

28. Flüsse in Netzen

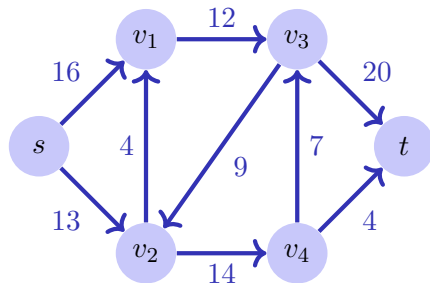
Flussnetzwerk, Maximaler Fluss, Schnitt, Restnetzwerk, Max-flow Min-cut Theorem, Ford-Fulkerson Methode, Edmonds-Karp Algorithmus, Maximales Bipartites Matching [Ottman/Widmayer, Kap. 9.7, 9.8.1], [Cormen et al, Kap. 26.1-26.3]

Motivation

- Modelliere Fluss von Flüssigkeiten, Bauteile auf Fließbändern, Strom in elektrischen Netzwerken oder Information in Kommunikationsnetzwerken.
- Konnektivität von Kommunikationsnetzwerken, Bipartites Matching, Zirkulationen, Scheduling, Image Segmentation, Baseball Elimination...

Flussnetzwerk

- **Flussnetzwerk** $G = (V, E, c)$: gerichteter Graph mit **Kapazitäten**
- Antiparallele Kanten verboten:
 $(u, v) \in E \Rightarrow (v, u) \notin E$.
- Fehlen einer Kante (u, v) auch modelliert durch $c(u, v) = 0$.
- **Quelle** s und **Senke** t : spezielle Knoten. Jeder Knoten v liegt auf einem Pfad zwischen s und t : $s \rightsquigarrow v \rightsquigarrow t$



Fluss

Ein **Fluss** $f : V \times V \rightarrow \mathbb{R}$ erfüllt folgende Bedingungen:

- **Kapazitätsbeschränkung:**

Für alle $u, v \in V$: $f(u, v) \leq c(u, v)$.

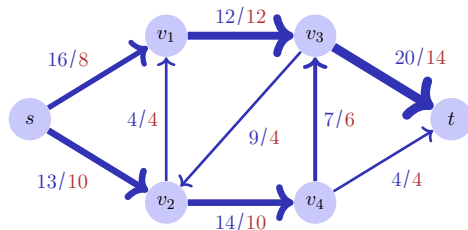
- **Schiefssymmetrie:**

Für alle $u, v \in V$: $f(u, v) = -f(v, u)$.

- **Flusserhaltung:**

Für alle $u \in V \setminus \{s, t\}$:

$$\sum_{v \in V} f(u, v) = 0.$$



Wert w des Flusses:

$$|f| = \sum_{v \in V} f(s, v).$$

Hier $|f| = 18$.

Wie gross kann ein Fluss sein?

Begrenzende Faktoren: Schnitte

- **s von t trennender Schnitt:** Partitionierung von V in S und T mit $s \in S$, $t \in T$.
- **Kapazität** eines Schnittes: $c(S, T) = \sum_{v \in S, v' \in T} c(v, v')$
- **Minimaler Schnitt:** Schnitt mit minimaler Kapazität.
- **Fluss über Schnitt:** $f(S, T) = \sum_{v \in S, v' \in T} f(v, v')$

Implizites Summieren

Notation: Seien $U, U' \subseteq V$

$$f(U, U') := \sum_{\substack{u \in U \\ u' \in U'}} f(u, u'), \quad f(u, U') := f(\{u\}, U')$$

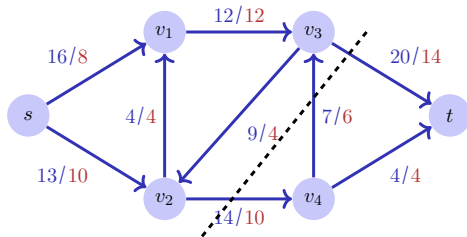
Somit

- $|f| = f(s, V)$
- $f(U, U) = 0$
- $f(U, U') = -f(U', U)$
- $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$, wenn $X \cap Y = \emptyset$.
- $f(R, V) = 0$ wenn $R \cap \{s, t\} = \emptyset$. [Flusserhaltung!]

