Datenstrukturen und effiziente Algorithmen

Markus Vieth David Klopp

29. Januar 2016

Inhaltsverzeichnis

1 Vorlesung

1.1 Restnetzwerk $G_f = (V, E_f)$

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{für } (u,v) \in E \\ f(v,u) & \text{für } (v,u) \in E \\ 0 & \text{sonst} \end{cases}$$

1.2 Max-Flow-Min-Cut-Theorem

1. $|f|$ ist maximal $\Rightarrow$ Es gibt keinen f-verbessernden Pfad

2. Es gibt einen (S,T)-Schnitt, so dass $c(S,T) = |f|$

3. $c(S,T) = |f| \Rightarrow f$ ist maximal und $c(S,T)$ minimal

1.2.1 Beweis: 1. $\Rightarrow$ 2.

Gegeben sein ein maximaler Fluss $f \Rightarrow t$ ist ein G_f nicht von s erreichbar.

$$\text{Sei } S = \{v \in V | \exists\, p: \ s \rightsquigarrow v \text{ in } G_f\}, T = V \setminus S, t \in T$$

1.2.2 Idee

Zeige: Alle Vorwärtskanten über den Schnitt S, T sind saturiert, d.h. $f(u,v) = c(u,v)$ Alle Rückwärtskanten von T nach S tragen den Flusswert 0.

Beweis Sei $(u,v) \in S \times T \cap E$

Anmerkung $f(u,v) < c(u,v)$

$$\Rightarrow (u,v) \in E_f \text{ mit } c_f(u,v) = c(u,v) - f(u,v) > 0$$

$$\Rightarrow v \text{ ist von } s \text{ aus erreichbar:} s \rightsquigarrow u \rightarrow v \lightning$$

Sei $(v,u) \in T \times S$

Anmerkung: $f(v,u) > 0$

$$\Rightarrow (u,v) \in E_f \text{ mit } c_f(u,v) = f(v,u) > 0$$

$$\Rightarrow v \text{ ist auf dem Weg } s \rightsquigarrow u \to v \text{ in } G_f \text{ erreichbar.} \lightning$$

$$|f| = \sum_{u \in S, v \in T} f(u,v) - \sum_{u \in S, v \in T} f(v,u) = \sum_{u \in S, v \in T} c(u,v) - 0 = c(S,T)$$

1. $\Rightarrow$ 2.
q.e.d.

$$|f| = c(S,T) \Rightarrow f \text{ maximal und } c(S,T) \text{ minimal.}$$

Offene Frage Unter welchen Bedingungen terminiert der Ford-Fulkerson-Algorithmus?

Anmerkung $c : E \to \mathbb{N}$

$$\Rightarrow |f^*|^{\mathrm{I}} \in \mathbb{N}, c_{min}(p) \in \mathbb{N} \text{ für } p \text{ flussverbessernder Pfad} \geq 1$$

$$\Rightarrow \text{ Es genügen } |f^*| \text{ viele Iterationen zur Flussverbesserung}$$

$$\Rightarrow \text{Laufzeit}(|f^*| \cdot |E|)$$

1.2.3 Zwick

$$\overline{\phi} = \frac{\sqrt{5}-1}{2} \quad X > 1000 \quad |f^*| = 2X + 1$$

Invariante $\overline{\phi} = 0{,}613\ldots$

$$1 - \overline{\phi} = \overline{\phi}^{\,2}$$
$$\overline{\phi}^{\,k-1} - \overline{\phi}^{\,k} = \overline{\phi}^{\,k+1}$$

1. Schritt Schicke $\overline{\phi}^{\,k}$ Flusseinheiten entlang Pfad B

2. Schritt Schicke $\overline{\phi}^{\,k}$ entlang C

3. Schritt Schicke $\overline{\phi}^{\,k+1}$ entlang B

2. Schritt Schicke $\overline{\phi}^{\,k+1}$ entlang A Wir iterieren diese 4 Schritte unendlich oft

$$|f| = 1 + 2 \cdot \sum_{k=1}^{\infty} \overline{\phi}^{\,k} + 2 \cdot \sum_{k=1}^{\infty} \overline{\phi}^{\,k+1} < 7 \leq 2X + 1 \qquad \sum_{k=0}^{\infty} \overline{\phi}^{\,k} = \frac{1}{1 - \overline{\phi}}$$

[I] maximaler Fluss

2 Vorlesung

2.1 Edmonds-Karp Algorithmus

$$G = (V, E) \quad , c : R \to \mathbb{R}_0^+, G_F = (V, E_f), G_f^L = (V, E_f^L)$$

```
f = 0;
while(∃ p ⇝ t ∈ G_f^L = (V, E_f^L)) {
    sei c_min(p) die kleinste Restkapazität auf p
    f(u,v) = { f(u,v) + c_min(p)   falls (u,v) ∈ p
             { f(u,v) − c_min(p)   falls (v,u) ∈ p
}
```

$\delta_f(s, v)$ die kleinste Zahl von Kanten, die in G_f^L benötigt werden, um von s nach v zu gelangen.

