Datenstrukturen, Algorithmen und Programmierung 2 (DAP2)

Graphen

Definition(gerichteter Graph)

- Ein *gerichteter Graph* ist ein Paar (V, E), wobei V eine endliche Menge ist und $E \subseteq V \times V$.
- V heißt Knotenmenge des Graphen
- Die Elemente aus V sind die Knoten des Graphen
- E heißt Kantenmenge des Graphen
- Die Elemente aus E sind die Kanten des Graphen

Graphen

Definition(ungerichteter Graph)

- Ein *ungerichteter Graph* ist ein Paar (V, E), wobei V eine endliche Menge ist und $E \subseteq V \times V$.
- V heißt Knotenmenge des Graphen
- Die Elemente aus V sind die Knoten des Graphen
- E heißt Kantenmenge des Graphen
- Die Elemente aus E sind die Kanten des Graphen
- Wir stellen Kanten aus V wie im gerichteten Fall durch (u, v) dar und nehmen an, dass die Kante (u, v) gleich der Kante (v, u) ist
- Manchmal repräsentieren wir einen ungerichteten Graph durch einen gerichteten, indem wir jede Kante (u, v) durch die gerichteten Kanten (u, v) und (v, u) ersetzen

Graphen

Definition(Inzidenz und Adjazenz)

- Ist (u, v) eine Kante in einem gerichteten Graph, so sagen wir,
- (u, v) ist inzident von u (oder verlässt Knoten u)
- (u, v) ist inzident zu v (oder zeigt auf Knoten v)

- Ist (u, v) eine Kante in einem ungerichteten Graph, so sagen wir,
- (u, v) ist inzident an u und v (oder (u, v) liegt an u und v an)

- Ist (u, v) eine Kante in einem Graph, so sagen wir u ist adjazent zu v

Graphen

Definition(Knotengrad)

- Der Ausgangsgrad eines Knotens in einem gerichteten Graph ist die Anzahl Kanten, die den Knoten verlassen
- Der Eingangsgrad eines Knotens in einem gerichteten Graph ist die Anzahl Kanten, die auf den Knoten zeigen
- Der Grad eines Knotens in einem ungerichteten Graph ist die Anzahl Kanten die an dem Knoten anliegen

Graphen

Definition(Pfad)

- Ein Pfad (oder Weg) der Länge k von Knoten u zu Knoten v in einem Graph $G = (V, E)$ ist eine Sequenz von $k + 1$ Knoten $(v_0, \dots, v_k)$ mit $u = v_0$ und $v = v_k$ und $(v_i, v_{i+1}) \in E$ für $i = 1, \dots, k$.
- Wir sagen, dass v von u erreichbar ist, wenn es einen Pfad von v nach u gibt
- Ein Pfad heißt einfach, wenn kein Knoten auf dem Pfad mehrfach vorkommt

Graphen

Definition(Zyklus)

- Ein Pfad $(v_0, \ldots, v_k)$ in einem gerichteten Graph mit $v_0 = v_k$ und mindestens einer Kante heißt Zyklus
- Ein gerichteter Graph, der keinen Zyklus enthält, heißt azyklisch

Graphen

Definition(Kreis)

- Ein Pfad $(v_0, \ldots, v_k)$ in einem ungerichteten Graph heißt Kreis, falls $v_0 = v_k$ und $(v_0, \ldots, v_{k-1})$ ein einfacher Pfad ist
- Ein kreisfreier Graph heißt Wald

Graphen

Definition(Zusammenhang)

- Ein gerichteter Graph heißt stark zusammenhängend, wenn es von jedem Knoten einen Pfad zu jedem anderen Knoten im Graph gibt
- Ein ungerichteter Graph heißt zusammenhängend, wenn es von jedem Knoten einen Pfad zu jedem anderen Knoten im Graph gibt
- Die starken Zusammenhangskomponenten eines Graphen sind die Äquivalenzklassen der Relation „ist beidseitig erreichbar“
- Die Zusammenhangskomponenten eines Graphen sind die Äquivalenzklassen der Relation „ist erreichbar“

Graphen

Definition(Baum)