Wie gross kann ein Fluss sein?

Es gilt für jeden Fluss und jeden Schnitt, dass $f(S, T) = |f|$:

$$\begin{aligned} f(S, T) &= f(S, V) - \underbrace{f(S, S)}_0 = f(S, V) \\ &= f(s, V) + \underbrace{f(S - \{s\}, V)}_{\not\ni t, \not\ni s} = |f|. \end{aligned}$$

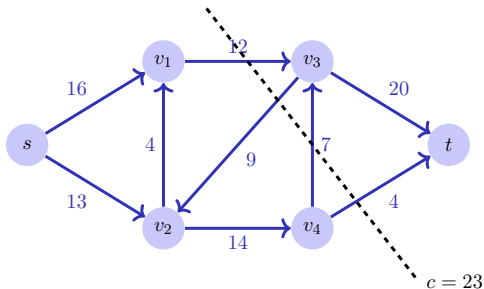


Maximaler Fluss ?

Es gilt insbesondere für alle Schnitte (S, T) von V .

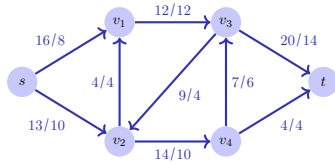
$$|f| \leq \sum_{v \in S, v' \in T} c(v, v') = c(S, T)$$

Werden sehen, dass Gleichheit gilt für $\min_{S, T} c(S, T)$.



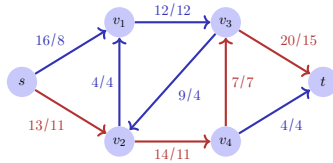
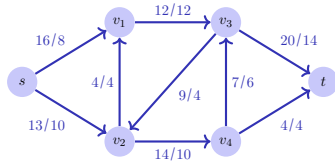
Maximaler Fluss ?

Naives Vorgehen:



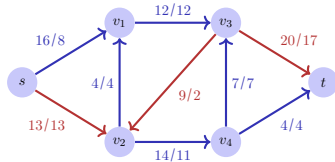
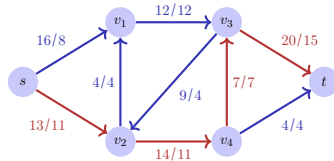
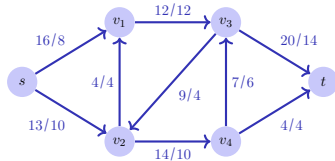
Maximaler Fluss ?

Naives Vorgehen:



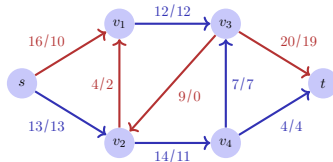
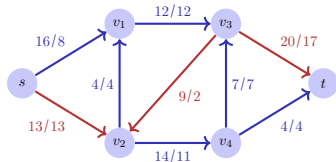
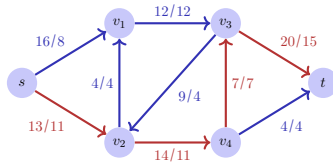
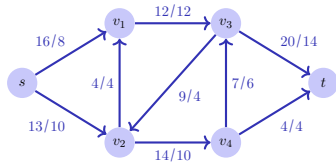
Maximaler Fluss ?

Naives Vorgehen:



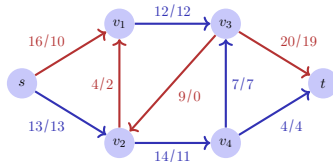
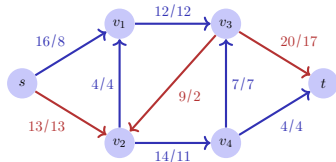
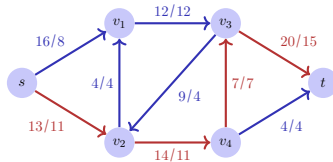
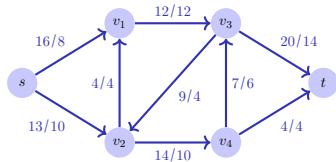
Maximaler Fluss ?

