Algorithmentheorie

Daniel Neuen (Universität Bremen)

WiSe 2023/24

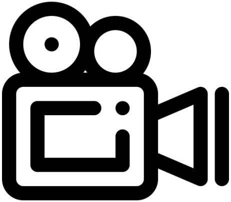
Netzwerkflüsse

11. Vorlesung

Aufzeichnung der Vorlesung

Diese Vorlesung wird aufgezeichnet und live gestreamt.

- Aufzeichnungen nur der Lehrenden durch sich selbst.
- Bei Rückfragen aus dem Auditorium und Diskussion bitte deutlich anzeigen, falls das Mikro stumm geschaltet werden soll.



Organisatorisches

Verbleibende Vorlesungen:

- VL 11 (11.01): Netzwerkflüsse
- VL 12 (18.01): Matchings
- VL 13 (25.01): Ausblick
- VL 14 (01.02): Wiederholung & Klausurvorbereitung

Verbleibende Übungen:

- 08-12.01: Hausaufgaben 5
- 15-19.01: Präsenzaufgaben 6
- 22-26.01: Hausaufgaben 6
- 29.01-02.02: Klausurvorbereitung

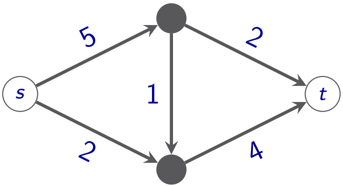
Netzwerkflüsse modellieren . . .

... und vieles mehr.

Netzwerk

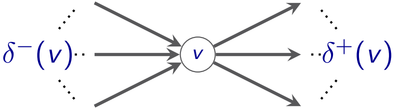
Netzwerk

- $G = (V, A, c)$: gewichteter Digraph
- $s \in V$: Quelle
- $t \in V$: Senke
- $c : A \to \mathbb{R}_+$: Kapazität



Zur Erinnerung: Für Knoten $v \in V$:

- $\delta^-(v) := \{(u, v) \in A\}$ Menge eingehender Kanten in v
- $\delta^+(v) := \{(v, u) \in A\}$ Menge ausgehender Kanten aus v



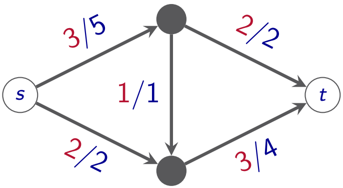
Universität Bremen

Maximale Flüsse und minimale Schnitte

s-t-Fluss

Netzwerk

- $G = (V, A, c)$: gewichteter Digraph
- $s \in V$: Quelle
- $t \in V$: Senke
- $c : A \to \mathbb{R}_+$: Kapazität



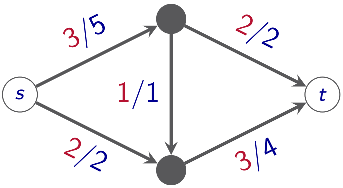
Ein **zulässiger s-t-Fluss** ist eine Funktion $f : A \to \mathbb{R}_+$, $a \in A$, die folgende Eigenschaften erfüllt:

- Kapazitätsbedingung: $0 \leq f(a) \leq c(a), \qquad \forall\, a \in A$
- Flusserhaltungsbedingung:
$$\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a), \qquad \forall v \in V \setminus \{s, t\}$$

Max s-t-Flussproblem

Der Überschuss (engl. excess) eines Flusses f in $v \in V$ ist

$$ex_f(v) := \sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a).$$



Der Wert eines (ausgehenden) Flusses ist $val(f) = ex_f(s)$.

Wegen der Flusserhaltung gilt $ex_f(s) = -ex_f(t)$, und in jedem anderen Knoten $v \neq s, t$ gilt $ex_f(v) = 0$.

Max s-t-Flussproblem

Gegeben sei ein Netzwerk $N = (V, A, c, s, t)$. Finde einen zulässigen s-t-Fluss f mit maximalem Flusswert $val(f)$.

Max s-t-Flussproblem

Bemerkung: Falls $f(a) \in \mathbb{N}$, dann ist f ein ganzzahliger Fluss. Im Allgemeinen ist $f(a)$ nicht ganzzahlig (fraktional).

$\longrightarrow$ Egal bspw. bei Zulieferung von Gas, Wasser, Elektrizität, aber nicht beim Routing von LKWs in Strassennetzwerk.

