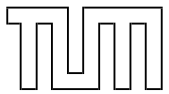
# Fortgeschrittene Netzwerk- und Graph-Algorithmen

Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Wintersemester 2009/10



# Übersicht

1. Algorithmen für Zentralitätsindizes
   - Basis-Algorithmen
   - Berechnung der Betweenness-Zentralitäten

# Kürzeste Pfade: SSSP / Dijkstra

**Algorithmus 1** : Dijkstra-Algorithmus (SSSP)

**Input** : $G = (V, E), \quad \omega : E \to \mathbb{R}, \quad s \in V$
**Output** : Distanzen $d(s, v)$ zu allen $v \in V$

$P = \emptyset, T = V;$
$d(s, v) = \infty$ for all $v \in V; \quad d(s, s) = 0; \quad pred(s) = 0;$
**while** $P \neq V$ **do**
  $v = \text{argmin}_{v \in T}\{d(s, v)\};$
  $P := P \cup v; \quad T := T \setminus v;$
  **for** $w \in N(v)$ **do**
    **if** $d(s, w) > d(s, v) + \omega(v, w)$ **then**
      $d(s, w) := d(s, v) + \omega(v, w);$
      $pred(w) := v;$

# Kürzeste Pfade: SSSP / Dijkstra

- Datenstruktur: Prioritätswarteschlange
  (z.B. Fibonacci Heap: amortisierte Komplexität $\mathcal{O}(1)$ für `insert` und `decreaseKey`, $\mathcal{O}(\log n)$ `deleteMin`)

- Komplexität:
  - $n-1$ `insert`
  - $n-1$ `deleteMin`
  - $\mathcal{O}(m)$ `decreaseKey`

  $\Rightarrow \mathcal{O}(m + n \log n)$

- aber: nur für nichtnegative Kantengewichte, sonst Bellman/Ford-Algorithmus in $\mathcal{O}(mn)$

# Kürzeste Pfade: APSP / Floyd-Warshall

**Algorithmus 2** : Floyd-Warshall APSP Algorithmus

**Input** : Graph $G = (V, E)$, edge weights $\omega : E \longrightarrow R$
**Output** : Shortest path distances $d(u, v)$ between all $u, v \in V$

$d(u, v) = \infty, pred(u, v) = 0$ for all $u, v \in V$;
$d(v, v) = 0$ for all $v \in V$;
$d(u, v) = \omega(u, v), pred(u, v) = u$ for all $\{u, v\} \in E$;
**for** $v \in V$ **do**
  **for** $\{u, w\} \in V \times V$ **do**
    **if** $d(u, w) > d(u, v) + d(v, w)$ **then**
      $d(u, w) := d(u, v) + d(v, w)$;
      $pred(u, w) := pred(v, w)$;

# Kürzeste Pfade: APSP / Floyd-Warshall

- Komplexität: $\mathcal{O}(n^3)$
- funktioniert auch, wenn Kanten mit negativem Gewicht existieren
- Kreise negativer Länge werden nicht direkt erkannt (wie z.B. beim Bellman-Ford-Alg.) und verfälschen das Ergebnis, sind aber indirekt am Ende an negativen Diagonaleinträgen der Distanzmatrix erkennbar.

# Betweenness Centrality

$$c_B(v) = \sum_{s \in V \setminus \{v\}} \sum_{t \in V \setminus \{v\}} \delta_{st}(v) = \sum_{s \in V \setminus \{v\}} \sum_{t \in V \setminus \{v\}} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Mögliche Berechnung:

1. Berechne Länge und Anzahl kürzester Pfade zwischen allen Knotenpaaren
2. Betrachte zu jedem Knoten $v$ alle möglichen Paare $s, t$ und berechne den Anteil kürzester Pfade durch $v$
   Bedingung(Bellman-Kriterium):

$$d(s,t) = d(s,v) + d(v,t)$$

# Betweenness Centrality

1. Modifiziere BFS bzw. Dijkstras Algorithmus
   - Ersetze einzelne Vorgängerknoten durch eine Vorgängermenge $\text{pred}(s, v) = \{u \in V : \{u, v\} \in E, d(s, v) = d(s, u) + w(u, v)\}$
   - Es gilt dann

$$\sigma_{sv} = \sum_{u \in \text{pred}(s,v)} \sigma_{su}$$

$\Rightarrow$ Für einen Startknoten $s \in V$ kann die Anzahl kürzester Pfade zu jedem anderen Knoten in $\mathcal{O}(m + n \log n)$ für gewichtete und in $\mathcal{O}(m)$ für ungewichtete Graphen berechnet werden.

