

On-line Page Importance Computation (OPIC)

*basierend auf der gleichnamigen Veröffentlichung von
Serge Abiteboul, Mihai Preda, Gregory Cobena*

Dozent: Prof. Dr. Ulrik Brandes
Betreuer: Christian Pich
Vortragender: Simon Endeke

Zusammenfassung des Vortrags vom 20.12.2005

Seminar *Methoden der Netzwerkanalyse*

Konstanz, 25.01.2006

Inhaltsverzeichnis

1	Einführung	2
1.1	Motivation	2
1.2	Beschreibungsmöglichkeiten für Seitenwichtigkeiten	2
1.3	PageRank-Algorithmus	3
2	OPIC-Algorithmus für statische Graphen	5
2.1	Idee	5
2.2	OPIC-Algorithmus	5
2.3	Vor- und Nachteile	7
3	Crawling-Strategien	7
3.1	Fehlerfaktor	7
3.2	Beispielstrategien	8
3.3	Experimentelle Ergebnisse	8
4	Adaptive OPIC	9
4.1	Änderung im Graphen	9
4.2	Window policies	9
4.3	Experimentelle Ergebnisse	10
5	Diskussion	11
5.1	Unterschiede im Modell zu PageRank	11
5.2	Kritische Sicht auf die Vorteile von OPIC	13

1 Einführung

1.1 Motivation

Internet-Suchseiten wie Google liefern bei einer Suchanfrage oft sehr große Treffermengen. Daher benötigen sie Bewertungskriterien für die Wichtigkeit von Seiten, nach denen sie ihre Suchergebnisse sortieren können.

Neben anderen Methoden (z. B. String-Vergleiche im Linktext) werden auch Methoden der Netzwerktheorie auf das Web angewendet, um die Wichtigkeit von Webseiten zu messen. Dabei wird das Internet als gerichteter Graph aufgefasst: Internetseiten sind Knoten und Links zwischen den Seiten sind gerichtete Kanten. Da eine Seite mehrere Links auf eine andere Seite haben kann, handelt es sich sogar um einen Multigraphen.

Der vorgestellte Algorithmus hat auch noch andere Anwendungsgebiete. Den Algorithmus von Google zu verbessern, sollte jedoch als Motivation ausreichen.

1.2 Beschreibungsmöglichkeiten für Seitenwichtigkeiten

Intuitive Beschreibung: Einfluss

Eine Seite wird umso wichtiger eingestuft, je wichtiger die Seiten, die einen Link auf diese Seite haben, in der Summe sind. Sie hat also selbst keinen Einfluss auf ihre eigene Wichtigkeit, sondern die Seiten beeinflussen nur die Wichtigkeit ihrer Nachfolger durch ihre eigene. Dies läuft auf eine balancierte Verteilung der Wichtigkeiten hinaus.

Intuitive Beschreibung: Irrfahrt

Eine Seite ist umso wichtiger, je höher die Aufenthaltswahrscheinlichkeit auf dieser Seite bei einer zufälligen Irrfahrt durch den Webgraphen ist: Von einem Knoten aus springt ein Surfer gleichverteilt über alle ausgehenden Kanten zu einem Nachfolger-Knoten. Zur Verdeutlichung siehe nachfolgende Abbildung 1.

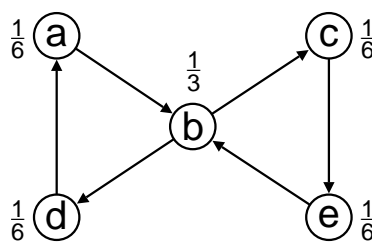


Abbildung 1: Beispielgraph mit Aufenthaltswahrscheinlichkeiten bei einer zufälligen Irrfahrt durch den Graphen

Berechnung

Genau diese Anforderungen werden durch Rückkopplungszentralitäten realisiert, z. B. Einfluss-, Eigenvektorzentralität, Hubs & Authorities und eben PageRank, die im Sommersemester in der Vorlesung behandelt wurden.

Berechnet werden Rückkopplungszentralitäten typischerweise näherungsweise durch Iteration. Wie dies genau funktioniert, werden wir in 1.3 sehen.

