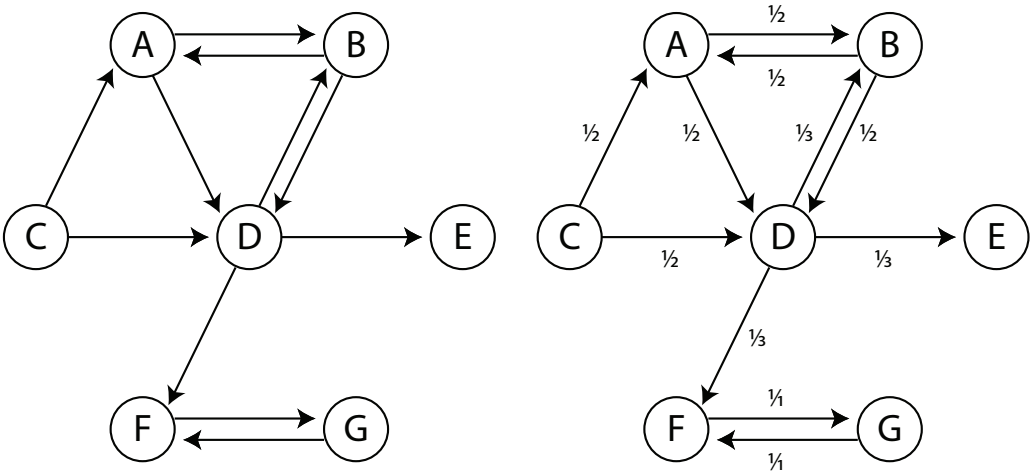
3.1 Representation av Internet

PageRank utgår ifrån en modell där Internet (förenklat i figur 3.1) representeras som en graf (3.2a)—varje webbsida representeras av en nod i grafen och hyperlänkar från en webbsida till en annan representeras av en riktad kant. Förekomster av dubletter av hyperlänkar på en webbsida ignoreras (de dubbla hyperlänkarna från C till D i figur 3.1 representeras endast som en kant i figur 3.2a).

Ingen hänsyn tas till webbsidans utformning (språkligt innehåll, layout, m.m.). Således antas alla hyperlänkar på en webbsida vara lika viktiga och besitta samma sannolikhet för att bli följd (jfr bli klickad på) av en besökare. Detta representeras i grafen genom att kantvikten för varje kant från en nod med n utgående kanter likafördelat sätts till $1/n$ (figur 3.2b).



Figur 3.1: Visualisering av enkel hyperlänkstruktur



(a) Utan kantvikter (b) Med kantvikter

Figur 3.2: Grafrepresentation av figur 3.1

3.1.1 Markovmodell

Som en del i utvecklandet av PageRank formulerades en modell kring idén av en ”slumpmässig surfare” (eng. ”random surfer”) [11]. Modellen utgår ifrån att en besökare börjar på slumpvist vald sida (med likafördelad sannolikhet i den förenklade modellen) och sedan successivt följer en slumpvist vald hyperlänk på varje sida den besöker. Sannolikheten att en besökare följer en viss hyperlänk på en given webbsida ges av motsvarande kantvikt i grafrepresentationen (d.v.s. likafördelat $1/n$, där n avser antalet *unika* utgående hyperlänkar på den givna webbsidan). Grundkravet för en markovmodell är därmed uppfyllt—sannolikheten för att nå ett givet nästa tillstånd x_{i+1} är endast beroende av det nuvarande tillståndet x_i och oberoende av de föregående tillstånden x_{i-j} där $j > 0$.

$$\mathbf{A} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\ \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{3.1}$$

Formel (3.1) är en viktad grannmatris och representerar grafen i figur 3.2b. Då radsumman för $\mathbf{A}_E$ är 0 (ty inga utgående hyperlänkar) uppfyller den inte kraven för en stokastisk övergångsmatris och kan inte utan modifieringar användas i markovmodellen.

Två problem uppstår med den förenklade modellen ur ett beteendeperspektiv. Det första problemet är att det är osannolikt att en verklig besökare som når en sida utan utgående hyperlänkar (webbsida E i figur 3.1) slutar att surfa. Denna typ av webbsidor kallas ”dinglande länkar” (eng. ”dangling links”) [11]. Istället förutsätts besökaren i ett sådant fall skriva in en URL manuellt i adressfältet och därmed *börja om från början* på slumpvist vald webbsida på Internet. Ur detta erhåller man övergångsmatrisen $\mathbf{P}$ i formel (3.2). För det andra är det osannolikt att en besökare som når en liten mängd sidor som endast har hyperlänkar till varandra (irreducibel delmängd) kommer att följa dessa hyperlänkar i oändlighet utan att bli uttråkad (webbsidorna F och G i figur 3.1). Med en viss sannolikhet α antas besökaren fortsätta följa hyperlänkarna och annars istället skriva in en URL manuellt i adressfältet och därmed *börja om från början*, precis som för det första fallet (se formel (3.8)).

$$\mathbf{P} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\ \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 & 0 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{3.2}$$

Ur grafrepresentationen av Internet har nu härletts en komplett och korrekt övergångsmatris för en markovmodell som kan användas för beräkningarna utifrån beteendet av en ”slumpmässig surfare”.

3.1.2 Heuristik

PageRank avser att med endast Internets länkstruktur som grund avgöra samtliga webbsidors *relativa informationsvärde* [8, 11].

Utgångspunkten är att genom att en webbsida hyperlänkar till en annan webbsida samtidigt avlägger en röst på att denna är viktig. Varje webbsida avlägger röster med ett sammanlagt värde motsvarande sitt eget PageRank-värde som likafördelas på de utgående hyperlänkarna (genom multiplikation med $1/n$). Därmed ges en ingående hyperlänk från en *viktig* webbsida (d.v.s. högt PageRank-värde) högre vikt än från en *oviktig*. Genom att en webbsida hyperlänkar till fler webbsidor (högre n-värde) sker dock en utspädning av röstens vikt—fler ska dela på samma kaka.

Det uträknade PageRank-värdet är ett mått på en sidas *viktighet* på Internet—även benämnt relativt informationsvärde—och *inte* dess relevans, som det lätt kan misstolkas som. Relevans är subjektivt och beroende av en sökterm medan PageRank är ett förberäknat, söktermsoberoende mått.

3.2 Implementation av PageRank

Låt u vara en webbsida på Internet. Mängden webbsidor *som u hyperlänkar till* benämns som F_u (eng. ”forward links from u”). Mängden webbsidor som istället *hyperlänkar till u* benämns som B_u (eng. ”backward links to u”).

Den förenklade definitionen av PageRank för given sida u kan då skrivas enligt formel (3.3). Observera att denna förenklade definition inte ger ett normaliserat värde samt att den inte beaktar problemen—och lösningarna som ges för dessa—som behandlas i avsnitt 3.1.1.

$$R(u) = \sum_{v \in B_u} \frac{R(v)}{|F_v|} \tag{3.3}$$

Som trivialt inses är denna definition rekursiv ty $R(v)$, mer eller mindre långsökt, kommer att vara beroende av $R(u)$. Detta leder fram till en mer tydlig rekursiv definition;

$$R^{(i)}(u) = \sum_{v \in B_u} \frac{R^{(i-1)}(v)}{|F_v|}, \quad \text{där } i = 1, 2, 3... \text{ och } R^{(0)} = \text{ godtyckligt positivt tal} \tag{3.4}$$

Antag att det finns k webbsidor och låt oss benämna dessa w_j $(0 \leq j < k)$. Genom att skapa en vektor $\mathbf{r}^{(i)} = [R^{(i)}(w_0), R^{(i)}(w_1), R^{(i)}(w_2), ..., R^{(i)}(w_{k-2}), R^{(i)}(w_{k-1})]$ kan man med hjälp av övergångsmatrisen $\mathbf{P}$ (formel (3.2)) skriva om den rekursiva definitionen i formel (3.4) som

$$\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)}\mathbf{P} \tag{3.5}$$

Den önskade PageRank-vektorn erhålles när man låter $i \to \infty$ och $\mathbf{r}^{(i)}$ konvergerar mot $\mathbf{r}$ (mer om konvergens i avsnitt 3.2.1).

$$\mathbf{r} = \lim_{i \to \infty} \mathbf{r}_i \tag{3.6}$$

$$\mathbf{r} = \mathbf{r}\mathbf{P}, \qquad \mathbf{r}\mathbf{e}^T = 1, \qquad (\mathbf{e} \text{ är en radvektor med ettor}) \tag{3.7}$$

Formel (3.6) implicerar formel (3.7) (givet att (3.6) konvergerar), d.v.s. den stationära fördelningen för markovkedjan med övergångsmatris $\mathbf{P}$. Detta innebär att problemet med att beräkna PageRank är att hitta den dominanta egenvektorn till $\mathbf{P}$, d.v.s. lösa det homogena linjära ekvationssystemet $\mathbf{r}(\mathbf{I} - \mathbf{P}) = \mathbf{0}$ med $\mathbf{r}\mathbf{e}^T = 1$.