$\Rightarrow$ Die Berechnung von $\sigma_{st}$ für alle Paare $s, t \in V$ kann in $\mathcal{O}(mn + n^2 \log n)$ bzw. $\mathcal{O}(mn)$ berechnet werden.

# Betweenness Centrality

2. Im Fall $d(s,t) = d(s,v) + d(v,t)$ gilt

$$\sigma_{st}(v) = \sigma_{sv} \cdot \sigma_{vt}$$

ansonsten $(d(s,t) < d(s,v) + d(v,t))$ ist $\sigma_{st}(v) = 0$

$\Rightarrow$ naive Berechnung:
$\mathcal{O}(n^2)$ pro Knoten $v$ (Summation über alle $s \neq v \neq t$),
also insgesamt Zeit $\mathcal{O}(n^3)$ und Platz $\mathcal{O}(n^2)$ für Speicherung aller $d(s,t)$ und $\sigma_{st}$

# Betweenness Centrality (Brandes)

Abhängigkeit eines Paares $(s, t)$ von $v$:

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

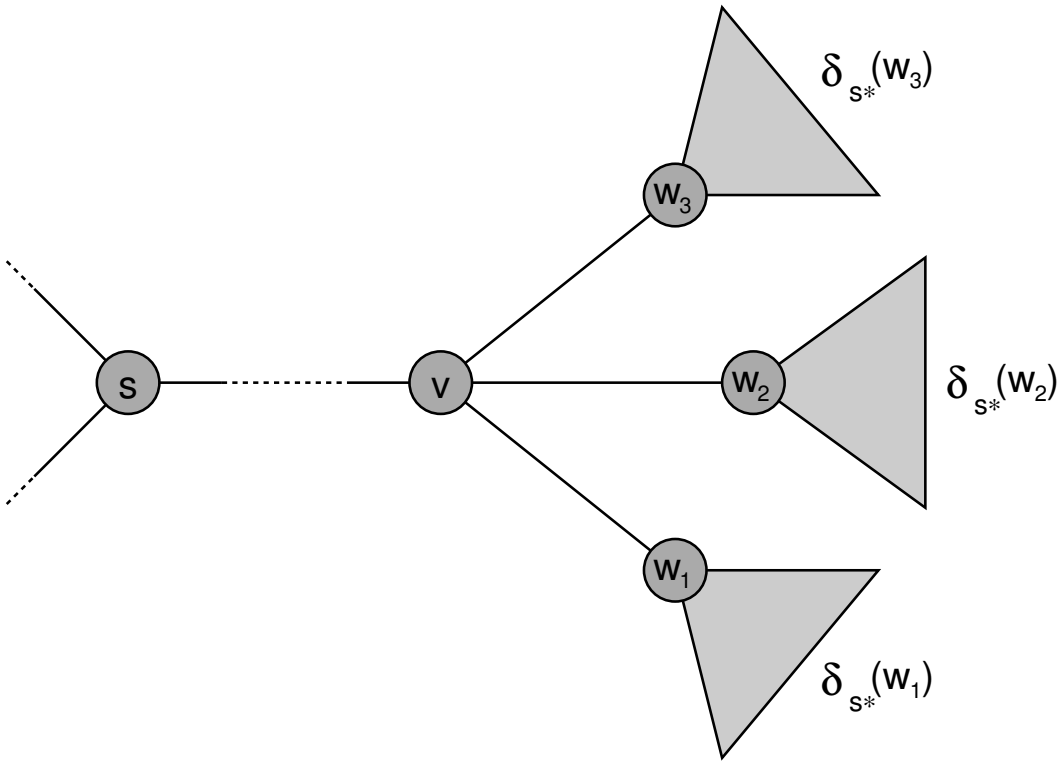
Abhängigkeit eines Startknotens $s$ von $v$:

$$\delta_{s*}(v) = \sum_{t \in V \setminus \{v\}} \delta_{st}(v)$$

Betweenness von $v$:

