Netzwerk- und Flussprobleme

Wie viel passt durch die Leitung?

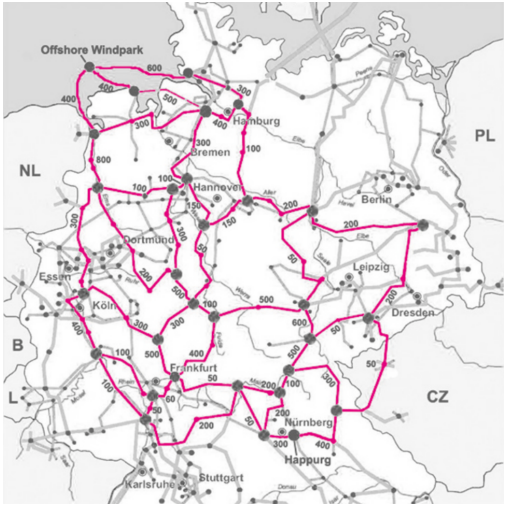
Georg Mix Jan Jakob

Universität Heidelberg

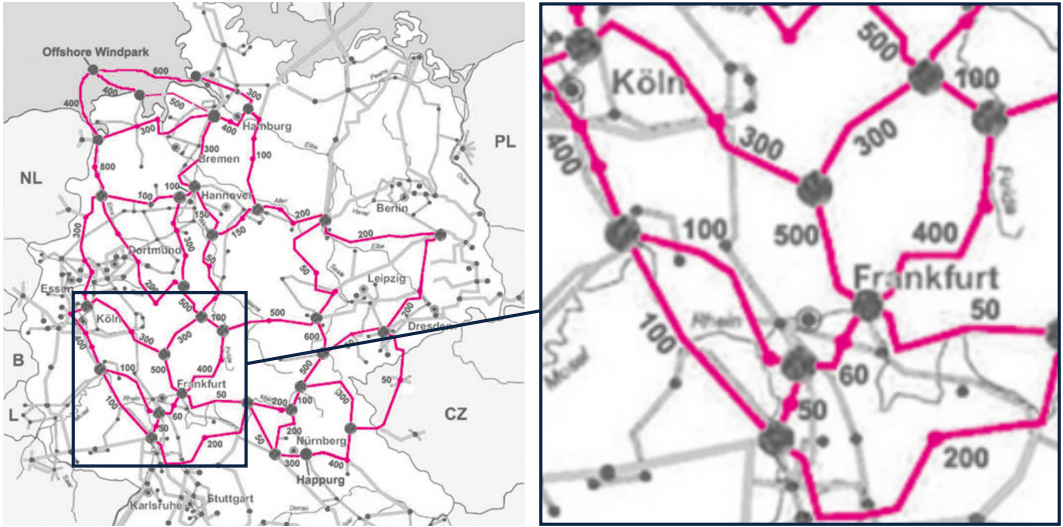
30. Mai 2021

Motivation - Anwendungsbeispiel

Motivation - Energietransport

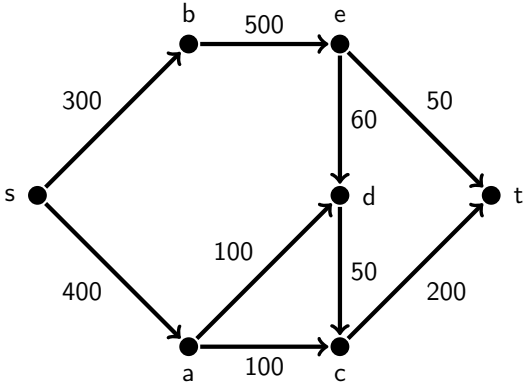


Motivation - Energietransport

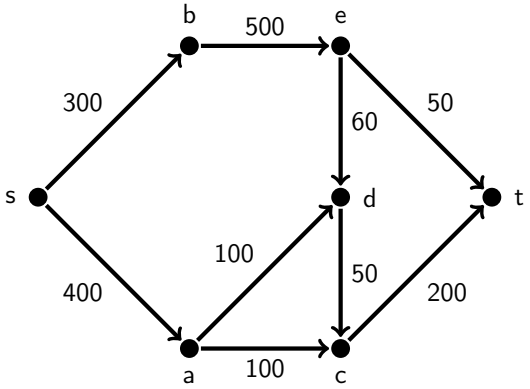


Motivation - Energietransport

Motivation - Energietransport

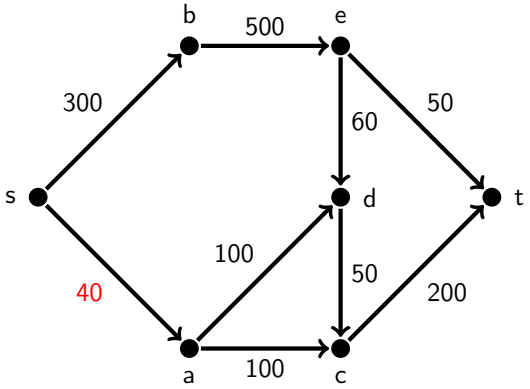


Motivation - Energietransport



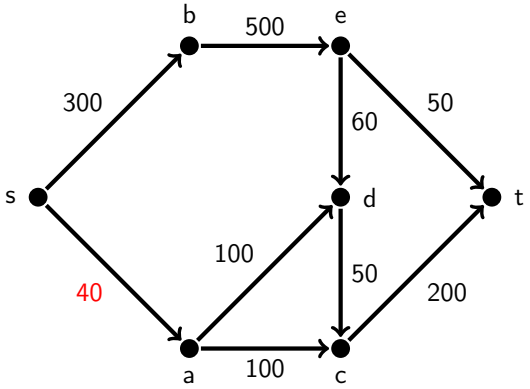
- Wir betrachten die Hochspannungsleitungen zwischen Köln (s) und Schweinfurt (t).
- Auf der Hochspannungsleitung zwischen Köln und Koblenz (s, a) liegt eine Störung vor, sodass dort nur noch 40 kV anstatt von 400 kV.
- Wir wollen wissen wie viel Strom wir von Köln nach Schweinfurt transportieren können.

Motivation - Energietransport



- Wir betrachten die Hochspannungsleitungen zwischen Köln (s) und Schweinfurt (t).
- Auf der Hochspannungsleitung zwischen Köln und Koblenz (s, a) liegt eine Störung vor, sodass dort nur noch 40 kV anstatt von 400 kV.
- Wir wollen wissen wie viel Strom wir von Köln nach Schweinfurt transportieren können.

Motivation - Energietransport



- Wir betrachten die Hochspannungsleitungen zwischen Köln (s) und Schweinfurt (t).
- Auf der Hochspannungsleitung zwischen Köln und Koblenz (s, a) liegt eine Störung vor, sodass dort nur noch 40 kV anstatt von 400 kV.
- Wir wollen wissen wie viel Strom wir von Köln nach Schweinfurt transportieren können.

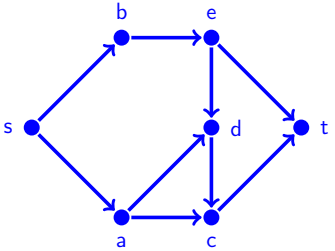
Die Begriffe Netzwerk und Fluss

Definition - Netzwerk

Definition - Netzwerk

Ein **s-t-Netzwerk** N ist ein Tupel $N = (G, c, s, t)$ bestehend aus:

- einem **gerichteten Graphen** G = (V, E, α, ω) mit $\alpha : E \to V$ Anfangsknoten und $\omega : E \to V$ Endknoten einer Kante
- $c : E \to \mathbb{R}_+$, einer **Kapazitätsfunktion** auf den Kanten mit nicht negativen Werten
- $s, t \in V$, zwei ausgezeichneten Knoten, der **Quelle** s und der **Senke** t mit $s \neq t$.