I avsnitt 3.1.1 formulerades $\mathbf{P}$ som en korrekt övergångsmatris ur den viktade grannmatrisen $\mathbf{A}$. Problemet med irreducibla delmängder kvarstår dock vilket leder till avsaknad av unik lösning för den stationära fördelningsvektorn $\mathbf{r}$ (läs mer i avsnitt 3.2.2). Genom att införa sannolikheten α i formeln (berörd i avsnitt 3.1.1) kan man tvinga fram irreducibilitet i matrisen (mer i avsnitt 3.2.2). Vilket värde på α som Google använder är hemligt men antas vara $\alpha = 0.85$ [8]. Genom att samtidigt införa en perturbationsmatris[1] $\mathbf{E}$ kan man forma den s.k. Google-matrisen $\mathbf{G}$ (formel (3.8)).

$$\mathbf{G} = \alpha\mathbf{P} + (1 - \alpha)\mathbf{E}, \qquad \text{där} \qquad \mathbf{E} = \frac{\mathbf{e}^T\mathbf{e}}{k} \qquad \text{och} \qquad 0 < \alpha < 1 \tag{3.8}$$

Intuitivt inses att perturbationsmatrisen $\mathbf{E}$ skiljer sig ganska kraftigt från verkligt beteende—det är mer sannolikt att en besökare manuellt går till exempelvis *http://www.kth.se/* än till *http://edu.bth.se/utbildning/*. För att korrigera denna felaktighet i modellen, införde L. Page, S. Brin m.fl. något de kallar ”Personalized

[1]Perturbering innebär att man tar ett lösbart problem som är svårt att lösa och transformerar det till ett problem som möjligvis inte går att lösa men som är enkelt att approximera

PageRank" [11]. Låt oss utgå från vår förenklade visualisering av Internet i figur 3.1 innehållande sju webbsidor. Men den ursprungliga *demokratiska* infallsvinkeln anses sannolikheten vara 1/7 att en besökare manuellt går till respektive webbsida. Det ger initialvektorn $\mathbf{v} = [1/7, 1/7, ..., 1/7]$. Genom att istället definiera en egen *personaliserad* initialvektor $\overline{\mathbf{v}}$ kan modellen både göras mer realistisk och anpassas efter behov (exempelvis favorisera eller bestraffa utvalda webbsidor).

$$\overline{\mathbf{v}} = \begin{bmatrix} 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \end{bmatrix} \tag{3.9}$$

$$\overline{\mathbf{E}} = \mathbf{e}^T \overline{\mathbf{v}} \tag{3.10}$$

$$\overline{\mathbf{E}} = \begin{pmatrix} 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \\ 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \\ 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \\ 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \\ 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \\ 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \\ 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \end{pmatrix} \tag{3.11}$$

Genom att ersätta $\mathbf{A}_E$ (se formel (3.1)) med $\overline{\mathbf{v}}$ istället för $\mathbf{v}$ erhålls en anpassad övergångsmatris $\overline{\mathbf{P}}$.

$$\overline{\mathbf{P}} = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 & 0 \\ 3/28 & 5/28 & 3/14 & 1/4 & 1/28 & 1/14 & 1/7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.12}$$

Den slutgiltigt dynamiskt anpassade Google-matrisen $\overline{\mathbf{G}}$ kan då skrivas som

$$\overline{\mathbf{G}} = \alpha \overline{\mathbf{P}} + (1 - \alpha) \overline{\mathbf{E}} \tag{3.13}$$

3.2.1 Noggrannhet och konvergens

Genom att lägga till perturbationsmatrisen $\mathbf{E}$ (i [11] kallad "rank source") garanteras PageRank-vektorn $\mathbf{r}$ alltid konvergera (se formel (3.6)). Noggrannheten för $\mathbf{r}^{(i)}$ ges av normen av dess residualvektor $\delta = ||\mathbf{r}^{(i)} - \mathbf{r}^{(i-1)}||_1$. Tillräckligt noggrann konvergens mot $\mathbf{r}$ uppnås när $\delta < \epsilon$.

För att justera hur snabbt den konvergerar kan man variera hur stor vikt $\mathbf{E}$ spelar in genom att ändra värdet på α—lägre värde på α ger snabbare konvergens. Problemet med ett lägre värde på α är att noggrannheten blir sämre, ty den manuellt definierade vektorn $\mathbf{v}$ får ett större inflytande.

3.2.2 Påtvingad irreducibilitet och unikhet

Det är rimligt att anta att Internet består av flertalet irreducibla delmängder, där en liten mängd sidor endast hyperlänkar till varandra. Detta fenomen kallas ”ranksänka” (eng. ”rank sink”) [11] ty denna delmängd skulle ackumulera PageRank till oändligheten utan att distribuera ut den till resten av Internet. En markovkedja baserad på en reducibel övergångsmatris (d.v.s en övergångsmatris med fler än en irreducibel delmängd) har den (i detta fall otrevliga) egenskapen att lösningen inte är unik (beror på starttillståndet i). Genom att införa perturbationsmatrisen (”rankkällan”) $\mathbf{E}$ blir grafen som matrisen $\mathbf{G}$ representerar komplett (kanter mellan samtliga noder). Därmed blir den också irreducibel och en unik lösning på ekvationen garanteras.

3.3 Exempel

Låt oss först utgå ifrån $\mathbf{G} = \alpha\mathbf{P} + (1-\alpha)\mathbf{E}$. Då erhålls $\mathbf{G}$ enligt formel (3.14) som för presentationens skull avrundats till fem decimaler (beräkningarna sker med femton decimaler).

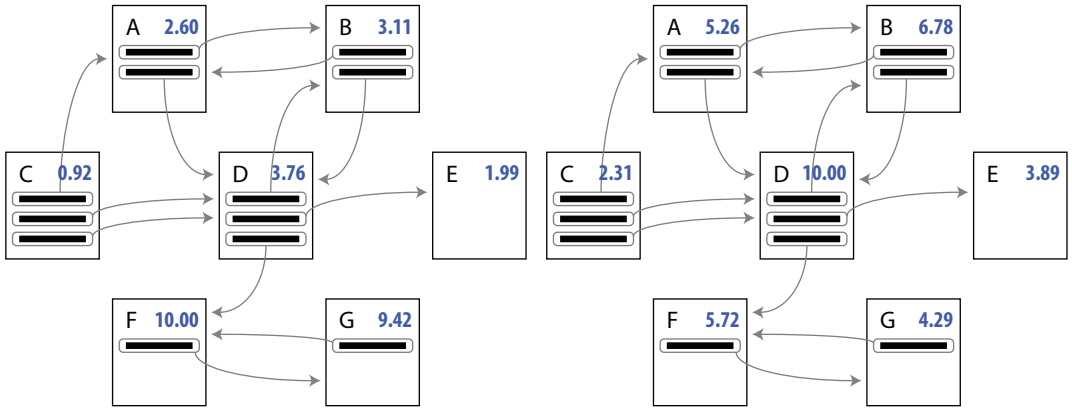
$$\mathbf{G} = \begin{pmatrix} 0.02143 & 0.44643 & 0.02143 & 0.44643 & 0.02143 & 0.02143 & 0.02143 \\ 0.44643 & 0.02143 & 0.02143 & 0.44643 & 0.02143 & 0.02143 & 0.02143 \\ 0.44643 & 0.02143 & 0.02143 & 0.44643 & 0.02143 & 0.02143 & 0.02143 \\ 0.02143 & 0.30476 & 0.02143 & 0.02143 & 0.30476 & 0.30476 & 0.02143 \\ 0.14286 & 0.14286 & 0.14286 & 0.14286 & 0.14286 & 0.14286 & 0.14286 \\ 0.02143 & 0.02143 & 0.02143 & 0.02143 & 0.02143 & 0.02143 & 0.87143 \\ 0.02143 & 0.02143 & 0.02143 & 0.02143 & 0.02143 & 0.87143 & 0.02143 \end{pmatrix} \tag{3.14}$$

Låt $\mathbf{r}^{(0)} = \mathbf{v} = \mathbf{e}/k = [1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7]$. Potensmetoden (formel (3.5), läs mer i avsnitt 3.4.1) ger

$$\begin{aligned} \mathbf{r}^{(1)} &= [0.16020, 0.13997, 0.03878, 0.22092, 0.07925, 0.20068, 0.16020] \\ \mathbf{r}^{(2)} &= [0.10702, 0.16173, 0.03105, 0.17510, 0.09365, 0.22982, 0.20163] \\ &\vdots \\ \mathbf{r}^{(35)} &= [0.08286, 0.09769, 0.02901, 0.11808, 0.06247, 0.31399, 0.29590] \end{aligned} \tag{3.15}$$

Efter 35 iterationer erhålls konvergens med fem decimalers noggrannhet. Värdena är normaliserade (summeras till ett). Önskas istället att skalan ska vara 0.00–10.00, där sidan med högst PageRank på Internet har 10.00 erhålls den slutgiltiga PageRankvektorn enligt

$$\begin{aligned} \mathbf{r} &= \frac{\mathbf{r}^{(35)}}{max(\mathbf{r}^{(35)})} \cdot 10.00 \\ &= [2.60, 3.11, 0.92, 3.76, 1.99, 10.00, 9.42] \end{aligned} \tag{3.16}$$



(a) $\mathbf{r}$ med $\alpha = 0.85$ och $\mathbf{v}$ (b) $\overline{\mathbf{r}}$ med $\alpha = 0.75$ och $\overline{\mathbf{v}}$

Figur 3.3: Uträknade PageRank-värden

Sidorna A–E erhåller PageRank-värden som intuitivt anses rimliga utifrån figur 3.3a. Sida F och G erhåller däremot orimligt höga värden. Detta beror på att när den slumpmässiga surfaren hamnat i dess irreducibla delmängd fortsätter att hoppa mellan sidorna med sannolikheten $\alpha = 0.85$ och med sannolikheten 0.15 hoppar till slumpvist vald sida. Då exempelstrukturen är så liten, är det 2/7 chans att man i ett sådant fall hoppar direkt tillbaka till delmängden och ackumulerar PageRank. Till detta kommer att man via sida D hoppar dit. Sammantaget är sannolikheten större än 0.93 att den slumpmässiga surfaren kommer att ackumulera PageRank i F och G när han/hon väl nått ranksänkan. I en verklig representation av webben uppstår inte detta problem då andelen små irreducibla delmängder antas väldigt liten i förhållande till hela storleken på Internet.