Satz

Das Max s-t-Flussproblem mit ganzzahligen Kapazitäten $c(a) \in \mathbb{N}$, $a \in A$, hat immer eine ganzzahlige Optimallösung.

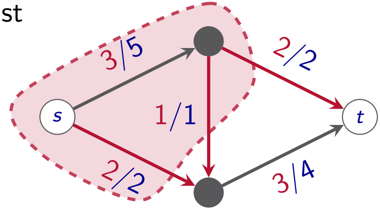
Beweis. (später)
Folgt aus Analyse des Algorithmus (bzw. aus Unimodularität).

s-t-Schnitt

Für $U \subseteq V$ sei $\delta^+(U) := \{(u,v) \in A \mid u \in U, v \in V \setminus U\}$.

Sei $U \subseteq V$ mit $s \in U$ und $t \notin U$, dann ist $C := \delta^+(U)$ ein s-t-Schnitt (Cut). Die Schnittkapazität ist

$$\text{cap}(C) := \sum_{a \in \delta^+(U)} c(a).$$



Min s-t-Schnittproblem

Gegeben sei ein Netzwerk. Finde einen s-t-Schnitt C minimaler Kapazität $\text{cap}(C)$.

Schnittkapazität offensichtlich obere Schranke an den maximalen Fluss.
$\longrightarrow$ formal?!

Anwendungen für Schnitte und Flüsse

Maximale Flüsse

- Lieferkapazitäten von Gas, Wasser, Elektrizität, Öl,...
- Produktionskapazitäten von Fertigungsstraßen
- Aufnahmekapazitäten von Abwassersystemen
- Lieferkapazitäten von Logistiknetzwerken
- Netzkapazitäten von Telekommunikationsnetzwerken

- typisches Unterproblem für ähnliche, aber kompliziertere Probleme

Minimale Schnitte

- Analyse von Bottlenecks in den obigen Netzwerken
- Robustheit/Störungsanfälligkeit der obigen Netze

Flusswert $\leq$ Schnittkapazität

Satz

Sei $N = (V, A, c, s, t)$ ein Netzwerk. Sei f ein s-t-Fluss und $C \subseteq A$ ein s-t-Schnitt. Dann gilt

$$val(f) \leq \text{cap}(C).$$

Beweis. Betrachte Schnitt $C = \delta^+(U)$ für $U \subset V$ mit $s \in U$, $t \notin U$.

- Nach Definition: $val(f) = ex_f(s) = \sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a)$.
- Wegen Flusserhaltung gilt:
  $\sum_{v \in U \setminus \{s\}} ex_f(v) = \sum_{v \in U \setminus \{s\}} \left(\sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a) \right) = 0$
- Also gilt

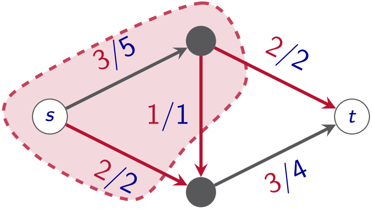
$$val(f) = ex_f(s) = \sum_{v \in U} ex_f(v) = \sum_{v \in U} \left(\sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a) \right)$$
$$= \sum_{a \in \delta^+(U)} f(a) - \sum_{a \in \delta^-(U)} f(a) \leq \text{cap}(C) \qquad \square$$

Max-Flow = Min-Cut

Tatsächlich gilt die deutlich stärkere Aussage. . .

Satz (Max-Fluss Min-Schnitt Theorem)

Gegeben sei ein Netzwerk $N = (V, A, c, s, t)$ mit Kapazitäten $c(a) \geq 0$, $a \in A$. Dann ist der Wert eines maximalen s-t-Flusses gleich der minimalen s-t-Schnittkapazität.



max Fluss = min Schnitt

Beweis. später; zunächst einige Vorbereitungen.

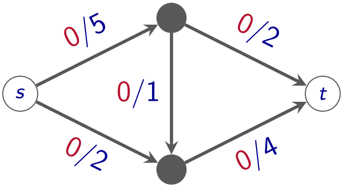
Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

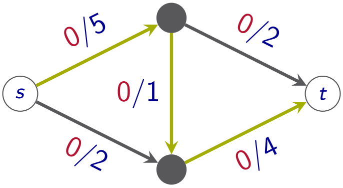
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität “erschöpft” ist.



Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

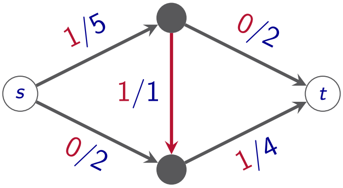
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität "erschöpft" ist.



Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

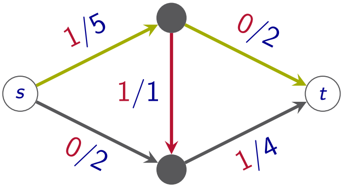
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität "erschöpft" ist.



Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

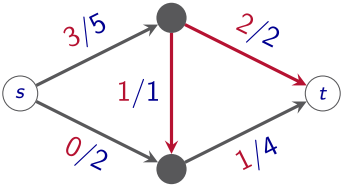
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität "erschöpft" ist.



Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

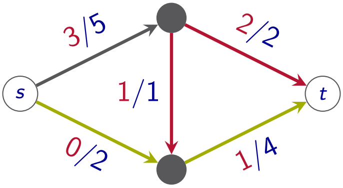
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität “erschöpft” ist.



Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

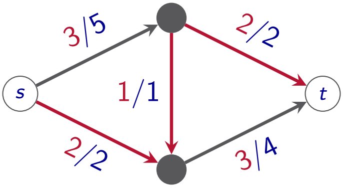
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität "erschöpft" ist.



Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

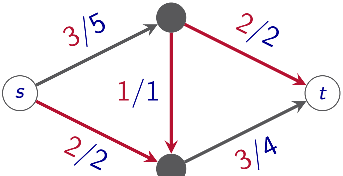
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität "erschöpft" ist.



Berechnung von Maximalen Flüssen (Idee)

Wie können wir einen maximalen Fluss in einem Netzwerk finden?

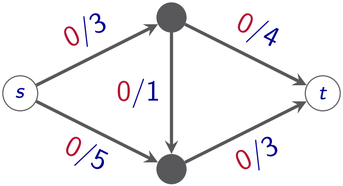
Greedy-Ansatz: Finde wiederholt einen Pfad P von s and t und schicke Fluss maximal möglichen Fluss entlang von P; ignoriere Kanten deren Kapazität “erschöpft” ist.



Problem: Wir erhalten nicht immer einen maximalen Fluss.

Berechnung von Maximalen Flüssen (Idee)

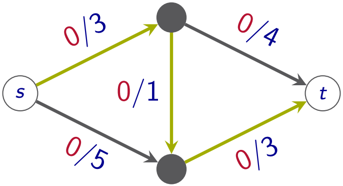
Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Berechnung von Maximalen Flüssen (Idee)

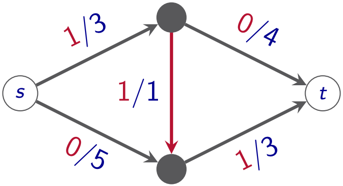
Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Berechnung von Maximalen Flüssen (Idee)

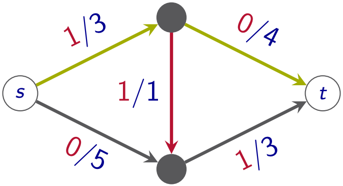
Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Berechnung von Maximalen Flüssen (Idee)

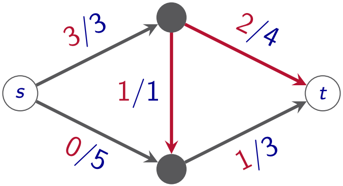
Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Berechnung von Maximalen Flüssen (Idee)

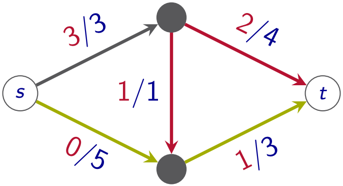
Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Berechnung von Maximalen Flüssen (Idee)

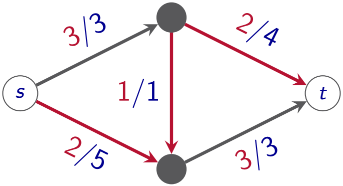
Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Berechnung von Maximalen Flüssen (Idee)

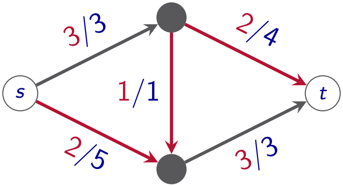
Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Berechnung von Maximalen Flüssen (Idee)

Problem: Wir erhalten nicht immer einen maximalen Fluss.