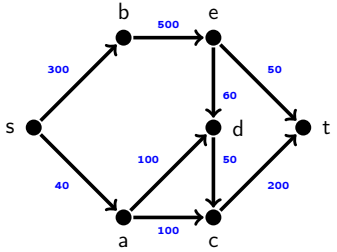


Definition - Netzwerk

Definition - Netzwerk

Ein **s-t-Netzwerk** N ist ein Tupel $N = (G, c, s, t)$ bestehend aus:

- einem **gerichteten Graphen** G = (V, E, α, ω) mit $\alpha : E \to V$ Anfangsknoten und $\omega : E \to V$ Endknoten einer Kante
- $c : E \to \mathbb{R}_+$, einer **Kapazitätsfunktion** auf den Kanten mit nicht negativen Werten
- $s, t \in V$, zwei ausgezeichneten Knoten, der **Quelle** s und der **Senke** t mit $s \neq t$.

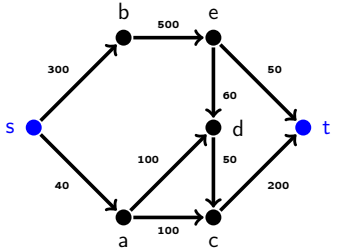


Definition - Netzwerk

Definition - Netzwerk

Ein **s-t-Netzwerk** N ist ein Tupel $N = (G, c, s, t)$ bestehend aus:

- einem **gerichteten Graphen** G = (V, E, α, ω) mit $\alpha : E \to V$ Anfangsknoten und $\omega : E \to V$ Endknoten einer Kante
- $c : E \to \mathbb{R}_+$, einer **Kapazitätsfunktion** auf den Kanten mit nicht negativen Werten
- $s, t \in V$, zwei ausgezeichneten Knoten, der **Quelle s** und der **Senke t** mit $s \neq t$.



Definition - Fluss

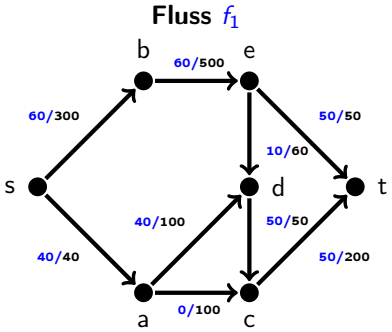
Definition - s-t-Fluss

Ein **s-t-Fluss** ist eine Funktion $f : E \to \mathbb{R}$, die jeder Kante eines Netzwerks N eine reelle Zahl zuordnet und folgende Bedingungen erfüllt:

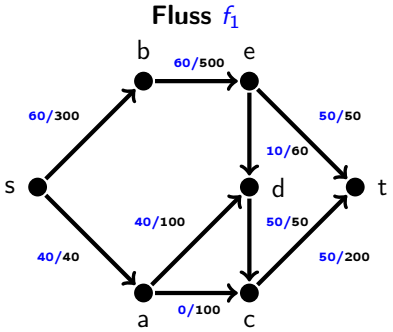
1. $0 \leq f(e) \qquad \forall e \in E$ (**Nichtnegativität**)
2. $f(e) \leq c(e) \quad \forall e \in E$ (**Kapazitätsbeschränkung**)
3. $f^+(v) = f^-(v) \qquad \forall v \in V \setminus \{s, t\}$ (**Erhaltungssatz**)

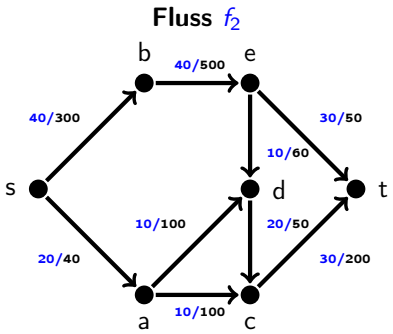
Dabei ist $f^+(v) = \sum_{e \in \delta^+(v)} f(e)$; $\delta^+(v) = \{e \in E : \alpha(e) = v, \omega(e) \neq v\}$ die **Flussmenge**, die aus einem Knoten $v \in V$ herausfließt und $f^-(v) = \sum_{e \in \delta^-(v)} f(e)$; $\delta^-(v) = \{e \in E : \alpha(e) \neq v, \omega(e) = v\}$ die Flussmenge, die hineinfließt.

Definition - Fluss

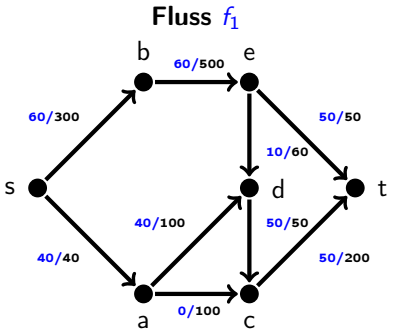


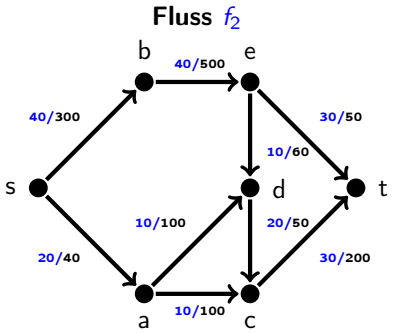
Definition - Fluss





Definition - Fluss





Definition - Wert eines s-t-Flusses

Der **Wert** $\mathcal{W}_f$ eines s-t-Flusses f ist definiert als $\mathcal{W}_f := f^+(s) - f^-(s)$. Dies entspricht gerade dem Gesamtfluss aus der Quelle s heraus. Außerdem gilt nach der Definition von f: $\mathcal{W}_f = f^-(t) - f^+(t)$

Definition - Problem des größten Flusses

Definition - maximaler s-t-Fluss

Ein Fluss f^* heißt **maximaler s-t-Fluss**, wenn er den maximalen Wert unter allen s-t-Flüssen besitzt: $\mathcal{W}_{f^*} = \max\{\mathcal{W}_f \mid f \text{ ist s-t-Fluss}\}$

Definition - Problem des größten Flusses

Die Aufgabe, zu einem Netzwerk $N = (G, c, s, t)$ mit $n = |V|$ Knoten und $m = |E|$ Kanten einen maximalen s-t-Fluß $f^* \in \mathbb{R}^m$ zu finden, heißt **Problem des größten Flusses (max flow problem)**

Existenz - Problem des größten Flusses

Existiert zu einem gegebenen Netzwerk N überhaupt ein maximaler s-t-Fluss, hat das max flow problem also stets eine maximale Lösung? -> JA, Beweis mittels Stetigkeitsaussagen!

Definition - Problem des größten Flusses

Definition - maximaler s-t-Fluss

Ein Fluss f^* heißt **maximaler s-t-Fluss**, wenn er den maximalen Wert unter allen s-t-Flüssen besitzt: $\mathcal{W}_{f^*} = \max\{\mathcal{W}_f \mid f \text{ ist s-t-Fluss}\}$

Definition - Problem des größten Flusses

Die Aufgabe, zu einem Netzwerk $N = (G, c, s, t)$ mit $n = |V|$ Knoten und $m = |E|$ Kanten einen maximalen s-t-Fluß $f^* \in \mathbb{R}^m$ zu finden, heißt **Problem des größten Flusses (max flow problem)**

Existenz - Problem des größten Flusses

Existiert zu einem gegebenen Netzwerk N überhaupt ein maximaler s-t-Fluss, hat das max flow problem also stets eine maximale Lösung? -> JA, Beweis mittels Stetigkeitsaussagen!

