Script generated by TTT

Title: Seidl: GAD (22.06.2016)

Date: Wed Jun 22 13:24:37 CEST 2016

Duration: 41:49 min

Pages: 19

APSP / Kantengewichte

Bessere Strategie:

- reduziere n Aufrufe des Bellman-Ford-Algorithmus auf n Aufrufe des Dijkstra-Algorithmus

Problem:

- Dijkstra-Algorithmus funktioniert nur für nichtnegative Kantengewichte

Lösung:

- Umwandlung in nichtnegative Kantenkosten ohne Verfälschung der kürzesten Wege

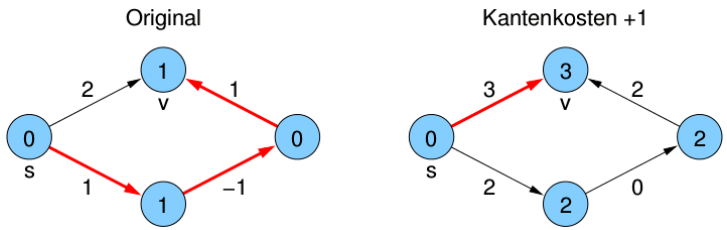
$O(n^2 \log n + n \cdot m)$ $O(n^2 \cdot m)$

Naive Modifikation der Kantengewichte

Naive Idee:

- negative Kantengewichte eliminieren, indem auf jedes Kantengewicht der gleiche Wert c addiert wird

$\Rightarrow$ verfälscht kürzeste Pfade



Knotenpotential

Sei $\Phi : V \mapsto \mathbb{R}$ eine Funktion, die jedem Knoten ein Potential zuordnet.

Modifizierte Kantenkosten von $e = (v, w)$:

$$\bar{c}(e) = \Phi(v) + c(e) - \Phi(w)$$

Lemma

Seien p und q Wege von v nach w in G.

$c(p)$ und $c(q)$ bzw. $\bar{c}(p)$ und $\bar{c}(q)$ seien die aufsummierten Kosten bzw. modifizierten Kosten der Kanten des jeweiligen Pfads.

Dann gilt für jedes Potential Φ:

$$\bar{c}(p) < \bar{c}(q) \quad \Leftrightarrow \quad c(p) < c(q)$$

Knotenpotential

Beweis.

Sei $p = (v_1, \ldots, v_k)$ beliebiger Weg und $\forall i: \; e_i = (v_i, v_{i+1}) \in E$

Es gilt:

$$\begin{aligned} \bar{c}(p) &= \sum_{i=1}^{k-1} \bar{c}(e_i) \\ &= \sum_{i=1}^{k-1} (\Phi(v_i) + c(e_i) - \Phi(v_{i+1})) \\ &= \Phi(v_1) + c(p) - \Phi(v_k) \end{aligned}$$

d.h. modifizierte Kosten eines Pfads hängen nur von ursprünglichen Pfadkosten und vom Potential des Anfangs- und Endknotens ab.
(Im Lemma ist $v_1 = v$ und $v_k = w$) □

Potential für nichtnegative Kantengewichte

Lemma

Annahme:

- *Graph hat keine negativen Kreise*
- *alle Knoten von s aus erreichbar*

Sei für alle Knoten v das Potential $\Phi(v) = d(s, v)$.

Dann gilt für alle Kanten e: $\bar{c}(e) \geq 0$

Beweis.

- für alle Knoten v gilt nach Annahme: $d(s, v) \in \mathbb{R}$ (also $\neq \pm\infty$)
- für jede Kante $e = (v, w)$ ist

$$\begin{aligned} d(s, v) + c(e) &\geq d(s, w) \\ d(s, v) + c(e) - d(s, w) &\geq 0 \end{aligned}$$

□

Johnson-Algorithmus für APSP

- füge neuen Knoten s und Kanten (s, v) für alle v hinzu mit $c(s, v) = 0$

$\Rightarrow$ alle Knoten erreichbar

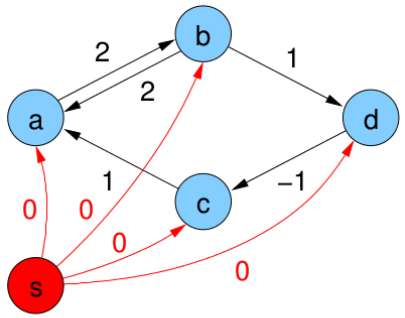
- berechne $d(s, v)$ mit Bellman-Ford-Algorithmus
- setze $\Phi(v) = d(s, v)$ für alle v
- berechne modifizierte Kosten $\bar{c}(e)$

$\Rightarrow$ $\bar{c}(e) \geq 0$, kürzeste Wege sind noch die gleichen

- berechne für alle Knoten v die Distanzen $\bar{d}(v, w)$ mittels Dijkstra-Algorithmus mit modifizierten Kantenkosten auf dem Graph ohne Knoten s
- berechne korrekte Distanzen $d(v, w) = \bar{d}(v, w) + \Phi(w) - \Phi(v)$

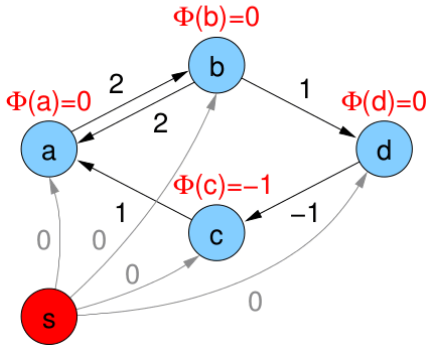
Johnson-Algorithmus für APSP

1. künstlicher Startknoten s:



