

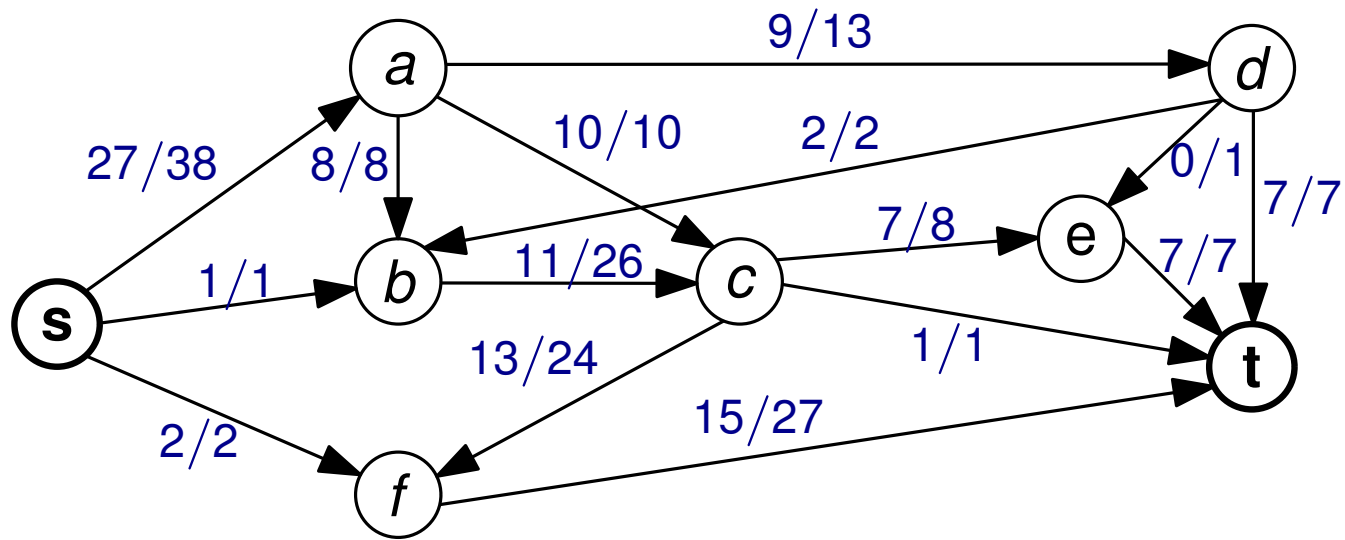
Algorithmen II

Vorlesung am 24.10.2013

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER



Flussprobleme und Dualität



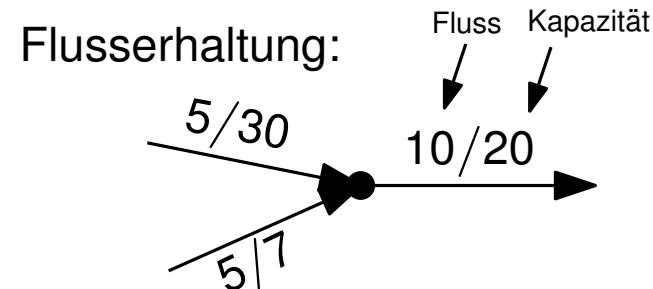
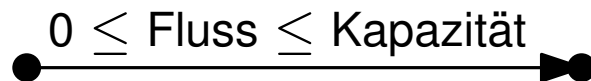
Definition:

- gegeben:
- Einfacher gerichteter Graph $D = (V, E)$.
 - Kantengewichtsfunktion $c: E \rightarrow \mathbb{R}_0^+$.
 - Zwei ausgezeichnete Knoten s (Quelle) und t (Senke).

Das Tupel (D, s, t, c) heißt *Netzwerk*. Eine Abbildung $f: E \rightarrow \mathbb{R}_0^+$ heißt *Fluss*, wenn folgende Bedingungen gelten:

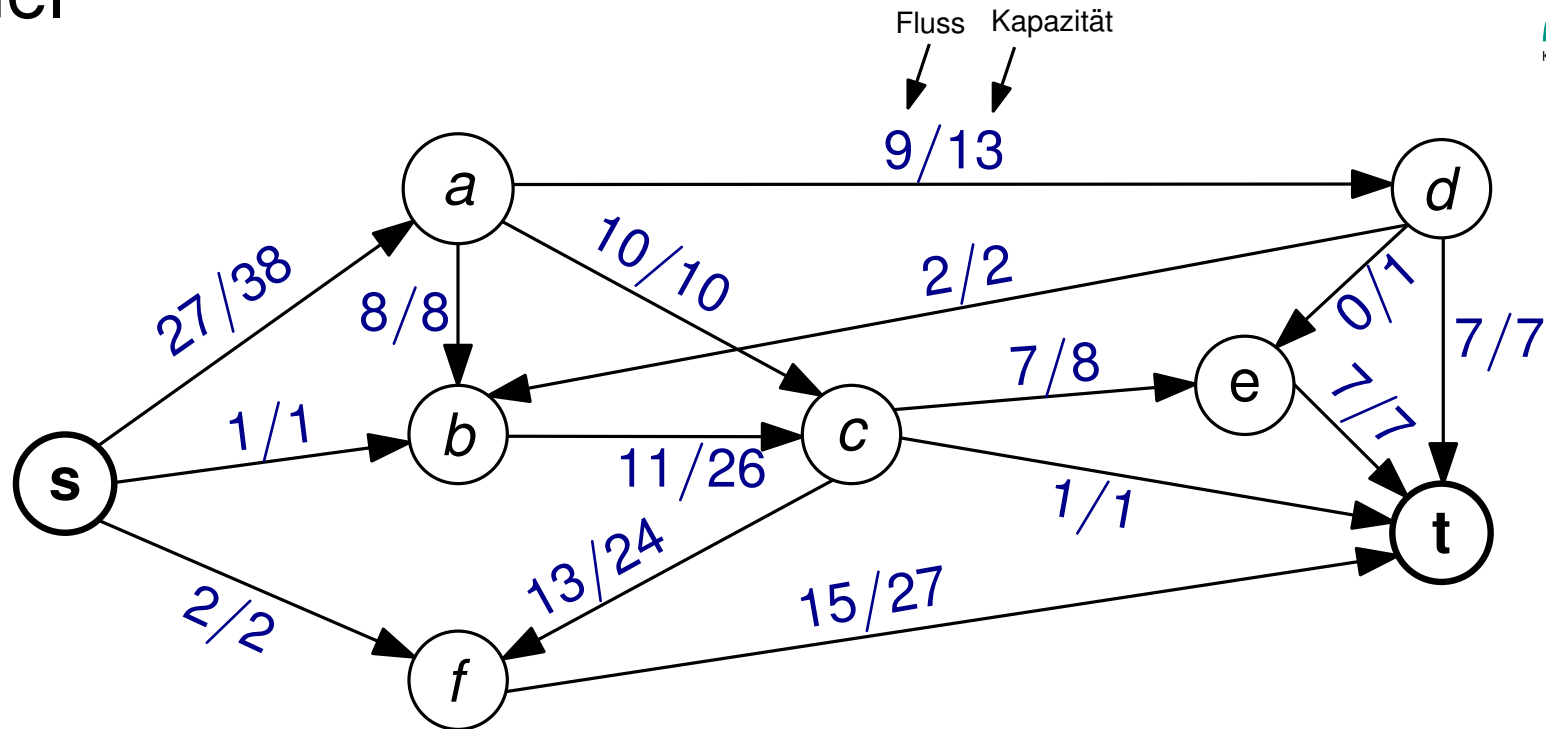
1. *Kapazitätsbedingung*: für alle $(i, j) \in E$ gilt $0 \leq f(i, j) \leq c(i, j)$
2. *Flusserhaltung*: für alle $i \in V \setminus \{s, t\}$ gilt $\sum_{(i,j) \in E} f(i, j) - \sum_{(j,i) \in E} f(j, i) = 0$

Kapazitätsbedingung:



Zwischenknoten können Fluss weder konsumieren noch produzieren.

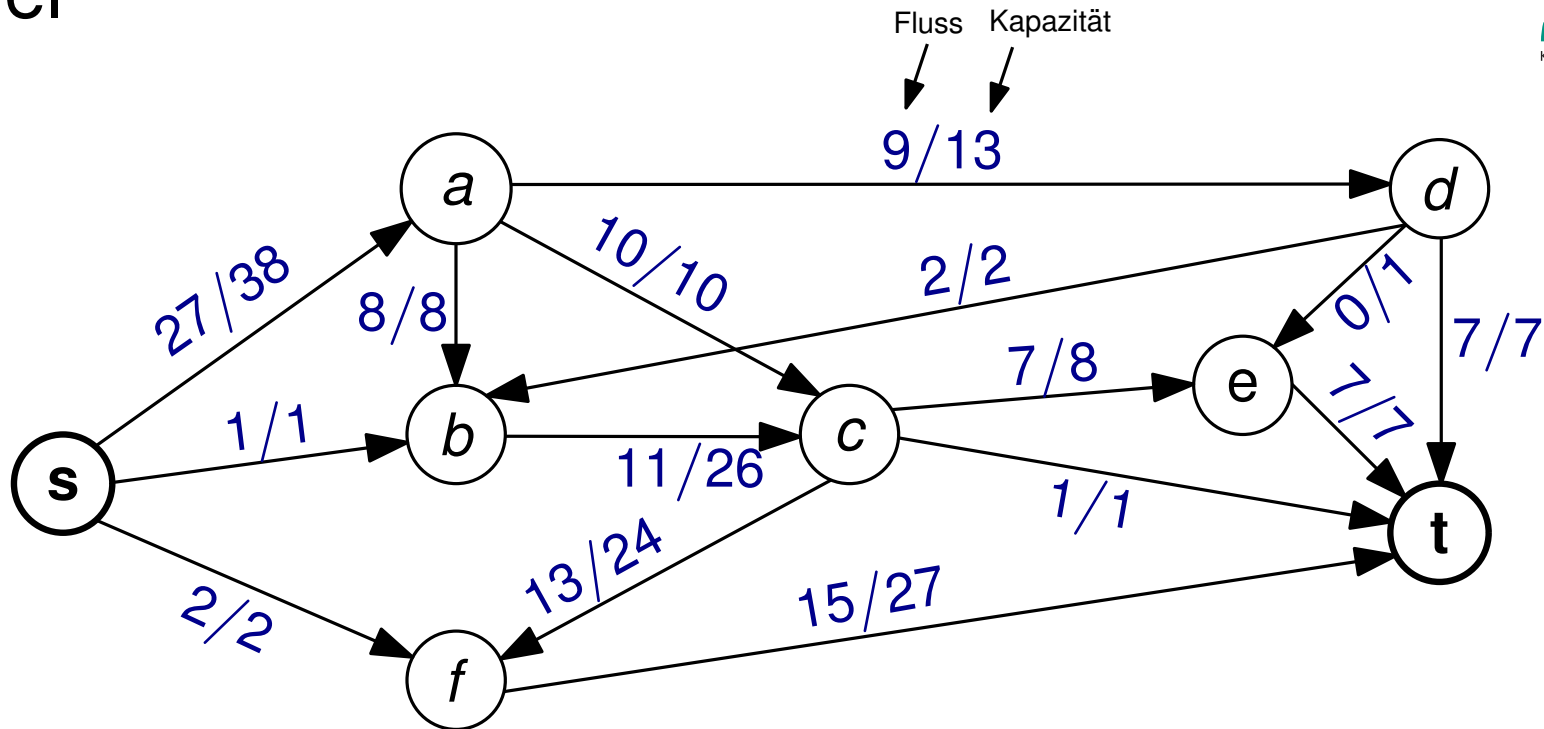
Beispiel



Zuweisung bildet Fluss, denn sowohl Kapazitätsbedingung als auch Flusserhaltung gelten:

1. *Kapazitätsbedingung:* für alle $(i, j) \in E$ gilt $0 \leq f(i, j) \leq c(i, j)$
2. *Flusserhaltung:* für alle $i \in V \setminus \{s, t\}$ gilt $\sum_{(i,j) \in E} f(i, j) - \sum_{(j,i) \in E} f(j, i) = 0$

Beispiel



Zuweisung bildet Fluss, denn sowohl Kapazitätsbedingung als auch Flusserhaltung gelten:

1. *Kapazitätsbedingung*: für alle $(i, j) \in E$ gilt $0 \leq f(i, j) \leq c(i, j)$
2. *Flusserhaltung*: für alle $i \in V \setminus \{s, t\}$ gilt $\sum_{(i,j) \in E} f(i, j) - \sum_{(j,i) \in E} f(j, i) = 0$

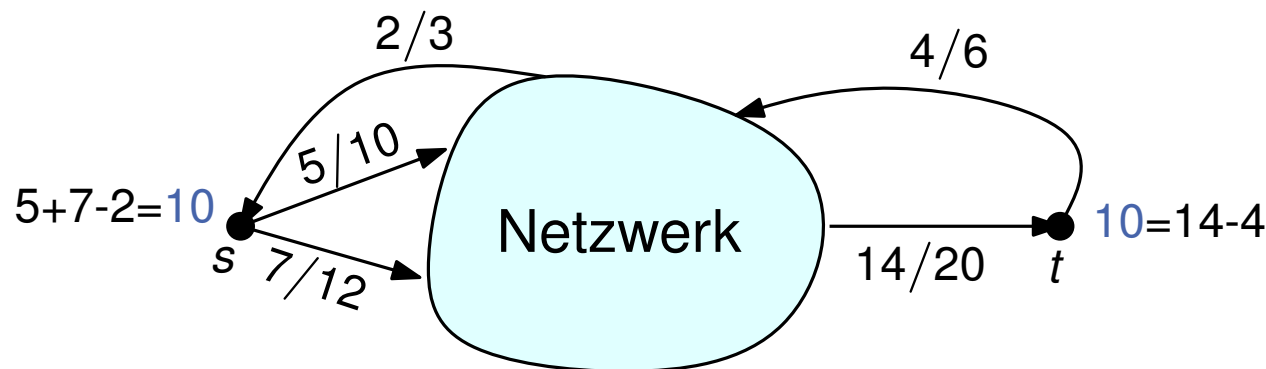
Bisher sind nicht alle Kapazitäten erschöpft: Gibt es einen besseren Fluss?