Definition - Problem des größten Flusses

Definition - maximaler s-t-Fluss

Ein Fluss f^* heißt **maximaler s-t-Fluss**, wenn er den maximalen Wert unter allen s-t-Flüssen besitzt: $\mathcal{W}_{f^*} = \max\{\mathcal{W}_f \mid f \text{ ist s-t-Fluss}\}$

Definition - Problem des größten Flusses

Die Aufgabe, zu einem Netzwerk $N = (G, c, s, t)$ mit $n = |V|$ Knoten und $m = |E|$ Kanten einen maximalen s-t-Fluß $f^* \in \mathbb{R}^m$ zu finden, heißt **Problem des größten Flusses (max flow problem)**

Existenz - Problem des größten Flusses

Existiert zu einem gegebenen Netzwerk N überhaupt ein maximaler s-t-Fluss, hat das max flow problem also stets eine maximale Lösung? -> JA, Beweis mittels Stetigkeitsaussagen!

Definition - Problem des größten Flusses

Definition - maximaler s-t-Fluss

Ein Fluss f^* heißt **maximaler s-t-Fluss**, wenn er den maximalen Wert unter allen s-t-Flüssen besitzt: $\mathcal{W}_{f^*} = \max\{\mathcal{W}_f \mid f \text{ ist s-t-Fluss}\}$

Definition - Problem des größten Flusses

Die Aufgabe, zu einem Netzwerk $N = (G, c, s, t)$ mit $n = |V|$ Knoten und $m = |E|$ Kanten einen maximalen s-t-Fluß $f^* \in \mathbb{R}^m$ zu finden, heißt **Problem des größten Flusses (max flow problem)**

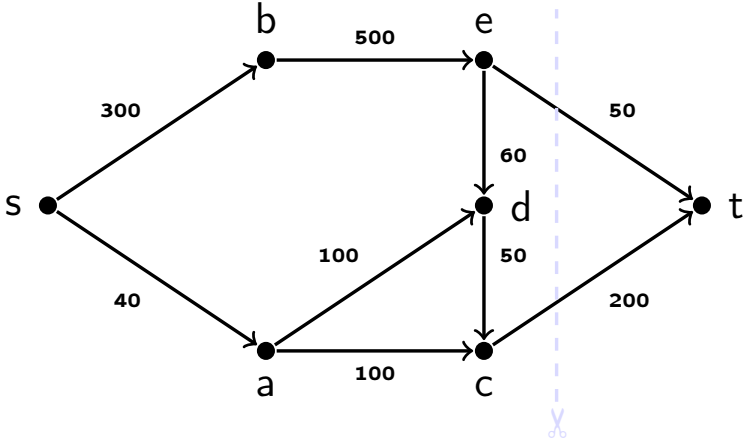
Existenz - Problem des größten Flusses

Existiert zu einem gegebenen Netzwerk N überhaupt ein maximaler s-t-Fluss, hat das max flow problem also stets eine maximale Lösung? -> JA, Beweis mittels Stetigkeitsaussagen!

Bestimmung maximaler Flüsse - Theoretische Betrachtungen

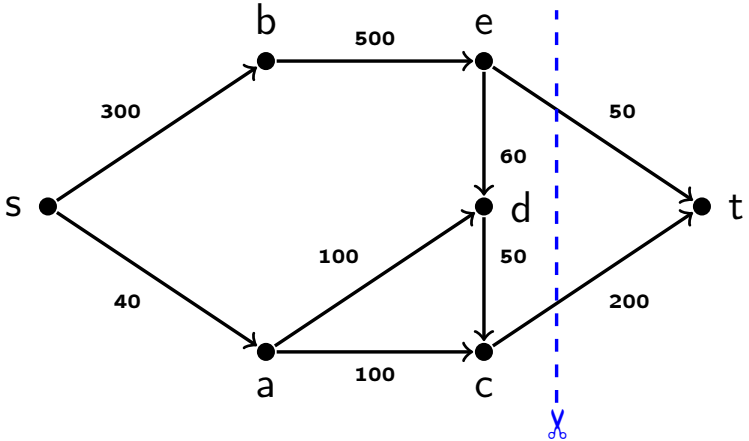
Anwendungsbeispiel

Was ist eine (möglichst kleine) obere Schranke des Gesamtflusses?



Anwendungsbeispiel

Was ist eine (möglichst kleine) obere Schranke des Gesamtflusses?



Netzwerkschnitte

Definition - Schnitt

Ein **Schnitt** (S, T) in einem Graphen $G = (V, E, \alpha, \omega)$ teilt die Knoten in zwei Partitionen $S, T \subset V$.
i.e. $V = S \cup T$, $S \cap T = \emptyset$ und $S, T \neq \emptyset$

Definition - s-t-Schnitt

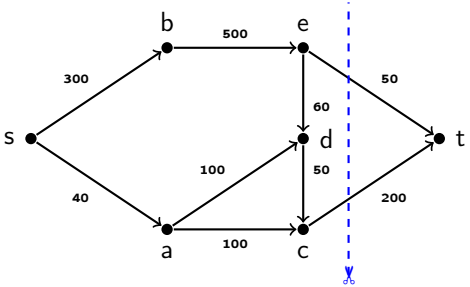
Ein Schnitt (S, T) mit $s \in S$ und $t \in T$.

Definition - Kapazität des Schnittes (S, T)

$$c(S, T) := \sum_{e \in (S,T)} c(e)$$

In unserem Beispiel:

$S = \{s, a, b, c, d, e\}, \; T = \{t\}$



$c(S, T) = 200 + 50$

Netzwerkschnitte

Definition - Schnitt

Ein **Schnitt** (S, T) in einem Graphen $G = (V, E, \alpha, \omega)$ teilt die Knoten in zwei Partitionen $S, T \subset V$.
i.e. $V = S \cup T$, $S \cap T = \emptyset$ und $S, T \neq \emptyset$

Definition - s-t-Schnitt

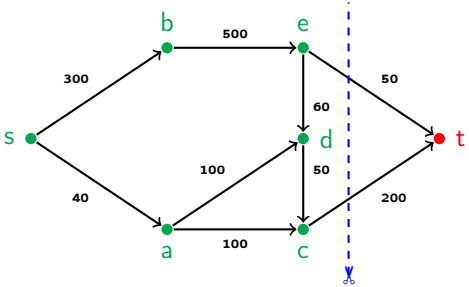
Ein Schnitt (S, T) mit $s \in S$ und $t \in T$.

Definition - Kapazität des Schnittes (S, T)

$$c(S, T) := \sum_{e \in (S,T)} c(e)$$

In unserem Beispiel:

$S = \{s, a, b, c, d, e\}$, $T = \{t\}$



$c(S, T) = 200 + 50$

Netzwerkschnitte

Definition - Schnitt

Ein **Schnitt** (S, T) in einem Graphen $G = (V, E, \alpha, \omega)$ teilt die Knoten in zwei Partitionen $S, T \subset V$.
i.e. $V = S \cup T$, $S \cap T = \emptyset$ und $S, T \neq \emptyset$

Definition - s-t-Schnitt

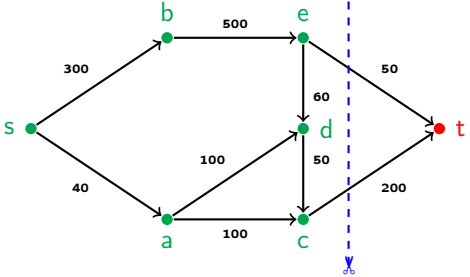
Ein Schnitt (S, T) mit $s \in S$ und $t \in T$.

Definition - Kapazität des Schnittes (S, T)

$$c(S, T) := \sum_{e \in (S,T)} c(e)$$

In unserem Beispiel:

$S = \{s, a, b, c, d, e\}$, $T = \{t\}$



$c(S, T) = 200 + 50$

Netzwerkschnitte

Definition - Schnitt

Ein **Schnitt** (S, T) in einem Graphen $G = (V, E, \alpha, \omega)$ teilt die Knoten in zwei Partitionen $S, T \subset V$.
i.e. $V = S \cup T$, $S \cap T = \emptyset$ und $S, T \neq \emptyset$

Definition - s-t-Schnitt

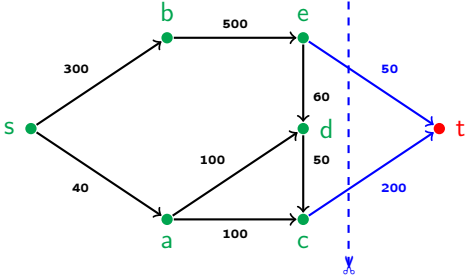
Ein Schnitt (S, T) mit $s \in S$ und $t \in T$.