$$\Rightarrow \qquad c_B(v) = \sum_{s \in V \setminus \{v\}} \delta_{s*}(v)$$

# Betweenness Centrality (Brandes)



# Betweenness Centrality (Brandes)

## Lemma

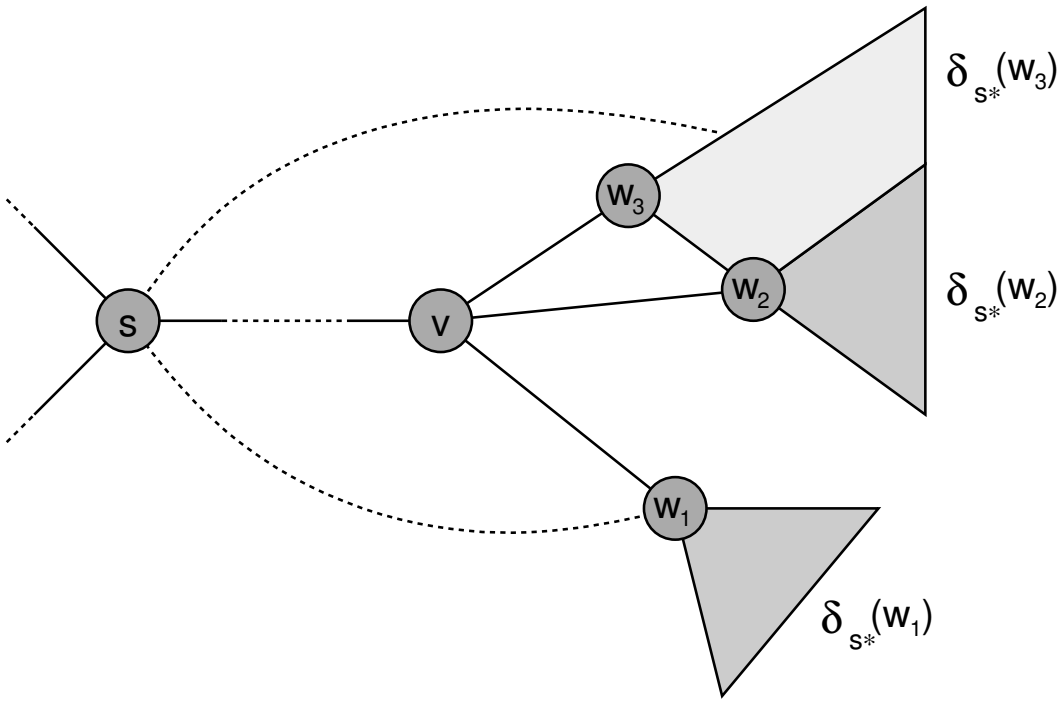
*Für den Fall, dass es von $s \in V$ genau einen kürzesten Pfad zu jedem $t \in V$ gibt, gilt*

$$\delta_{s*}(v) = \sum_{w:\ v \in pred(s,w)} (1 + \delta_{s*}(w))$$

## Beweis.

- Die kürzesten Wege von $s$ formen einen Baum.
- Also liegt $v$ auf *allen* kürzesten Wegen von $s$ zu einem $t$ oder auf *keinem*, d.h. $\delta_{st}(v)$ ist 1 oder 0.
- $v$ liegt auf allen kürzesten Wegen zu den Nachfolgern, sowie auf allen kürzesten Pfaden, auf denen diese Nachfolger liegen.

# Betweenness Centrality (Brandes)




Im allgemeinen Fall werden die Anteile der Abhängigkeiten von den Nachfolgern über die Kanten des Kürzeste-Wege-DAGs propagiert.

# Betweenness Centrality (Brandes)

## Satz

*Für die Abhängigkeit $\delta_{s*}(v)$ eines Startknotens $s \in V$ von den anderen Knoten $v \in V$ gilt:*

$$\delta_{s*}(v) = \sum_{w:\ v \in pred(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}}(1 + \delta_{s*}(w))$$

# Betweenness Centrality (Brandes)

**Beweis.**

- $\delta_{st}(v) > 0$ gilt nur für die $t \in V \setminus \{s\}$, für die $v$ auf mindestens einem kürzesten Pfad von $s$ nach $t$ liegt.
- Jeder solche Pfad hat exakt eine Kante $\{v, w\}$ mit $v \in \text{pred}(s, w)$ (siehe Abbildung).