- Ein ungerichteter, zusammenhängender, kreisfreier Graph heißt Baum

Graphalgorithmen

Darstellung von Graphen

- Adjazenzlisten (dünne Graphen, $|E| \ll |V|^2$)
- Adjazenzmatrix (dichte Graphen, $|E|$ nah an $|V|^2$)

Arten von Graphen

- Ungerichtet, gerichtet
- Ungewichtet, gewichtet

Graphalgorithmen

Adjazenzmatrixdarstellung

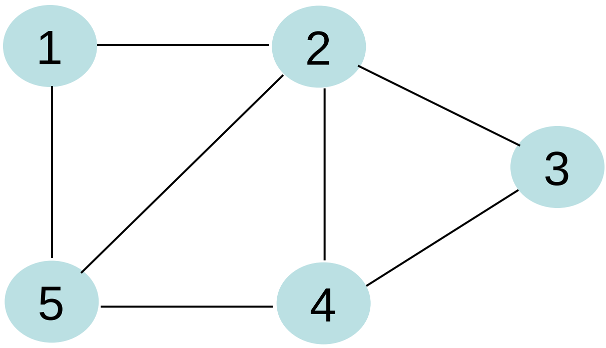
- Knoten sind nummeriert von 1 bis $|V|$
- $|V| \times |V|$ Matrix $A = (a_{ij})$ mit

$$a_{ij} = \begin{cases} 1 & \text{, falls } (i,j) \in E \\ 0 & \text{, sonst} \end{cases}$$

- Bei ungerichteten Graphen gilt $A = A^T$

Graphalgorithmen

Beispiel



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Graphalgorithmen

Adjazenzlistendarstellung

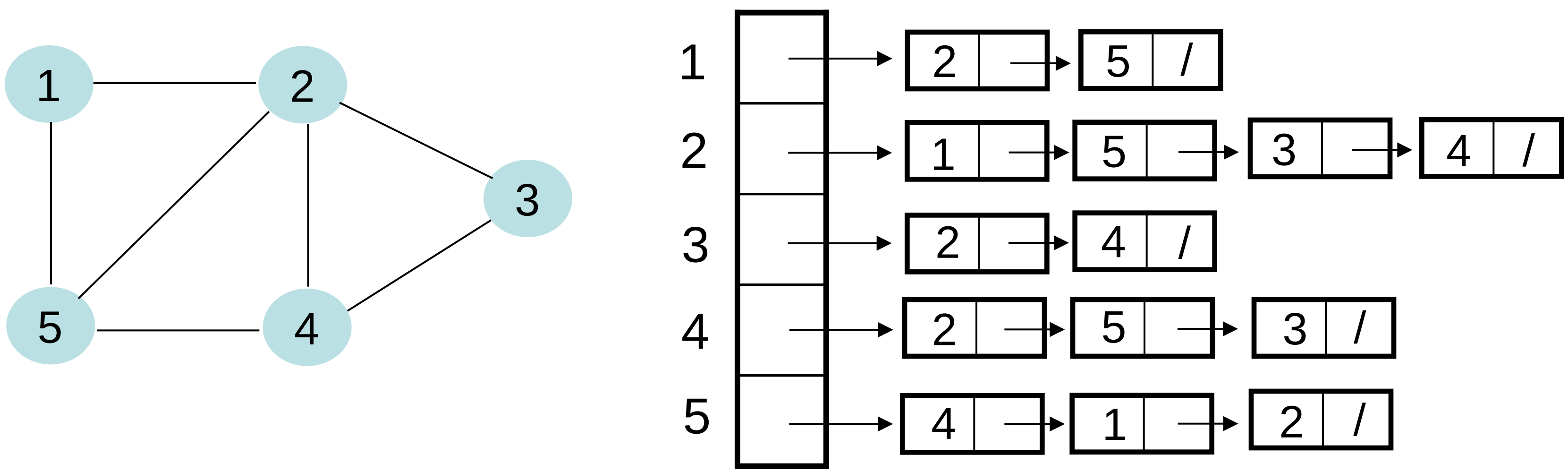
- Feld Adj mit $|V|$ Listen (eine pro Knoten)
- Für Knoten v enthält $\mathrm{Adj}[v]$ eine Liste aller Knoten u mit $(v,u) \in E$
- Die Knoten in $\mathrm{Adj}[u]$ heißen zu u adjazent
- Ist G ungerichtet, so gilt: $v \in \mathrm{Adj}[u] \Leftrightarrow u \in \mathrm{Adj}[v]$