Was heißt besser? / Wie Fluss messen?

Lemma 4.2: Für einen Fluss f in einem Netzwerk (D, s, t, c) gilt

$$\sum_{(s,i) \in E} f(s,i) - \sum_{(i,s) \in E} f(i,s) = \sum_{(i,t) \in E} f(i,t) - \sum_{(t,i) \in E} f(t,i)$$

Intuition: Das was an s entsteht muss bei t verbraucht werden (und umgekehrt).



$$w(f) := \sum_{(s,i) \in E} f(s,i) - \sum_{(i,s) \in E} f(i,s) \text{ heißt } \textit{Wert} \text{ des Flusses } f$$

Lemma 4.2: Für einen Fluss f in einem Netzwerk (D, s, t, c) gilt

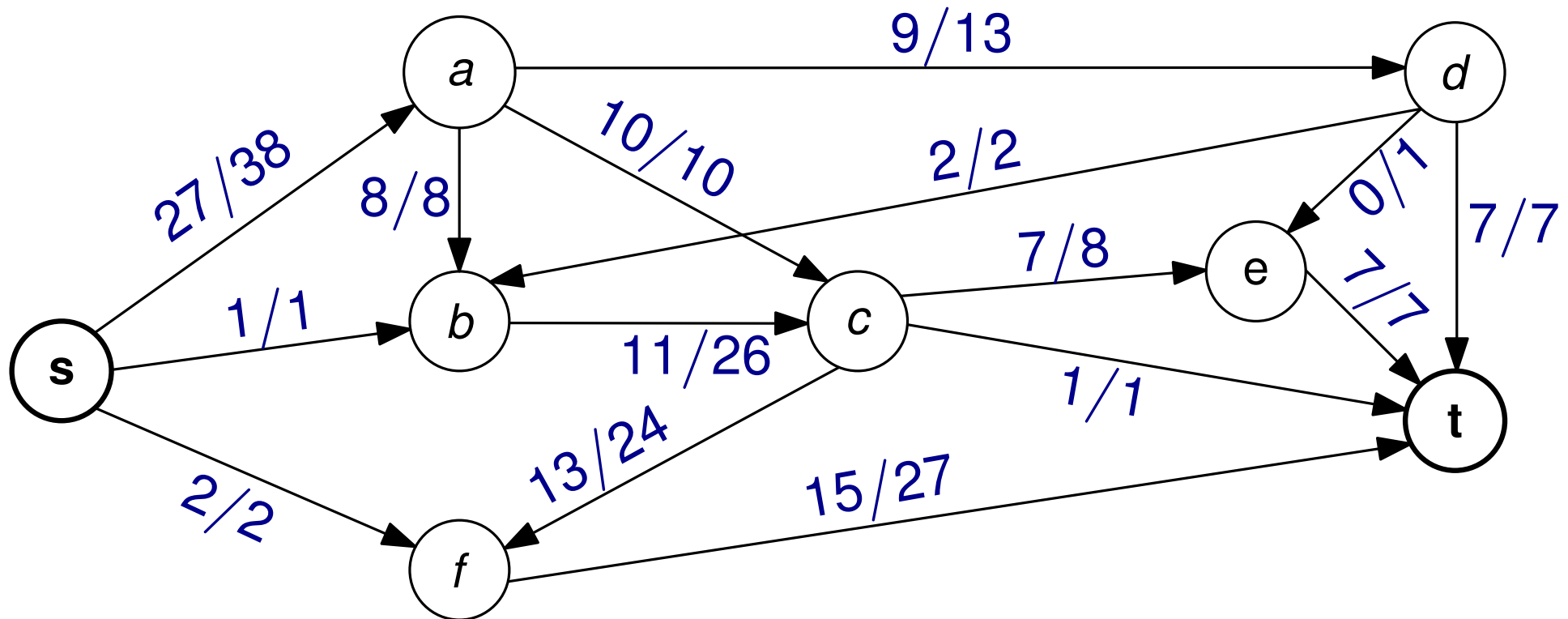
$$\sum_{(s,i) \in E} f(s,i) - \sum_{(i,s) \in E} f(i,s) = \sum_{(i,t) \in E} f(i,t) - \sum_{(t,i) \in E} f(t,i)$$

$w(f) := \sum_{(s,i) \in E} f(s,i) - \sum_{(i,s) \in E} f(i,s)$ heißt **Wert** des Flusses f .

Problemstellung:

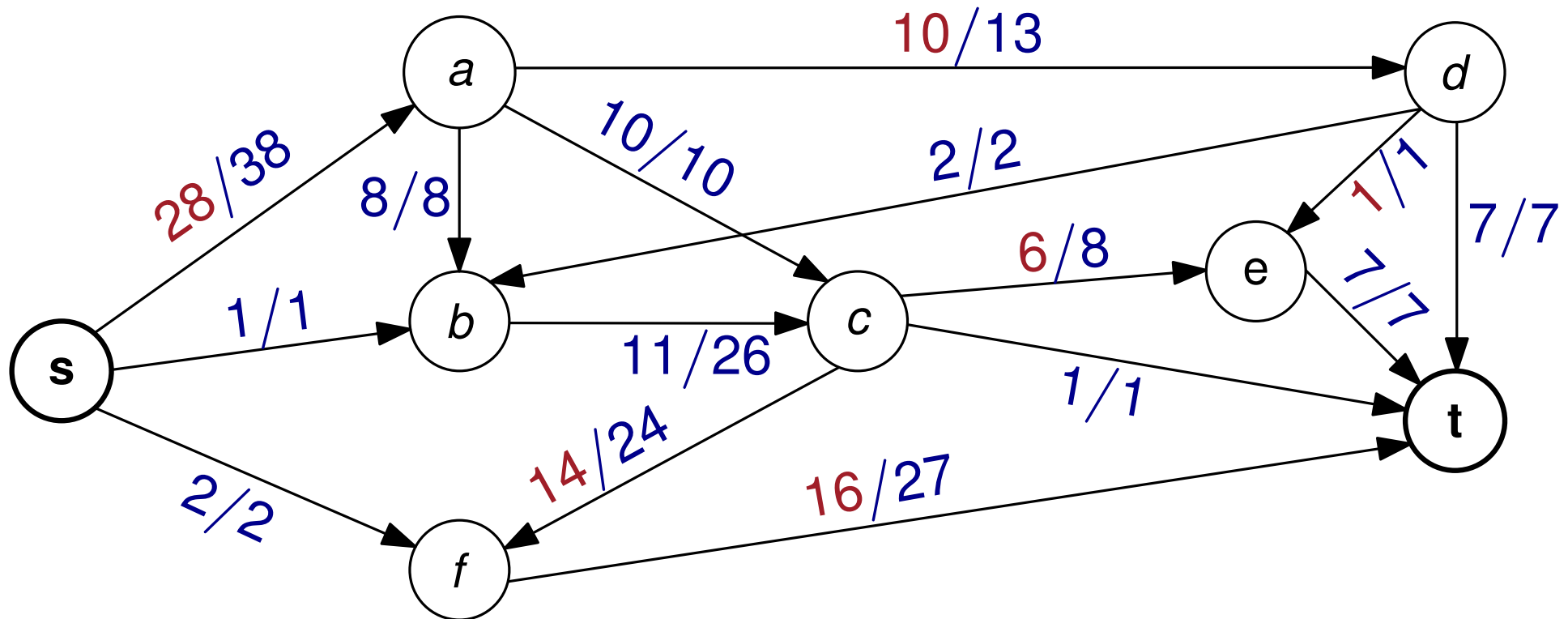
Finde in einem gegebenen Netzwerk (D, s, t, c) einen Maximalfluss f , d.h. für alle anderen Flüsse f' in dem Netzwerk gilt $w(f') \leq w(f)$.

Beispiel

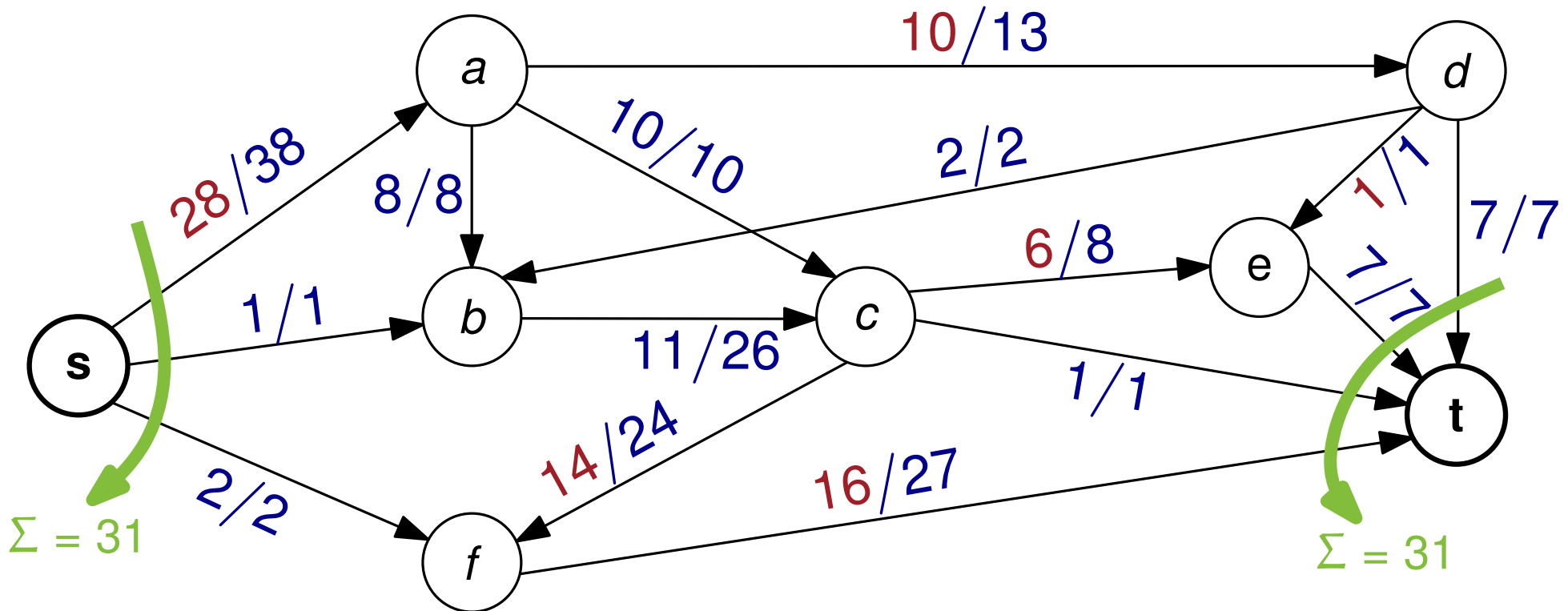


Wie Fluss verbessern, dass er maximal ist?

Beispiel



Welchen Wert besitzt dieser Fluss?



Welchen Wert besitzt dieser Fluss?

Antwort: 31

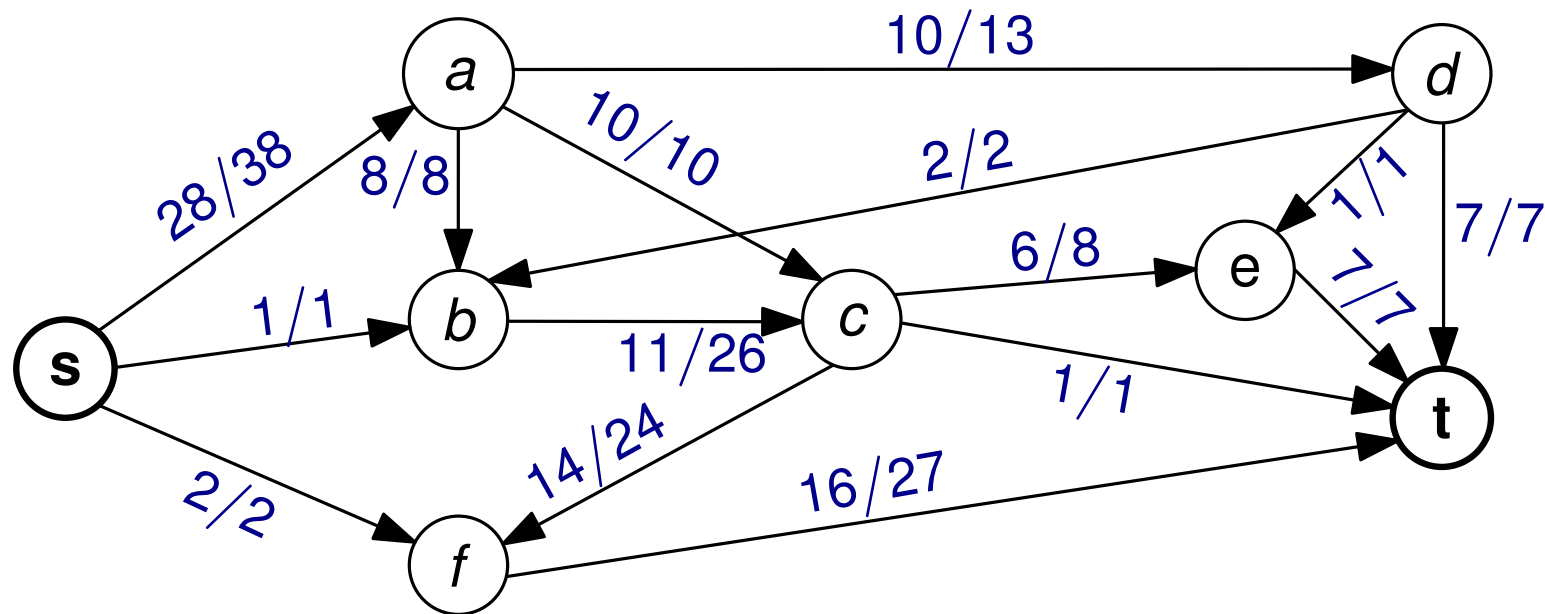
Im Folgendem: Wie zeigen, dass Fluss wirklich maximal ist?
 Welchen Zusammenhang gibt es zu Schnitten in Graphen?
 Wie Problem algorithmisch lösen?