2.1.1 Lemma:

Im Verlauf des Edmonds-Karp Algorithmus gilt:

$$\delta_{f'}(s, v) \geq \delta_f(s, v)$$

wobei der Fluss f' durch eine Flussverbesserung aus f hervorgegangen ist.

2.1.2 Beweis durch Widerspruch

Annamhe

$$\exists v \in V : \delta_{f'}(s, v) < \delta_f(s, v) \qquad (**)$$

sei v so gewählt, dass $\delta_{f'}(s, v)$ minimal.
Sei $s \rightsquigarrow u \to v$ ein kürzester Weg in $G_{f'}^L$.

$$\delta_f(s, u) \leq \delta_{f'}(s, u) = \delta_{f'}(s, v) - 1 \qquad (*)$$

Behauptung

$$(u, v) \notin E_f^L$$

Beweis durch Widerspruch

Annahme $(u, v) \in E_f^L$

$$\delta_f(s, v) \leq^{\mathrm{I}} \delta_f(s, u) + 1 \leq^{\mathrm{II}} \delta_{f'}(s, v) \quad \lightning \text{ zu } (**)$$

[I] Dreiecksungleichung
[II] wegen $(*)$

$$\Rightarrow (u,v) \notin E_f^L \text{ aber } (u,v) \in E_{f'}^L$$

d.h. Bei der Flussverbesserung von f zu f' wurde die Kante (v,u) benutzt in G_f^L.

$$\begin{aligned}\delta_f(s,u) &= \delta_f(s,v) + 1 \\ &\overset{(**)}{>} \delta_{f'}(s,v) + 1 \\ &\overset{(*)}{\geq} \delta_f(s,u) + 2 \lightning\end{aligned}$$

q.e.d.

2.1.3 Lemma

Eine kante (u,v) kann un den Level-Rest-Netzwerken höchstens $\frac{|V|}{2}$ mal saturiert werden und damit temporär aus dem jeweiligen Rest-Netzwerk verschwinden

2.1.4 Beweis

$$\delta_f(s,v) = \delta_f(s,u) + 1$$
$$\delta_{f'}(s,v) = \delta_{f'}(s,u) - 1$$

$$\delta_f(s,u) = \delta_f(s,v) - 1 \leq \delta_{f'}(s,v) - 1 = \delta_{f'}(s,u) - 2$$
$$\Rightarrow \delta_{f'}(s,u) \geq \delta_f(s,u) + 2$$

q.e.d.

2.1.5 Laufzeitanalyse von Edmonds-Karp Algorithmus

Bei jeder Flussverbesserung wird mindestens eine Kante saturiert. Jede einzelne Kante kann aber höchstens $\frac{|V|}{2}$ mal saturiert werden.
$\Rightarrow$ Es gibt höchstens $\mathcal{O}(|E| \cdot |V|)$ viele Flussverbesserungen, Jede Flussverbesserung kann in $\mathcal{O}(|E|)$ ausgeführt werden.
$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(|E|^2 \cdot |V|)$

2.2 Algorithmus von Dinic

2.2.1 Sperrfluss (blocking flow)

$$G_f^L = (V, E_f^L)$$

Wir konstruieren einen Sperrfluss g für einen Graphen H, indem wir wiederholt entlang von (s,t)-Pfaden Fluss von s nach t transportieren.
Bevor wir diesen Prozess wiederholen, löschen wir saturierte Kanten aus H. Läuft man bei der Wegesuche in eine Sackgasse, so muss diese aus H entfernt werden, damit man zu einem späteren Zeitpunkt nicht wieder in diese Sackgasse gerät.

Ziel Algorithmus zur Sperrflussberechnung in Zeit $\mathcal{O}(|V| \cdot |E|)$

Abbildungsverzeichnis