Probleme mit dieser Definition

Es wird gefordert, dass der Graph stark zusammenhängend und nicht bipartit ist. Dies erscheint logisch, wenn man beispielsweise einen Knoten ohne ausgehende Kanten betrachtet: Die Aufenthaltswahrscheinlichkeit auf diesem Knoten müsste 1 sein, da man vom ihm nicht mehr weg kommt. Zum einen erscheint dies unfair, zum anderen ist das Modell bei zwei Knoten mit dieser Eigenschaft nicht mehr wohldefiniert. Dass der Graph nicht bipartit sein darf, hat einen eher technischen Hintergrund: Bei der Berechnung können die Werte oszillieren und konvergieren dadurch nicht.

Vom Internet kann man jedoch nicht erwarten, dass es stark zusammenhängend und nicht bipartit ist.

Lösung

Um dieses Problem zu lösen, werden bei PageRank *dünne* Kanten eingefügt: Auf einer Seite springt ein Surfer mit einer gewissen Sprungwahrscheinlichkeit ω zu einer beliebigen URL. Mit der Gegenwahrscheinlichkeit $1 - \omega$ verfolgt er einen Link auf der Seite. Dadurch wird der Graph sogar vollständig, d. h. er ist auf jeden Fall stark zusammenhängend und bei mindestens 3 Knoten auch nicht bipartit.

1.3 PageRank-Algorithmus

Definition (Vielfachheit einer Kante)

In einem Multigraphen $G = (V, E)$ bezeichnen wir mit $\# : E \rightarrow \mathbb{N}_0$ die Vielfachheit einer Kante. Kurz: $\#(v, w)$ gibt an, wie oft die Kante (v, w) in der Kantenmultimenge E enthalten ist.

Definition (Ausgangs-/Eingangsgrad eines Knotens)

Mit $d^+(v)$ bezeichnen wir den Ausgangsgrad und mit $d^-(v)$ den Eingangsgrad, also die Mächtigkeit der Nachfolger- bzw. Vorgängermultimenge des Knotens v .

Grobe Funktionsweise des PageRank-Algorithmus

Es werden zwei Knotenarrays c und c' allokiert. In jedem Iterationsschritt berechnen wir aus den Werten des letzten Schritts (c') für einen Knoten v die neue Näherung $c[v]$ durch...

- die Wahrscheinlichkeit $\frac{\omega}{n}$, dass wir zufällig auf diese Seite gesprungen sind...
- plus $1 - \omega$ mal die Werte seiner Vorgänger – jeweils gewichtet nach Anteil an deren ausgehenden Kanten.

Formal:

$$c[v] \leftarrow \frac{\omega}{n} + (1 - \omega) \cdot \sum_{u \in N^-(v)} \frac{\#(u, v)}{d^+(u)} \cdot c'[u]$$

Der Faktor $\frac{\#(u,v)}{d^+(u)}$ steht hier für den Anteil an allen ausgehenden Kanten von u , die nach v führen.

PageRank-Algorithmus

Algorithmus 1 : PageRank näherungsweise (Brin und Page 1998) [3]

Eingabe : Multigraph $G = (V, E)$, Sprungwahrscheinlichkeit $0 < \omega < 1$
 Abbruchkriterium $\varepsilon > 0$

Daten : Knotenarray c' (letzte Näherung)

Ausgabe : Knotenarray c (PageRank)

```

foreach  $v \in V$  do
   $c[v] \leftarrow \frac{1}{n}$  //Initialisierung: Gleichverteilung
repeat
   $c' \leftarrow c$ 
  foreach  $v \in V$  do
     $c[v] \leftarrow \frac{\varepsilon}{n}$ 
    foreach  $u \in N^-(v)$  do
       $c[v] \leftarrow c[v] + (1 - \omega) \cdot \frac{\#(u,v)}{d^+(u)} \cdot c'[u]$ 
  until  $\|c - c'\| < \varepsilon$ 

```

Anmerkungen

Je kleiner ω gewählt wird, desto exakter ist die Berechnung, allerdings konvergiert sie auch umso langsamer. Die Wahl von ω ist also ein Abwägen von Genauigkeit des Ergebnisses gegen Berechnungszeit. Google verwendet aus „mysteriösen Gründen“ $\omega = 0,2$.

PageRank ist nur Grundlage des Algorithmus, den Google heute verwendet, und wie erwähnt nicht das einzige Maß für die Wichtigkeit einer Seite.