Johnson-Algorithmus für APSP

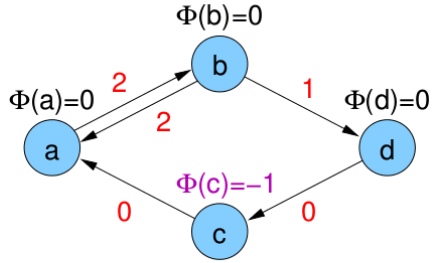
2. Bellman-Ford-Algorithmus auf s:



Johnson-Algorithmus für APSP

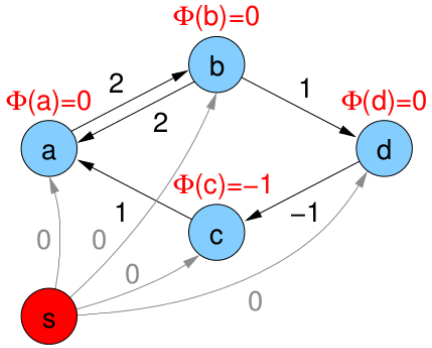
3. $\bar{c}(e)$-Werte für alle $e = (v, w)$ berechnen:

$$\bar{c}(e) = \Phi(v) + c(e) - \Phi(w)$$



Johnson-Algorithmus für APSP

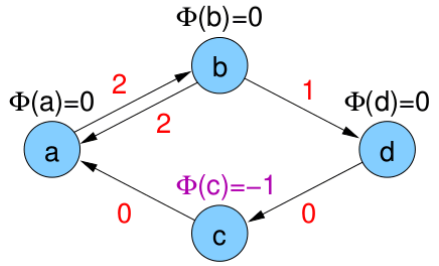
2. Bellman-Ford-Algorithmus auf s:



Johnson-Algorithmus für APSP

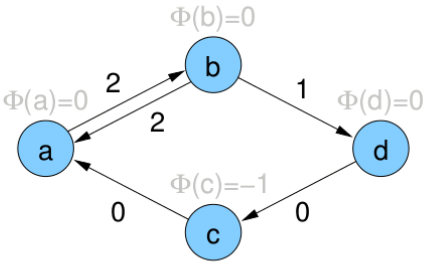
3. $\bar{c}(e)$-Werte für alle $e = (v, w)$ berechnen:

$$\bar{c}(e) = \Phi(v) + c(e) - \Phi(w)$$



Johnson-Algorithmus für APSP

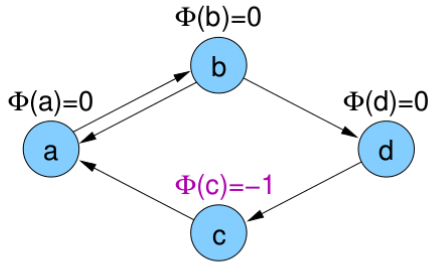
4. Distanzen $\bar{d}$ mit modifizierten Kantengewichten via Dijkstra:



$\bar{d}$	a	b	c	d
a	0	2	3	3
b	1	0	1	1
c	0	2	0	3
d	0	2	0	0

Johnson-Algorithmus für APSP

5. korrekte Distanzen berechnen: $d(v,w) = \bar{d}(v,w) + \Phi(w) - \Phi(v)$



d	a	b	c	d
a	0	2	2	3
b	1	0	0	1
c	1	3	0	4
d	0	2	−1	0

Johnson-Algorithmus für APSP

Laufzeit:

$$\begin{aligned} T_{\text{Johnson}}(n,m) &= O(T_{\text{Bellman-Ford}}(n+1, m+n) + n \cdot T_{\text{Dijkstra}}(n,m)) \\ &= O((m+n)\cdot(n+1) + n\cdot(n\log n + m)) \\ &= O(m\cdot n + n^2 \log n) \end{aligned}$$

(bei Verwendung von Fibonacci Heaps)

Floyd-Warshall-Algorithmus für APSP

Grundlage:

- geht der kürzeste Weg von u nach w über v, dann sind auch die beiden Teile von u nach v und von v nach w kürzeste Pfade zwischen diesen Knoten
- Annahme: alle kürzesten Wege bekannt, die nur über Zwischenknoten mit Index kleiner als k gehen

$\Rightarrow$ kürzeste Wege über Zwischenknoten mit Indizes bis einschließlich k können leicht berechnet werden:

  - entweder der schon bekannte Weg über Knoten mit Indizes kleiner als k
  - oder über den Knoten mit Index k (hier im Algorithmus der Knoten v)

Floyd-Warshall-Algorithmus für APSP

Algorithmus Floyd-Warshall: löst APSP-Problem

Eingabe : Graph $G = (V, E)$, $c : E \mapsto \mathbb{R}$
Ausgabe : Distanzen $d(u, v)$ zwischen allen $u, v \in V$

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

Graphen | Kürzeste Wege

Floyd-Warshall-Algorithmus für APSP

Algorithmus Floyd-Warshall: löst APSP-Problem

Eingabe : Graph $G = (V, E)$, $c : E \mapsto \mathbb{R}$
Ausgabe : Distanzen $d(u, v)$ zwischen allen $u, v \in V$

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

H. Seidl (TUM) GAD SS'16 519

Floyd-Warshall-Algorithmus für APSP

- Komplexität: $O(n^3)$
- funktioniert auch, wenn Kanten mit negativem Gewicht existieren
- Kreise negativer Länge werden nicht direkt erkannt und verfälschen das Ergebnis, sind aber indirekt am Ende an negativen Diagonaleinträgen der Distanzmatrix erkennbar

Minimaler Spannbaum

Frage: Welche Kanten nehmen, um mit minimalen Kosten alle Knoten zu verbinden?