Schnitte in Netzwerken

Definition: Sei $S \subset V$. Die Partition $(S, V \setminus S)$ heißt *Schnitt* im Graphen $D = (V, E)$. Im Netzwerk (D, s, t, c) heißt $(S, V \setminus S)$ ein *s-t-Schnitt*, falls $s \in S$ und $t \in V \setminus S$.

Die *Kapazität* eines Schnittes $(S, V \setminus S)$ ist definiert als $c(S, V \setminus S) := \sum_{\substack{(i,j) \in E \\ i \in S, j \in V \setminus S}} c(i, j)$

Ein Schnitt $(S, V \setminus S)$ heißt *minimal*, wenn $c(S, V \setminus S)$ minimalen Wert unter allen Schnitten $(S', V \setminus S')$ hat.

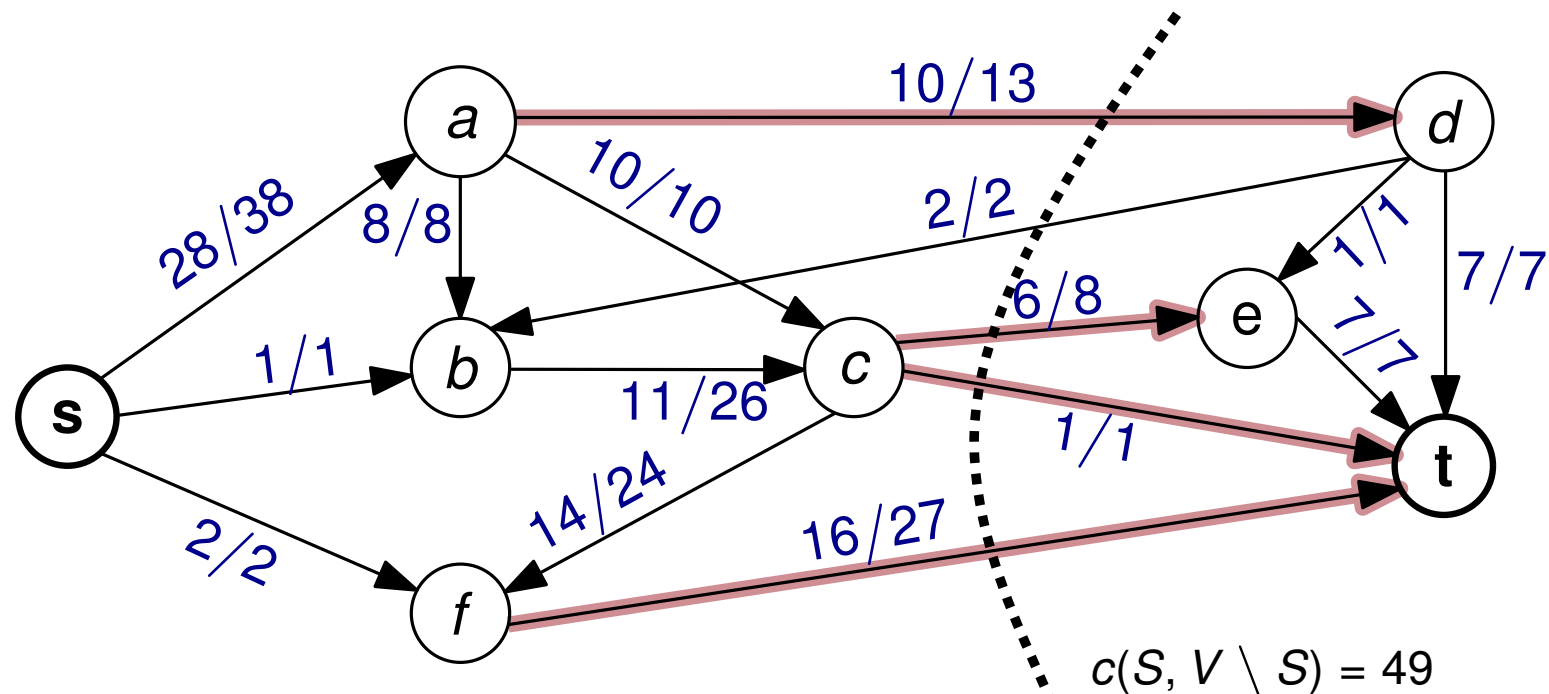


Schnitte in Netzwerken

Definition: Sei $S \subset V$. Die Partition $(S, V \setminus S)$ heißt *Schnitt* im Graphen $D = (V, E)$. Im Netzwerk (D, s, t, c) heißt $(S, V \setminus S)$ ein *s-t-Schnitt*, falls $s \in S$ und $t \in V \setminus S$.

Die *Kapazität* eines Schnittes $(S, V \setminus S)$ ist definiert als $c(S, V \setminus S) := \sum_{\substack{(i,j) \in E \\ i \in S, j \in V \setminus S}} c(i, j)$

Ein Schnitt $(S, V \setminus S)$ heißt *minimal*, wenn $c(S, V \setminus S)$ minimalen Wert unter allen Schnitten $(S', V \setminus S')$ hat.

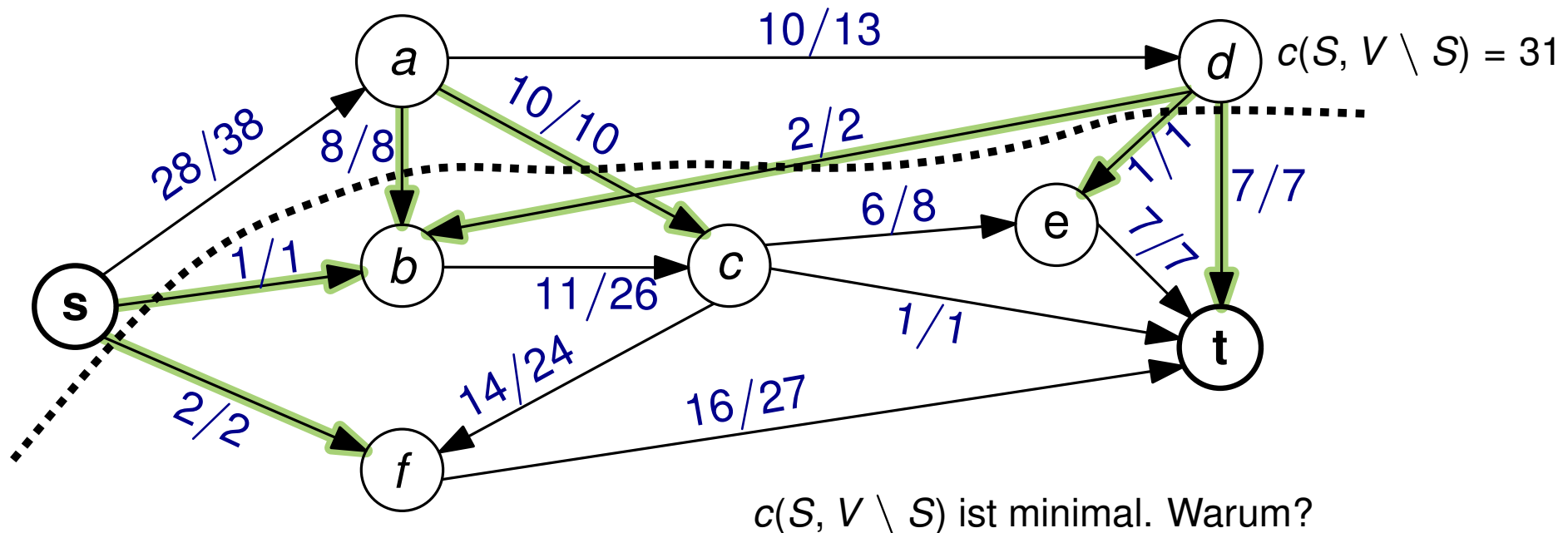


Schnitte in Netzwerken

Definition: Sei $S \subset V$. Die Partition $(S, V \setminus S)$ heißt *Schnitt* im Graphen $D = (V, E)$. Im Netzwerk (D, s, t, c) heißt $(S, V \setminus S)$ ein *s-t-Schnitt*, falls $s \in S$ und $t \in V \setminus S$.

Die *Kapazität* eines Schnittes $(S, V \setminus S)$ ist definiert als $c(S, V \setminus S) := \sum_{\substack{(i,j) \in E \\ i \in S, j \in V \setminus S}} c(i,j)$

Ein Schnitt $(S, V \setminus S)$ heißt *minimal*, wenn $c(S, V \setminus S)$ minimalen Wert unter allen Schnitten $(S', V \setminus S')$ hat.



Definition: Sei $S \subset V$. Die Partition $(S, V \setminus S)$ heißt *Schnitt* im Graphen $D = (V, E)$. Im Netzwerk (D, s, t, c) heißt $(S, V \setminus S)$ ein *s-t-Schnitt*, falls $s \in S$ und $t \in V \setminus S$.

Die *Kapazität* eines Schnittes $(S, V \setminus S)$ ist definiert als $c(S, V \setminus S) := \sum_{\substack{(i,j) \in E \\ i \in S, j \in V \setminus S}} c(i, j)$

Ein Schnitt $(S, V \setminus S)$ heißt *minimal*, wenn $c(S, V \setminus S)$ minimalen Wert unter allen Schnitten $(S', V \setminus S')$ hat.

Lemma 4.5: Sei $(S, V \setminus S)$ ein *s-t-Schnitt* im Netzwerk (D, s, t, c) . Für jeden Fluss f gilt, dass

$$w(f) = \sum_{\substack{(i,j) \in E \\ i \in S, j \in V \setminus S}} f(i, j) - \sum_{\substack{(i,j) \in E \\ j \in S, i \in V \setminus S}} f(i, j)$$

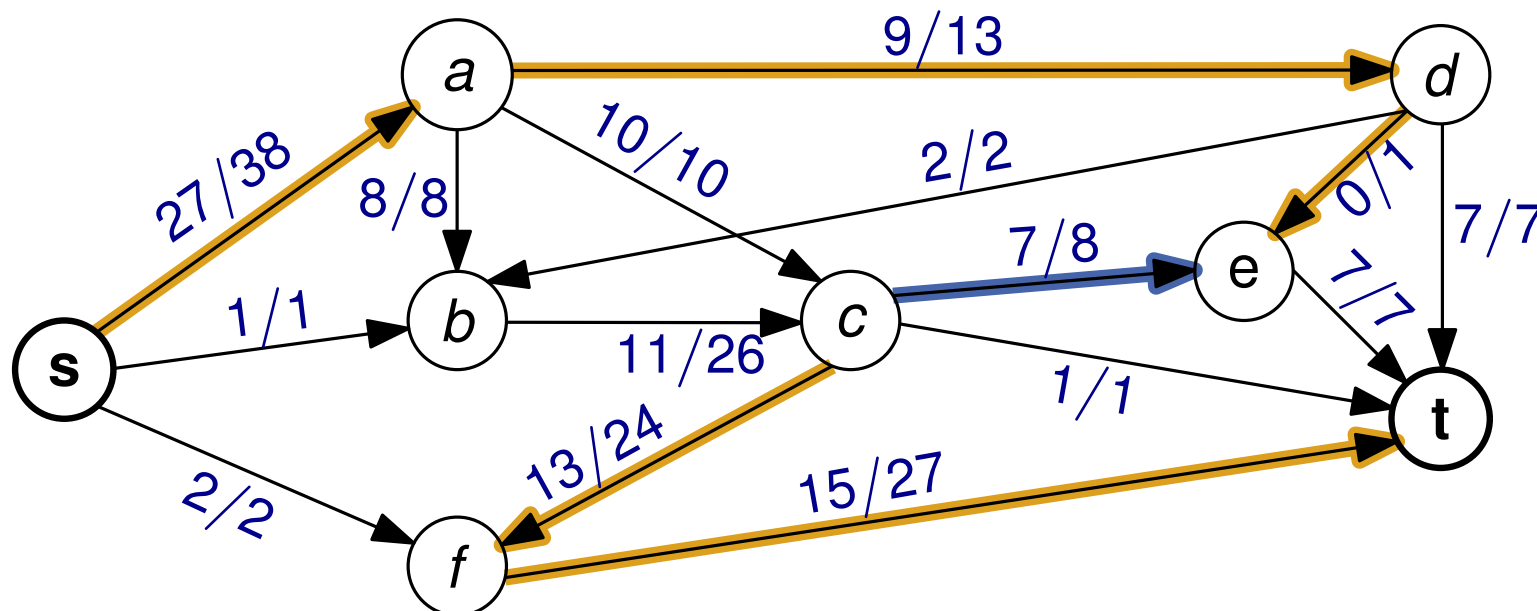
Insbesondere gilt $w(f) \leq c(S, V \setminus S)$.

Definition: Betrachte zu einem Fluss f im Netzwerk (D, s, t, c) einen ungerichteten Weg P von s nach t :

Alle Kanten auf P , die von s nach t gerichtet sind, heißen *Vorwärtskanten* und alle anderen *Rückwärtskanten*.

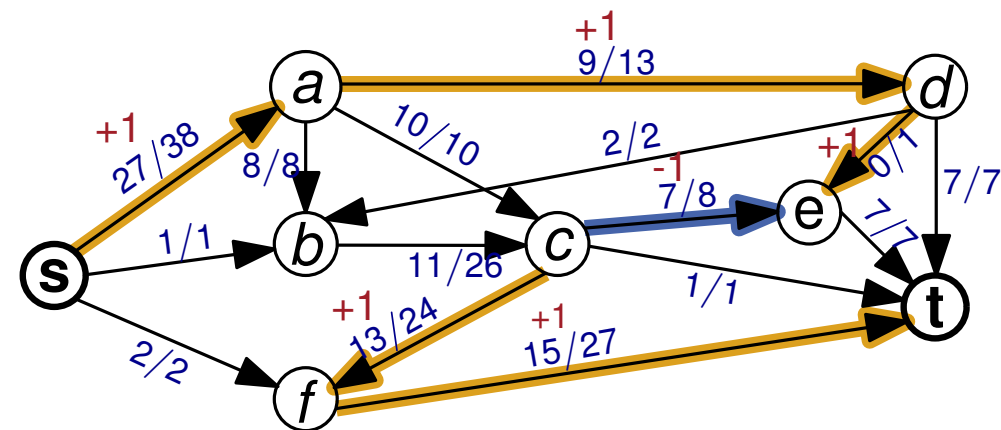
Der Weg P heißt *erhöhender Weg* bezüglich f , wenn

1. für jede Vorwärtskante (i, j) des Weges $f(i, j) < c(i, j)$ gilt, und
2. für jede Rückwärtskante (i, j) des Weges $0 < f(i, j)$ gilt.