Nachteile

- Es wird vorausgesetzt, dass zu jedem Knoten all seine Vorgänger bekannt sind (also zu einer Seite all diejenigen Seiten, die einen Link auf diese Seite haben), d. h. das **gesamte** Web (bzw. der Teil, auf dem später gesucht werden kann) muss gecrawlt und als Graph **abgespeichert** werden.
- **Erst dann** kann der PageRank darauf berechnet werden, d. h. Crawling und Berechnen der Zentralität sind strikt getrennt; der Algorithmus arbeitet *offline*.
- Bei einer Änderung des Graphen muss der PageRank komplett neu berechnet werden.

2 OPIC-Algorithmus für statische Graphen

2.1 Idee

Wir „drehen“ die Berechnung um: Wenn ein Knoten „an der Reihe“ ist, wird nicht der Wert berechnet, den er **bekommt**, sondern sein Wert wird auf seine Nachbarn **verteilt**.

Dadurch müssen wir nicht zuerst das gesamte Internet crawlen und als Graph abspeichern um darauf die Zentralität zu berechnen, sondern wir berechnen während dem Crawlen.

In diesem Abschnitt wird der Algorithmus für statische Graphen erläutert (OPIC). In Abschnitt 4 werden wir sehen, wie man diesen Algorithmus modifizieren kann, dass er auf dynamischen Graphen läuft (Adaptive OPIC).

2.2 OPIC-Algorithmus

Grobe Funktionsweise des Algorithmus

Das Internet wird (theoretisch) unendlich lang durchgescrawlt.

- Wir führen eine Währung für die Wichtigkeit der Knoten ein, den *Cash*: Wenn ein Knoten gecrawlt wird, verteilt er seinen *Cash* an seine Nachbarn. Die Summe des gesamten *Cashes* sei 1, analog zum Aufenthaltswahrscheinlichkeitsmodell.
- Daneben besitzt jeder Knoten eine *History*: Beim Crawlen des Knotens wird der verteilte *Cash* zur *History* hinzuaddiert, d. h. sie speichert den *Cash*, der bisher durch den Knoten hindurch geflossen ist.
- Außerdem speichern wir die Summe der *History* aller Knoten in einer Variablen g , d. h. sie speichert den bisher insgesamt durch alle Knoten geflossenen *Cash*.

Initialisiert werden g und $H[v]$ für alle Knoten v mit 0, da zu Beginn ja noch kein *Cash* geflossen ist. C kann wie bei PageRank mit einer Gleichverteilung initialisiert werden, d. h. wir setzen $C[v]$ auf $\frac{1}{n}$ für alle Knoten v . Es sind – ebenso bei PageRank – auch andere Initialisierungen möglich.

Anmerkung: Während die Summe des *Cashes* gleich bleibt, streben g und die *History*-Werte der einzelnen Knoten gegen unendlich.

Wie werden dabei die *dünnen* Kanten realisiert?

Wir führen eine *virtuelle Seite* ein, zu der alle Seiten einen Link haben und die einen Link zu allen Seiten hat. Das heißt, wir stellen keine direkte Verbindung zwischen allen Seiten her, sondern bündeln diese *dünnen* Kanten und lassen sie über diese *virtuelle Seite* laufen. Dadurch hat jeder Knoten mindestens einen Nachfolger, nämlich diese *virtuelle Seite*. Auch sie muss einmal an die Reihe kommen und verteilt dann ihren *Cash* gleichmäßig an alle Knoten.

Spezialfall siehe 4.1.

OPIC-Algorithmus

Algorithmus 2 : Online Page Importance Computation (OPIC) [1]

Eingabe : Initiale Anzahl Knoten n

Daten : Knotenarray C (*Cash*), Knotenarray H (*History*)
gesamte *History* g ($= |H|$)

foreach $v \in V$ **do**

$C[v] \leftarrow \frac{1}{n}$
 $H[v] \leftarrow 0$

$g \leftarrow 0$

do forever

 choose some node $v \in V$ //jeder Knoten unendlich oft

$H[v] \leftarrow H[v] + C[v]$

$g \leftarrow g + C[v]$

foreach $w \in N^+(v)$ **do**

$C[w] \leftarrow \frac{C[v]}{d^+(v)}$ //u.a. die virtuelle Seite

$C[v] \leftarrow 0$

Berechnung der Wichtigkeit

Zu jedem Zeitpunkt kann (eine Näherung für) die Wichtigkeit einer Seite v berechnet werden durch:

$$\frac{H[v] + C[v]}{g + 1}$$

Da die *virtuelle Seite* keine Wichtigkeit besitzen soll, müsste der Term eigentlich korrekterweise lauten:

$$\frac{H[v] + C[v]}{g + 1 - C[\text{virtual page}]}$$

Wahl der Knoten

Es ist zu beachten, dass im Algorithmus „choose some node v “ steht und nicht „foreach node v “, d. h. es gibt keine strenge Reihenfolge, in der die Knoten ausgewählt werden müssen.