Genom att öka sannolikheten att den slumpmässiga surfaren hoppar till slumpmässig sida, istället för att följa länkarna, och samtidigt skapa en personaliserad initialvektor där F och G straffas hårt, kan ett mer önskat resultat uppnås. Låt $\alpha = 0.75$ och $\overline{\mathbf{v}} = [0.14814, 0.18517, 0.18517, 0.37034, 0.11110, 0.00004, 0.00004]$. På samma sätt erhålls då istället

$$\overline{\mathbf{r}} = [5.26, 6.78, 2.31, 10.00, 3.89, 5.72, 4.29] \tag{3.17}$$

3.4 Beräkningsalgoritmer

Detta avsnitt går igenom tre algoritmer för beräkning av PageRank. Den första algoritmen, potensmetoden, är rättfram medan de andra anses mycket mer komplexa och kräver markant högre förkunskaper för fullständig förståelse. Rekommenderat är att endast läsa dessa övergripande för att förstå infallsvinkeln. Den nyfikne hänvisas till källorna för vidare läsning och djupare förståelse.

3.4.1 Potensmetoden

Potensmetoden (eng. ”power method”) är en approximation för att beräkna en matris dominanta egenvektor. För att beräkna PageRank med potensmetoden genomförs iterativ multiplikation av PageRank-vektorn med den justerade övergångsmatrisen $\mathbf{G}$ tills dess att PageRank-vektorn konvergerar, se ekvation (3.18). [8]

$$\begin{aligned}
\mathbf{r}^{(1)} &= \mathbf{r}^{(0)}\mathbf{G} \\
\mathbf{r}^{(2)} &= \mathbf{r}^{(1)}\mathbf{G} = (\mathbf{r}^{(0)}\mathbf{G})\mathbf{G} = \mathbf{r}^{(0)}\mathbf{G}^2 \\
\mathbf{r}^{(3)} &= \mathbf{r}^{(2)}\mathbf{G} = (\mathbf{r}^{(0)}\mathbf{G}^2)\mathbf{G} = \mathbf{r}^{(0)}\mathbf{G}^3 \\
&\vdots \\
\mathbf{r}^{(i)} &= \mathbf{r}^{(i-1)}\mathbf{G} = (\mathbf{r}^{(0)}\mathbf{G}^{(i-1)})\mathbf{G} = \mathbf{r}^{(0)}\mathbf{G}^i
\end{aligned} \tag{3.18}$$

För tillräckligt stora värden på i (antalet iterationer) kommer PageRank-vektorn att konvergera och ge en god approximation på den justerade övergångsmatrisens egenvektor som även är den stationära fördelningen för markovkedjan.

3.4.2 Extrapolationsmetoden

Extrapolation är ett alternativ på optimerande modifikationer att utvidga konceptet PageRank med för att erhålla snabbare konvergens. Den typ av extrapolation som kommer behandlas här är kvadratisk extrapolation (eng. ”quadratic extrapolation”). Den kvadratiska extrapolationen bygger på att approximation av den dominanta egenvektorn (stationära fördelningen) görs med hjälp av tidigare tre iterationer och att man antar att övergångsmatrisen har tre dominerande egenvektorer. Självklart har övergångsmatrisen fler än tre egenvektorer, men antagandet gör att man kan få en bra approximation på den dominanta egenvektorn för vidare beräkningar med hjälp av potensmetoden.

Extrapolationen bygger på att man antar att $\mathbf{r}^{(k-3)}$ kan skrivas som en linjär kombination av övergångsmatrisens tre egenvektorer, enligt ekvation (3.19)

$$\begin{aligned}
\mathbf{r}^{(k-3)} &= \mathbf{u}_1 + \mathbf{u}_2 + \mathbf{u}_3 \\
\mathbf{r}^{(k-2)} &= \mathbf{r}^{(k-3)}\mathbf{G} \\
\mathbf{r}^{(k-1)} &= \mathbf{r}^{(k-2)}\mathbf{G} \\
\mathbf{r}^{(k)} &= \mathbf{r}^{(k-1)}\mathbf{G}
\end{aligned} \tag{3.19}$$

Där $\mathbf{u}_1$ kommer vara approximationen av den dominanta egenvektorn och approximationen för $\mathbf{r}^{(k)}$ som kommer användas för vidare iterationer i potensmetoden.

Eftersom vi har antagit att övergångsmatrisen $\mathbf{G}$ har tre egenvektorer så kommer det karaktäristiska polynomet för $\mathbf{G}$ att bli enligt ekvation (3.20). Vad som även är känt är att det första egenvärdet är 1, ty $\mathbf{G}$ är en markovmatris och att $p_{\mathbf{G}}(1) = 0$, vilket medför ekvation (3.21). Vad som även är känt är att $\mathbf{G}$ satisfierar sitt egna

karaktäristiska polynom, $p_{\mathbf{G}}(\mathbf{G}) = 0$, med hjälp av Cayley-Hamiltons sats så gäller ekvation (3.22) för alla radvektorer $\mathbf{z}$ i $\mathbb{R}^n$.

$$p_{\mathbf{G}}(\lambda) = \gamma_0 + \gamma_1\lambda + \gamma_2\lambda^2 + \gamma_3\lambda^3 \tag{3.20}$$

$$p_{\mathbf{G}}(1) = 0 \Rightarrow \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3 = 0 \tag{3.21}$$

$$\begin{aligned} \mathbf{z}p_{\mathbf{G}}(G) &= \mathbf{z}\Big[\gamma_0\mathbf{I} + \gamma_1\mathbf{G} + \gamma_2\mathbf{G}^2 + \gamma_3\mathbf{G}^3\Big] = 0 \\ \Rightarrow \mathbf{r}^{(k-3)}&\Big[\gamma_0\mathbf{I} + \gamma_1\mathbf{G} + \gamma_2\mathbf{G}^2 + \gamma_3\mathbf{G}^3\Big] = 0 \end{aligned} \tag{3.22}$$

Följande definitioner införs:

$$\begin{aligned} \mathbf{y}^{(k-2)} &= \mathbf{r}^{(k-2)} - \mathbf{r}^{(k-3)} \\ \mathbf{y}^{(k-1)} &= \mathbf{r}^{(k-1)} - \mathbf{r}^{(k-3)} \\ \mathbf{y}^{(k)} &= \mathbf{r}^{(k)} - \mathbf{r}^{(k-3)} \end{aligned} \tag{3.23}$$

Med hjälp ekvationerna (3.18), (3.21), (3.22) samt definitionerna i (3.23) kan ekvationen (3.24) härledas.

$$\gamma_1\mathbf{y}^{(k-2)} + \gamma_2\mathbf{y}^{(k-1)} + \gamma_3\mathbf{y}^{(k)} = 0 \tag{3.24}$$

Eftersom den triviala lösningen $\gamma_i = 0$ så begränsas den ledande termen i det karaktäristiska polynomet till $\gamma_3 = 1$. Ett överbestämt ekvationssytem kommer då erhållas som löses med minstakvadrat-metoden och ekvation (3.25). Plustecknet betecknar pseudoinversen.

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = \begin{pmatrix} \mathbf{y}^{(k-2)} & \mathbf{y}^{(k-1)} \end{pmatrix}^{+} \mathbf{y}^{(k)} \tag{3.25}$$

Alla koefficenter i (3.20) är nu kända och definitionerna i ekvation (3.26) införs. Sedan kommer man fram till lösningen med hjälp av Cayley-Hamiltons sats, se ekvation (3.27).

$$\begin{aligned} \beta_0 &= \gamma_1 + \quad \gamma_2 + \gamma_3 \\ \beta_1 &= \qquad \gamma_2 + \gamma_3 \\ \beta_2 &= \qquad\quad \gamma_3 \end{aligned} \tag{3.26}$$

$$\mathbf{u}_1 = \beta_0\mathbf{r}^{k-2} + \beta_1\mathbf{r}^{k-1} + \beta_2\mathbf{r}^{k} \tag{3.27}$$

Lösningen på $\mathbf{u}_1$ kommer som sagt att användas i nästa iteration av potensmetoden. Den kvadratiska extrapolationen kommer genomföras periodiskt för att snabba upp konvergensen. Kvadratisk extrapolation berörs i artikeln [6] och för en mer ingående förklaring hänvisas läsaren till originalartikeln samt den experimentella delen där beräkningsalgoritmen implementeras.