Definition - Kapazität des Schnittes (S, T)

$$c(S, T) := \sum_{e \in (S,T)} c(e)$$

In unserem Beispiel:

$S = \{s, a, b, c, d, e\}$, $T = \{t\}$



$c(S, T) = 200 + 50$

Netzwerkschnitte

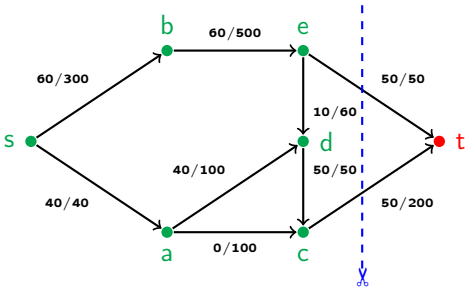
- **Nettofluss über die Knoten** eines Schnittes:

$$f^+(S) - f^-(S) := \sum_{v \in S} f^+(v) - f^-(v)$$

- **Nettofluss über die Kanten** des Schnittes:

$$f(S,T) - f(T,S) := \sum_{e \in (S,T)} f(e) - \sum_{e \in (T,S)} f(e)$$

In unserem Beispiel:



Netzwerkschnitte

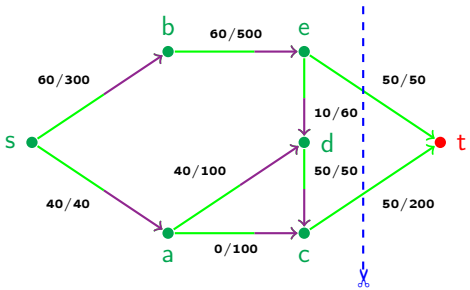
- **Nettofluss über die Knoten** eines Schnittes:

$$f^+(S) - f^-(S) := \sum_{v \in S} f^+(v) - f^-(v)$$

- **Nettofluss über die Kanten** des Schnittes:

$$f(S,T) - f(T,S) := \sum_{e \in (S,T)} f(e) - \sum_{e \in (T,S)} f(e)$$

In unserem Beispiel:



Netzwerkschnitte

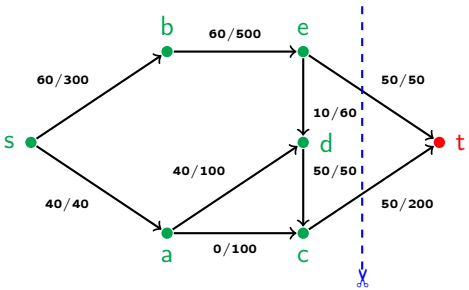
- **Nettofluss über die Knoten** eines Schnittes:

$$f^+(S) - f^-(S) := \sum_{v \in S} f^+(v) - f^-(v)$$

- **Nettofluss über die Kanten** des Schnittes:

$$f(S,T) - f(T,S) := \sum_{e \in (S,T)} f(e) - \sum_{e \in (T,S)} f(e)$$

In unserem Beispiel:



Netzwerkschnitte

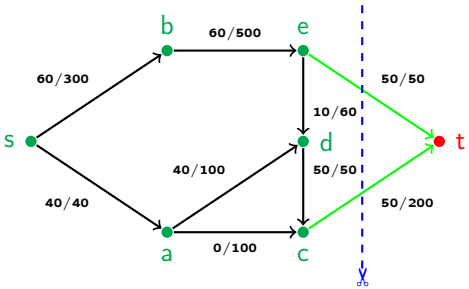
- **Nettofluss über die Knoten** eines Schnittes:

$$f^+(S) - f^-(S) := \sum_{v \in S} f^+(v) - f^-(v)$$

- **Nettofluss über die Kanten** des Schnittes:

$$f(S,T) - f(T,S) := \sum_{e \in (S,T)} f(e) - \sum_{e \in (T,S)} f(e)$$

In unserem Beispiel:



Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

Beweis:

$$\begin{aligned}
\mathcal{W}_f = f^-(t) - f^+(t) &= f^+(s) - f^-(s) && \text{(Einfluss = Ausfluss)} \\
&= \sum_{v \in S} f^+(v) - f^-(v) && \text{(Erhaltungssatz)} \\
&= \sum_{e \in (S,T)} f(e) + \sum_{e \in (S,S)} f(e) - \sum_{e \in (S,S)} f(e) - \sum_{e \in (T,S)} f(e) \\
&= f(S,T) - f(T,S)
\end{aligned}$$

Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

- Nettofluss über die Knoten/Kanten eines Schnitts ist kleiner als die Kapazität des Schnitts

$$f^+(S) - f^-(S) = f(S,T) - f(T,S) \leq c(S,T)$$

Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

- Nettofluss über die Knoten/Kanten eines Schnitts ist kleiner als die Kapazität des Schnitts

$$f^+(S) - f^-(S) = \underbrace{f(S,T)}_{\leq c(S,T)} - \underbrace{f(T,S)}_{\geq 0} \leq c(S,T)$$

Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

- Nettofluss über die Knoten/Kanten eines Schnitts ist kleiner als die Kapazität des Schnitts

$$f^+(S) - f^-(S) = f(S,T) - f(T,S) \leq c(S,T)$$

- Beschränktheit des Flusses

$$\max(\mathcal{W}_f) \leq \min(c(S,T))$$

Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

- Nettofluss über die Knoten/Kanten eines Schnitts ist kleiner als die Kapazität des Schnitts

$$f^+(S) - f^-(S) = f(S,T) - f(T,S) \leq c(S,T)$$

- Beschränktheit des Flusses

$$\max(\mathcal{W}_f) \leq \min(c(S,T))$$

Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

- Nettofluss über die Knoten/Kanten eines Schnitts ist kleiner als die Kapazität des Schnitts

$$f^+(S) - f^-(S) = f(S,T) - f(T,S) \leq c(S,T)$$

- Beschränktheit des Flusses

$$\max(\mathcal{W}_f) \leq \min(c(S,T))$$

Lemma

Sei f ein zulässiger s-t-Fluss und (S,T) ein s-t-Schnitt, mit

$$\mathcal{W}_f = c(S,T)$$

Dann ist f ein maximaler s-t-Fluss und (S,T) ein minimaler s-t-Schnitt.

Eigenschaften von s-t-Schnitten

- Gesamtfluss ist gleich dem Nettofluss über die Knoten/Kanten eines Schnitts

$$\mathcal{W}_f = f^+(S) - f^-(S) = f(S,T) - f(T,S)$$

- Nettofluss über die Knoten/Kanten eines Schnitts ist kleiner als die Kapazität des Schnitts

$$f^+(S) - f^-(S) = f(S,T) - f(T,S) \leq c(S,T)$$

- Beschränktheit des Flusses

$$\max(\mathcal{W}_f) \leq \min(c(S,T))$$

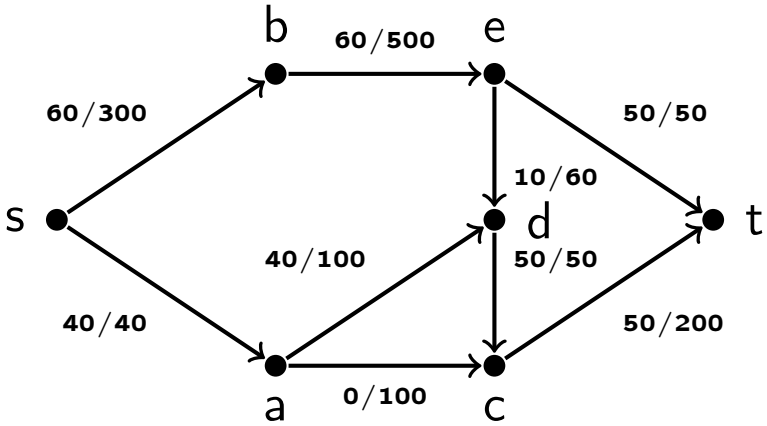
Lemma

Sei f ein zulässiger s-t-Fluss und (S,T) ein s-t-Schnitt, mit

$$\mathcal{W}_f = c(S,T)$$

Dann ist f ein maximaler s-t-Fluss und (S,T) ein minimaler s-t-Schnitt.