Gewichtete Graphen

- Kanten haben Gewicht gegeben durch Funktion $w: E \rightarrow \mathbb{R}$
- Gewicht $w(u,v)$ von Kante (u,v) wird mit Knoten v in u‘s Adjazenzliste gespeichert

Graphalgorithmen

Beispiel



Big Data

Suchmaschinen

- Grundwerkzeug, um auf Inhalte im WWW zuzugreifen
- Es werden fast nur die ersten paar Ergebnisse angeklickt

Herausforderung

- Wie kann man automatisiert die Qualität von Webseiten bewerten?

Big Data

„Prä-Google“ Zeit

- Charakterisierung über Wörter, die häufig auf einer Webseite auftreten, aber insgesamt eher selten sind
- Extrem einfach zu manipulieren

Grundidee von Pagerank

- Betrachte Links als „Stimmen“
- Ein eingehender Link ist eine Stimme für den Knoten
- Hoffnung: Eine Seite mit vielen eingehenden Links ist wichtiger als eine Seite mit wenigen eingehenden Links
- Eine Stimme einer qualitativ guten Seite zählt mehr als eine Stimme einer schlechten Seite

⇒ Rekursive Formulierung

Big Data

Vereinfachter Pagerank

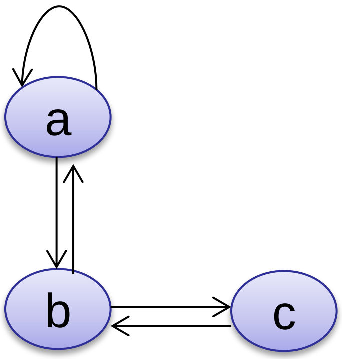
- Modelliere das Web als gerichteten Graph
- Knoten entsprechen Webseiten
- Gerichtete Kanten entsprechen Links
- Jeder Link hat ein Gewicht proportional zur Wichtigkeit des Ausgangsknotens
- Wenn Knoten v mit Wichtigkeit $r(v)$ genau n ausgehende Kanten(Links) besitzt, dann erhält jeder Link $r(v)/n$ Stimmen
- Die Wichtigkeit von Knoten v ist die Summe der Stimmen auf den eingehenden Kanten

Big Data

Vereinfachter Pagerank - Flussformulierung

- Gerichteter Graph $G = (V, E)$
- Sei $d(v)$ der Ausgangsgrad von Knoten v
- Definiere „Rang“ von Knoten v als

$$r(v) = \sum_{(u,v)\in E} \frac{r(u)}{d(u)}$$



$$\begin{aligned} r(a) &= ½\, r(a) + ½\, r(b) \\ r(b) &= ½\, r(a) + r(c) \\ r(c) &= ½\, r(b) \end{aligned}$$

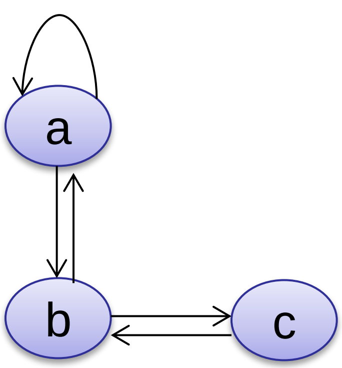
Big Data

Vereinfachter Pagerank - Flussformulierung

- Gerichteter Graph $G = (V, E)$
- Sei $d(v)$ der Ausgangsgrad von Kn
- Definiere „Rang“ von Knoten v als

$r(v)$

Gleichungssystem hat keine eindeutige Lösung



$$r(a) = ½\, r(a) + ½\, r(b)$$
$$r(b) = ½\, r(a) + r(c)$$
$$r(c) = ½\, r(b)$$

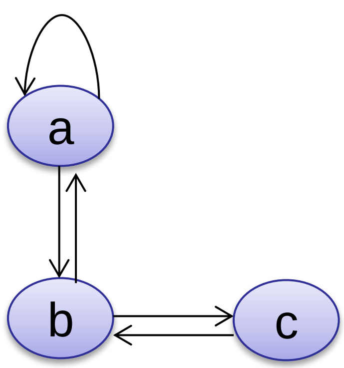
Big Data

Vereinfachter Pagerank - Flussformulierung

- Gerichteter Graph $G = (V, E)$
- Sei $d(v)$ der Ausgangsgrad von Kn
- Definiere „Rang“ von Knoten v als