3.4.3 IAD-metoden med Tarjans omsorteringsalgoritm

Tarjans omsorteringsalgoritm syftar till att utifrån en reducibel kvadratisk matris med starkt kopplade (irreducibla) delmängder hitta en permutation så att den blir blockvis övre triangulär (se ekvation (3.28)) [4, 13]. Intuitivt inses att en komplett matris (exempelvis $\mathbf{G}$) inte kan permuteras till denna form, ty avsaknad av noll-element. Istället appliceras denna omsortering lämpligen på den riktade matrisen $\mathbf{A}$. Hur omsorteringen går till beskrivs i [4].

$$\mathbf{T} = \begin{pmatrix} \mathbf{T_0} & \mathbf{T_1} \\ \mathbf{0} & \mathbf{T_2} \end{pmatrix} \tag{3.28}$$

En alternativ metod för att lösa ekvation (3.7) är den s.k. ”iterativ aggregering-disaggregering”-metoden (eng. ”iterative aggregation-disaggregation method”), förkortat kallad IAD-metoden.

$$\overline{\mathbf{A}} = \begin{pmatrix} \mathbf{A}_0 & \dots & \mathbf{0} \\ \mathbf{0} & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{m-1} \end{pmatrix} \tag{3.29}$$

Antag en av $\mathbf{A}$ permuterad matris $\overline{\mathbf{A}}$ (omsorterad med hjälp av Tarjans algoritm) enligt (3.29). Varje delmatris $\mathbf{A_i}$ $(0 \leq i < m)$ representerar en starkt kopplad (irreducibel) delmängd. Låt oss definiera en aggregeringsmatris $\mathbf{X}_{m\times k}$ och disaggregeringsmatris $\mathbf{Y}(r)_{k\times m}$ enligt

$$\mathbf{X}_{i,j} = \begin{cases} 1 & j \in \mathbf{A_i} \\ 0 & \text{f.ö.} \end{cases} \tag{3.30}$$

$$\mathbf{Y}(r)_{j,i} = \begin{cases} \dfrac{r_j}{\sum_{s\in\mathbf{A_i}} r_s} & j \in \mathbf{A_i} \\ 0 & \text{f.ö.} \end{cases} \tag{3.31}$$

Notera att $\mathbf{XY}(r) = \mathbf{I}$. Utan vidare förklaring (djupare förklaring för den nyfikne finnes i [12]) skapas

$$\mathbf{T} = \mathbf{M}^{-1}\mathbf{W} \tag{3.32}$$

$$\mathbf{I} - \overline{\mathbf{A}} = \mathbf{M} - \mathbf{W} \tag{3.33}$$

där $\mathbf{M}^{-1} \geq 0$ och $\mathbf{M}^{-1}\mathbf{W} \geq 0$. Nu kan PageRank-vektorn beräknas med IAD-metoden.

IAD-algoritm

Steg 1 Bestäm en initialapproximation för $\mathbf{r}$ (lämpligen $\mathbf{r}^{(0)} = \mathbf{v}$) och sätt $i = 0$.

Steg 2 Beräkna den aggregerade matrisen $\mathbf{X}\overline{\mathbf{A}}^s\mathbf{Y}(\mathbf{r}^{(i)})$ för något positivt heltal s och lös ekvationen

$$\mathbf{X}\overline{\mathbf{A}}^s\mathbf{Y}(\mathbf{r}^{(i)})\overline{\mathbf{r}} = \overline{\mathbf{r}} \tag{3.34}$$

Steg 3 Beräkna den nya approximationen för $\mathbf{r}$ med ett lämpligt positivt heltal t

$$\mathbf{r}^{(i+1)} = \mathbf{T}^t\mathbf{Y}(\mathbf{r}^{(i)})\overline{\mathbf{r}} \tag{3.35}$$

Steg 4 Upprepa algoritmen från steg 2 med $i = i + 1$ tills konvergens uppnås (ändringen tillräckligt liten).

Kapitel 4

Andra rankningsalgoritmer

Det här kapitlet kommer behandla två stycken andra algoritmer som används för att lösa problemet med hur man rangordnar sidor på Internet. Den första algoritmen, HITS, skiljer sig en del från PageRank och kommer beskrivas mer i detalj senare. Sist så kommer algoritmen SALSA att kortfattat nämnas då den är en kombination av idéer från PageRank och HITS. Syftet med kapitlet är främst för att ge läsaren en inblick över vad det finns för lösningar utöver PageRank ur ett allmänt bildande perspektiv. Algoritmen som berörs kommer även kortfattat att jämföras med PageRank för att framhäva styrkor och svagheter hos respektive.

4.1 HITS

HITS[1] är en algoritm för att rangordna sidor på Internet och definierar två olika begrepp för rangordningen: Nav (eng. hub) och auktoritet (eng. authority). Ett nav är en sida med endast utlänkar och en auktoritet är en sida med inlänkar. En sida på Internet kommer att inneha både en *navpoäng* och en *auktoritetspoäng* där tesen med HITS är att sidor med god auktoritet pekas på från goda nav och goda nav pekar på goda auktoriteter. [8]

För att beräkna nav- samt auktoritetspoäng över en mängd sidor med hjälp av HITS-algoritmen så behövs en matris som representerar kopplingar mellan sidorna i mängden, en så kallad grannmatris. Låt $\mathbf{L}$ vara en grannmatris över en mängd med n sidor, se figur 4.1.

$$\mathbf{L} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & \dots & n \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & \dots & 0 \\ 1 & 0 & 1 & \dots & 1 \\ 1 & 0 & 0 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix} \end{array} \tag{4.1}$$

[1] Hyper-induced Topic Search

Där kanterna representeras av ettor i matrisen enligt följande definition [8]:

$$\mathbf{L}_{i,j} = \begin{cases} 1 & \text{om det finns en länk från i till j} \\ 0 & \text{annars} \end{cases} \tag{4.2}$$

Då grannmatrisen för den undersökta mängden av sidor finns tillgänglig kan man beräkna respektive sidas nav- samt auktoritetspoäng. Navpoängen beräknas genom att man summerar alla auktoritetspoänger hos sidorna som navet pekar på. Auktoritetspoängen för en sida beräknas genom att man summerar alla navpoänger hos de nav som pekar på sidan. Auktoritetspoängen för samtliga sidor i mängden representeras av en vektor $\mathbf{x}$ och navpoängen för samtliga sidor i mängden representeras av en vektor $\mathbf{y}$. På följande sätt beräknas sidornas rankning med avseende på auktoritetspoäng och navpoäng: [8]

$$\mathbf{x}^{(k)} = \mathbf{L}^T\mathbf{y}^{(k-1)} \tag{4.3}$$

$$\mathbf{y}^{(k)} = \mathbf{L}\mathbf{x}^{(k)} \tag{4.4}$$

Ser man till ekvation (4.3) transponeras vektorn eftersom det är inlänkarnas navpoäng som är intressant, med andra ord så kommer i:te värdet i vektorn $\mathbf{x}$ bestå av summan av produkterna mellan i:te kolumnen i $\mathbf{L}$ och vektorn med alla navpoäng, se ekvation (4.5).

$$\mathbf{x}_i = \mathbf{L}_{1,i} \cdot \mathbf{y}_1 + \mathbf{L}_{2,i} \cdot \mathbf{y}_2 + ... + \mathbf{L}_{n,i} \cdot \mathbf{y}_n \tag{4.5}$$

I ekvation (4.4) så beräknas navpoängen och där transponeras inte grannmatrisen eftersom man är intresserad av utlänkarna i grannmatrisen. Det i:te värdet i vektorn $\mathbf{y}$ kommer då bestå av summan av produkterna i i:te raden i $\mathbf{L}$ och vektorn med alla auktoritetspoäng, se ekvation (4.6).

$$\mathbf{y}_i = \mathbf{L}_{i,1} \cdot \mathbf{x}_1 + \mathbf{L}_{i,2} \cdot \mathbf{x}_2 + ... + \mathbf{L}_{i,n} \cdot \mathbf{x}_n \tag{4.6}$$

Enligt (4.3) och (4.4) så är HITS-algoritmen en iterativ process och de slutliga rankningarna fås då vektorerna konvergerat. Om man genomför en enkel substitution mellan (4.3) och (4.4) så kan man beräkna auktoritets- respektive navpoäng med hjälp av egenvektorer, enligt ekvation (4.7) och (4.8). [8]

$$\mathbf{x}^{(k)} = \mathbf{L}^T\mathbf{L}\mathbf{x}^{(k-1)} \tag{4.7}$$

$$\mathbf{y}^{(k)} = \mathbf{L}\mathbf{L}^T\mathbf{y}^{(k-1)} \tag{4.8}$$

Dessa ekvationer löses genom att man beräknar de dominanta egenvektorerna med hjälp av potensmetoden för $\mathbf{L}^T\mathbf{L}$ (auktoritetsmatrisen) och $\mathbf{L}\mathbf{L}^T$ (navmatrisen). [8]

Till skillnad från PageRank så har man inga förberäknade värden med HITS-algoritmen utan den beräknas beroende på vilka sidor som innehåller den givna

söktermen och dess grannar. Dessa sidor kommer att befolka **L**-matrisen. Beräkningen kommer därmed endast göras med avseende på sidor som är relaterade till söktermen och inte innefatta hela Internet, vilket underlättar beräkningsprocessen.

