03.07.12

Grundkonzept für die folgenden Algorithmen:

Flüsse erhöhen oder erniedrigen!

Beispiel:

(siehe 49)!

Interpretation: Fluss erniedrigen ist gleichbedeutend mit „Fluss in entgegengesetzte Richtung schicken".

- Konsequenz dieser Idee:

<u>Definition 4.2</u> (Residualgraph, augmentierende Pfade)

„Rest" „verbessernde"

(1) Für Digraph G ist

$$\overleftrightarrow{G} := \left(V(G),\ E(G) \,\dot\cup\, \{\overleftarrow{e} \mid e \in E(G)\} \right) \quad \text{mit}$$

$$\overleftarrow{e} = (w,v) \text{ für } e = (v,w) \quad \text{(die „Gegenkante" von } e\text{)}$$

der doppelt gerichtete Graph zu G.

(2) Für einen Fluss f in einem Digraphen G mit Kapazitäten $u: E(G) \to \mathbb{R}_+$ definieren wir die

Residualkapazitäten u_f: $\begin{cases} u(e) - f(e) & \text{für } e \in E(G) \text{ Vorwärtskanten} \\ f(\overleftarrow{e}) & \text{für Rückwärtskanten} \end{cases}$

~~Interpretation~~

(3) Der Residualgraph G_f ist der Graph

$$\left(V(G),\ \{ e \in E(\overleftrightarrow{G}) \mid u_f(e) > 0 \} \right).$$

(4) Interpretation des Residualgraphen:

u_f auf Vorwärtskanten beschreibt, um wieviel man f noch erhöhen kann

u_f auf Rückwärtskanten beschreibt, um wieviel man f erniedrigen kann.

(5) Um für einen Fluss f und einen Pfad (oder Kreis) P in G_f den Fluss entlang P um γ zu augmentieren, muss man den Fluss für Vorwärtskanten um γ erhöhen, für Rückwärtskanten um γ reduzieren.

(6) Ein f-augmentierender Pfad in einem Netzwerk (G, u, s, t) mit einem Fluss f ist ein s-t-Pfad im Residualgraphen G_f.

Natürliche Konsequenz:

Algorithmus 4.3 (Ford-Fulkerson 1956)

Eingabe: Netzwerk (G, u, s, t)

Ausgabe: Ein s-t-Fluss von maximalem Wert

1. Setze $f(e) = 0$ für alle $e \in E(G)$
2. Finde einen f-augmentierenden Pfad P in G_f.
   Falls keiner existiert: STOP
3. Bestimme $\gamma := \min_{e \in E(P)} u_f(e)$
   Erhöhe f entlang P um γ, GO TO (2)

In VL mit WHILE geschrieben!

In VL eingefügt:
- Problem MinCut (4.6)
- Dualität (4.7)

Satz 4.5

Ein s-t-Fluss f ist optimal $\Leftrightarrow$ Es gibt keinen f-augmentierenden Pfad

Also stoppt der Algorithmus mit einem Optimalwert, wenn er stoppt.
(Wie wir noch sehen werden ist das „wenn" und „wann" u.U. etwas problematisch!)

Beweis:

„$\Rightarrow$" Wenn es einen augmentierenden Pfad gibt, wird von Algorithmus 4.3 in (3) ein besserer Fluss berechnet, also kann f nicht maximal sein.

„$\Leftarrow$" Wenn es keinen augmentierenden Pfad gibt, dann ist t in G_f nicht von s aus erreichbar.
Sei R die Menge der von s aus in G_f erreichbaren Knoten. Nach Definition von G_f ist also
für alle $e \in \delta_G^+(R)$: $f(e) = u(e)$
für alle $e \in \delta_G^-(R)$: $f(e) = 0$.

Lemma 4.4 (Schranken durch Schnitte)

Für $A \subseteq V(G)$ mit $s \in A$, $t \notin A$ gilt für jeden st-Fluss f:

(a) $$\text{Wert}(f) = \sum_{e \in \delta^+(A)} f(e) - \sum_{e \in \delta^-(A)} f(e)$$

(b) $$\text{Wert}(f) \leq \sum_{e \in \delta^+(A)} u(e)$$

Beweis: (a) Da für alle $v \in A \setminus \{s\}$ Flusserhaltung gilt, ist

$$\text{Wert}(f) = \sum_{e \in \delta^+(s)} f(e) - \sum_{e \in \delta^-(s)} f(e)$$

$$= \sum_{v \in A} \left(\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) \right)$$

$$= \sum_{e \in \delta^+(A)} f(e) - \sum_{e \in \delta^-(A)} f(e) .$$

(b) Mit $0 \leq f(e) \leq u(e)$ ergibt sich das aus (a)

Also ist nach Lemma 4.4 (a)

$$\text{Wert}(f) = \sum_{e \in \delta^+(R)} f(e) - \sum_{e \in \delta^-(R)} f(e) = \sum_{e \in \delta^+(R)} u(e) .$$

Nach Lemma 4.4 (b) ist $\sum_{e \in \delta^+(R)} u(e)$ aber eine obere Schranke fürs Optimum, also f optimal.

$\square$

Interpretation:

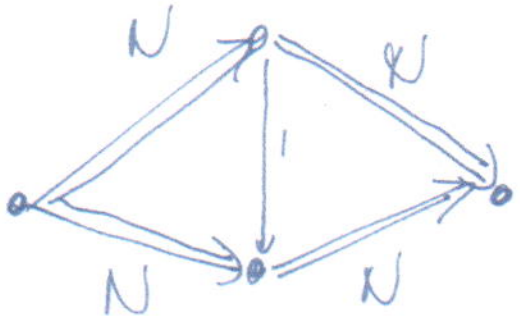
Satz 4.8 (Max Flow - Min Cut)

In jedem Netzwerk ist der maximale Wert eines s-t-Flusses gleich dem minimalen Kapazität eines s-t-Schnittes.

Historisch: Harris + Ross 1955 untersuchten einen minimalen Schnitt im selben sowjetischen Eisenbahnnetz, für das Tolstoi 1930 einen maximalen Fluss bestimmt hatte!

Schwierigkeit mit Ford-Fulkerson:

Beispiel 4.9



Bei „schlechter" Wahl der augmentierenden Pfade kann der Algorithmus u.U. sehr lange benötigen – hier 2N Augmentierungen!

Lösung:

ALGORITHMUS 4.10 (Edmonds-Karp 1972)

Eingabe : (G, u, s, t)

Ausgabe : Ein s-t-Fluss maximalen Werts

(1) Setze $f(e) = 0$ für alle $e \in E(G)$

(2) Bestimme einen kürzesten f-augmentierenden Pfad P
Wenn es keinen gibt : STOP

(3) Berechne $\gamma := \min_{e \in E(P)} u_f(e)$.
Augmentiere f entlang P um γ, GO TO (2)

Also sollte man (2) von Algorithmus 4.3 mit BFS implementieren!

Ohne Beweise:

Satz 4.11 (Edmonds und Karp 1972)

Unabhängig von den Kapazitäten stoppt der Edmonds-Karp-Algorithmus nach höchstens $\frac{mn}{2}$ Augmentierungen.

Korollar 4.12

Der Edmonds-Karp-Algorithmus terminiert in $O(m^2 n)$ Schritten.

Beweis:

Jede Augmentierung kann mit BFS in $O(m)$ durchgeführt werden.