Warum ist es theoretisch egal, welchen Knoten wir wählen?

- Intuitiv: Die Summe von *Cash* und *History* eines Knotens v ändert sich beim Crawlen von v nicht und somit auch nicht der Zähler in der Wichtigkeit des Knotens. Es verändert sich zwar g , dies wirkt sich jedoch auf alle Knoten gleichermaßen aus.
- Eine gewisse *Fairness* muss jedoch gegeben sein: Jeder Knoten muss irgendwann drankommen, da sich sonst der *Cash* bei einem oder mehreren Knoten ansammelt und somit der *Cash-Fluss* versiegt.

Wie man die Knoten geschickt wählen kann, werden wir in Abschnitt 3 sehen.

2.3 Vor- und Nachteile

Nachteile

- Der abgespeicherte Webgraph ist nützlich für andere Dinge (z. B. die Angabe der Backlinks einer Seite).

Vorteile

- Wie bereits erwähnt, müssen wir nicht das gesamte Web crawlen und abspeichern. Dies spart Ressourcen und Zeit.
- Die Berechnung ist näher am Crawling-Prozess und damit aktueller.
- Die Wichtigkeit einer Seite ist schon näherungsweise bekannt, bevor sie gecrawlt wurde, da sie schon *Cash* von ihren Vorgängern angesammelt hat, sofern diese schon gecrawlt wurden.
- Implementationsdetail: Wir speichern C im Arbeitsspeicher und H auf der Festplatte. Dadurch wird nur genau ein Festplattenzugriff benötigt, wenn ein Knoten gecrawlt wird. Solch eine Trennung ist bei PageRank nicht möglich.
- Der Algorithmus erlaubt *Focused Crawling*. Hiermit beschäftigt sich der nächste Abschnitt.

3 Crawling-Strategien

3.1 Fehlerfaktor

Definition

C_t , H_t und g_t bezeichnen die Werte von C , H und g nach Ende des Schrittes t . C_0 , H_0 , g_0 sind somit die Werte der Variablen nach der Initialisierung.

Definition (Fehlerfaktor)

Zu einem Zeitpunkt t bezeichnet

$$\frac{1}{g_t} = \frac{1}{|H_t|} = \frac{1}{\sum_{k \in V} H_t[k]}$$

den Fehlerfaktor. Eigentlich ist dies nicht der Fehlerfaktor, sondern eine obere Schranke dafür und damit eher ein Indikator für die Abweichung von den konvergierten Werten.

Da g_t gegen unendlich wächst für $t \rightarrow \infty$, strebt der Fehlerfaktor gegen 0. Wählt man jedoch ungeschickterweise immer Knoten mit geringem *Cash*, geschieht dies langsamer.

3.2 Beispielstrategien

Cycle:

Die Knoten werden immer wieder in einer festgesetzten Reihenfolge durchlaufen.

Fairness: Alle Knoten kommen genau gleich oft dran. Dies entspricht noch am ehesten dem PageRank-Algorithmus (d. h. der Iteration).

Random:

Der nächste Knoten wird zufällig mit gleicher Wahrscheinlichkeit über alle Knoten gewählt.

Fairness: Alle Knoten sollten gleich oft drankommen.

Greedy:

Als nächster Knoten wird derjenige mit dem höchsten *Cash* gewählt. Dadurch minimiert sich der Fehlerfaktor.

Fairness ist nur gegeben, wenn der Graph stark zusammenhängend ist, weil dann jeder Knoten irgendwann so viel *Cash* angesammelt hat, dass er drankommt. Durch die *virtuelle Seite* wird dies jedoch gewährleistet.