En av nackdelarna med HITS gentemot PageRank är dock att för varje sökanrop så kommer en grannmatris behöva framställas med avseende på given sökterm innan beräkningar kan göras. Vad som även är negativt med HITS är att en användare kan påverka både sin nav- och auktoritetspoäng genom att ha mycket utlänkar på sin sida eftersom nav- och auktoritetspoäng är beroende av varandra. En annan nackdel är att sidor som inte alls relaterar till given sökterm kan tas i beakt om den sidan är anknuten till en sida som innehåller given sökterm.

4.2 SALSA

SALSA[2] innefattar idéer från båda HITS och PageRank. Det som är gemensamt med HITS och SALSA är att man använder sig av konceptet med nav och auktoriteter. SALSA använder dock markovkedjor för att sedan beräkna rankningen liksom PageRank. Att SALSA använder två starka egenskaper från både HITS och PageRank gör den intressant i området rankningsalgoritmer. [8]

En fördel med SALSA är att man ärver möjligheten att rangordna både nav och auktoriter från HITS och genom att man använder sig av markovkedjor liksom PageRank så minskas beroendet mellan nav- och auktoritetspoäng. En klar nackdel med SALSA är att även den är beroende av söktermen liksom HITS. [8]

[2]Stochastic Approach for Link Structure Analysis

Del II

Experimentell del

Kapitel 5

Metod

Den experimentella delen innefattar implementation och prestandamätningar av PageRank-konceptet. Implementationen utförs i MATLAB [9].

Datamängden som representerar Internet är en kopia av hur Stanfords webbplats (Stanford.edu) såg ut 2002 [14]. Sidstrukturen på webbplatsen återges av en grafrepresentation som innehåller 281903 noder (sidor) och 2312497 kanter (länkar). På grund av prestandaskäl används endast 1/50 av noderna vilket medför att även kanter bortfaller i samma utsträckning. Vid selektionen av denna 1/50-del antas grafens noder inte ha någon sortering utan vara slumpmässigt ordnade. Med hänsyn till det antagandet har grafens första block innehållande de 1/50 första noderna valts ut som datamängd att utföra beräkningarna på.

Prestandamätningarna avser både antalet iterationer och tiden det tar för beräkningarna att konvergera. Tiden är en särskilt relevant storhet i sammanhanget då olika beräkningsmetoder ska jämföras och att dessa utför olika många beräkningar i varje iteration—en minskning i antal iterationer kan fortfarande vara tidsmässigt sämre. Tidsperspektivet är något som tidigare forskning inte belyst då man—medvetet eller omedvetet—eftersträvar att glorifiera sin egen metod genom att visa hur pass många färre iterationer som görs. Konvergenskravet för beräkningarna är när 1-normen av residualvektorn understiger $\epsilon = 10^{-6}$.

Nedan specifieras den hårdvara som användes vid testerna.

Processor:	Intel Core 2 Duo T7500
Processorns klockfrekvens:	2.2 GHz
Antal kärnor:	2
L2 Cache:	4 MB
Busshastighet:	800 MHz
GFLOPS:	17.6
Minne:	2 GB DDR2 SDRAM
GPU Chipset:	GeForce 8600M GT
VRAM:	128 MB

Kapitel 6

Genomförande

Tidigare teori implementeras med hjälp av existerande implementationer och tillgänglig pseudokod. MATLAB-koden presenteras i sin helhet tillsammans med adekvat dokumentation. Resultaten presenteras som grafer (se figur 6.1-6.5) där residualvärdets storlek i förhållande till antal iterationer och förlupen tid presenteras.

6.1 Förberedelser

6.1.1 MATLAB-kod

```
% @file: GetMatrix.m
% @authors: David Tornquist, Fredrik Cedervall
%
% Load the matrix defined in file (of the tab separated sparse
% matrix format "N1 N2") and extract 1/subset that is transformed
% into the directed adjency matrix A (1 = edge, 0 = no edge)
function [A] = GetMatrix(file, num_nodes, num_edges, subset)
     % Load the matrix from file
     M = dlmread(file, '\t');

     subset_ratio = num_nodes/subset;

     % Allocate memory for the adjency matrix
     A = zeros(floor(subset_ratio), floor(subset_ratio));

     % Add all edges to the adjacent matrix
     for i=1 : i_num_edges
          if M(i,1) < (subset_ratio) && M(i,2) < (subset_ratio)
               A(M(i,1), M(i,2)) = 1;
          end
     end
end
```

```
% @file: FindZeroRows.m
% @authors: David Tornquist, Fredrik Cedervall
%
% Find zero rows in the matrix M. The results will be
% indicated in the vector z with a 1 or 0 for zero-rows
% and non-zero-rows, respectively.
function [z] = FindZeroRows(M)
     % Sum each row in M
     z = sum(M, 2);
     % Make the zero rows = 1 and non-zero rows = 0
     z = sign(-z)+1;
end
```

```
% @file: Initialize.m
% @authors: David Tornquist, Fredrik Cedervall
%
% To prevent the time-wasting work of loading the matrix
% every time something is changed or tested, this initialization
% procedure is extracted to this file.

clear, clc
format long

% Graph to use
graph = 'web-Stanford-revised.txt';
num_nodes = 281903;
num_edges = 2312497;

% Only use 1/subset of the graph
subset = 50;

% Get the matrix
A = GetMatrix(graph, num_nodes, num_edges, subset);

% Size of the adjency matrix
k = size(A, 2);

% Row vector of ones
e = ones(1, k);

% Initial vector ("teleportation vector")
v = e / k;

% Pertubation matrix
E = e' * v;

% Vector which indicates zero rows (1 if zero row, 0 otherwise)
z = FindZeroRows(A);

% Fill zero rows with teleportation vector
P = A + diag(z) * E;

% Normalization matrix
N = repmat(sum(P, 2), 1, floor(num_nodes/subset));

% Normalize P row wise so that each row sum is 1
P = P ./ N;

% Generate the Google matrix from the transition matrix
a = 0.85;
G = a * P + (1 - a) * E;
```

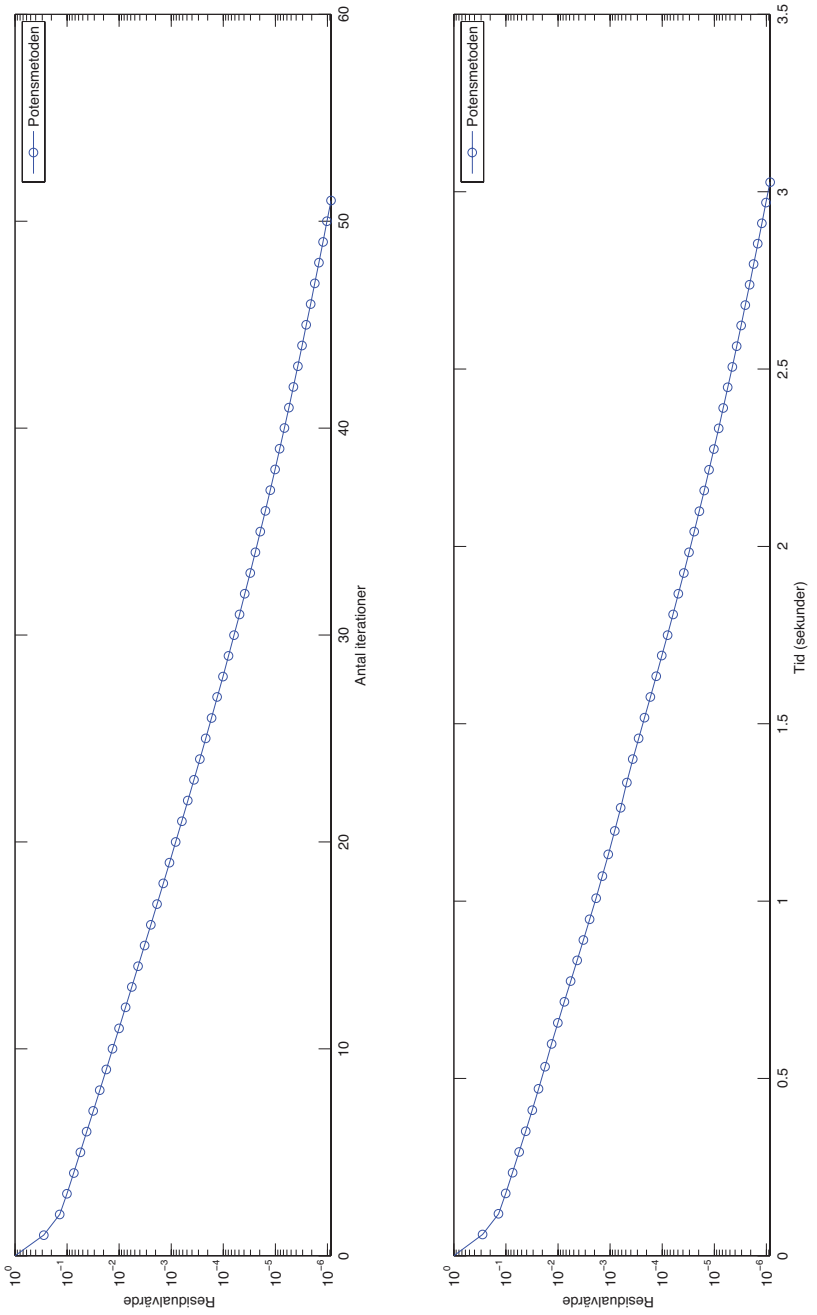
6.2 Potensmetoden

Potensmetoden är implementerad utifrån pseudokoden från [6] och Molers kapitel om PageRank [10].