$r(v)$

Gleichungssystem hat eindeutige Lösung



$r(a) = ½\, r(a) + ½\, r(b)$
$r(b) = ½\, r(a) + r(c)$
$r(c) = ½\, r(b)$
$r(a) + r(b) + r(c) = 1$ (Normierung)

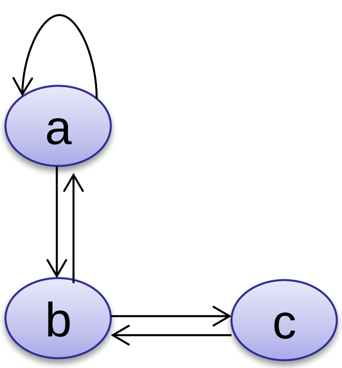
Big Data

Vereinfachter Pagerank - Flussformulierung

- Gerichteter Graph $G = (V, E)$
- Sei $d(v)$ der Ausgangsgrad von Kn
- Definiere „Rang“ von Knoten v als

$r(v)$

$r(a) = 2/5$
$r(b) = 2/5$
$r(c) = 1/5$



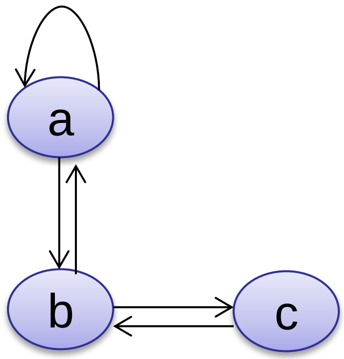
$r(a) = ½\, r(a) + ½\, r(b)$
$r(b) = ½\, r(a) + r(c)$
$r(c) = ½\, r(b)$
$r(a) + r(b) + r(c) = 1$ (Normierung)

Big Data

Vereinfachter Pagerank - Matrixschreibweise

- Gerichteter Graph $G = (V, E)$
- Sei $d(v)$ der Ausgangsgrad von Knoten v
- Definiere „Rang“ von Knoten v als

$$r(v) = \sum_{(u,v)\in E} \frac{r(u)}{d(u)}$$



$$M = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{pmatrix}$$

Big Data

Vereinfachter Pagerank - Matrixschreibweise

- Gerichteter Graph $G = (V, E)$
- Sei $d(v)$ der Ausgangsgrad von Knoten v
- Definiere „Rang“ von Knoten v als

$$r(v) = \sum_{(u,v) \in E} \frac{r(u)}{d(u)}$$

$$r = M \cdot r$$

- r ist also Eigenvektor von M mit zugehörigem Eigenwert 1
  (Ein Vektor $r \neq 0$ heißt Eigenvektor einer quadratischen Matrix M mit zugehörigem Eigenwert λ, wenn $M \cdot r = \lambda \cdot r$ gilt)

Big Data

Lösung des linearen Gleichungssystems

Gauss Elimination (schlechte Laufzeit)

Simulation des Abstimmprozesses

- Wie genau?
- Warum bzw. unter welchen Voraussetzungen terminiert der Prozess?

Big Data

PowerIteration(M, ε)

1. $r^{(0)} = (1/n, \dots, 1/n\,)^T$
2. **repeat**
3. $\quad r^{i+1} = M \cdot r^{(i)}$
4. **until** $\left\| r^{(i+1)} - r^{(i)} \right\|_1 < \varepsilon$
5. **return** $r^{(i+1)}$

Definition (l_1 –Norm)

$$\|r - t\|_1 = \sum |r(i) - t(i)|$$

Big Data

Satz 42 (Konvergenz von PowerIteration)

Sei M eine Matrix mit Eigenwerten $\lambda_1, \ldots, \lambda_n$ und $1 = \lambda_1 > |\lambda_2| > \cdots > |\lambda_n|$ und zugehörigen linear unabhängigen Eigenvektoren $v_1, \ldots, v_n$ der Länge 1. Falls $r^{(0)} = \sum c_i \cdot v_i$ mit $c_1 \neq 0$ gilt, so gilt $\lim\limits_{k\to\infty} r^{(k)} / \|r^{(k)}\| = v_1$