3.3 Experimentelle Ergebnisse

Abbildung 2 zeigt den Einfluss von Crawling-Strategien auf die Konvergenz des Algorithmus. Als Fehlermaß wurde hier die durchschnittliche Abweichung der Wichtigkeit der Knoten von den konvergierten Werten des offline-Algorithmus (PageRank) gewählt.

Angewandt wurden die Algorithmen auf Zufallsgraphen mit $N = 100\,000$ Knoten. Um die Werte von PageRank mit denen von OPIC vergleichen zu können, wurden N gecrawlte Seiten wie eine Iteration gezählt.

Wie man den Grafiken entnehmen kann, konvergieren die Werte von wichtigen Seiten – vor allem bei Greedy – besonders schnell. Dies ist eine nützliche Eigenschaft, da wichtige Seiten öfter aktualisiert werden und (theoretisch) mehr User darauf zugreifen.

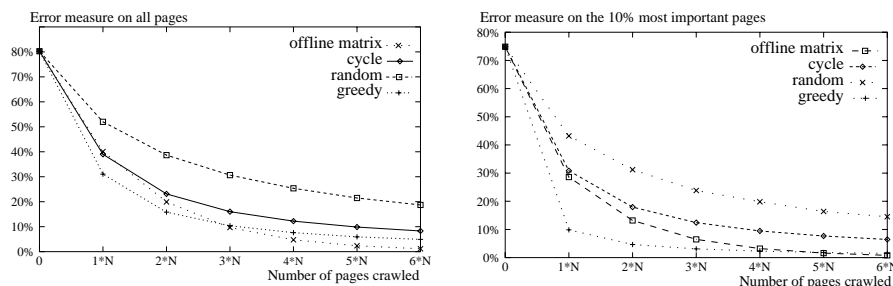


Abbildung 2: Einfluss der Crawling-Strategie auf die Konvergenz [1]

4 Adaptive OPIC

4.1 Änderung im Graphen

Das Internet ändert sich ständig, Seiten und Links entstehen und verschwinden.

Beim Vergrößern des Graphen ergibt sich die (Zitat:) „*fast schon philosophische Frage*“: Wie viel *Cash* und *History* bekommt ein neuer Knoten?

Lösung

Ein neuer Knoten bekommt...

- *Cash* von der *virtuellen Seite*, damit der gesamte *Cash* 1 bleibt,
- eine *default history*, die aus kürzlich eingefügten Seiten ermittelt wird.

Das bleibende Problem mit der *History* ist jedoch, dass die Vergangenheit eines Knotens eine große Rolle spielt. War ein Knoten beispielsweise lange sehr zentral, ist es aber plötzlich nicht mehr (z. B. durch Verschwinden von eingehenden Kanten), so bleibt seine *History* jedoch trotzdem sehr hoch, d. h. seine Wichtigkeit bleibt ihm lange erhalten. Ebenso kann ein neu eingefügter Knoten, der womöglich sehr zentral ist, fälschlicherweise eine zu niedrige *default history* erhalten. Es dauert sehr lange, bis er Knoten, die gleich zentral aber schon länger im Graph enthalten sind, einholt.

Aus diesen Gründen wird die *History* immer über ein bestimmtes Zeitfenster gemittelt. Dadurch wirkt sich nur der in diesem Zeitraum gesammelte *Cash* auf die Wichtigkeit des Knotens aus.

Verkleinern des Graphen (Entfernen eines Knotens) wird im Paper nicht erwähnt. Vermutlich wird der *Cash* des entfernten Knotens der *virtuellen Seite* hinzugefügt, damit dessen Summe 1 bleibt.

4.2 Window policies

Da g , die Summe aller *Historys*, monoton wächst, können wir uns g als Uhr vorstellen. Im Folgenden wird unter einem Zeitpunkt ein Wert von g verstanden.

Definition (Messung)

Ein Paar $(C[v], g)$ nennen wir eine *Messung*, wobei $C[v]$ der *Cash* ist, der beim *Crawlen* des Knotens v zum Zeitpunkt g verteilt und auf seine *History* addiert wird.

Anmerkung: Statt $(C[v], g)$ könnten wir auch $(H[v], g)$ abspeichern, da sich $C[v]$ aus $H[v]$ als Differenz zur letzten Messung berechnen lässt.

Window policies

Fixed Window (mit Zeitraum T):

Für jede Seite speichern wir eine Messung für jedes *Crawlen* der Seite, das höchstens T zurückliegt.

Die Anzahl Speicherungen variiert zwischen den Seiten.