6.2.1 MATLAB-kod

```
% @file: PowerMethod.m
% @authors: David Tornquist, Fredrik Cedervall
%
% Calculate PageRank with the power method
function [r, numiter, time] = PowerMethod(r_0, G, epsilon)
     % Start clock
     tic;

     % Initial PageRank-vector
     r_1 = r_0;

     % Initial diff value (make sure greater than epsilon)
     delta = epsilon + 1;

     % Iterations
     i = 0;

     % Loop until residual is small enough
     while(delta > epsilon)
          % Calculate next PageRank-vector
          r_1 = r_0 * G;

          % Calculate the residual value as the 1-norm
          delta = norm(r_1 - r_0, 1);

          % Set the new PageRank vector for next iteration
          r_0 = r_1;

          % Increment iteration variable
          i = i + 1;
     end

     % Normalize rows (sum 1)
     r = r_1 / sum(r_1);

     % Number of iterations performed
     numiter = i;

     % Stop clock
     time = toc;
end
```

6.2.2 Resultat



Figur 6.1: Potensmetoden. Konvergens efter 51 iterationer (2.98 sekunder) ger 505 unika PageRank-värden.

6.3 Extrapolationsmetoden

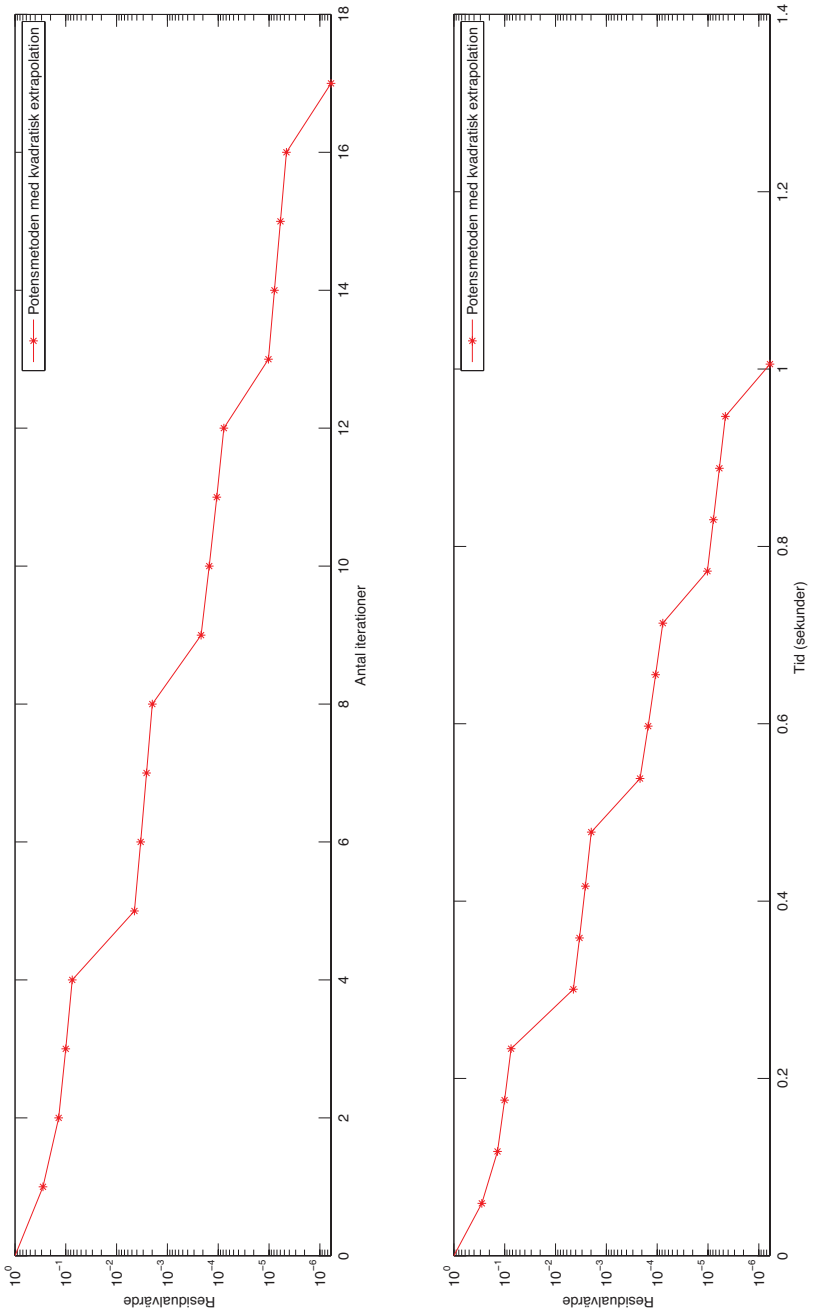
Potensmetoden med kvadratisk extrapolation har implementerats med hjälp av pseudokoden från [6] och en liknande implementation i MATLAB av Amy N. Langville [7].