Beweisskizze

- Es gilt $r^{(1)} = Mr^{(0)} = M \cdot (\sum c_i \cdot v_i) = \sum c_i M v_i = \sum c_i \lambda_i v_i$
- Es gilt $r^{(k)} = \sum c_i (\lambda_i)^k v_i = c_1 (\lambda_1)^k v_1 + \sum_{i>1} c_i (\lambda_i)^k v_i$
- Für $i > 1$ geht λ_i^k gegen 0 (für k gegen ∞)
- Also geht $r^{(k)}$ gegen $c_1 \cdot v_1$
- Normieren ergibt den Satz

Wir werden dies aber nicht vollständig beweisen können.

Big Data

Satz 42 (Konvergenz von PowerIteration)

Sei M eine Matrix mit Eigenwerten $\lambda_1, \dots, \lambda_n$ und $1 = \lambda_1 > |\lambda_2| > \cdots > |\lambda_n|$ und zugehörigen linear unabhängigen Eigenvektoren $v_1, \dots, v_n$ der Länge 1. Falls $r^{(0)} = \sum c_i \cdot v_i$ mit $c_1 \neq 0$ gilt, so gilt $\lim\limits_{k\to\infty} r^{(k)} / \|r^{(k)}\| = v_1$

Beweisskizze

- Es gilt $r^{(1)} = Mr^{(0)} = M \cdot (\sum c_i \cdot v_i) = \sum c_i M v_i = \sum c_i \lambda_i v_i$
- Es gilt $r^{(k)} = \sum c_i (\lambda_i)^k v_i = c_1 (\lambda_1)^k v_1 + \sum_{i>1} c_i (\lambda_i)^k v_i$
- Für $i > 1$ geht λ_i^k gegen 0 (für k gegen ∞)
- Also geht $r^{(k)}$ gegen $c_1 \cdot v_1$
- Normieren ergibt den Satz

Der allgemeine Pagerank Algorithmus stellt sicher, dass diese Eigenschaften erfüllt sind

Big Data

Satz 42 (Konvergenz von PowerIteration)

Sei M eine Matrix mit Eigenwerten $\lambda_1, \ldots, \lambda_n$ und $1 = \lambda_1 > |\lambda_2| > \cdots > |\lambda_n|$ und zugehörigen linear unabhängigen Eigenvektoren $v_1, \ldots, v_n$ der Länge 1. Falls $r^{(0)} = \sum c_i \cdot v_i$ mit $c_1 \neq 0$ gilt, so gilt $\lim_{k\to\infty} r^{(k)} / \|r^{(k)}\| = v_1$

Beweisskizze

- Es gilt $r^{(1)} = Mr^{(0)} = M \cdot (\sum c_i \cdot v_i) = \sum c_i M v_i = \sum c_i \lambda_i v_i$
- Es gilt $r^{(k)} = \sum c_i (\lambda_i)^k v_i = c_1 (\lambda_1)^k v_1 + \sum_{i>1} c_i (\lambda_i)^k v_i$
- Für $i > 1$ geht λ_i^k gegen 0 (für k gegen ∞)
- Also geht $r^{(k)}$ gegen $c_1 \cdot v_1$
- Normieren ergibt den Satz

Big Data

Satz 42 (Konvergenz von PowerIteration)

Sei M eine Matrix mit Eigenwerten $\lambda_1, \dots, \lambda_n$ und $1 = \lambda_1 > |\lambda_2| > \cdots > |\lambda_n|$ und zugehörigen linear unabhängigen Eigenvektoren $v_1, \dots, v_n$ der Länge 1. Falls $r^{(0)} = \sum c_i \cdot v_i$ mit $c_1 \neq 0$ gilt, so gilt $\lim_{k\to\infty} r^{(k)} / \left\| r^{(k)} \right\| = v_1$

Beweisskizze

- Es gilt $r^{(1)} = M r^{(0)} = M \cdot (\sum c_i \cdot v_i) = \sum c_i M v_i = \sum c_i \lambda_i v_i$
- Es gilt $r^{(k)} = \sum c_i (\lambda_i)^k v_i = c_1 (\lambda_1)^k v_1 + \sum_{i>1} c_i (\lambda_i)^k v_i$
- Für $i > 1$ geht λ_i^k gegen 0 (für k gegen ∞)
- Also geht $r^{(k)}$ gegen $c_1 \cdot v_1$
- Normieren ergibt den Satz