Naives Vorgehen:



Maximaler Fluss ?

Naives Vorgehen:



Folgerung: Greedy Flusserhöhung löst das Problem nicht.

Die Ford-Fulkerson Methode

- Starte mit $f(u, v) = 0$ für alle $u, v \in V$
- Bestimme Restnetzwerk* G_f und Erweiterungspfad in G_f
- Erhöhe Fluss über den Erweiterungspfad*
- Wiederholung bis kein Erweiterungspfad mehr vorhanden.

$$\begin{aligned}G_f &:= (V, E_f, c_f) \\c_f(u, v) &:= c(u, v) - f(u, v) \quad \forall u, v \in V \\E_f &:= \{(u, v) \in V \times V \mid c_f(u, v) > 0\}\end{aligned}$$

*Wird im Folgenden erklärt

Flusserhöhung, negativ

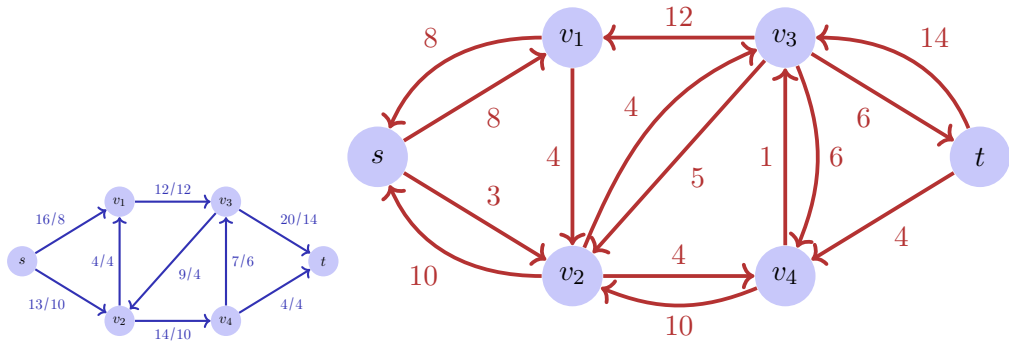
Sei ein Fluss f im Netzwerk gegeben.

Erkenntnis:

- Flusserhöhung in Richtung einer Kante möglich, wenn Fluss entlang der Kante erhöht werden kann, also wenn $f(u, v) < c(u, v)$.
Restkapazität $c_f(u, v) = c(u, v) - f(u, v) > 0$.
- Flusserhöhung **entgegen** der Kantenrichtung möglich, wenn Fluss entlang der Kante verringert werden kann, also wenn $f(u, v) > 0$.
Restkapazität $c_f(v, u) = f(u, v) > 0$.

Restnetzwerk

Restnetzwerk G_f gegeben durch alle Kanten mit Restkapazität:



Restnetzwerke haben dieselben Eigenschaften wie Flussnetzwerke, ausser dass antiparallele Kapazitäten-Kanten zugelassen sind.

Theorem 33

Sei $G = (V, E, c)$ ein Flussnetzwerk mit Quelle s und Senke t und f ein Fluss in G . Sei G_f das dazugehörige Restnetzwerk und sei f' ein Fluss in G_f . Dann definiert $f \oplus f'$ mit

$$(f \oplus f')(u, v) = f(u, v) + f'(u, v)$$

einen Fluss in G mit Wert $|f| + |f'|$.