Unser Greedy-Ansatz findet einen Fluss mit Wert 5, der maximale Fluss hat aber Wert 6.

Idee: Erlaube “zurück schicken” von Fluss um Fehlentscheidungen zu korrigieren.
$\rightarrow$ Residualgraph

Der Residualgraph

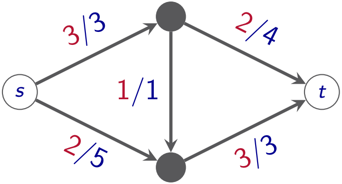
Konstruktion

- Rückwärtskanten einführen: $\overleftarrow{A} := \{\overleftarrow{a} \,:\, a \in A\}$
- Residualkapazitäten für $a \in \overleftrightarrow{A} := A \cup \overleftarrow{A}$ sind definiert als

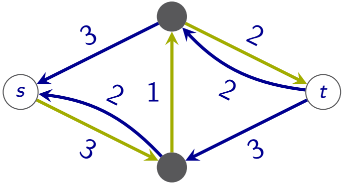
$$\bar{c}_f(a) := \begin{cases} c(a) - f(a) & \text{falls } a \in A \text{ (Vorwärtskante)} \\ f(a) & \text{falls } a \in \overleftarrow{A} \text{ (Rückwärtskante)} \end{cases}$$

Kanten mit $\bar{c}_f(a) = 0$ werden gelöscht.

Residualgraph $D_f = (V, A_f)$: $A_f := \{a \in \overleftrightarrow{A} \,:\, \bar{c}_f(a) > 0\}$



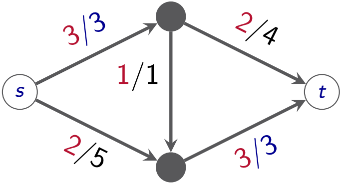
Digraph D mit Fluss f



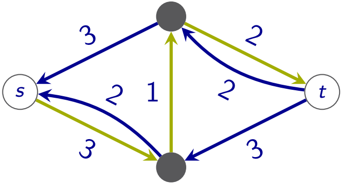
Residualgraph D_f

Universität Bremen

Augmentierende Wege



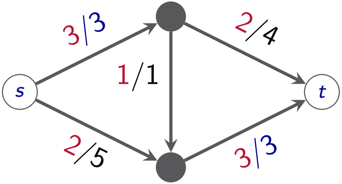
Digraph D mit Fluss f



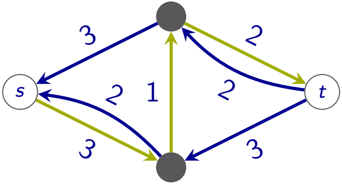
Residualgraph D_f

- f-augmentierender Weg P: s-t-Weg in D_f

Augmentierende Wege



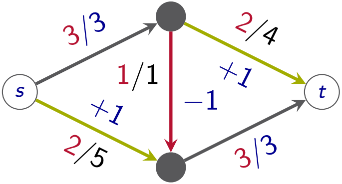
Digraph D mit Fluss f



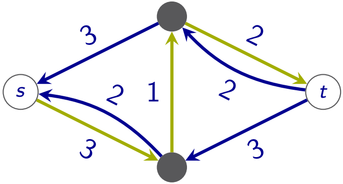
Residualgraph D_f

- f-augmentierender Weg P: s-t-Weg in D_f
- Bottleneck-Kapazität von P: $\gamma := \min_{a \in P} \bar{c}_f(a)$

Augmentierende Wege



Digraph D mit Fluss f

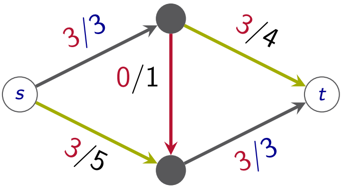


Residualgraph D_f

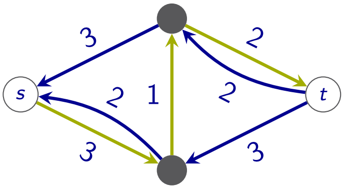
- f-augmentierender Weg P: s-t-Weg in D_f
- Bottleneck-Kapazität von P: $\gamma := \min_{a \in P} \bar{c}_f(a)$
- Erhöhung des Flusses f entlang P um γ ergibt Fluss f' in D:

$$f'(a) := \begin{cases} f(a) + \gamma & \text{falls } a \in P \\ f(a) - \gamma & \text{falls } \overleftarrow{a} \in P \\ f(a) & \text{sonst} \end{cases}$$