Big Data

Das Random Surfer Modell (Random Walk)

- Der Surfer startet o.b.d.A. zu Zeitpunkt 0 bei Knoten 1
- Zum Zeitpunkt t steht der Surfer auf Knoten v
- Zum Zeitpunkt $t+1$ wählt er eine von v ausgehenden Kante zufällig und gleichverteilt und folgt dieser zu Nachbarknoten u
- Wiederhole den Prozess unendlich lange

Der Verteilungsvektor

- Sei $p(t)$ ein Vektor, dessen Eintrag für Knoten v die Wahrscheinlichkeit angibt, dass der Surfer zum Zeitpunkt t an Knoten v steht
- $p(t)$ ist eine Wahrscheinlichkeitsverteilung über den Knoten

Big Data

Wie entwickelt sich die Verteilung p(t)?

Es gilt $p(t+1) = M \cdot p(t)$

Definition

Eine Verteilung die $p(t+1) = M \cdot p(t) = p(t)$ erfüllt, heißt *stationäre Verteilung* des Random Walks.

Beobachtung

Da dann $M \cdot p(t) = p(t)$ ist, ist $p(t)$ ein Eigenvektor zum Eigenwert 1, d.h. $p(t)$ entspricht unserem Rangvektor r

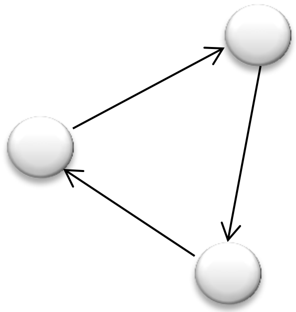
Big Data

Satz 43 (Konvergenz von Random Walks)

Ist ein Graph stark zusammenhängend und aperiodisch, so ist die stationäre Verteilung eindeutig und ein Random Walk konvergiert gegen diese Verteilung (unabhängig von der Ausgangsverteilung zu Zeitpunkt 0).

Definition

Für Knoten v sei $N(v)$ die Menge aller j, so dass ein Weg von v nach v der Länge j existiert. Ein Graph heißt periodisch, wenn es einen Knoten v gibt, so dass der ggT von $N(v)$ ungleich 1 ist. Ansonsten heißt G aperiodisch.



Beispiel: Ein periodischer Graph

Big Data

Zwischenfazit

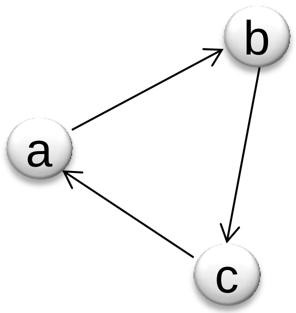
- Der Rangvektor, den der vereinfachte Pagerank Algorithmus berechnet, kann als iterierter Wahlvorgang interpretiert werden
- Gleichzeitig entspricht der Rangvektor einer stationären Verteilung eines Random Walks (Random Surfer Modell)
- Mit Hilfe des PowerIteration Algorithmus kann man den Rangvektor bis auf einen kleinen Fehler berechnen

Probleme

Wir wissen nur, dass Random Walks für stark zusammenhängende, aperiodische Graphen konvergieren.

Big Data

Probleme (periodische Graphen)



$$r(v)^{(t+1)} = \sum_{(u,v)\in E} \frac{r^{(t)}(u)}{d(u)}$$

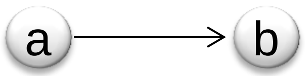
time	1	2	3	4	5	6	7	8
$r(a)$	1	0	0	1	0	0	1	0
$r(b)$	0	1	0	0	1	0	0	1
$e(c)$	0	0	1	0	0	1	0	0

Big Data

$$r(v)^{(t+1)} = \sum_{(u,v)\in E} \frac{r^{(t)}(u)}{d(u)}$$

Probleme (Sackgassen)

- Knoten ohne ausgehende Kanten
- (G nicht stark zusammenhängend)