Variable Window (der Größe k):

Wir speichern für jede Seite die letzten k Messungen.
Der Zeitpunkt der letzten Messung variiert zwischen den Seiten.

Interpolation (über Zeitfenster T):

Wir speichern den Zeitpunkt (also den Wert von g) des letzten Crawls und eine interpolierte *History*, die den im zurückliegenden Zeitfenster T gesammelten *Cash* abschätzt. Zur Verdeutlichung siehe Abbildung 3.

Für jede Seite muss nur eine Messung gespeichert werden.



Abbildung 3: Interpolation

4.3 Experimentelle Ergebnisse

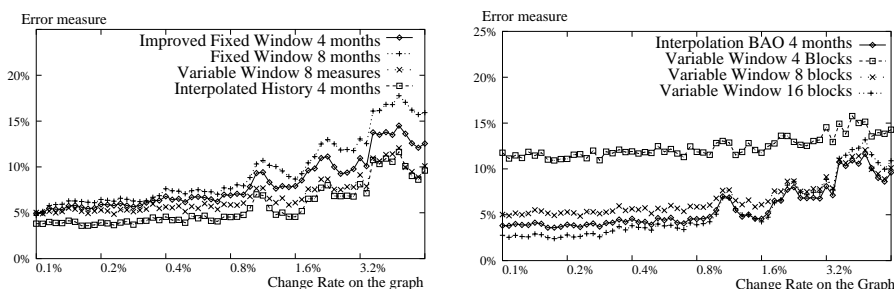


Abbildung 4: Einfluss der Fensterarten und -größen [1]

Abbildung 4 zeigt den Einfluss der Wahl der *window policy* und der Fenstergröße auf die Fehlerrate.

Als Fehler-Maß wurde wieder dasselbe wie in 3.3 verwendet. Die Änderungsrate bezeichnet hier den Anteil an den N Seiten, deren Eingangsgrad sich in den letzten N Crawling-Schritten mindestens halbiert oder verdoppelt hat.

Es ist zu erkennen, dass die Qualität der Ergebnisse der Interpolations-Methode besser als die von *Fixed Window* mit $T = 8$ Monaten und *Variable*

Window mit $k = 8$ ist. Wenn man sie mit *Variable Window* vergleicht, liegen die Werte zwischen $k = 8$ und $k = 16$, d. h. obwohl bei der Interpolations-Methode immer nur 1 Messung gespeichert wird, liegen die Ergebnisse zwischen den *Variable Window*-Ergebnissen mit 8 und 16 gespeicherten Messungen.

5 Diskussion

5.1 Unterschiede im Modell zu PageRank

Bei OPIC sind die Kanten zur *virtuellen Seite* nicht alle gleich „dünn“, sondern hängen vom Grad des Knotens ab. Dadurch ist das Modell näher am Irrfahrt-Modell, da der virtuelle Surfer auf jeden Fall springt, wenn es keine ausgehenden Links auf der Seite gibt.

Wir wollen hier untersuchen, welcher Art und Größe dieser Unterschied ist.

Es lässt sich beweisen, dass nach jedem Schritt t für jede Seite v gilt:

$$H_t[v] + C_t[v] = C_0[v] + \sum_{u \in N^-(v)} \frac{\#(u, v)}{d^+(u)} \cdot H_t[u]$$

Daraus erhält man:

$$\frac{H_t[v] + C_t[v]}{g + 1} = \frac{C_0[v]}{g + 1} + \sum_{u \in N^-(v)} \frac{\#(u, v)}{d^+(u)} \cdot \frac{H_t[u]}{g + 1}$$

Der Term $\frac{C_0[v]}{g+1}$ kann vernachlässigt werden, da für $t \rightarrow \infty$ alle *History*-Werte gegen unendlich streben, die *Cash*-Werte jedoch immer im Intervall $[0, 1]$ liegen und somit nicht mehr ins Gewicht fallen. Aus dem selben Grund kann $\frac{H_t[u]}{g+1}$ als die Wichtigkeit des Knotens u aufgefasst werden.