Augmentierende Wege



Digraph D mit Fluss f

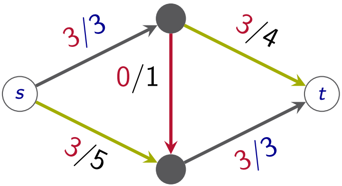


Residualgraph D_f

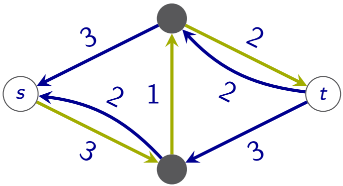
- f-augmentierender Weg P: s-t-Weg in D_f
- Bottleneck-Kapazität von P: $\gamma := \min_{a \in P} \bar{c}_f(a)$
- Erhöhung des Flusses f entlang P um γ ergibt Fluss f' in D:

$$f'(a) := \begin{cases} f(a) + \gamma & \text{falls } a \in P \\ f(a) - \gamma & \text{falls } \overleftarrow{a} \in P \\ f(a) & \text{sonst} \end{cases}$$

Augmentierende Wege



Digraph D mit Fluss f



Residualgraph D_f

- f-augmentierender Weg P: s-t-Weg in D_f
- Bottleneck-Kapazität von P: $\gamma := \min_{a \in P} \bar{c}_f(a)$
- Erhöhung des Flusses f entlang P um γ ergibt Fluss f' in D:

$$f'(a) := \begin{cases} f(a) + \gamma & \text{falls } a \in P \\ f(a) - \gamma & \text{falls } \overleftarrow{a} \in P \\ f(a) & \text{sonst} \end{cases}$$

Satz

f' ist ein s-t-Fluss in D mit Flusswert $val(f') = val(f) + \gamma$.

Optimialitätskriterium

Satz

Sei $N = (V, A, c, s, t)$ ein Netzwerk und f ein zulässiger s-t-Fluss. Dann gilt:

f maximal $\Leftrightarrow \nexists$ f-augmentierender s-t-Weg im Residualgraphen.

Beweis. (Bild an Tafel)
"$\Rightarrow$": klar, da ein f-augmentierender s-t-Weg den Flusswert erhöhen würde.
"$\Leftarrow$": Sei $U \subset V$ die Menge der Knoten, die im Residualgraphen von s aus über gerichtete Wege erreichbar. $\delta^+(U) = C$ ist ein s-t-Schnitt, denn $s \in U$ und $t \notin U$.

$$\Rightarrow val(f) = ex_f(s) \leq \text{cap}(C). \qquad (*)$$

- Für Kante $a = (x, y) \in \delta^+(U)$ in N gilt $f(a) = c(a)$ (sonst $y \in U$).
- Für Kante $a = (u, v) \in \delta^-(U)$ in N gilt $f(a) = 0$ (sonst $u \in U$).

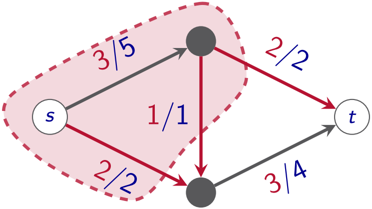
$$val(f) = ex_f(s) = \sum_{a \in \delta^+(U)} f(a) - \sum_{a \in \delta^-(U)} f(a) = \sum_{a \in \delta^+(U)} c(a) = \text{cap}(C)$$

Mit $(*)$ folgt, dass f maximal ist. $\square$

Max-Fluss Min-Schnitt Theorem

Satz (Max-Fluss Min-Schnitt Theorem)

Gegeben sei ein Netzwerk $N = (V, A, c, s, t)$ mit Kapazitäten $c(a) \geq 0$, $a \in A$. Dann ist der Wert eines maximalen s-t-Flusses gleich der minimalen s-t-Schnittkapazität.



max Fluss = min Schnitt

Beweis. Sei f ein maximaler Fluss. Dann gibt es nach vorigem Satz keinen f-augmentierenden s-t-Weg im Residualgraphen. Konstruiere Schnitt C wie im vorigen Satz. Dann $val(f) = \text{cap}(C)$.