Ist der Fluss maximal?



Residualnetzwerk

Definition - Residualnetzwerk

Für einen s-t-Fluss f im Netzwerk $N = (G, c, s, t)$ definieren wir das **Residualnetzwerk** $N_f := (G_f, c_f, s, t)$ mit **Residualgraph** $G_f = (V, E_f, \alpha', \omega')$ und **Residualkapazität** $c_f : E_f \to \mathbb{R}_{\geq 0}$ wie folgt:

- Für alle $e \in E$ mit $f(e) < c(e)$ existiert $+e \in E_f$ mit $\alpha'(+e) = \alpha(e)$, $\omega'(+e) = \omega(e)$ und $c_f(+e) = c(e) - f(e)$
- Für alle $e \in E$ mit $f(e) > 0$ existiert $-e \in E_f$ mit $\alpha'(-e) = \omega(e)$, $\omega'(-e) = \alpha(e)$ und $c_f(-e) = f(e)$

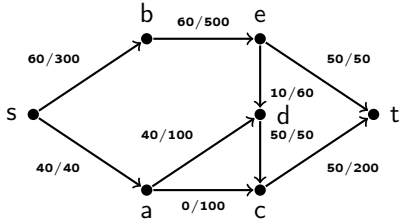
Residualnetzwerk

Definition - Residualnetzwerk

Für einen s-t-Fluss f im Netzwerk $N = (G, c, s, t)$ definieren wir das **Residualnetzwerk** $N_f := (G_f, c_f, s, t)$ mit **Residualgraph** $G_f = (V, E_f, \alpha', \omega')$ und **Residualkapazität** $c_f : E_f \to \mathbb{R}_{\geq 0}$ wie folgt:

- Für alle $e \in E$ mit $f(e) < c(e)$ existiert $+e \in E_f$ mit $\alpha'(+e) = \alpha(e)$, $\omega'(+e) = \omega(e)$ und $c_f(+e) = c(e) - f(e)$
- Für alle $e \in E$ mit $f(e) > 0$ existiert $-e \in E_f$ mit $\alpha'(-e) = \omega(e)$, $\omega'(-e) = \alpha(e)$ und $c_f(-e) = f(e)$

Flussnetzwerk



Residualnetzwerk

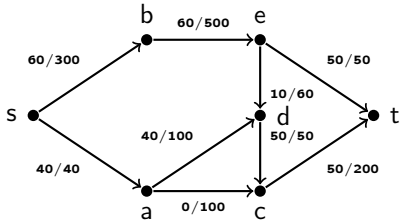
Residualnetzwerk

Definition - Residualnetzwerk

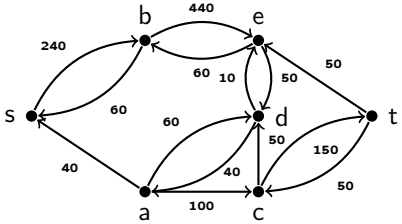
Für einen s-t-Fluss f im Netzwerk $N = (G, c, s, t)$ definieren wir das **Residualnetzwerk** $N_f := (G_f, c_f, s, t)$ mit **Residualgraph** $G_f = (V, E_f, \alpha', \omega')$ und **Residualkapazität** $c_f : E_f \to \mathbb{R}_{\geq 0}$ wie folgt:

- Für alle $e \in E$ mit $f(e) < c(e)$ existiert $+e \in E_f$ mit $\alpha'(+e) = \alpha(e)$, $\omega'(+e) = \omega(e)$ und $c_f(+e) = c(e) - f(e)$
- Für alle $e \in E$ mit $f(e) > 0$ existiert $-e \in E_f$ mit $\alpha'(-e) = \omega(e)$, $\omega'(-e) = \alpha(e)$ und $c_f(-e) = f(e)$

Flussnetzwerk



Residualnetzwerk



Augmenting Path Theorem

augmentierender/flussvergrößernder Weg: Ein s-t-Weg P im Residualnetzwerk G_f

Residualkapazität des Weges P:

$$\Delta P := \min_{r \in P} c_f(r)$$

Augmenting Path Theorem

Sei f ein zulässiger s-t-Fluss in G. Dann gilt:

f ist ein maximaler Fluss $\Leftrightarrow$ es existiert kein augmentierender Weg in G_f

Augmenting Path Theorem

augmentierender/flussvergrößernder Weg: Ein s-t-Weg P im Residualnetzwerk G_f

Residualkapazität des Weges P:

$$\Delta P := \min_{r \in P} c_f(r)$$

Augmenting Path Theorem

Sei f ein zulässiger s-t-Fluss in G. Dann gilt:

f ist ein maximaler Fluss $\quad \Leftrightarrow \quad$ es existiert kein augmentierender Weg in G_f

Beweis Augmenting Path Theorem (1)

f ist ein maximaler Fluss $\Leftrightarrow$ es existiert kein augmentierender Weg in G_f

"$\Rightarrow$" Klar, wenn ein augmentierender Weg existiert kann f nicht maximal sein

"$\Leftarrow$" Idee: Wir geben einen s-t-Schnitt (S, T) an mit $\mathcal{W}_f = c(S, T)$
$S = \{v \in V :$ es existiert ein s-v-Weg in $G_f\}$, $\quad T = V \setminus S$

$$\mathcal{W}_f = c(S,T) \quad \Leftrightarrow \quad \sum_{e \in (S,T)} f(e) - \sum_{e \in (T,S)} f(e) = \sum_{e \in (S,T)} c(e)$$

Es genügt zu zeigen, dass:

$$f(e) = c(e) \qquad \forall e \in (S,T) \tag{1}$$
$$f(e) = 0 \qquad \forall e \in (T,S) \tag{2}$$

Beweis Augmenting Path Theorem (1)

f ist ein maximaler Fluss $\Leftrightarrow$ es existiert kein augmentierender Weg in G_f

"$\Rightarrow$" Klar, wenn ein augmentierender Weg existiert kann f nicht maximal sein

"$\Leftarrow$" Idee: Wir geben einen s-t-Schnitt (S, T) an mit $\mathcal{W}_f = c(S, T)$
$S = \{v \in V :$ es existiert ein s-v-Weg in $G_f\}$, $\quad T = V \setminus S$

$$\mathcal{W}_f = c(S,T) \quad \Leftrightarrow \quad \sum_{e \in (S,T)} f(e) - \sum_{e \in (T,S)} f(e) = \sum_{e \in (S,T)} c(e)$$

Es genügt zu zeigen, dass:

$$f(e) = c(e) \qquad \forall e \in (S,T) \tag{1}$$
$$f(e) = 0 \qquad \forall e \in (T,S) \tag{2}$$

Beweis Augmenting Path Theorem (2)

f ist ein maximaler Fluss $\Leftrightarrow$ es existiert kein augmentierender Weg in G_f

$$f(r) = c(e) \qquad \forall e \in (S, T) \tag{1}$$
$$f(r) = 0 \qquad \forall e \in (T, S) \tag{2}$$

Beweis durch Widerspruch:

(1) Sei $f(e) < c(e)$ für ein $e \in (S, T)$. Dann wäre $+e \in G_f$ mit $\alpha'(+e) \in S$ und $\omega'(+e) \in T$.
Widerspruch zu $\omega'(+e)$ ist nicht von s erreichbar.