Beweis

$f \oplus f'$ ist ein Fluss in G :

■ Kapazitätsbeschränkung

$$(f \oplus f')(u, v) = f(u, v) + \underbrace{f'(u, v)}_{\leq c(u, v) - f(u, v)} \leq c(u, v)$$

■ Schiefsymmetrie

$$(f \oplus f')(u, v) = -f(v, u) + -f'(v, u) = -(f \oplus f')(v, u)$$

■ Flusserhaltung $u \in V - \{s, t\}$:

$$\sum_{v \in V} (f \oplus f')(u, v) = \sum_{v \in V} f(u, v) + \sum_{v \in V} f'(u, v) = 0$$

Wert von $f \oplus f'$

$$\begin{aligned} |f \oplus f'| &= (f \oplus f')(s, V) \\ &= \sum_{u \in V} f(s, u) + f'(s, u) \\ &= f(s, V) + f'(s, V) \\ &= |f| + |f'| \end{aligned}$$



Erweiterungspfade

Erweiterungspfad p : einfacher Pfad von s nach t im Restnetzwerk G_f .

Restkapazität $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ Kante in } p\}$

Theorem 34

Die Funktion $f_p : V \times V \rightarrow \mathbb{R}$,

$$f_p(u, v) = \begin{cases} c_f(p) & \text{wenn } (u, v) \text{ Kante in } p \\ -c_f(p) & \text{wenn } (v, u) \text{ Kante in } p \\ 0 & \text{sonst} \end{cases}$$

ist ein Fluss in G_f mit dem Wert $|f_p| = c_f(p) > 0$.

f_p ist ein Fluss (leicht nachprüfbar). Es gibt genau einen Knoten $u \in V$ mit $(s, u) \in p$. Somit $|f_p| = \sum_{v \in V} f_p(s, v) = f_p(s, u) = c_f(p)$.

Strategie für den Algorithmus:

Mit einem Erweiterungspfad p in G_f definiert $f \oplus f_p$ einen neuen Fluss mit Wert $|f \oplus f_p| = |f| + |f_p| > |f|$.

Max-Flow Min-Cut Theorem

Theorem 35

Wenn f ein Fluss in einem Flussnetzwerk $G = (V, E, c)$ mit Quelle s und Senke t ist, dann sind folgende Aussagen äquivalent:

1. f ist ein maximaler Fluss in G
2. Das Restnetzwerk G_f enthält keine Erweiterungspfade
3. Es gilt $|f| = c(S, T)$ für einen Schnitt (S, T) von G .

■ (3) \Rightarrow (1):

Es gilt $|f| \leq c(S, T)$ für alle Schnitte S, T . Aus $|f| = c(S, T)$ folgt also $|f|$ maximal.

■ (1) \Rightarrow (2):

f maximaler Fluss in G . Annahme: G_f habe einen Erweiterungsfad. Dann gilt $|f \oplus f_p| = |f| + |f_p| > |f|$. Widerspruch.

Beweis (2) \Rightarrow (3)

Annahme: G_f habe keinen Erweiterungsfad

Definiere $S = \{v \in V : \text{es existiert Pfad } s \rightsquigarrow v \text{ in } G_f\}$.

$(S, T) := (S, V \setminus S)$ ist ein Schnitt: $s \in S, t \in T$.

Sei $u \in S$ und $v \in T$. Dann $c_f(u, v) = 0$, also $c_f(u, v) = c(u, v) - f(u, v) = 0$.

Somit $f(u, v) = c(u, v)$.

Somit

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



Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$

Output: Maximaler Fluss f .

for $(u, v) \in E$ **do**

└ $f(u, v) \leftarrow 0$

while Existiert Pfad $p : s \rightsquigarrow t$ im Restnetzwerk G_f **do**

└ $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \in p\}$

└ **foreach** $(u, v) \in p$ **do**

└└ $f(u, v) \leftarrow f(u, v) + c_f(p)$

└└ $f(v, u) \leftarrow f(v, u) - c_f(p)$

Praktische Anmerkung

In einer Implementation des Ford-Fulkerson Algorithmus werden die negativen Flusskanten normalerweise nicht gespeichert, da ihr Wert sich stets als der negierter Wert der Gegenkante ergibt.