Algorithmen für das Max-Fluss Problem

Augmentierende-Wege Algorithmus

Algorithmus (Ford & Fulkerson, 1957)

1. $f := 0$
2. Solange ein f-augmentierender Pfad P im Residualgraphen existiert, erhöhe Fluss f entlang P um $\min_{a \in P} \bar{c}_f(a)$.
3. Return f.

…Beispiel an Tafel.

Augmentierende-Wege Algorithmus

Algorithmus (Ford & Fulkerson, 1957)

1. $f := 0$
2. Solange ein f-augmentierender Pfad P im Residualgraphen existiert, erhöhe Fluss f entlang P um $\min_{a \in P} \bar{c}_f(a)$.
3. Return f.

Theorem

Wenn die Kantenkapazitäten ganzzahlig sind, dann berechnet der Algorithmus einen ganzzahligen maximalen s-t-Fluss in Laufzeit $\mathcal{O}(m \cdot M)$, wobei M der maximale Flusswert ist.

Beweisskizze.

- Fluss ist maximal gdw. es keinen augmentierenden Weg gibt.
- Algorithmus erhöht/verringert Fluss einer Kante um ein γ, welches für $c(a) \in \mathbb{Z}_+$ ganzzahlig ist. $\Rightarrow$ Fluss ganzzahlig
- In jeder Iteration ist $\gamma > 0$ und Flusswert wird strikt erhöht. $\Rightarrow$ Fluss ist maximal wenn Alg. terminiert.

Laufzeit Augmentierende-Wege Algorithmus

Laufzeit Ford-Fulkerson $\mathcal{O}(|A| \cdot M)$, mit M maximaler Flusswert

M kann gross sein, etwa $|A| \cdot c_{\max}$ mit $c_{\max} := \max_{a \in A} c(a)$. Das ist nur pseudo-polynomial in Inputgröße.

... Beispiel in Übung.

Satz (Edmonds und Karp, 1969)

Die Variante des Ford-Fulkerson Algorithmus, die immer entlang des kürzesten augmentierenden Weges (bzgl. Anzahl Kanten) Fluss erhöht, hat eine Laufzeit von $\mathcal{O}(m^2 \cdot n)$.

$=$ polynomielle Laufzeit

Ohne Beweis.

Polynomialzeitalgorithmen

Verschiedene Polynomialzeitalgorithmen für das max Flussproblem:

- **Edmonds-Karp algorithm**: $\mathcal{O}(m^2 \cdot n)$
  wähle immer den kürzesten augmentierenden Weg
- **Dinic's Algorithmus**: $\mathcal{O}(mn \log n)$
  augmentiere entlang eines "blockierenden Flusses"
- **Fujishige's Algorithmus**: $\mathcal{O}(mn \log c_{\max})$
  "capacity scaling'
- **Goldberg-Tarjans Algorithmus**: $\mathcal{O}(n^2 \sqrt{m})$
  Preflow-Push
- Es gibt zwei **LP Formulierungen**.
- Und andere Algorithmen ...

Weitere Netzwerkflussprobleme

Mehrere Quellen und Senken

- Gegeben sei ein Netzwerk mit mehreren Quellen $s_1, \dots, s_k$ und Senken $t_1, \dots, s_\ell$.
- Flusserhaltung bedeutet:

$$\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a), \qquad \forall v \in V \setminus \{s_1, \dots, s_k, t_1, \dots, t_\ell\}$$

- Der Wert eines Flusses ist $val(f) = \sum_{i=1}^{k} ex_f(s_i)$.
- Wie findet man einen Fluss mit maximalem Flusswert?

Lösung:

- Füge Super-Quelle s und Supersenke t hinzu.
- Verbinde Super-Quelle mit Quellen ($e_i = (s, s_i)$) und setze $c(e_i) = \infty$.
- Verbinde Senken mit Supersenke ($e_i = (t_i, t)$) und setze $c(e_i) = \infty$.
- Berechne maximalen s-t-Fluss

Zusammenfassung

- Netzwerke und maximale Flüsse in Netzwerken
- Residualgraphen und augmentierende Wege
- Ford-Fulkerson Algorithmus (pseudopolynomielle Laufzeit)
- Edmonds-Karp Algorithmus: spezielle Wege-Wahl, polynomiell
- Max-Fluss Min-Schnitt Theorem
  - Frage: Gegebenen max Fluss, wie findet man einen min Schnitt?