(2) Sei $f(e) > 0$ für ein $e \in (T, S)$. Dann wäre $-e \in G_f$ mit $\alpha'(-e) \in S$ und $\omega'(-e) \in T$.
Widerspruch zu $\omega'(-r)$ ist nicht von s erreichbar.

Bestimmung maximaler Flüsse - Algorithmen

Algorithmus von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $\quad f(e) + \Delta P \quad \forall + e \in P$ ▷ *Flussnetzwerk aktualisieren*
5: $\quad f(e) - \Delta P \quad \forall - e \in P$
6: $\quad$ aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

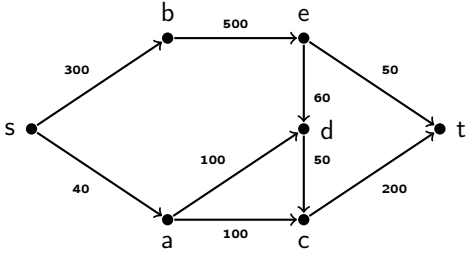
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $\quad f(e) + \Delta P \quad \forall + e \in P$ ▷ *Flussnetzwerk aktualisieren*
5: $\quad f(e) - \Delta P \quad \forall - e \in P$
6: $\quad$ aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk

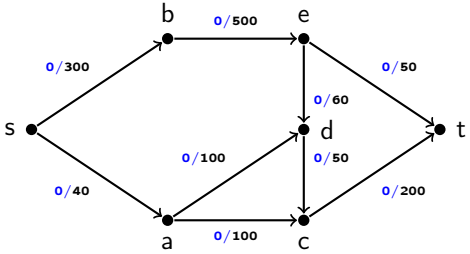
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk **Residualnetzwerk**



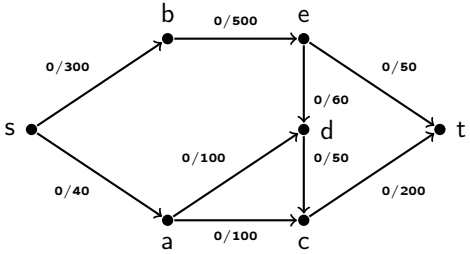
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

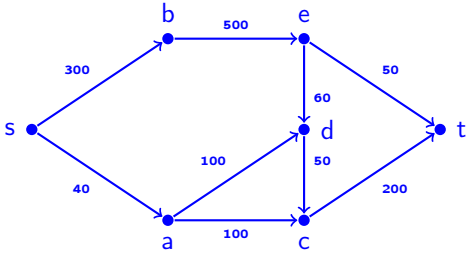
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $\quad f(e) + \Delta P \quad \forall + e \in P$
5: $\quad f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: $\quad$ aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



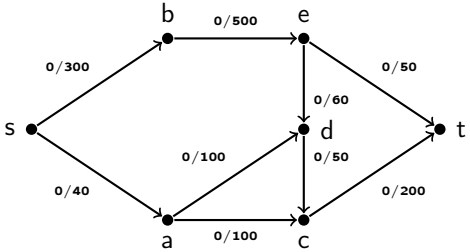
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

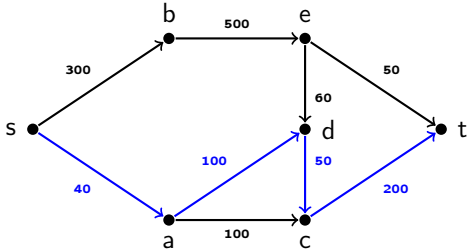
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk

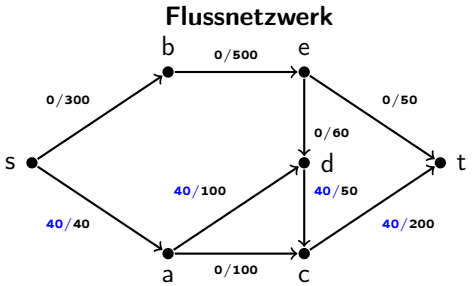


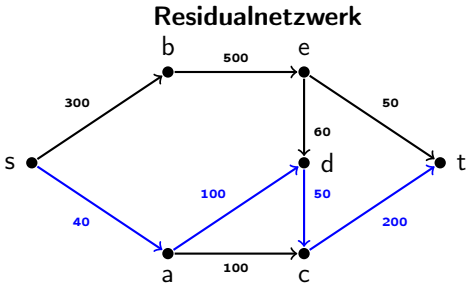
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**



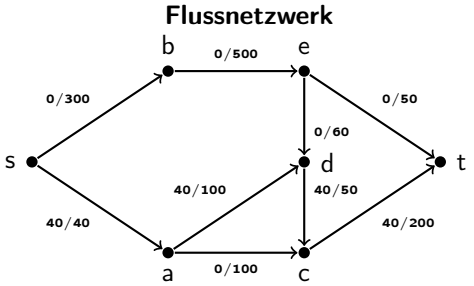


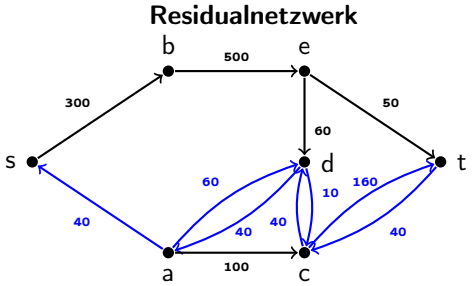
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**





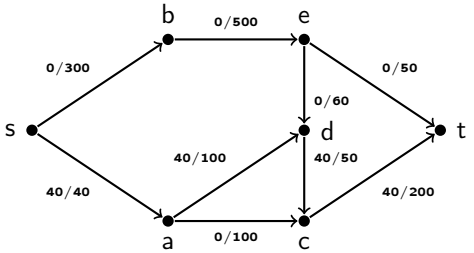
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

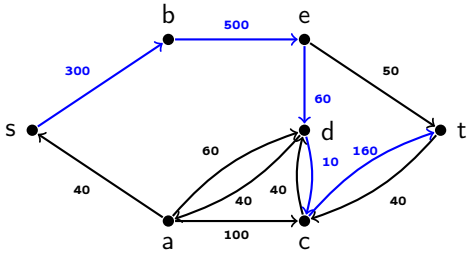
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk

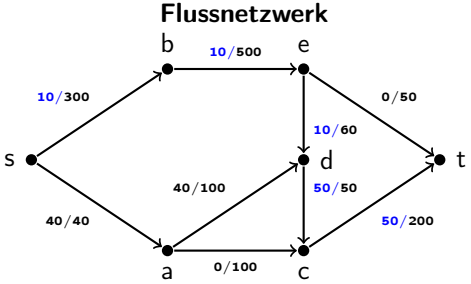


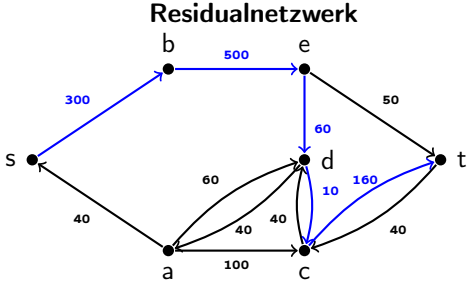
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**





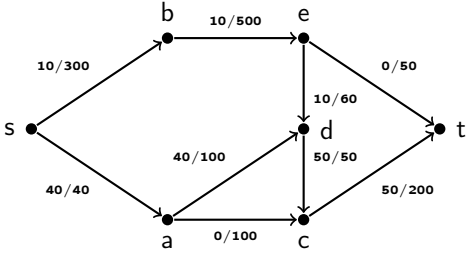
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