Somit erfüllen die Wichtigkeiten der Knoten für $t \rightarrow \infty$ folgende Gleichung:

$$c[v] = \sum_{u \in N^-(v)} \frac{\#(u, v)}{d^+(u)} \cdot c[u]$$

Zu beachten ist hier wieder, dass die *virtuelle Seite* Nachfolger jeder Seite ist. Wenn man diese explizit schreibt und aus der Summe heraus nimmt, sieht die Gleichung folgendermaßen aus:

$$c[v] = \frac{1}{n} \cdot c[virt] + \sum_{u \in N^-(v)} \frac{\#(u, v)}{d^+(u) + 1} \cdot c[u]$$

Bei PageRank dagegen wird nach einer ausreichenden Anzahl von Iterationen für jeden Knoten v folgende Gleichung erfüllt:

$$c_P[v] = \frac{\omega}{n} + (1 - \omega) \cdot \sum_{u \in N^-(v)} \frac{\#(u, v)}{d^+(u)} \cdot c_P[u]$$

Das ω in PageRank entspricht in OPIC der (variablen) Wichtigkeit der *virtuellen Seite*. Der Unterschied zwischen den beiden Maßen besteht hauptsächlich darin, dass die Abschwächung der Knotenzentralitäten nicht durch einen konstanten

Faktor geschieht, sondern wie oben erwähnt vom Ausgangsgrad des jeweiligen Knotens abhängt.

Die Wichtigkeit der virtuellen Seite berechnet sich in dieser expliziten Schreibweise durch:

$$c[virt] = \sum_{u \in V} \frac{1}{d^+(u) + 1} \cdot c[u]$$

Das heißt, sie hängt von den Knotengraden, also von der Dichte des Graphen ab.

Da die *virtuelle Seite* jedoch keine Wichtigkeit besitzen soll, müssen die Werte der Knoten nach der Berechnung noch durch $1 - c[virt]$ geteilt werden.

Beispielgraph

In Tabelle 1 sind die PageRank- und OPIC-Werte für den Graphen in Abbildung 5 zu sehen. Knoten *b* hat für $\omega = 0,2$ einen höheren PageRank-Wert als Knoten *c*. Bei den OPIC-Werten verhält sich dies jedoch anders herum, d. h. die Reihenfolge dieser Knoten stimmt nicht überein. Ebenso vertauscht sich für $\omega \geq 0,64$ die Ordnung der PageRank-Werte der beiden Knoten, wodurch die Reihenfolge der Knoten für beide Maße wieder identisch ist (Tab. 1 zeigt die PageRank-Werte für $\omega = 0,8$).

Dies ist ein anschauliches Beispiel dafür, dass die unterschiedlichen Abschwächungsmethoden der beiden Zentralitäten sehr wohl eine Auswirkung auf das Ranking der Knoten haben können. Sogar die Wahl von ω bei PageRank kann dies beeinflussen.

Knoten	PageRank mit $\omega = 0,2$	OPIC	PageRank mit $\omega = 0,8$
a	27,1	25,4	22,1
b	25,7	20,7	20,4
c	21,2	23,3	21,0
d	15,2	17,5	19,1
e	10,8	13,1	17,4

Tabelle 1: PageRank- und OPIC-Werte (in %) für Graph in Abb. 5.

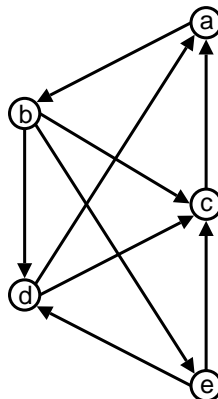


Abbildung 5: Vergleichsgraph für PageRank- und OPIC-Zentralität.

5.2 Kritische Sicht auf die Vorteile von OPIC

Bei PageRank können bei einer Änderung des Graphen auch die alten Werte als Initialisierung verwendet werden. Dadurch konvergieren die Werte schneller, da sie – vorausgesetzt, die Änderung des Graphen und somit der PageRank-Werte ist nicht allzu groß – schon näher an den konvergierten Werten liegen.

Zugegebener Maßen löst dies allerdings nicht das Problem, dass das komplette Web als Graph abgespeichert werden muss.

Literatur

- [1] Serge Abiteboul, Mihai Preda, Gregory Cobena: Adaptive On-Line Page Importance Computation. *Proc. WWW* 2003.
- [2] Sergey Brin, Lawrence Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: *Computer Networks and ISDN Systems* 1998.
- [3] <http://www.inf.uni-konstanz.de/algo/lehre/ss05/mna/skript/zentralitaeten.pdf>, Stand 25.01.2006