Satz vom erhöhenden Weg

Satz vom erhöhenden Weg: Ein Fluss f in einem Netzwerk (D, s, t, c) ist genau dann ein Maximalfluss, wenn es bezüglich f keinen erhöhenden Weg gibt.



Satz vom erhöhenden Weg

Satz vom erhöhenden Weg: Ein Fluss f in einem Netzwerk (D, s, t, c) ist genau dann ein Maximalfluss, wenn es bezüglich f keinen erhöhenden Weg gibt.

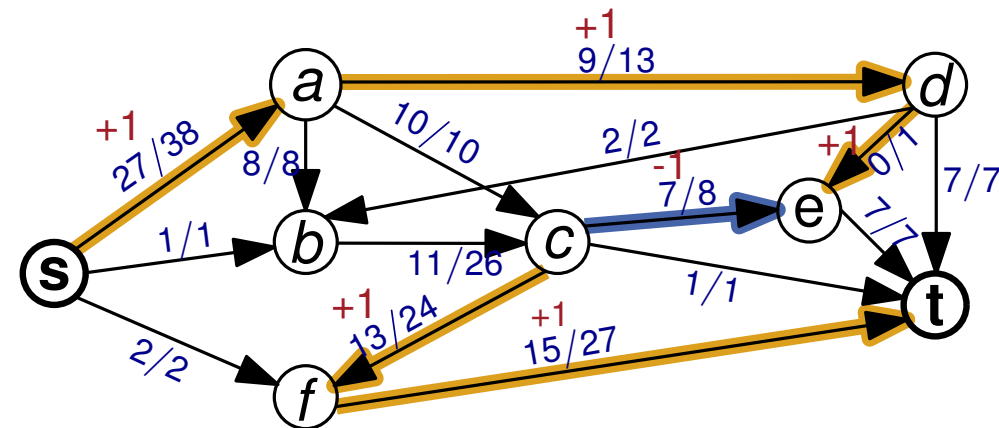
Beweis: " \Rightarrow " Annahme f ist maximal und es gibt erhöhenden Weg W .

Idee: Konstruiere mithilfe von f und W einen Fluss f' , sodass $w(f') > w(f)$.

Definiere für Kanten (i, j) von W :

$$\Delta(i, j) := \begin{cases} c(i, j) - f(i, j) & \text{falls } (i, j) \text{ Vorwärtskante} \\ f(i, j) & \text{falls } (i, j) \text{ Rückwärtskante} \end{cases}$$

$$\Delta := \min\{\Delta(i, j) \mid (i, j) \in W\} .$$



Satz vom erhöhenden Weg

Satz vom erhöhenden Weg: Ein Fluss f in einem Netzwerk (D, s, t, c) ist genau dann ein Maximalfluss, wenn es bezüglich f keinen erhöhenden Weg gibt.

Beweis: " \Rightarrow " Annahme f ist maximal und es gibt erhöhenden Weg W .

Idee: Konstruiere mithilfe von f und W einen Fluss f' , sodass $w(f') > w(f)$.


Definiere für Kanten (i, j) von W :

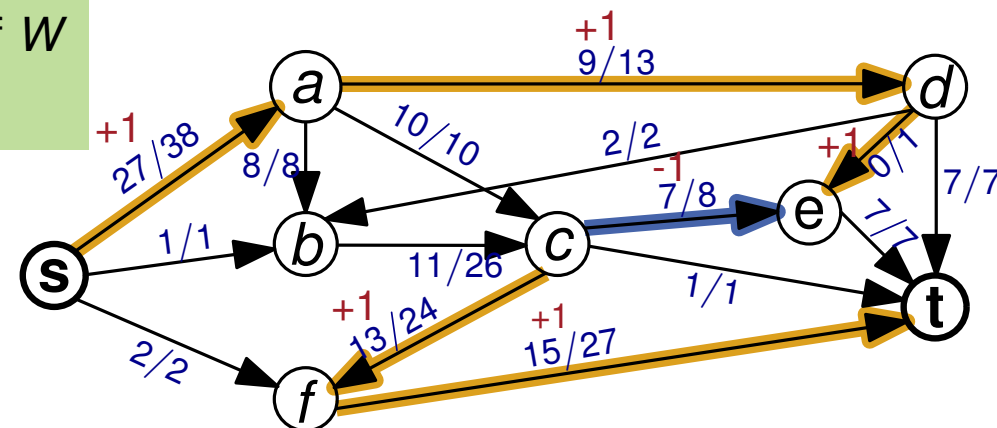
$$\Delta(i, j) := \begin{cases} c(i, j) - f(i, j) & \text{falls } (i, j) \text{ Vorwärtskante} \\ f(i, j) & \text{falls } (i, j) \text{ Rückwärtskante} \end{cases}$$

$$\Delta := \min\{\Delta(i, j) \mid (i, j) \in W\} .$$

Sei nun $f' : E \rightarrow \mathbb{R}_0^+$ definiert als

$$f' := \begin{cases} f(i, j) + \Delta & \text{falls } (i, j) \text{ Vorwärtskante auf } W \\ f(i, j) - \Delta & \text{falls } (i, j) \text{ Rückwärtskante auf } W \\ f(i, j) & \text{sonst.} \end{cases}$$

Weil $\Delta > 0$ gilt $w(f') > w(f)$  Annahme f ist maximal.

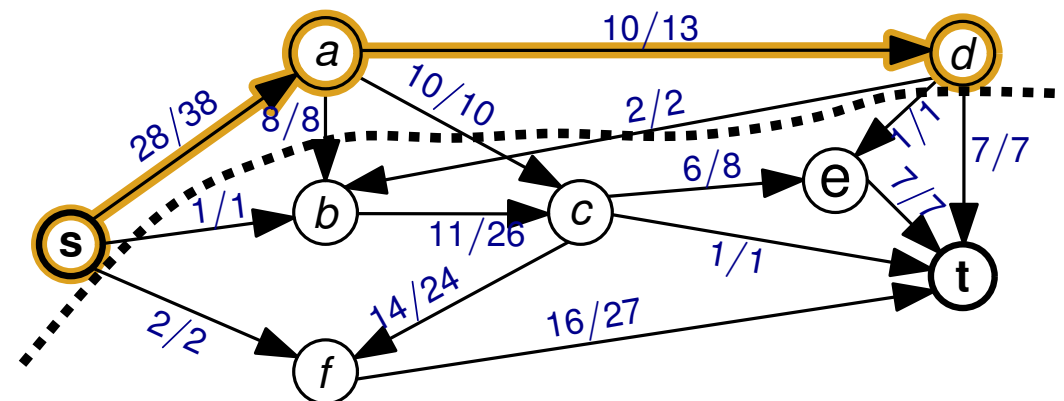


Satz vom erhöhenden Weg

Satz vom erhöhenden Weg: Ein Fluss f in einem Netzwerk (D, s, t, c) ist genau dann ein Maximalfluss, wenn es bezüglich f keinen erhöhenden Weg gibt.

Beweis: " \Leftarrow " Enthalte S alle Knoten, die bzgl. f auf einem erhöhenden Weg von s aus erreichbar sind.

Es gilt: $S \neq \emptyset$, weil $s \in S$ $S \neq V$, weil $t \notin S$



Satz vom erhöhenden Weg

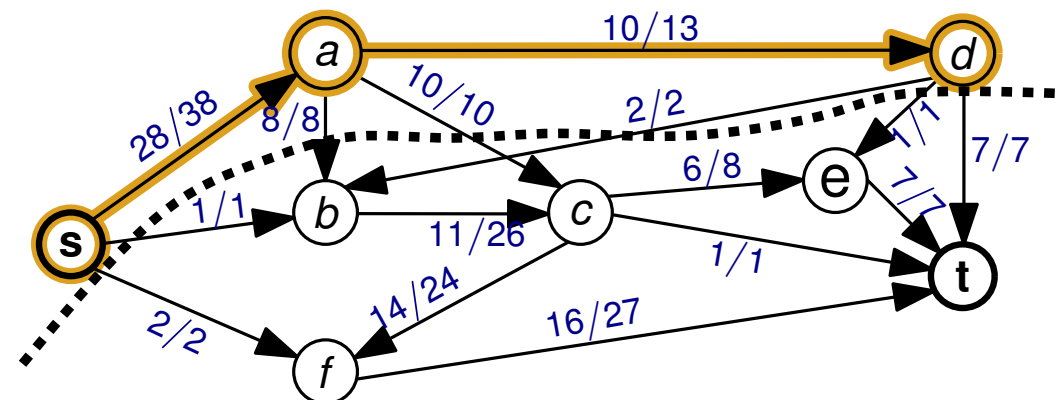
Satz vom erhöhenden Weg: Ein Fluss f in einem Netzwerk (D, s, t, c) ist genau dann ein Maximalfluss, wenn es bezüglich f keinen erhöhenden Weg gibt.

Beweis: " \Leftarrow " Enthalte S alle Knoten, die bzgl. f auf einem erhöhenden Weg von s aus erreichbar sind.

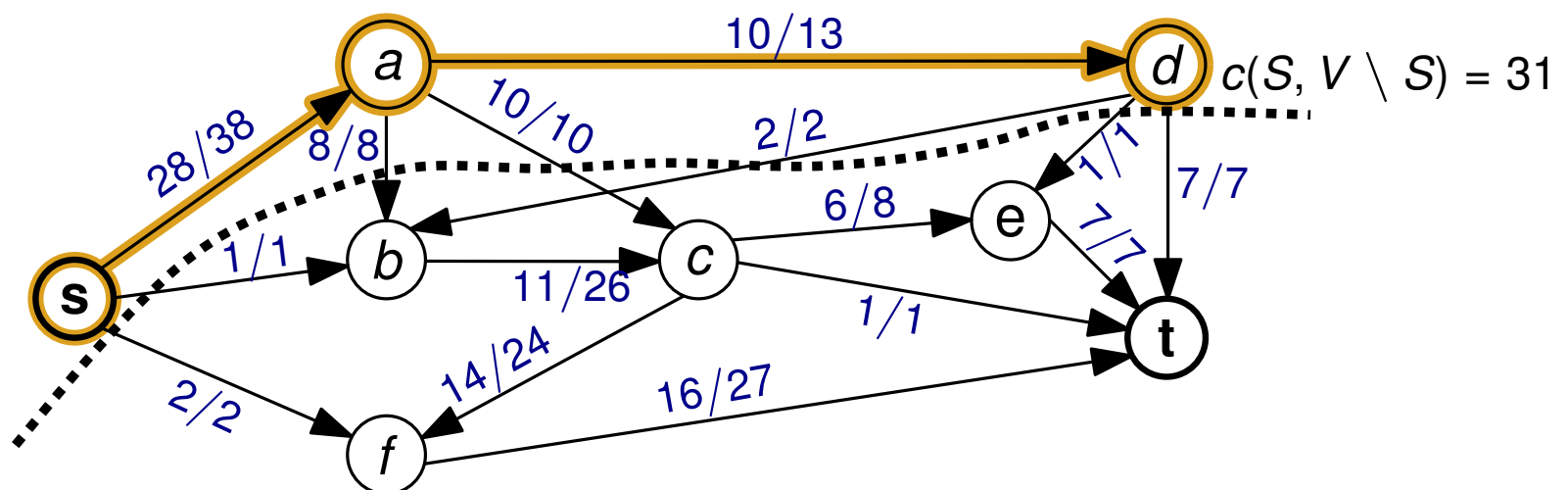
Es gilt: $S \neq \emptyset$, weil $s \in S$ $S \neq V$, weil $t \notin S$

- Es folgt:**
1. $(S, V \setminus S)$, ist s, t -Schnitt
 2. Alle Kanten von S nach $V \setminus S$ sind *saturiert*.
 3. Alle Kanten von $V \setminus S$ nach S sind *leer*.

Nach **Lemma 4.5** gilt $w(f) = c(S, V \setminus S)$ und somit ist f maximal.



Satz 4.9: In einem Netzwerk (D, s, t, c) ist der Wert eines Maximalflusses gleich der Kapazität eines minimalen s - t -Schnittes.



Max-Flow Min-Cut Theorem von Ford & Fulkerson

Satz 4.9: In einem Netzwerk (D, s, t, c) ist der Wert eines Maximalflusses gleich der Kapazität eines minimalen s - t -Schnittes.

Beweis: Folgt direkt aus dem Satz vom erhöhenden Weg:

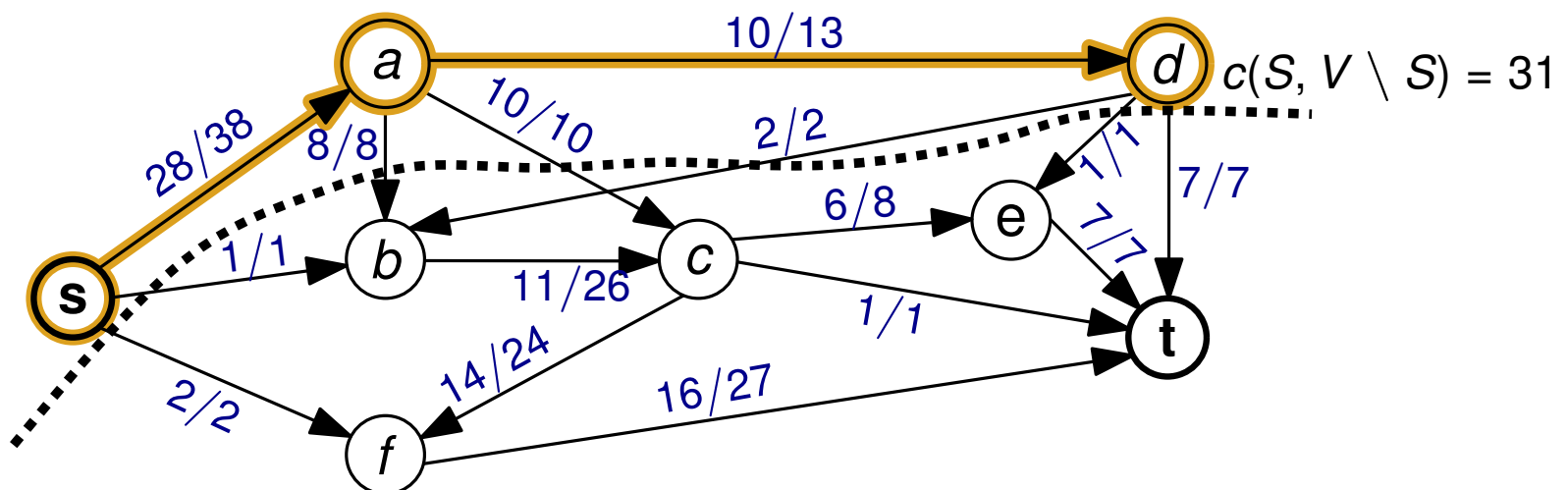
Da f Maximalfluss ist, gilt:

Es gibt einen Schnitt $(S, V \setminus S)$ mit $s \in S$ und $t \in V \setminus S$

(S enthält alle Knoten, die auf einem erhöhenden Weg von s aus erreichbar sind.)