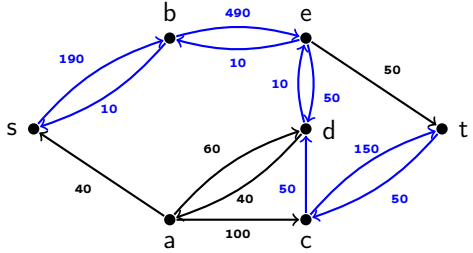
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



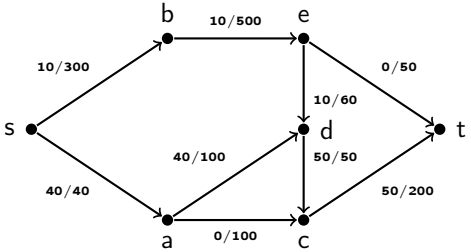
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

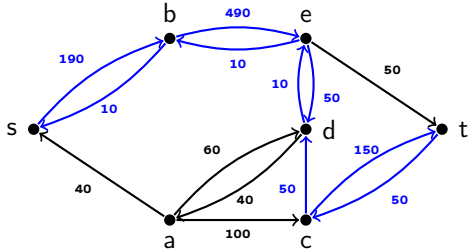
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $\quad f(e) + \Delta P \quad \forall + e \in P$
5: $\quad f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: $\quad$ aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



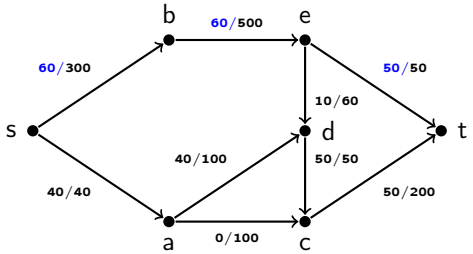
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

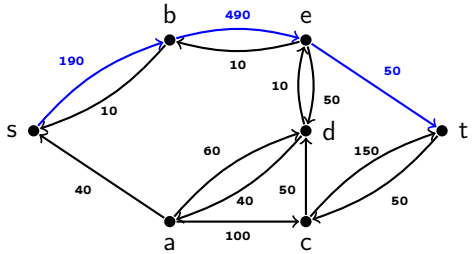
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



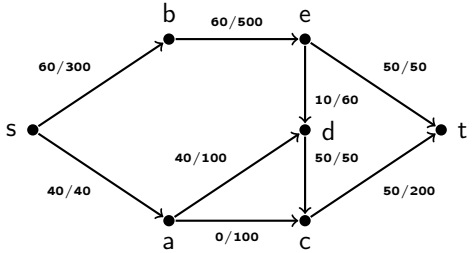
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

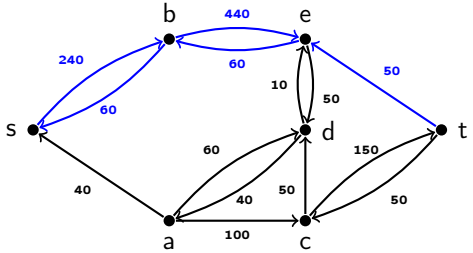
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $\quad f(e) + \Delta P \quad \forall + e \in P$
5: $\quad f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: $\quad$ aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



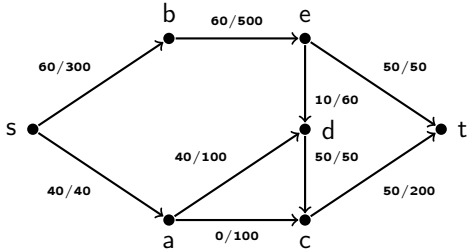
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

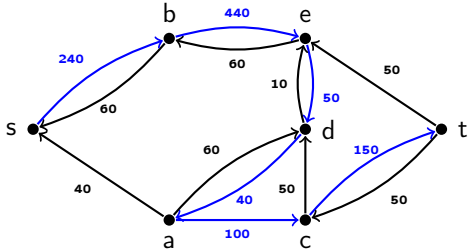
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



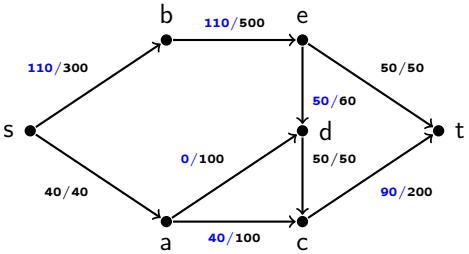
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

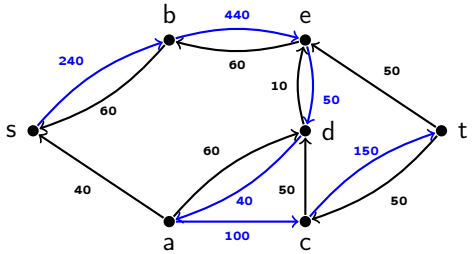
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $f(e) + \Delta P \quad \forall + e \in P$
5: $f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



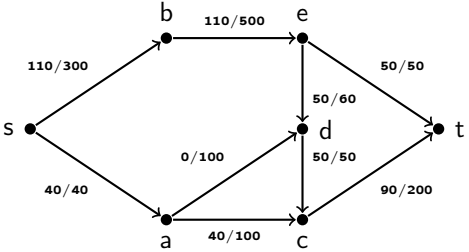
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

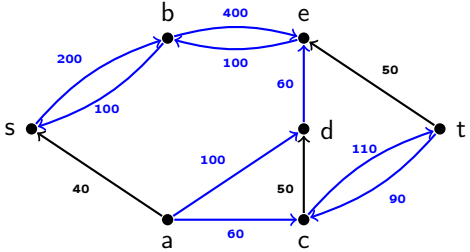
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg P in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $\quad f(e) + \Delta P \quad \forall + e \in P$
5: $\quad f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: $\quad$ aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



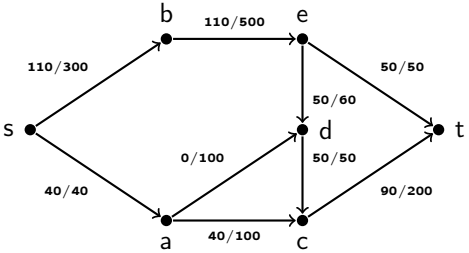
Anwendung von Ford und Fulkerson

Algorithm 1 Algorithmus von Ford und Fulkerson

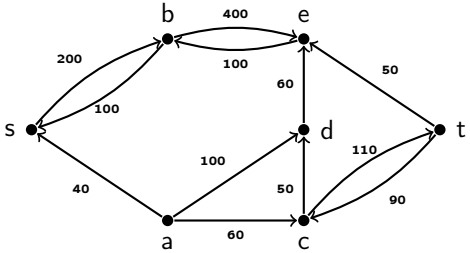
Input: gerichteter Graph $G = (V, E, \alpha, \omega)$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$, $s, t \in V, s \neq t$
Output: ein maximaler s-t-Fluss

1: $f(e) = 0 \quad \forall e \in E$ ▷ *Initialisierung des Flusses*
2: Bestimme das Residualnetzwerk G_f
3: **while** $\exists$ ein augmentierender Weg in G_f **do** ▷ *Abbruchbedingung: Augmenting Path Theorem*
4: $\quad f(e) + \Delta P \quad \forall + e \in P$
5: $\quad f(e) - \Delta P \quad \forall - e \in P$ ▷ *Flussnetzwerk aktualisieren*
6: $\quad$ aktualisiere G_f ▷ *Bestimmung des neuen Residualnetzwerks*
7: **end while**

Flussnetzwerk



Residualnetzwerk



Korrektheitsanalyse

Wenn der Ford-Fulkerson Algorithmus terminiert, ist der maximale s-t-Fluss gefunden.