# Betweenness Centrality (Brandes)

## Beweis.

- Definiere $\sigma_{st}(v, e)$: Anzahl kürzester $s$-$t$-Pfade die sowohl Knoten $v$ als auch Kante $e$ enthalten
- Definiere entsprechend

$$\delta_{st}(v, e) = \frac{\sigma_{st}(v, e)}{\sigma_{st}}$$

$$\delta_{s*}(v) = \sum_{t \in V \setminus \{v\}} \delta_{st}(v) = \sum_{t \in V \setminus \{v\}} \left( \sum_{w:\ v \in \text{pred}(s,w)} \delta_{st}(v, \{v, w\}) \right)$$

$$= \sum_{w:\ v \in \text{pred}(s,w)} \left( \sum_{t \in V \setminus \{v\}} \delta_{st}(v, \{v, w\}) \right)$$

# Betweenness Centrality (Brandes)

## Beweis.

- Betrachte Knoten $w$, so dass $v \in \text{pred}(s, w)$
- $\sigma_{sw}$ kürzeste Pfade von $s$ nach $w$,
  davon $\sigma_{sv}$ von $s$ nach $v$ gefolgt von Kante $\{v, w\}$
- Anteil $\sigma_{sv}/\sigma_{sw}$ der Anzahl kürzester Pfade von $s$ nach $t$ über $w$ benutzt auch die Kante $\{v, w\}$:

$$\delta_{st}(v, \{v, w\}) = \begin{cases} \frac{\sigma_{sv}}{\sigma_{sw}} & \text{falls } t = w \\ \frac{\sigma_{sv}}{\sigma_{sw}} \cdot \frac{\sigma_{st}(w)}{\sigma_{st}} & \text{falls } t \neq w \end{cases}$$

# Betweenness Centrality (Brandes)

## Beweis.

$$
\begin{aligned}
\delta_{s*}(v) &= \sum_{w:\ v \in \mathrm{pred}(s,w)} \left( \sum_{t \in V \setminus \{v\}} \delta_{st}(v, \{v,w\}) \right) \\
&= \sum_{w:\ v \in \mathrm{pred}(s,w)} \left( \frac{\sigma_{sv}}{\sigma_{sw}} + \sum_{t \in V \setminus \{v,w\}} \frac{\sigma_{sv}}{\sigma_{sw}} \cdot \frac{\sigma_{st}(w)}{\sigma_{st}} \right) \\
&= \sum_{w:\ v \in \mathrm{pred}(s,w)} \left( \frac{\sigma_{sv}}{\sigma_{sw}} + \frac{\sigma_{sv}}{\sigma_{sw}} \sum_{t \in V \setminus \{v,w\}} \frac{\sigma_{st}(w)}{\sigma_{st}} \right) \\
&= \sum_{w:\ v \in \mathrm{pred}(s,w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s*}(w))
\end{aligned}
$$

# Betweenness Centrality (Brandes)

## Beweis.

Erklärung:

- Zuerst wird die Summe in die beiden Fälle $t = w$ und $t \neq w$ aufgeteilt.
- Dann wird ausgeklammert.
- Es gilt

$$\sum_{t \in V \setminus \{v,w\}} \frac{\sigma_{st}(w)}{\sigma_{st}} = \sum_{t \in V \setminus \{w\}} \frac{\sigma_{st}(w)}{\sigma_{st}} = \delta_{s*}(w)$$

weil aufgrund der Annahme, dass $v$ ein Vorgänger von $w$ bezüglich Startknoten $s$ ist, für den Fall $t = v$ gilt, dass $\sigma_{st}(w) = \sigma_{sv}(w) = 0$

# Betweenness Centrality (Brandes)

- Berechne $n$ kürzeste-Wege-DAGs
  (einen für jeden Startknoten $s \in V$)
- Berechne nacheinander für alle $s \in V$ aus dem kürzeste-Wege-DAG von $s$ die Abhängigkeiten $\delta_{s*}(v)$ für alle anderen Knoten $v \in V$
  Vorgehen: rückwärts, von den Blättern im kürzeste-Wege-Baum bzw. von der entferntesten Schicht im kürzeste-Wege-DAG zum Startknoten hin
- Summiere die einzelnen Abhängigkeiten
  (kann schon parallel während der Berechnung aufsummiert werden, um nicht $\mathcal{O}(n^2)$ Platz zu verbrauchen)

# Betweenness Centrality (Brandes)

**Satz**

*Die Betweenness-Zentralität $c_B(v)$ für alle Knoten $v \in V$ kann*

- *für gewichtete Graphen in Zeit* $\mathcal{O}(n(m + n \log n)) = \mathcal{O}(nm + n^2 \log n)$
- *für ungewichtete Graphen in* $\mathcal{O}(mn)$

*berechnet werden.*
*Der Algorithmus benötigt dabei nur* $\mathcal{O}(n + m)$ *Speicherplatz.*

Bemerkung:
Die anderen auf kürzesten Wegen basierenden Zentralitäten kann man relativ einfach mit SSSP-Traversierung berechnen.