Für $(S, V \setminus S)$ gilt:

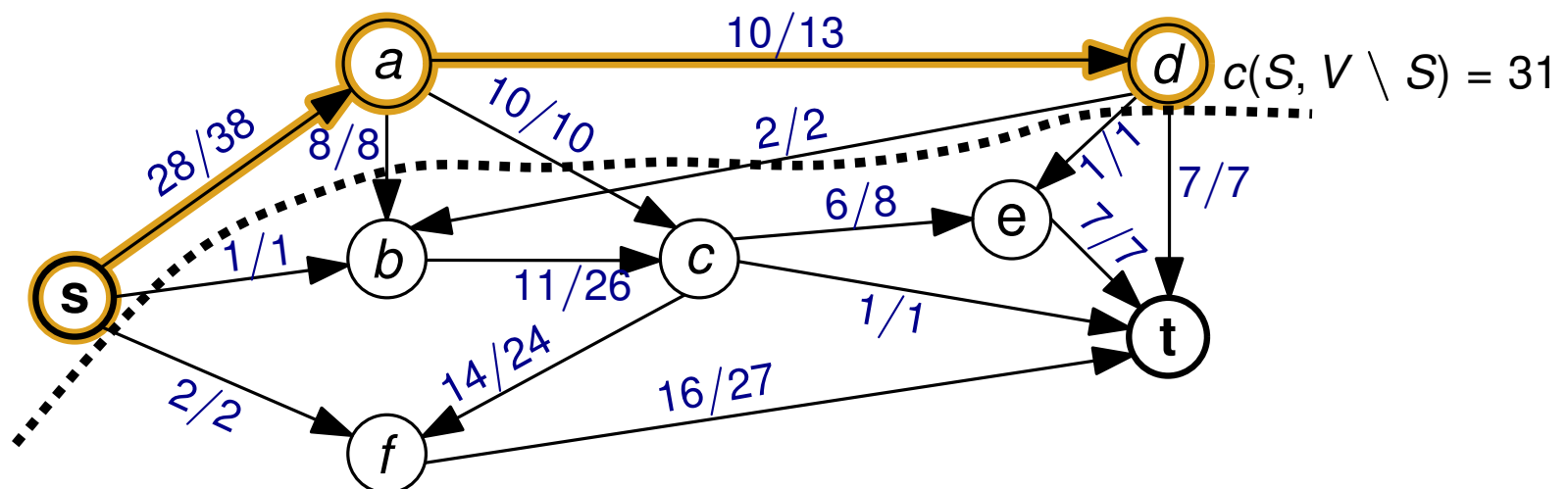
$$w(f) = c(S, V \setminus S) \text{ und } c(S, V \setminus S) = \min_{\substack{s \in S' \\ t \in V \setminus S'}} c(S', V \setminus S')$$



Satz 4.9: In einem Netzwerk (D, s, t, c) ist der Wert eines Maximalflusses gleich der Kapazität eines minimalen s - t -Schnittes.

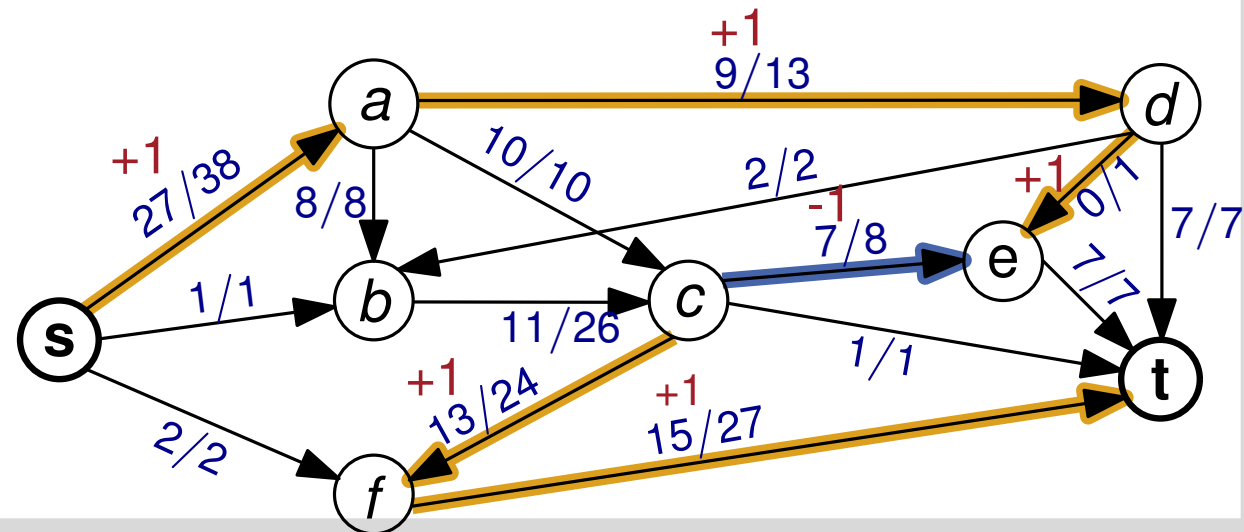
Bemerkungen: Für einen Fluss f in einem Netzwerk (D, s, t, c) sind die folgenden Aussagen äquivalent:

- Der Wert $w(f)$ ist maximal.
- Es gibt keinen bezüglich f erhöhenden Weg.
- Die Kapazität eines minimalen s - t -Schnittes $(S, V \setminus S)$ ist $w(f)$.



Ganzzahligkeitssatz

Satz 4.11: Sei (D, s, t, c) ein Netzwerk mit $c: E \rightarrow \mathbb{N}_0$. Dann gibt es einen Maximalfluss f mit $f(i, j) \in \mathbb{N}_0$ für alle $(i, j) \in E$ und damit $w(f) \in \mathbb{N}_0$



Satz 4.11: Sei (D, s, t, c) ein Netzwerk mit $c: E \rightarrow \mathbb{N}_0$. Dann gibt es einen Maximalfluss f mit $f(i, j) \in \mathbb{N}_0$ für alle $(i, j) \in E$ und damit $w(f) \in \mathbb{N}_0$

Beweis: Definiere Anfangsfluss f_0 mit $f_0(i, j) = 0$ für alle $(i, j) \in E$

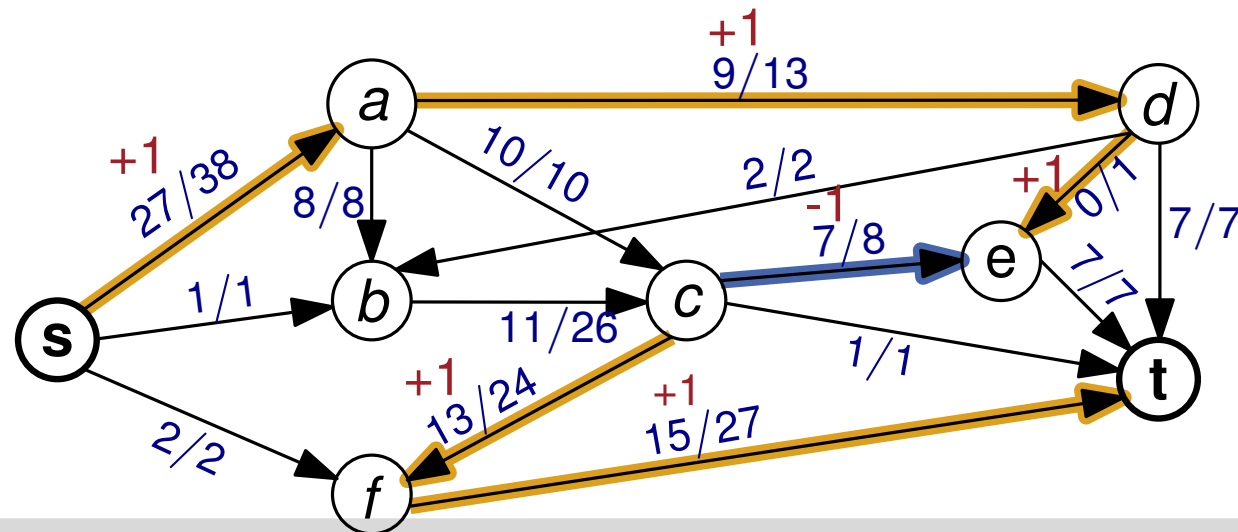
Falls f_0 nicht maximal, dann gibt es erhöhenden Weg W .

Definiere für Kanten (i, j) von W :

$$\Delta_0(i, j) := \begin{cases} c(i, j) - f_0(i, j) & \text{falls } (i, j) \text{ Vorwärtskante} \\ f_0(i, j) & \text{falls } (i, j) \text{ Rückwärtskante} \end{cases}$$

und

$$\Delta_0 := \min\{\Delta(i, j) \mid (i, j) \text{ auf erhöhendem Weg } W\} .$$



Ganzzahligkeitssatz

Satz 4.11: Sei (D, s, t, c) ein Netzwerk mit $c: E \rightarrow \mathbb{N}_0$. Dann gibt es einen Maximalfluss f mit $f(i, j) \in \mathbb{N}_0$ für alle $(i, j) \in E$ und damit $w(f) \in \mathbb{N}_0$

Beweis: Definiere Anfangsfluss f_0 mit $f_0(i, j) = 0$ für alle $(i, j) \in E$

Falls f_0 nicht maximal, dann gibt es erhöhenden Weg W .

Definiere für Kanten (i, j) von W :

$$\Delta_0(i, j) := \begin{cases} c(i, j) - f_0(i, j) & \text{falls } (i, j) \text{ Vorwärtskante} \\ f_0(i, j) & \text{falls } (i, j) \text{ Rückwärtskante} \end{cases}$$

und

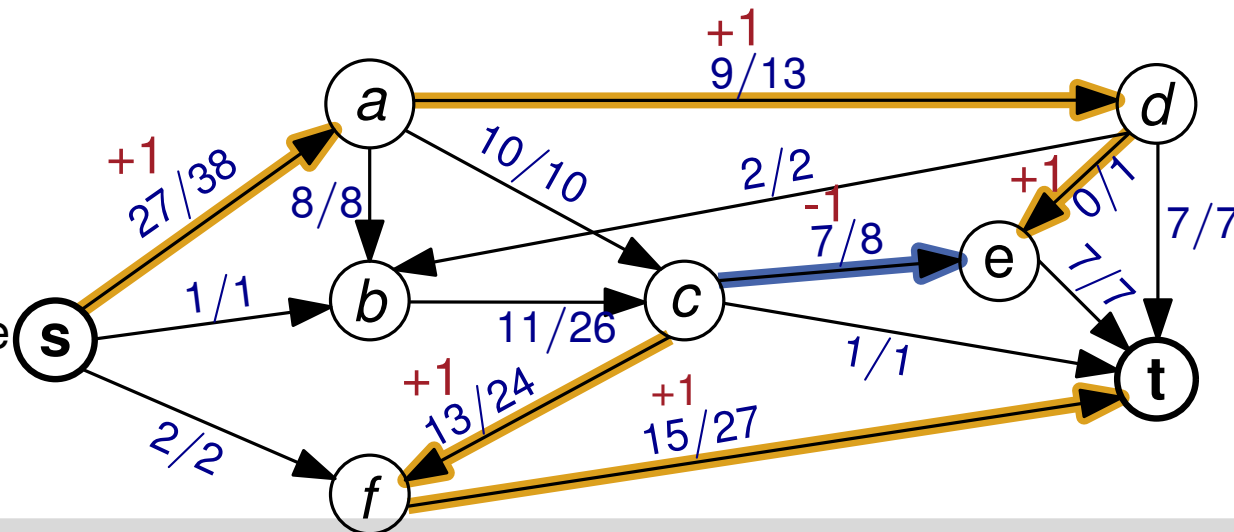
$$\Delta_0 := \min\{\Delta(i, j) \mid (i, j) \text{ auf erhöhendem Weg } W\} .$$

Offensichtlich gilt: $\Delta_0 \in \mathbb{N}$

Konstruiere entsprechend Fluss f_1 mit

$$w(f_1) = w(f_0) + \Delta_0$$

Wende Verfahren an, bis erhöhende Wege nicht mehr vorhanden sind.