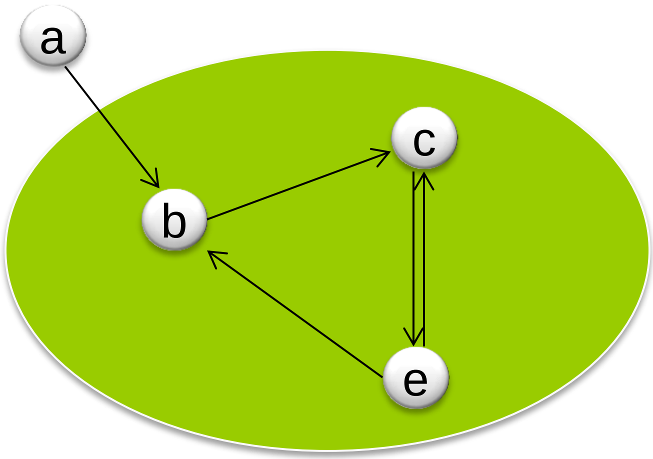
time	1	2	3	4
$r(a)$	1	0	0	0
$r(b)$	0	1	0	0

Big Data

$$r(v)^{(t+1)} = \sum_{(u,v)\in E} \frac{r^{(t)}(u)}{d(u)}$$

Probleme (Spinnenfallen)

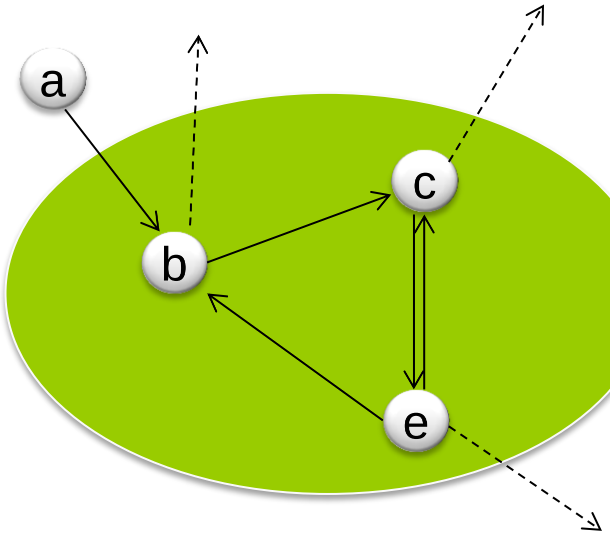
- Teilgraph ohne ausgehende Kanten
- (G nicht stark zusammenhängend)
- Irgendwann sammeln diese Knoten die gesamten „Stimmen“



Big Data

Lösung für Spinnenfallen

- Mit Wahrscheinlichkeit β folge zufälliger Kante
- Mit Wahrscheinlichkeit $1 - \beta$ springe zu zufälligem Knoten
- Typische Werte von β liegen bei 0.8 bis 0.9



Big Data

Lösung für Sackgassen

- Springe zu einem zufälligen Knoten

Auswirkung der Modifikationen

- Man kann die Modifikationen als Hinzufügen neuer Kanten betrachten
- Dadurch wird der Graph stark zusammenhängend und aperiodisch
- Allerdings werden nicht alle Kanten mit derselben Wahrscheinlichkeit benutzt
- Satz über Konvergenz von Random Walks gilt immer noch

Big Data

Der allgemeine Pagerank [Brin, Page, 98]

$$r(v) = \sum_{(u,v)\in E} \beta \cdot \frac{r(u)}{d(u)} + (1-\beta) \cdot \frac{1}{n}$$

- Annahme: Keine Sackgassen
- Bei Sackgassen werden Kanten zu allen anderen Knoten angenommen

Big Data

Der allgemeine Pagerank [Brin, Page, 98]

$$r(v) = \sum_{(u,v)\in E} \beta \cdot \frac{r(u)}{d(u)} + (1-\beta) \cdot \frac{1}{n}$$

Die Google Matrix

$$A = \beta M + (1-\beta) \begin{pmatrix} 1/n & \cdots & 1/n \\ \vdots & \ddots & \vdots \\ 1/n & \cdots & 1/n \end{pmatrix}$$

Der Pagerankvektor ergibt sich durch Lösen von $r = Ar$ => Poweriteration

Big Data

Implementierung von PowerIteration

- Hauptschritt: Berechnung von Ar
- Herausforderung: A ist quadratisch in der Anzahl der Webseiten und alle Einträge sind ungleich 0 !!!!