6.3.1 MATLAB-kod

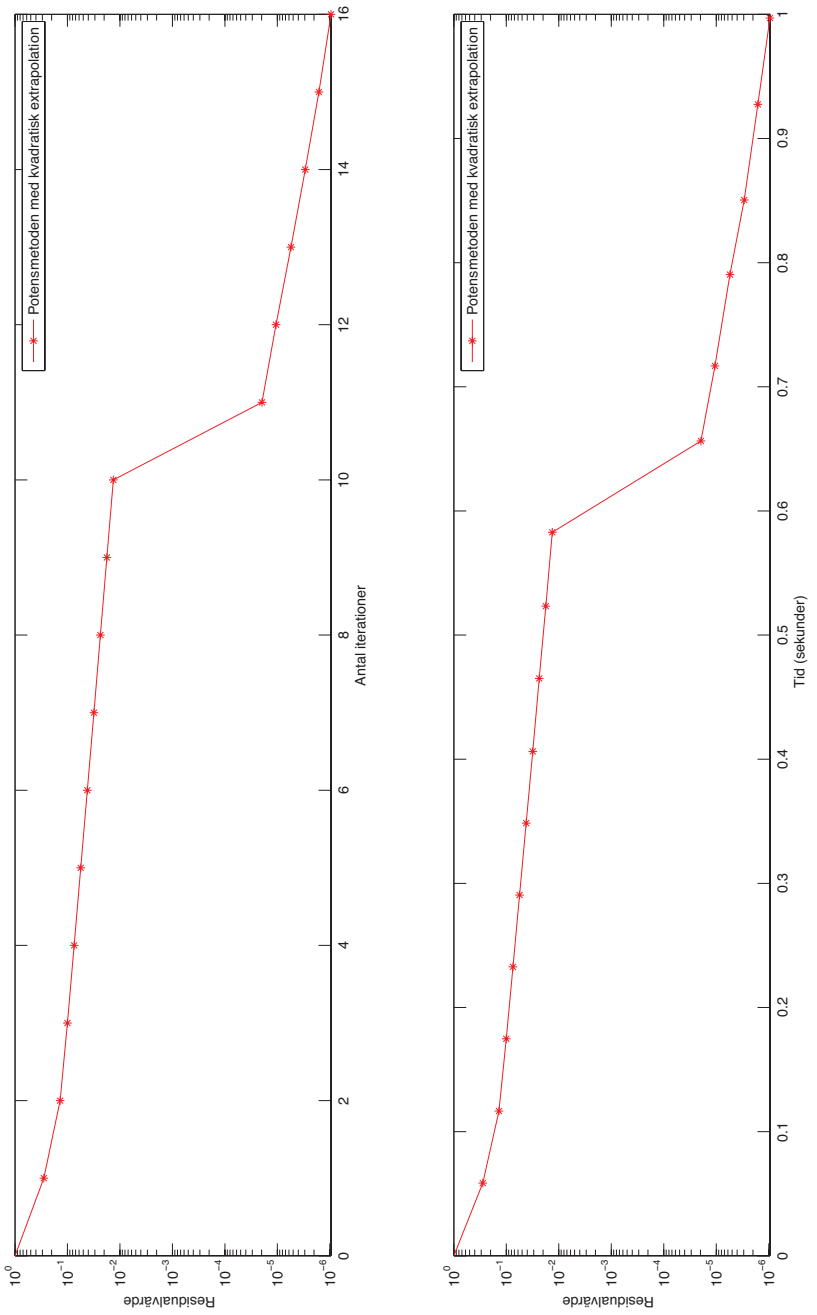
```
% @file: QuadraticExtraPolation.m
% @authors: David Tornquist, Fredrik Cedervall
%
% Quadratic extrapolation based on three previous values (r0, r1, r2)
% an estimate of the current value (r3).
function [r] = QuadraticExtraPolation (r_0, r_1, r_2, r_3)

    % Define y
    y_0         = r_1 - r_0;
    y_1         = r_2 - r_0;
    y_2         = r_3 - r_0;

    % Problem to solve with Gram-Schmidt
    Y = [y_0' y_1'];

    % Gram-Schmidt process ortho-normalizing a set of vectors
    [m, n] = size(Y);
    Q = zeros(m,n);
    R = zeros(n);

    for j=1:n
         R(j,j) = norm(Y(:,j));
         Q(:,j) = Y(:,j)/R(j,j);
         R(j,j+1:n) = Q(:,j)'*Y(:,j+1:n);
         Y(:,j+1:n) = Y(:,j+1:n) - Q(:,j)*R(j,j+1:n);
    end

    Q_y_2 = Q' * y_2';

    % Trivial solution is not interesting...
    gamma_3 = 1;
    % Calculate the other gamma values from Gram-Schmidt values
    gamma_2 = - Q_y_2(2) / R(2,2);
    gamma_1 = ( - Q_y_2(1) - gamma_2 * R(1,2)) / R(1,1);

    beta_0 = gamma_1 + gamma_2 + gamma_3;
    beta_1 = gamma_2 + gamma_3;
    beta_2 = gamma_3;

    % Based on the assumption that r is an linear combination of the
    % three eigenvectors.
    r = beta_0*r_1 + beta_1 * r_2 + beta_2 * r_3;

end
```

```
% @file: QuadraticPowerMethod.m
% @authors: David Tornquist, Fredrik Cedervall
%
% Calculate PageRank with the power method utilizing
% quadratic extrapolation.
function [r, numiter, time] = QuadraticPowerMethod(r_0, G, epsilon,
    extrapolation_frequence)

     % Start clock
     tic;

     % Initial PageRank-vectors
     r_1 = r_0 * G;
     r_2 = r_1 * G;

     % Initial diff value (make sure greater than epsilon)
     delta = epsilon + 1;

     % Iterations (two vectors already calculated above)
     i = 2;

     % Loop until residual is small enough
     while(delta > epsilon)

          % Calculate next PageRank vector
          r_3 = r_2 * G;

          % Calculate the residual value as the 1-norm
          delta = norm(r_3 - r_2, 1);

          % Execute quadratic extrapolation in a specified frequence
          if(mod(k, extrapolation_frequence) == 0)
               r_3 = QuadraticExtraPolation(r_0, r_1, r_2, r_3);
          end

          % Set PageRank vectors for next iteration
          r_0 = r_1;
          r_1 = r_2;
          r_2 = r_3;

          % Increment iteration variable
          i = i + 1;
     end

     % Normalize rows (sum 1)
     r = r_3 / sum(r_3);

     % Number of iterations performed
     numiter = i;

     % Stop clock
     time = toc;
end
```

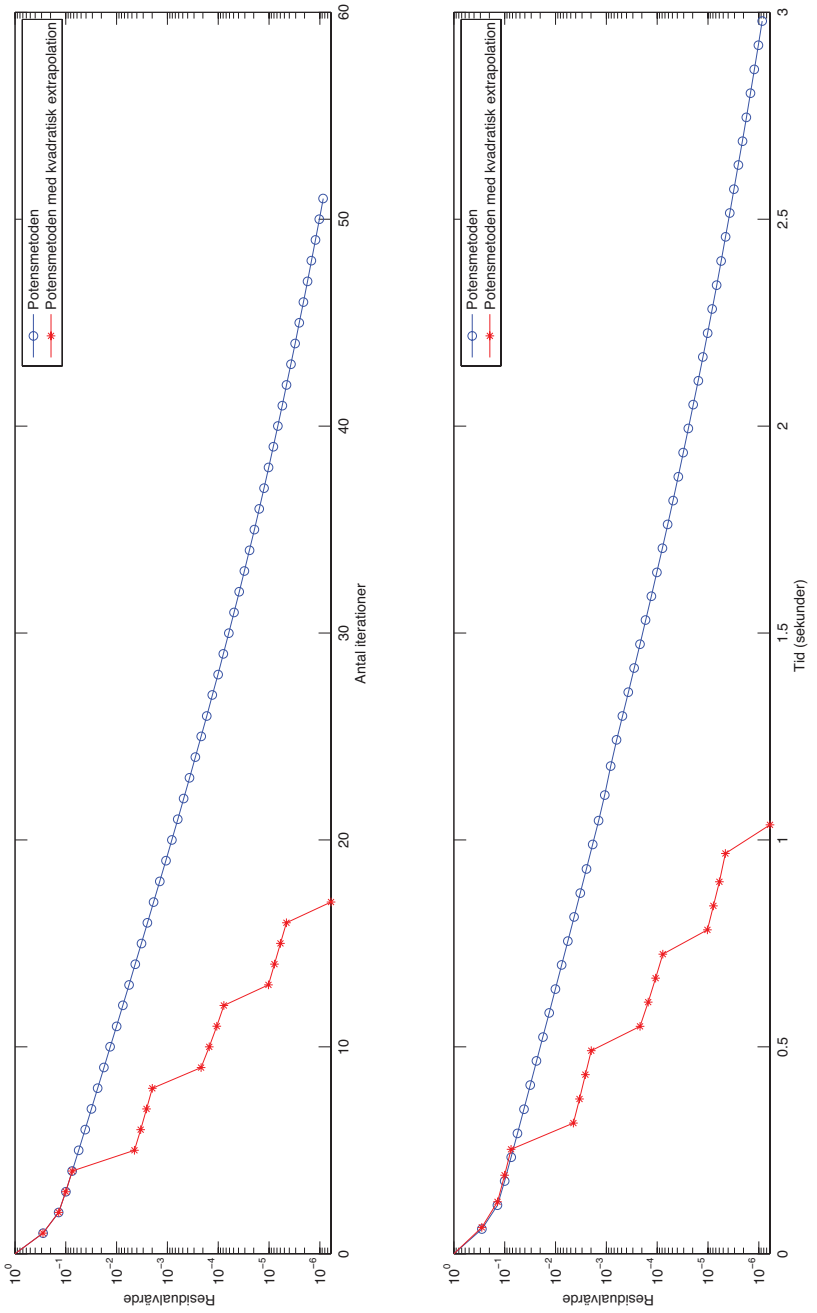
6.3.2 Resultat



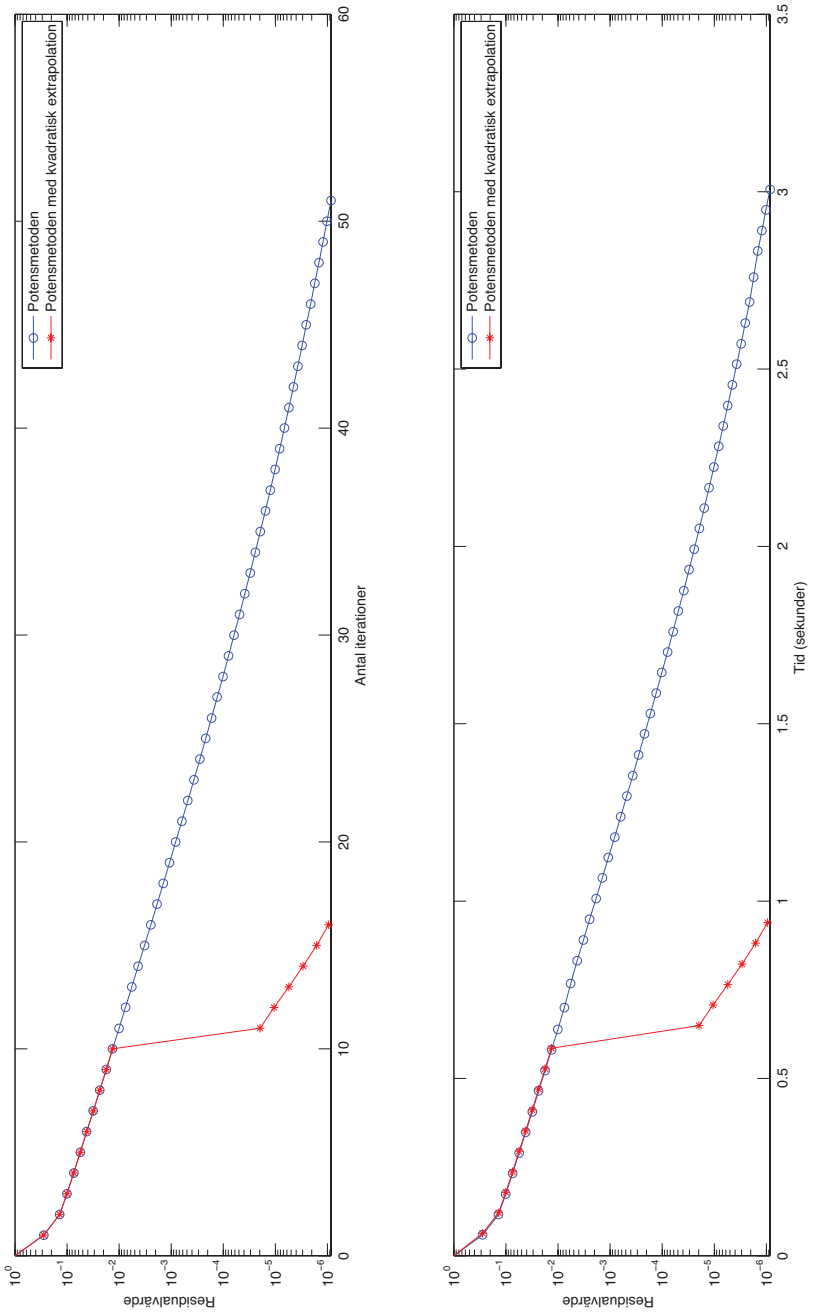
Figur 6.2: Potensmetoden med extrapolation var fjärde iteration. Konvergens efter 17 iterationer (1.04 sekunder) ger 487 unika PageRank-värden.