Laufzeitanalyse

Algorithm 18 Algorithmus von Ford und Fulkerson

Input: gerichteter Graph $G = (V, E)$

Anwendungen von Flussnetzwerken

Anwendungen

Flussnetzwerke haben ein sehr breites Anwendungsgebiet! Versuch einer Klassifizierung:

- Offensichtliche Problemstellungen, die sich mit Flussnetzwerken modellieren lassen
- Umformung bzw. Rückführung von Problemen auf Flussnetzwerke
- Dienen der Lösung anderer Problemstellungen der kombinatorischen Optimierung

Die **Schwierigkeit in der Praxis** liegt in der geeigneten Modellierung des Problems und / oder der Rückführung des Problems auf ein Flussnetzwerk!

Offensichtliche Anwendungen

- Infrastruktur, Versorgungsnetze
  - Waserfluss in Rohrnetzwerken ohne Speicher
  - Stromtransport in Elektrizitätsnetzen
  - Versendung von Datenpaketen in Firmennetzwerken
  - Ölpipelines
- Transportprobleme
- Airline scheduling
- Netzwerkanalyse

Die **Schwierigkeit in der Praxis** liegt in der geeigneten Modellierung des Problems. Nicht alle Transportprobleme lassen sich mit Flussnetzwerken lösen! :(

Limitationen der Modellierung

- Es kann nur eine Quelle und eine Senke modelliert werden.
- Modelle bei denen jede Leitung in beide Richtungen, jedoch nur in eine Richtung gleichzeitig fließen kann, kann modelliert werden mit

$$c(v, w) = c(w, v) \qquad v, w \in V$$

$$f(v, w) \cdot f(w, v) = 0$$

Die zweite Bedingung kann beim bestimmen des Flussnetzwerkes ignoriert werden, nach dem der maximale Fluss bestimmt ist, kann dieser angepasst werden, um $f(v, w) \cdot f(w, v) = 0$ zu erfüllen.

$$f(v, w) = f(v, w) - \min(f(v, w), f(w, v))$$
$$f(w, v) = f(w, v) - \min(f(v, w), f(w, v))$$

Limitationen der Modellierung

- Es kann nur eine Quelle und eine Senke modelliert werden.
- Modelle bei denen jede Leitung in beide Richtungen, jedoch nur in eine Richtung gleichzeitig fließen kann, kann modelliert werden mit

$$c(v, w) = c(w, v) \qquad v, w \in V$$

$$f(v, w) \cdot f(w, v) = 0$$

Die zweite Bedingung kann beim bestimmen des Flussnetzwerkes ignoriert werden, nach dem der maximale Fluss bestimmt ist, kann dieser angepasst werden, um $f(v, w) \cdot f(w, v) = 0$ zu erfüllen.

$$f(v, w) = f(v, w) - \min(f(v, w), f(w, v))$$
$$f(w, v) = f(w, v) - \min(f(v, w), f(w, v))$$

Max-Flow-Min-Cut-Theorem

MaxFlow-MinCut-Satz

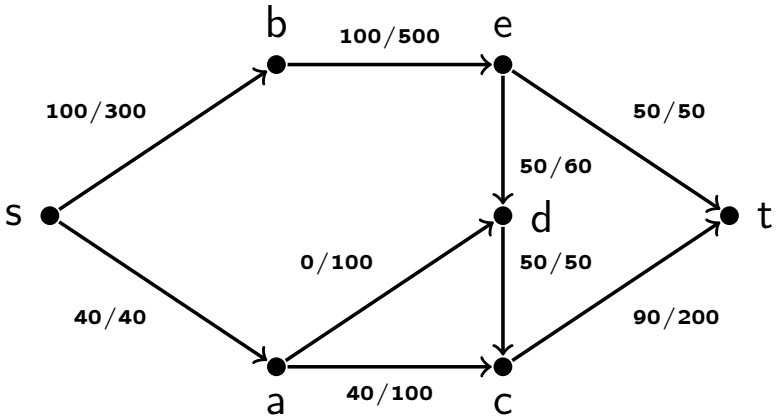
Der maximale Fluss in einem Netzwerk ist gleich der Kapazität des minimalen Schnitts:

$$\max \mathcal{W}_f = \min(c(S, T))$$

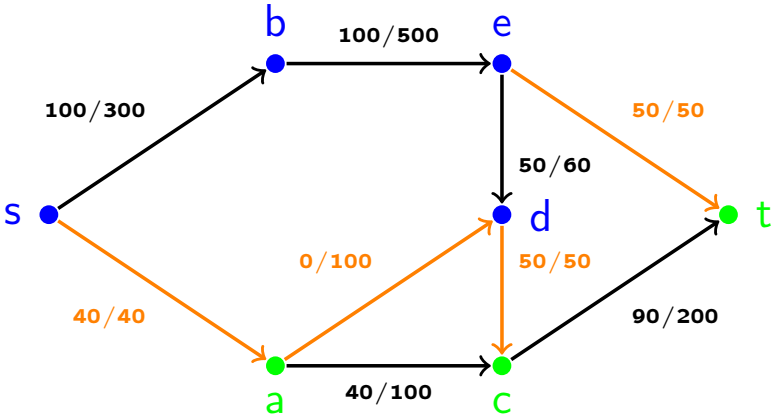
Beweis:
"$\leq$" entspricht der Beschränktheit des Flusses.
Wir können nun analog zum *Augmented Path Theorem* einen s-t-Schnitt (S, T) für den maximalen Fluss f angeben.
Wie bereits bewiesen gilt für diesen Schnitt $\mathcal{W}_f = c(S, T)$

Anwendungsbeispiel

Max Cut des Anwendungsbeispiels



Max Cut des Anwendungsbeispiels



Bemerkung zum Max-Flow-Min-Cut-Theorem

- Dualitätsaussage zwischen Max-Flow und Min-Cut
- Beweis auch möglich mittels Dualitätsaussagen aus der linearen Optimierung (starker Dualitätssatz)
  - Formulierung des Max-Flow-Problems als lineare Optimierungsaufgabe unter Nebenbedingungen und des Min-Cut-Problems als duales Problem dazu
  - **starker Dualitätssatz**: Besitzen das primale Problem eine optimale Lösung, so besitzt auch das duale Problem eine optimale Lösung und die beiden Lösungen stimmen überein.

Bemerkung zum Max-Flow-Min-Cut-Theorem

- Dualitätsaussage zwischen Max-Flow und Min-Cut
- Beweis auch möglich mittels Dualitätsaussagen aus der linearen Optimierung (starker Dualitätssatz)
  - Formulierung des Max-Flow-Problems als lineare Optimierungsaufgabe unter Nebenbedingungen und des Min-Cut-Problems als duales Problem dazu
  - **starker Dualitätssatz**: Besitzen das primale Problem eine optimale Lösung, so besitzt auch das duale Problem eine optimale Lösung und die beiden Lösungen stimmen überein.

Bemerkung zum Max-Flow-Min-Cut-Theorem

- Dualitätsaussage zwischen Max-Flow und Min-Cut
- Beweis auch möglich mittels Dualitätsaussagen aus der linearen Optimierung (starker Dualitätssatz)
  - Formulierung des Max-Flow-Problems als lineare Optimierungsaufgabe unter Nebenbedingungen und des Min-Cut-Problems als duales Problem dazu
  - **starker Dualitätssatz**: Besitzen das primale Problem eine optimale Lösung, so besitzt auch das duale Problem eine optimale Lösung und die beiden Lösungen stimmen überein.

Bemerkung zum Max-Flow-Min-Cut-Theorem

- Dualitätsaussage zwischen Max-Flow und Min-Cut
- Beweis auch möglich mittels Dualitätsaussagen aus der linearen Optimierung (starker Dualitätssatz)
  - Formulierung des Max-Flow-Problems als lineare Optimierungsaufgabe unter Nebenbedingungen und des Min-Cut-Problems als duales Problem dazu
  - **starker Dualitätssatz**: Besitzen das primale Problem eine optimale Lösung, so besitzt auch das duale Problem eine optimale Lösung und die beiden Lösungen stimmen überein.