$$f(u, v) \leftarrow f(u, v) + c_f(p)$$

$$f(v, u) \leftarrow f(v, u) - c_f(p)$$

wird dann zu

if $(u, v) \in E$ **then**

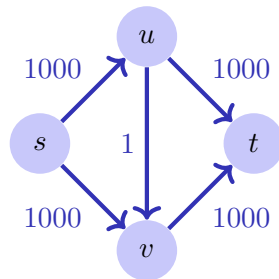
$$\quad | \quad f(u, v) \leftarrow f(u, v) + c_f(p)$$

else

$$\quad | \quad f(v, u) \leftarrow f(v, u) - c_f(p)$$

Analyse

- Der Ford-Fulkerson Algorithmus muss für irrationale Kapazitäten nicht einmal terminieren! Für ganze oder rationale Zahlen terminiert der Algorithmus.
- Für ganzzahligen Fluss benötigt der Algorithmus maximal $|f_{\max}|$ Durchläufe der While-Schleife (denn der Fluss erhöht sich mindestens um 1). Suche einzelner zunehmender Weg (z.B. Tiefensuche oder Breitensuche) $\mathcal{O}(|E|)$. Also $\mathcal{O}(f_{\max}|E|)$.



Bei schlecht gewählter Strategie benötigt der Algorithmus hier bis zu 2000 Iterationen.

Edmonds-Karp Algorithmus

Wähle in der Ford-Fulkerson-Methode zum Finden eines Pfades in G_f jeweils einen Erweiterungspfad kürzester Länge (z.B. durch Breitensuche).

Edmonds-Karp Algorithmus

Theorem 36

Wenn der Edmonds-Karp Algorithmus auf ein ganzzahliges Flussnetzwerk $G = (V, E)$ mit Quelle s und Senke t angewendet wird, dann ist die Gesamtanzahl der durch den Algorithmus angewendete Flusserhöhungen in $\mathcal{O}(|V| \cdot |E|)$.

\Rightarrow Gesamte asymptotische Laufzeit: $\mathcal{O}(|V| \cdot |E|^2)$

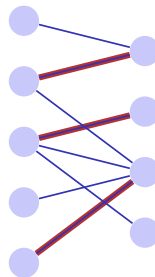
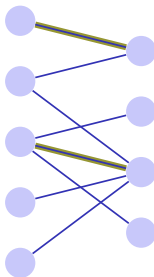
[Ohne Beweis]

Anwendung: Maximales bipartites Matching

Gegeben: bipartiter ungerichteter Graph $G = (V, E)$.

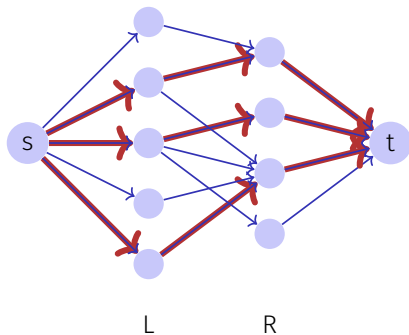
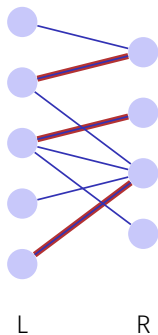
Matching M : $M \subseteq E$ so dass $|\{m \in M : v \in m\}| \leq 1$ für alle $v \in V$.

Maximales Matching M : Matching M , so dass $|M| \geq |M'|$ für jedes Matching M' .



Korrespondierendes Flussnetzwerk

Konstruiere zur einer Partition L, R eines bipartiten Graphen ein korrespondierendes Flussnetzwerk mit Quelle s und Senke t , mit gerichteten Kanten von s nach L , von L nach R und von R nach t . Jede Kante bekommt Kapazität 1.



Ganzzahligkeitstheorem

Theorem 37

Wenn die Kapazitäten eines Flussnetzwerks nur ganzzahlige Werte annehmen, dann hat der durch Ford-Fulkerson erzeugte maximale Fluss die Eigenschaft, dass der Wert von $f(u, v)$ für alle $u, v \in V$ eine ganze Zahl ist.

[ohne Beweis]

Folgerung: Ford Fulkerson erzeugt beim zum bipartiten Graph gehörenden Flussnetzwerk ein maximales Matching $M = \{(u, v) : f(u, v) = 1\}$.