Lösungsverfahren für Flussprobleme und minimale Schnitte

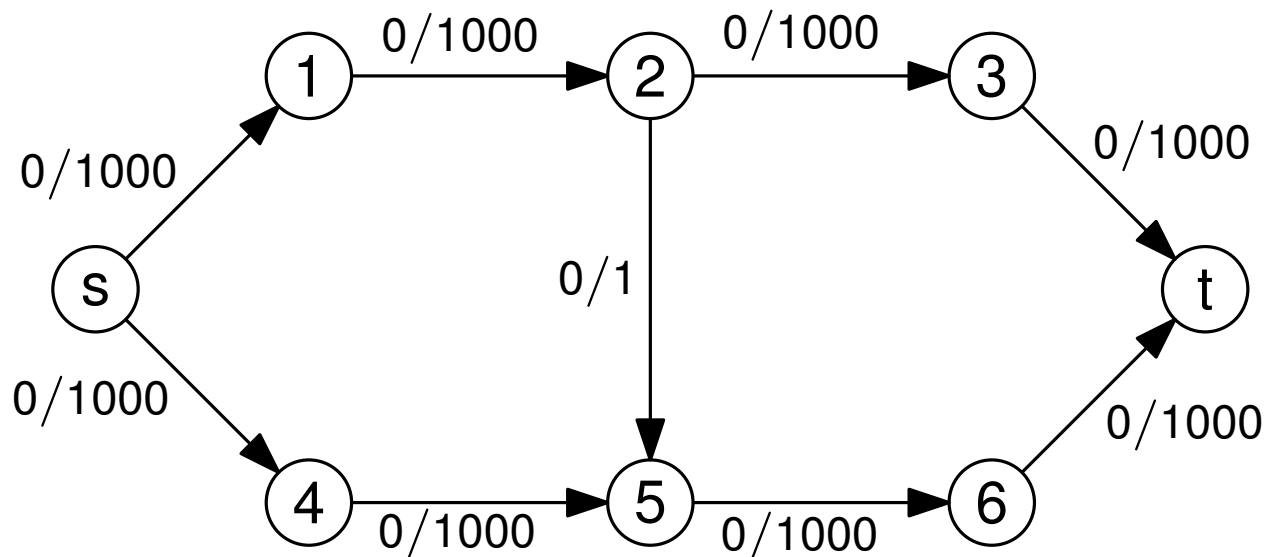
Schema des Ford-Fulkerson-Algorithmus

Eingabe: Netzwerk (D, s, t, c)

Ausgabe: Maximalfluss f bezüglich s und t

1. $f(i, j) \leftarrow 0$ für alle Kanten $(i, j) \in E$.
2. **Solange** ein erhöhender Weg $\langle e_1, e_2, \dots, e_k \rangle$ bezüglich f existiert **tue**
 - (a) $\delta \leftarrow \min(\{c(e_i) - f(e_i) \mid e_i \text{ ist Vorwärtskante}\} \cup \{f(e_i) \mid e_i \text{ ist Rückwärtskante}\})$
 - (b) Setze für alle e_i :

$$f(e_i) \leftarrow \begin{cases} f(e_i) + \delta & e_i \text{ ist Vorwärtskante} \\ f(e_i) - \delta & e_i \text{ ist Rückwärtskante} \end{cases}$$



Schema des Ford-Fulkerson-Algorithmus

Eingabe: Netzwerk (D, s, t, c)

Ausgabe: Maximalfluss f bezüglich s und t

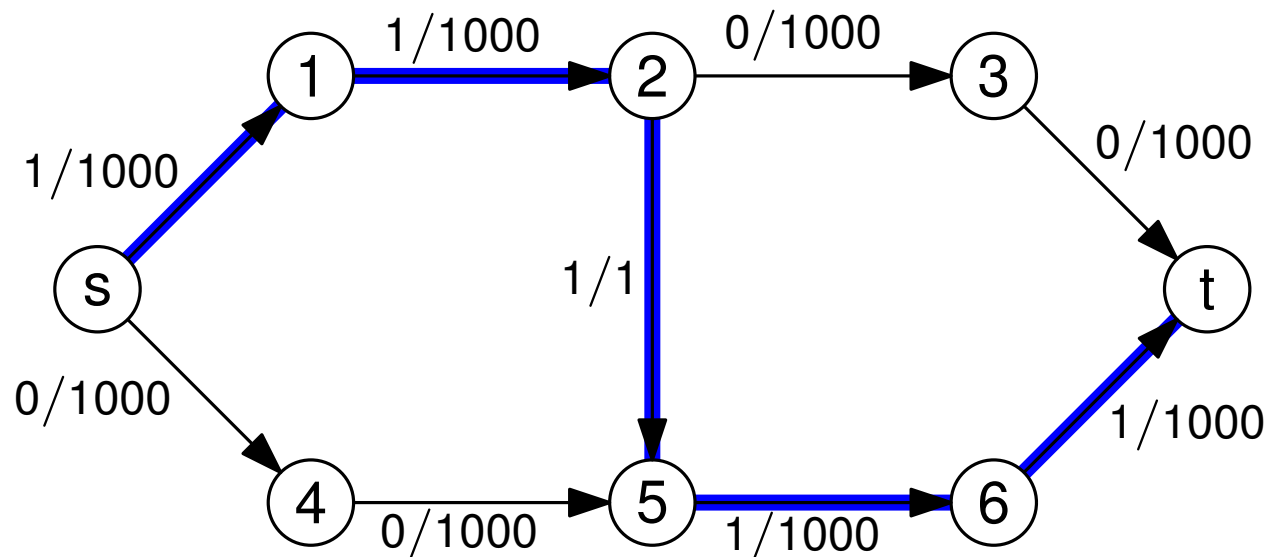
1. $f(i, j) \leftarrow 0$ für alle Kanten $(i, j) \in E$.

2. **Solange** ein erhöhender Weg $\langle e_1, e_2, \dots, e_k \rangle$ bezüglich f existiert **tue**

(a) $\delta \leftarrow \min(\{c(e_i) - f(e_i) \mid e_i \text{ ist Vorwärtskante}\} \cup \{f(e_i) \mid e_i \text{ ist Rückwärtskante}\})$

(b) Setze für alle e_i :

$$f(e_i) \leftarrow \begin{cases} f(e_i) + \delta & e_i \text{ ist Vorwärtskante} \\ f(e_i) - \delta & e_i \text{ ist Rückwärtskante} \end{cases}$$



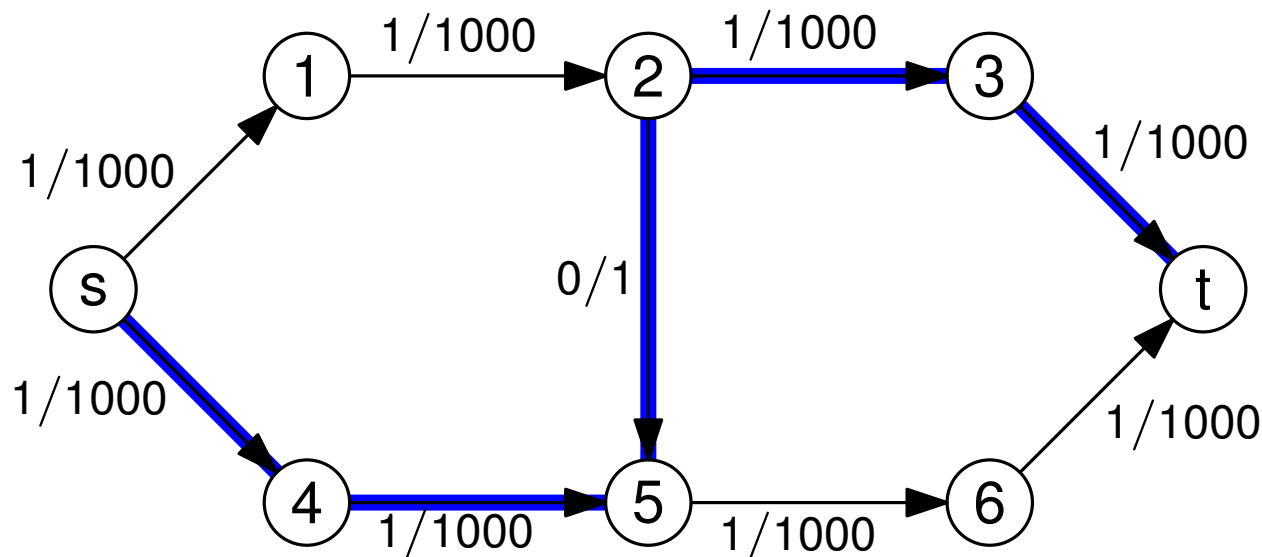
Schema des Ford-Fulkerson-Algorithmus

Eingabe: Netzwerk (D, s, t, c)

Ausgabe: Maximalfluss f bezüglich s und t

1. $f(i, j) \leftarrow 0$ für alle Kanten $(i, j) \in E$.
2. **Solange** ein erhöhender Weg $\langle e_1, e_2, \dots, e_k \rangle$ bezüglich f existiert **tue**
 - (a) $\delta \leftarrow \min(\{c(e_i) - f(e_i) \mid e_i \text{ ist Vorwärtskante}\} \cup \{f(e_i) \mid e_i \text{ ist Rückwärtskante}\})$
 - (b) Setze für alle e_i :

$$f(e_i) \leftarrow \begin{cases} f(e_i) + \delta & e_i \text{ ist Vorwärtskante} \\ f(e_i) - \delta & e_i \text{ ist Rückwärtskante} \end{cases}$$



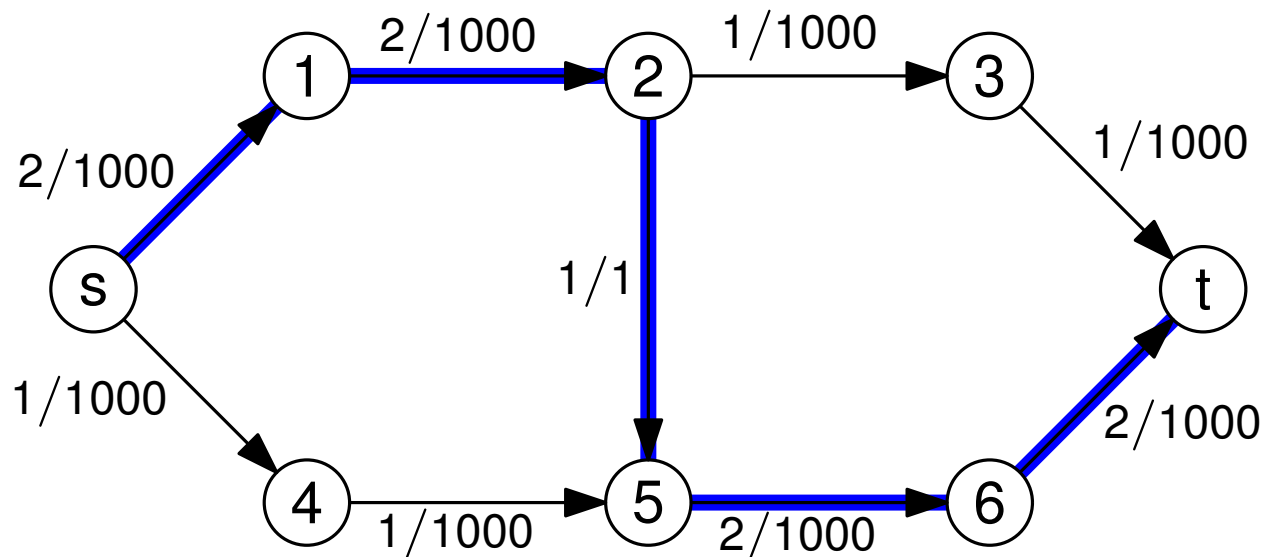
Schema des Ford-Fulkerson-Algorithmus

Eingabe: Netzwerk (D, s, t, c)

Ausgabe: Maximalfluss f bezüglich s und t

1. $f(i, j) \leftarrow 0$ für alle Kanten $(i, j) \in E$.
2. **Solange** ein erhöhender Weg $\langle e_1, e_2, \dots, e_k \rangle$ bezüglich f existiert **tue**
 - (a) $\delta \leftarrow \min(\{c(e_i) - f(e_i) \mid e_i \text{ ist Vorwärtskante}\} \cup \{f(e_i) \mid e_i \text{ ist Rückwärtskante}\})$
 - (b) Setze für alle e_i :

$$f(e_i) \leftarrow \begin{cases} f(e_i) + \delta & e_i \text{ ist Vorwärtskante} \\ f(e_i) - \delta & e_i \text{ ist Rückwärtskante} \end{cases}$$

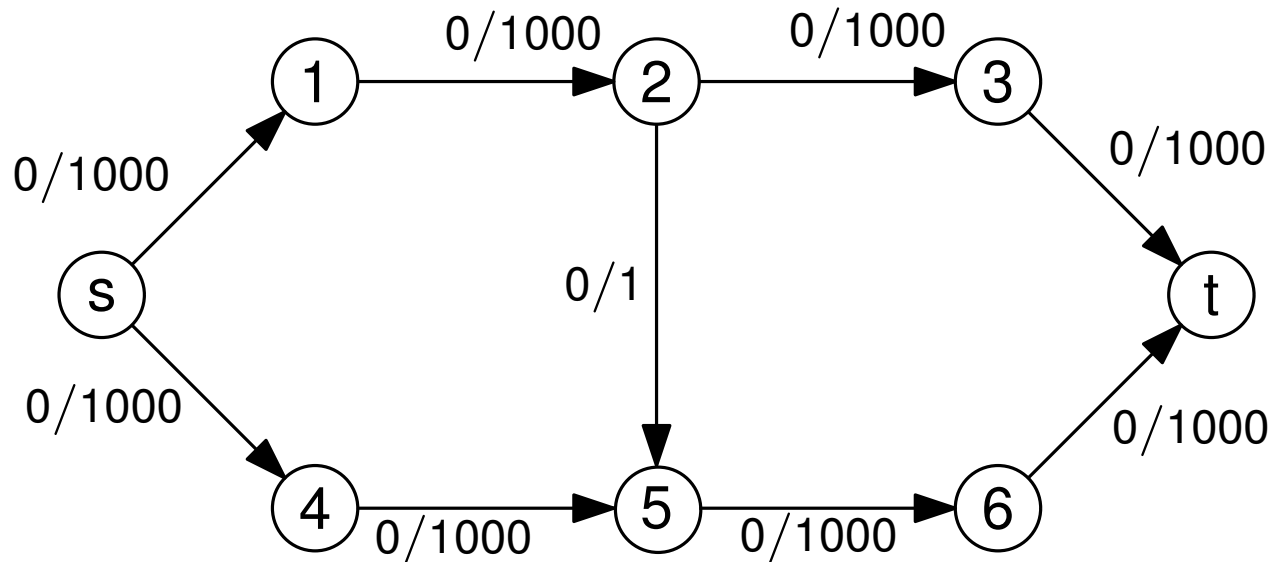


Besprechung des Algorithmus

- Die Laufzeit des Algorithmus hängt stark von der Wahl der erhöhenden Wege ab.
- Anzahl der Erhöhungen hängt auch von $\max\{c(i, j) \mid (i, j) \in E\}$ ab.
- Bei nicht rationalen Werten $c(i, j)$ ist nicht sicher gestellt, dass das Verfahren terminiert.
- Ansatzpunkt für Verbesserungen: Wahl der erhöhenden Wege.