Figur 6.3: Potensmetoden med extrapolation var tionde iteration. Konvergens efter 16 iterationer (0.94 sekunder) ger 409 unika PageRank-värden.



Figur 6.4: Potensmetoden med extrapolation var fjärde iteration jämfört med potensmetoden



Figur 6.5: Potensmetoden med extrapolation var tionde iteration jämfört med potensmetoden.

6.4 IAD-metoden med försortering med Tarjans algoritm

Pultarová beskriver i [12] hur PageRank kan beräknas med hjälp av IAD-metoden tillsammans med Tarjans algoritm. IAD-metoden är i sig själv rättfram och lättimplementerad. Förberedelserna med Tarjans algoritm, omsortering och skapandet av matriserna $\mathbf{X}$, $\mathbf{Y}$ och $\mathbf{T}$ (se avsnitt 3.4.3) är det däremot inte, även om den tillgängliga beskrivningen till synes är enkel.

I MATLAB finns funktionen `graphconncomp` som implementerar Tarjans algoritm för att finna samtliga starkt kopplade komponenter i en graf. Däremot ges ingen information om hur dessa ska placeras inbördes för att en *så diagonaliserad permutation som möjligt* ska uppnås. Ingen heuristisk lösning funnen för att lösa detta. Försöken resulterar i att den omsorterade matrisen inte får den önskade form som Pultarová beskriver i [12] utan istället en till synes godtycklig form.

Den beskrivna algoritmen för att beräkna PageRank med IAD-metoden innehåller klara praktiska brister. Då omsortering med hjälp av Tarjans algoritm endast har någon adekvat effekt på glesa matriser kan pertubationsmatrisen $\mathbf{E}$ inte införas, ty den är en komplett matris. Detta medför att den personaliserade initialvektorn $\overline{\mathbf{v}}$ inte kan införas i modellen. I avsnitt 3.3 visades vikten och styrkan med att kunna applicera denna vektor i modellen. Google är beroende av detta för att kunna *moderera* webben; länksamlingar som upprättats för att slussa högre PageRank-värde till utvalda sidor kan bestraffas och sidor som ligger i Googles intresse belönas (Google.com innehar såklart PageRank 10.0).

IAD-metodens bristande användbarhet, tillsammans med dess ofullständiga beskrivning vilket gör implementationen krävande utanför ramarna för denna uppsats, implementeras inte denna. Istället använder vi Pultarovás resultat [12] i diskussionen kring resultaten.

Del III

Resultat

Kapitel 7

Diskussion

Den här uppsatsen ämnade undersöka tre olika beräkningsmetoder av PageRank: Potensmetoden, potensmetoden med kvadratisk extrapolation samt IAD-metoden med Tarjans omsorteringsalgoritm. De två sistnämnda avser att vara prestandamässiga förbättringar jämfört med potensmetoden.

Vad som är särskilt relevant då olika beräkningsmetoder jämförs med varandra är den tidsmässiga aspekten. Tidigare forskning som refereras till i uppsatsen fokuserar istället på antalet iterationer, vilket kan vara väldigt missvisande ty varje iterations tidsmässiga *kostnad* inte framkommer. Att lyfta fram det praktiska perspektivet, och därmed tidsperspektivet, är av stor vikt i vår uppsats. Att tidigare rapporter utelämnar denna aspekt kan bero på att författarna avser glorifiera sina egna resultat.

Ur ett prestandamässigt perspektiv erhölls dock ungefär en 60-70%-ig förbättring, såväl iterations- som tidsmässigt, genom att använda sig av kvadratisk extrapolation tillsammans med potensmetoden jämfört med enbart potensmetoden. Resultatet varierade beroende på val av frekvens av extrapolationen. Som kompromiss till prestandaförbättringen får det antas att extrapolationen leder till något försämrad noggrannhet i de beräknade svaren—beräkningar med uppskattade värden bör rimligen hålla sämre noggrannhet än motsvarande värden som endast mattas av genom beräkningsnoggrannhet och cancellation. Antalet unika värden i den beräknade PageRank-vektorn ses som ett mått på resultatets noggrannhet. Detta motiveras med antagandet att det är orimligt att flera sidor har exakt samma PageRank-värde ty det skulle kräva att länkstrukturen på Internet är någorlunda symmetrisk med isomorfa delmängder. Vid användandet av extrapolation erhålls $< 20\%$ minskning av antalet unika värden i PageRank-vektorn.

IAD-metoden med Tarjans omsorteringsalgoritm implementerades inte i vår undersökning på grund av de praktiska brister den har kombinerat med komplexiteten av att implementera den, som tidigare diskuterat. De praktiska bristerna medför att IAD-metoden med Tarjans omsorteringsalgoritm har ett högre teoretiskt värde än ett praktiskt nyttovärde.

Enligt Pultarová går IAD-metoden med Tarjans omsorteringsalgoritm itera-

tionsmässigt snabbare än potensmetoden—ungefär 60-70% [12]. Vad som dock inte framgår är det tidsmässiga perspektivet på dels IAD-metodens iterationer samt tiden för Tarjans omsorteringsalgoritm. Antalet komplexa matrisoperationer indikerar dock att förberedelserna och iterationerna är relativt dyra. I sitt praktiska sammanhang har därför potensmetoden med kvadratisk extrapolation större signifikans, med såväl praktiskt nyttovärde som konkurrenskraftig prestandaförbättring.

Litteraturförteckning

[1] Howard Anton och Chris Rorres. *Elementary Linear Algebra*. John Wiley & Sons, 2005.

[2] Norman L. Biggs. *Discrete Mathematics*. Oxford University Press, 2002.

[3] Gunnar Blom, Jan Enger, Gunnar Englund, Jan Grandell, och Lars Holst. *Sannolikhetsteori och statistikteori med tillämpningar*. Studentlitteratur, 2005.

[4] Iain S. Duff och John K. Reid. An Implementation of Tarjan's Algorithm for the Block Triangularization of a Matrix. *ACM Transactions on Mathematical Software*, 4(2):137–147, June 1978.

[5] Jan Enger och Jan Grandell. Markovprocesser och köteori. 2006.

[6] Sepandar D. Kamvar, Taher H. Haveliwala, Christopher D. Manning, och Gene H. Golub. Extrapolation methods for accelerating pagerank computations. ACM, May 2003.

[7] Amy N. Langville. Matlab implementation of personalized pagerank power algorithm with quadratic extrapolation, 2010. URL `http://spinner.cofc.edu/~langvillea/PRDataCode/Lang-quadpagerank.m`.

[8] Amy N. Langville och Carl D. Meyer. A Survey of Eigenvector Methods for Web Information Retrieval. *SIAM Review*, 2005.

[9] Mathworks. Matlab, April 2010. URL `http://www.mathworks.com/products/matlab/`.

[10] Cleve Moler. Experiments with matlab, 2009. URL `http://www.mathworks.com/moler/exm/index.html`.

[11] Lawrence Page, Sergey Brin, Rajeev Motwani, och Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL `http://ilpubs.stanford.edu:8090/422/`. Previous number = SIDL-WP-1999-0120.

[12] Ivana Pultarová. Tarjan's Algorithm in Computing PageRank. Technical report, Department of Mathematics, Faculty of Civil Engineering, Czech Technical University in Prague, 2005.

[13] Ivana Pultarová. Implementation of Tarjan's algorithm in Matlab. Technical report, Department of Mathematics, Faculty of Civil Engineering, Czech Technical University in Prague, 2008.

[14] Stanford University. Stanford large network dataset collection, April 2010. URL `http://snap.stanford.edu/data/index.html`.