Idee

- Ausnutzen, dass M dünn besetzt ist

$$Ar = \beta M r + (1-\beta)\begin{pmatrix} 1/n & \cdots & 1/n \\ \vdots & \ddots & \vdots \\ 1/n & \cdots & 1/n \end{pmatrix} r = \beta M r + (1-\beta)\begin{pmatrix} 1/n \\ \vdots \\ 1/n \end{pmatrix}$$

- Man muss also nur Mr berechnen und M hat sehr viele Einträge gleich 0

Big Data

Sackgassen

- Gibt es Sackgassen, so geht die gesamte Wahrscheinlichkeit dieser Knoten in jeder Iteration verloren
- Wir wollten uns aber von jeder Sackgasse gleichverteilt zu einem Knoten bewegen
- Also wird die „verlorene Wahrscheinlichkeit“ einfach gleichmäßig auf alle Knoten verteilt

Big Data

AllgemeinerPagerank(G, β, ε)

for each $v \in V$ **do**
    $\text{new}(v) \leftarrow 1/n$
repeat
    for each $v \in V$ **do**
        $\text{old}(v) \leftarrow \text{new}(v)$
        $\text{new‘}(v) \leftarrow \sum_{(u,v)\in E} \beta\, \text{new}(u)/d(u)$
        if Eingangsgrad von v ist 0 **then** $\text{new‘}(v) = 0$
    Berechne $S \leftarrow \sum \text{new‘}(v)$
    for each $v \in V$ **do**
        $\text{new}(v) \leftarrow \text{new‘}(v) + (1 - S)/n$
until $\|\text{old} - \text{new}\| < \varepsilon$
return new

Big Data

AllgemeinerPagerank(G, β, ε)

1. **for each** $v \in V$ **do**
2. new(v) $\leftarrow 1/n$
3. **repeat**
4. **for each** $v \in V$ **do**
5. old(v) $\leftarrow$ new(v)
6. new‘(v) $\leftarrow \sum_{(u,v) \in E} \beta$ new(u)$/d(u)$
7. **if** Eingangsgrad von v ist 0 **then** new‘(v) $= 0$
8. Berechne $S \leftarrow \sum$ new‘(v)
9. **for each** $v \in V$ **do**
10. new(v) $\leftarrow$ new‘(v) $+ (1 - S)/n$
11. **until** ||old − new|| $< \varepsilon$
12. **return** new

Speichere G durch „umgedrehte“ Adjazenzlisten

Big Data

AllgemeinerPagerank(G, β, ε)

1. **for each** $v \in V$ **do**
2. new$(v) \leftarrow 1/n$
3. **repeat**
4. **for each** $v \in V$ **do**
5. old$(v) \leftarrow$ new(v)
6. new'$(v) \leftarrow \sum_{(u,v)\in E} \beta$ new$(u)/d(u)$
7. **if** Eingangsgrad von v ist 0 **then** new'$(v) = 0$
8. Berechne $S \leftarrow \sum$ new'(v)
9. **for each** $v \in V$ **do**
10. new$(v) \leftarrow$ new'$(v) + (1 - S)/n$
11. **until** $\|\text{old} - \text{new}\| < \varepsilon$
12. **return** new

Laufzeit pro Iteration:

$$\mathbf{O}(V) + \mathbf{O}\left(\sum \text{indeg}(v)\right) = \mathbf{O}(|V| + |E|)$$

Zusammenfassung

Pagerank

- Idee: Iterierte Qualitätsbewertung, die die Linkstruktur ausnutzt
- Probleme durch Sackgassen und Spinnenfallen können mit Hilfe von „Teleportation“ gelöst werden
- Implementierung mit Hilfe von PowerIteration benötigt Laufzeit $\mathbf{O}(|V| + |E|)$ bei geeigneter Speicherung des Graphen (einfache Matrix-Vektor Multiplikation würde $\mathbf{O}(|V|^2)$ benötigen)

Referenzen

(Sehr) frei nach Folien von

- http:// www.mmds.org
- Dort kann man auch auf das zugehörige Buch zugreifen