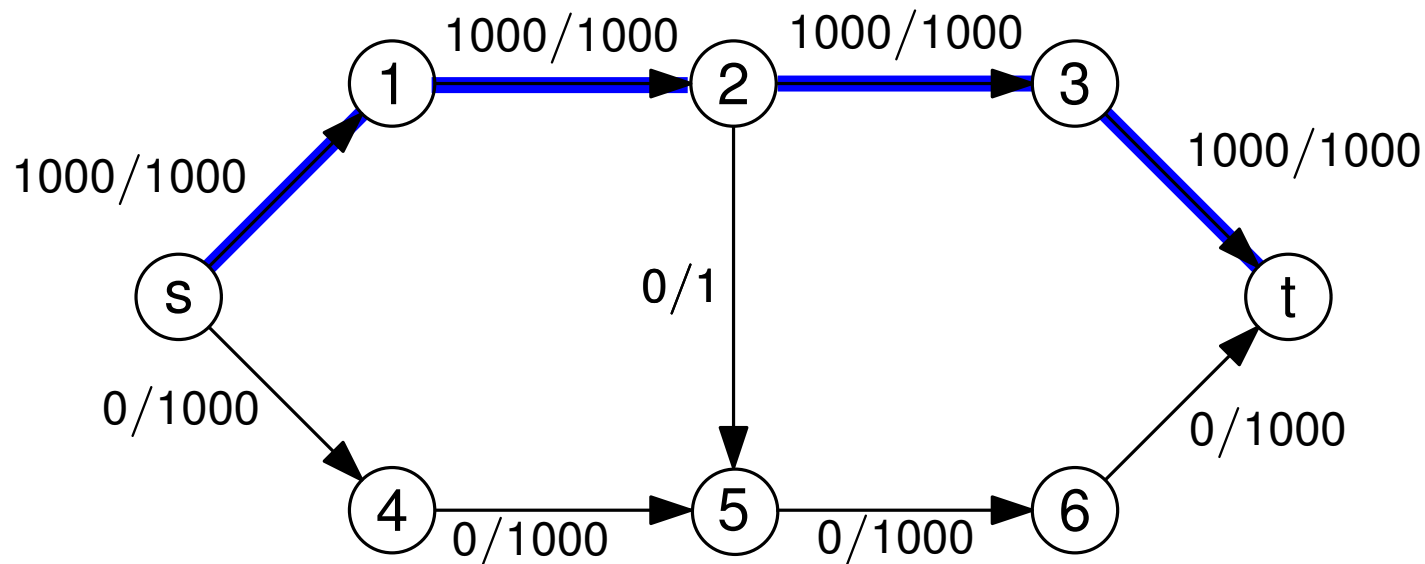
Algorithmus von Edmonds und Karp

- Verbesserung von Ford und Fulkerson Algorithmus.
- Wahl der erhöhenden Wege klar festgelegt:
 - Wähle immer kürzesten erhöhenden Weg (bezüglich Kantenzahl).
 - Kann mithilfe einer Art Breitensuche implementiert werden.
- Es werden $\mathcal{O}(|V| \cdot |E|)$ Erhöhungen durchgeführt, die jeweils $\mathcal{O}(|E|)$ Zeit kosten $\rightarrow \mathcal{O}(|V| \cdot |E|^2)$ Laufzeit. (Beweis siehe: Algorithmen – Eine Einführung, Cormen et al.)



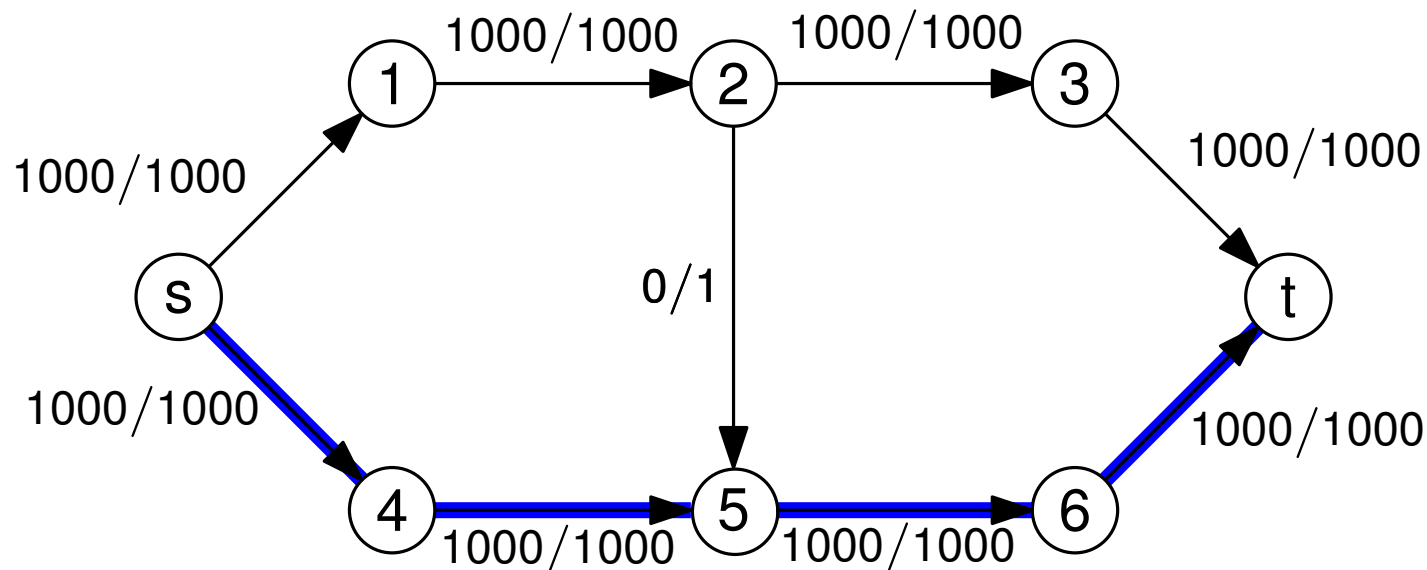
Algorithmus von Edmonds und Karp

- Verbesserung von Ford und Fulkerson Algorithmus.
- Wahl der erhöhenden Wege klar festgelegt:
 - Wähle immer kürzesten erhöhenden Weg (bezüglich Kantenzahl).
 - Kann mithilfe einer Art Breitensuche implementiert werden.
- Es werden $\mathcal{O}(|V| \cdot |E|)$ Erhöhungen durchgeführt, die jeweils $\mathcal{O}(|E|)$ Zeit kosten $\rightarrow \mathcal{O}(|V| \cdot |E|^2)$ Laufzeit. (Beweis siehe: Algorithmen – Eine Einführung, Cormen et al.)



Algorithmus von Edmonds und Karp

- Verbesserung von Ford und Fulkerson Algorithmus.
- Wahl der erhöhenden Wege klar festgelegt:
 - Wähle immer kürzesten erhöhenden Weg (bezüglich Kantenzahl).
 - Kann mithilfe einer Art Breitensuche implementiert werden.
- Es werden $\mathcal{O}(|V| \cdot |E|)$ Erhöhungen durchgeführt, die jeweils $\mathcal{O}(|E|)$ Zeit kosten $\rightarrow \mathcal{O}(|V| \cdot |E|^2)$ Laufzeit. (Beweis siehe: Algorithmen – Eine Einführung, Cormen et al.)



Lineare Programme (LP)

Ein lineares Programm besteht aus

1. **Variablen:**

$$\bar{x} = (x_1, \dots, x_n)^T$$

2. einer **linearen Zielfunktion:**

$$f(\bar{x}) = c_1 \cdot x_1 + \dots + c_n \cdot x_n$$

3. **Nebenbedingungen:**

$$a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 + \dots + a_{1,n} \cdot x_n \leq b_1$$

...

$$a_{m,1} \cdot x_1 + a_{m,2} \cdot x_2 + \dots + a_{m,n} \cdot x_n \leq b_m$$

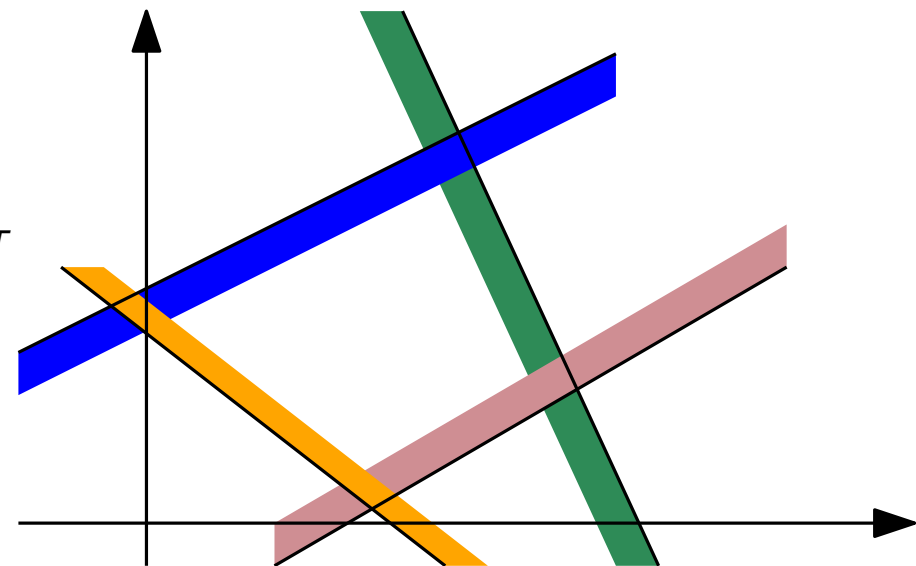
Ziel: bestimme x_1, \dots, x_n so, dass $f(\bar{x})$ maximal/minimal ist.

Matrixschreibweise des LP:

$$A\bar{x} \leq \bar{b} \text{ mit } A = (a_{i,j})$$

$$f(\bar{x}) = \bar{x}^T \bar{c}$$

$$\text{mit } \bar{c} = (c_1, \dots, c_n)^T \text{ und } \bar{b} = (b_1, \dots, b_m)^T$$



Lösungsraum 2-dimensionales LP

Beispiel: Bäckerei

	<i>Weizenmehl</i>	<i>Wasser</i>	<i>Mischkornschrot</i>
1 Kiste Weizenmischbrot (20 €)	12 kg	8 kg	0 kg
1 Kiste Mehrkornbrot (60 €)	6 kg	12 kg	10 kg
Kontingent	630 kg	620 kg	350 kg

Weitere Bedingungen:

- 10 Kisten Weizenmischbrote sind für Stammkunden reserviert.
- Bäcker möchte Gewinn maximieren.

Beispiel: Bäckerei

	<i>Weizenmehl</i>	<i>Wasser</i>	<i>Mischkornschrot</i>
1 Kiste Weizenmischbrot (20 €)	12 kg	8 kg	0 kg
1 Kiste Mehrkornbrot (60 €)	6 kg	12 kg	10 kg
Kontingent	630 kg	620 kg	350 kg

Weitere Bedingungen:

- 10 Kisten Weizenmischbrote sind für Stammkunden reserviert.
- Bäcker möchte Gewinn maximieren.

x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

$$\begin{array}{l}
 \text{Zielfunktion **ZF**:} \quad f(x_1, x_2) = \quad 20 \quad x_1 \quad + \quad 60 \quad x_2 \quad = \text{max!} \\
 \text{Nebenbedingungen **NB**:} \quad \quad \quad \quad 12 \quad x_1 \quad + \quad 6 \quad x_2 \quad \leq 630 \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad 8 \quad x_1 \quad + \quad 12 \quad x_2 \quad \leq 620 \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad 10 \quad x_2 \quad \leq 350 \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad x_1 \quad \geq 10 \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad x_1 \quad \geq 0 \\
 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad x_2 \quad \geq 0
 \end{array}$$

Geometrische Interpretation



x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion ZF:	$f(x_1, x_2) =$	20	x_1	+	60	x_2	= max!	
Nebenbedingungen NB:		12	x_1	+	6	x_2	≤ 630	Weizen
		8	x_1	+	12	x_2	≤ 620	Wasser
					10	x_2	≤ 350	Körner
			x_1				≥ 10	Stammkunden
			x_1				≥ 0	
			x_2				≥ 0	

Geometrische Interpretation



x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion **ZF:** $f(x_1, x_2) = 20x_1 + 60x_2 = \max!$

Nebenbedingungen **NB:** $12x_1 + 6x_2 \leq 630$ Weizen

$8x_1 + 12x_2 \leq 620$ Wasser

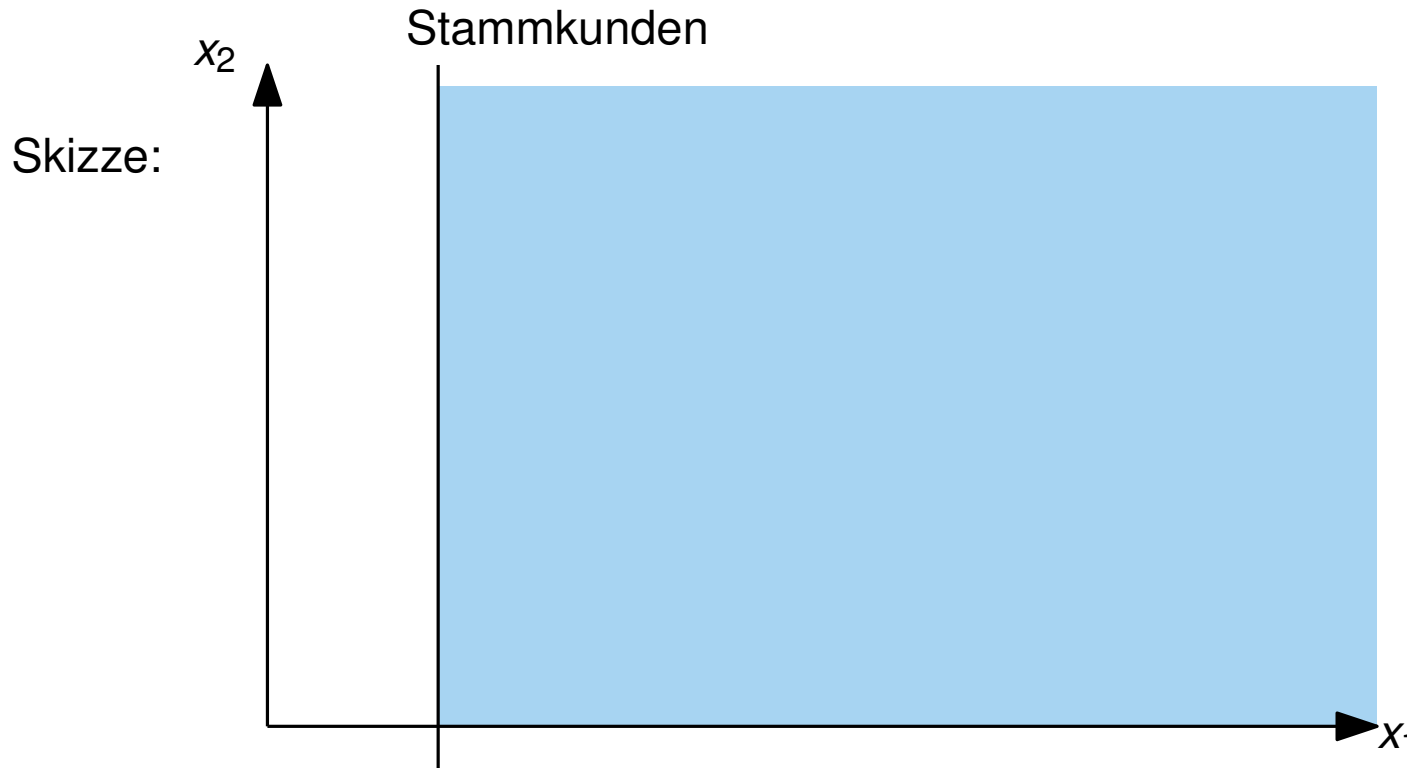
$10x_2 \leq 350$ Körner

$x_1 \geq 10$ Stammkunden

$x_1 \geq 0$

$x_2 \geq 0$

Geometrische Interpretation



x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion **ZF:** $f(x_1, x_2) = 20x_1 + 60x_2 = \max!$

Nebenbedingungen **NB:** $12x_1 + 6x_2 \leq 630$ Weizen

$8x_1 + 12x_2 \leq 620$ Wasser

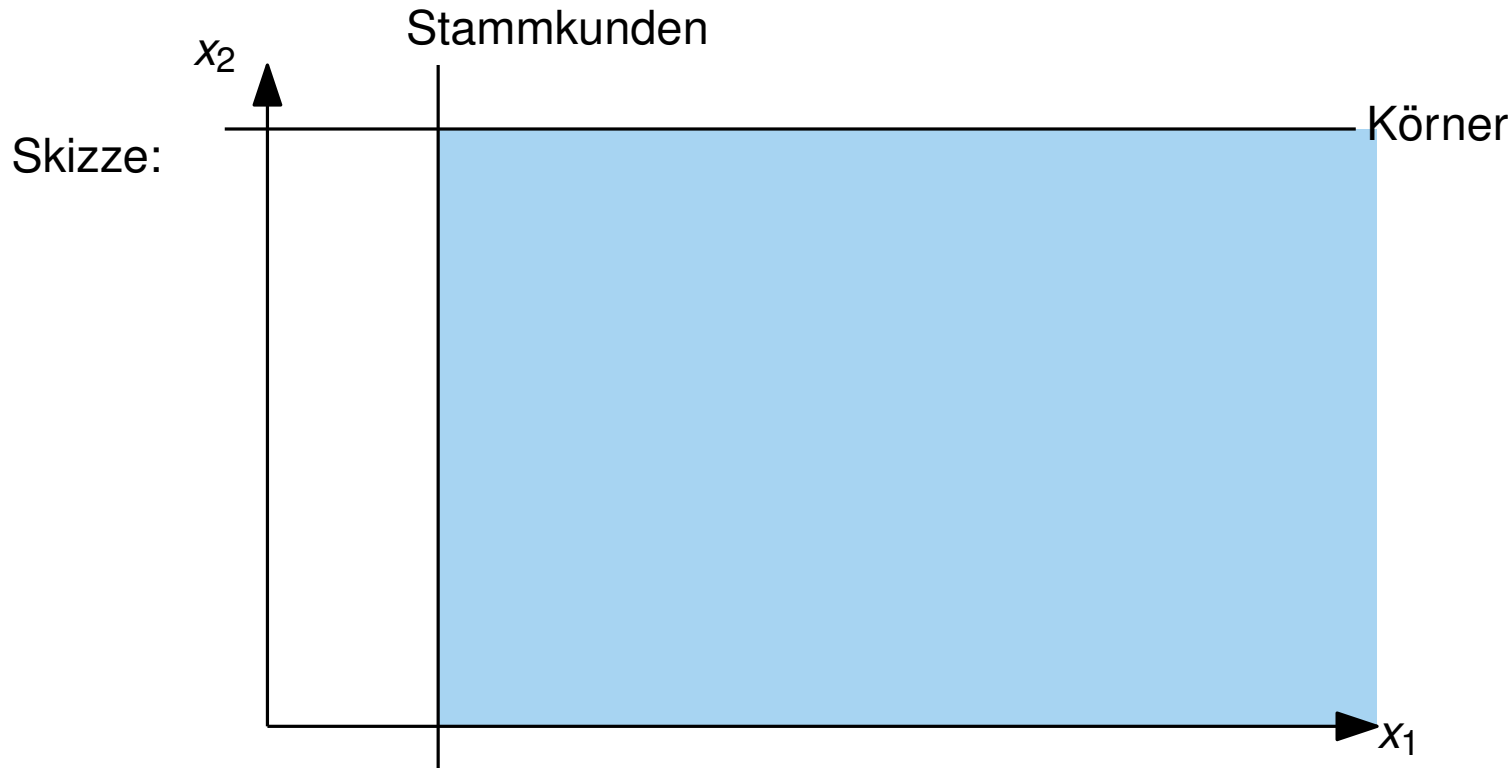
$10x_2 \leq 350$ Körner

$x_1 \geq 10$ Stammkunden

$x_1 \geq 0$

$x_2 \geq 0$

Geometrische Interpretation



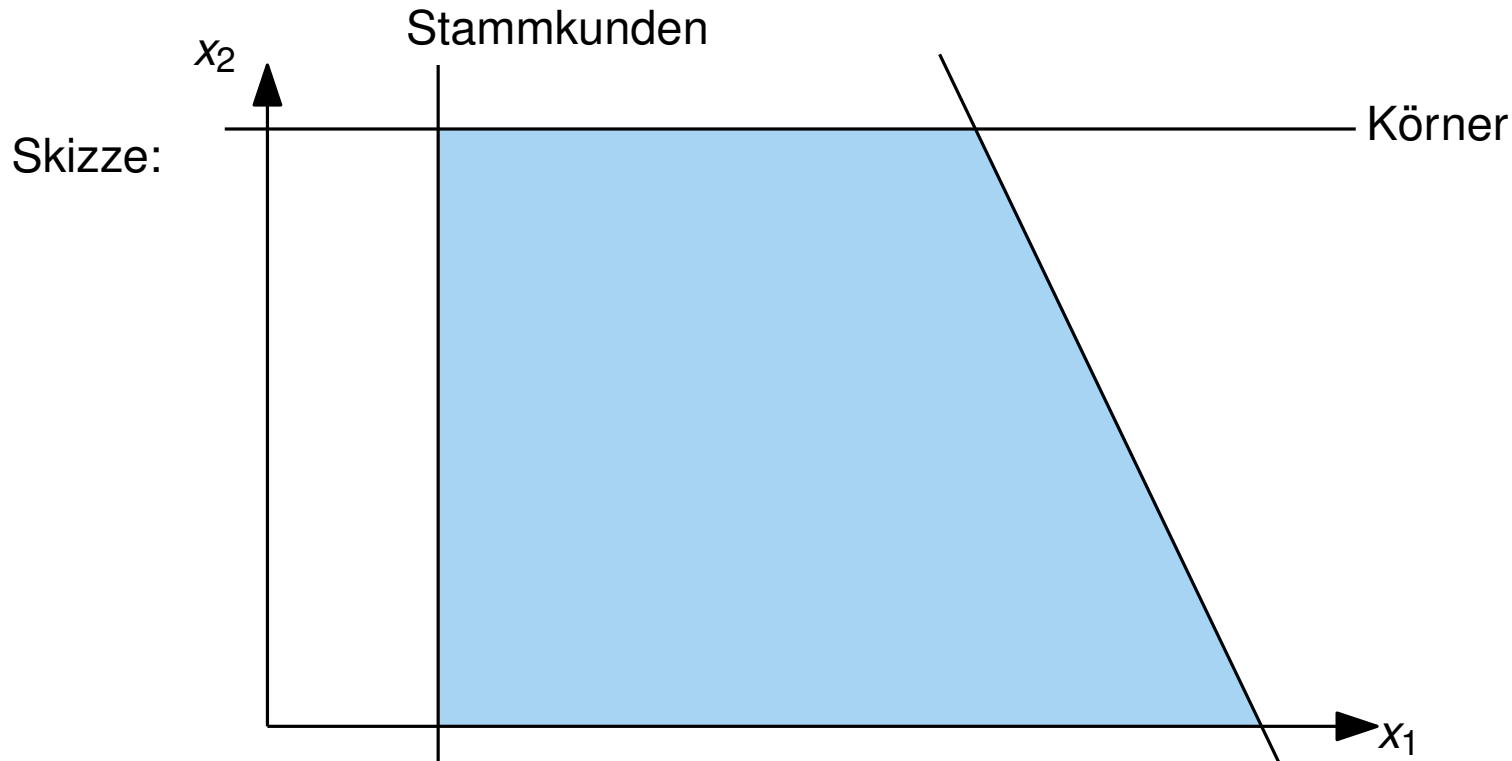
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion **ZF:** $f(x_1, x_2) = 20x_1 + 60x_2 = \max!$

Nebenbedingungen **NB:**

12	x_1	+	6	x_2	≤ 630	Weizen
8	x_1	+	12	x_2	≤ 620	Wasser
			10	x_2	≤ 350	Körner
	x_1				≥ 10	Stammkunden
	x_1				≥ 0	
	x_2				≥ 0	

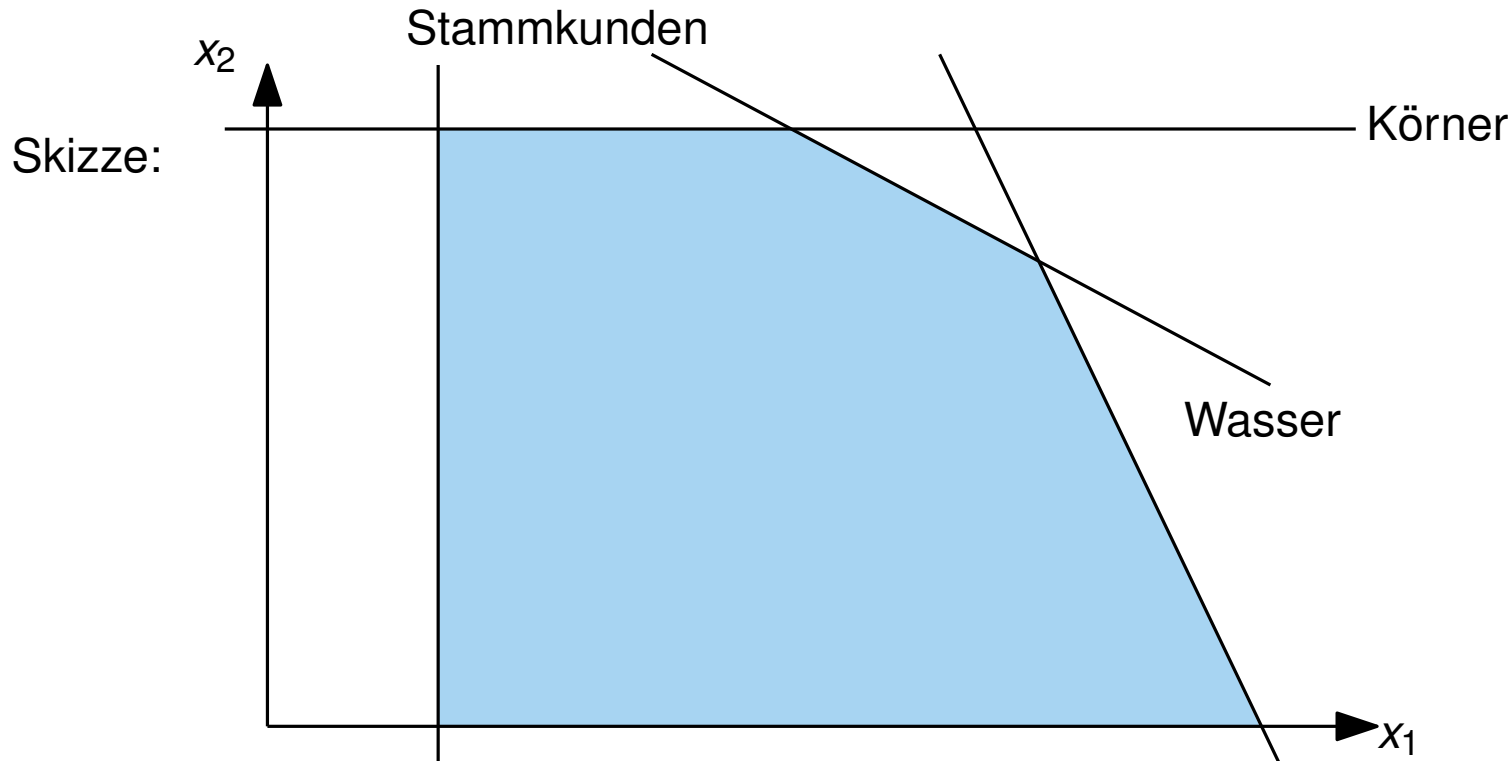
Geometrische Interpretation



x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion ZF:	$f(x_1, x_2) =$	20	x_1	+	60	x_2	= max!	
Nebenbedingungen NB:		12	x_1	+	6	x_2	≤ 630	Weizen
		8	x_1	+	12	x_2	≤ 620	Wasser
					10	x_2	≤ 350	Körner
			x_1				≥ 10	Stammkunden
			x_1				≥ 0	
			x_2				≥ 0	

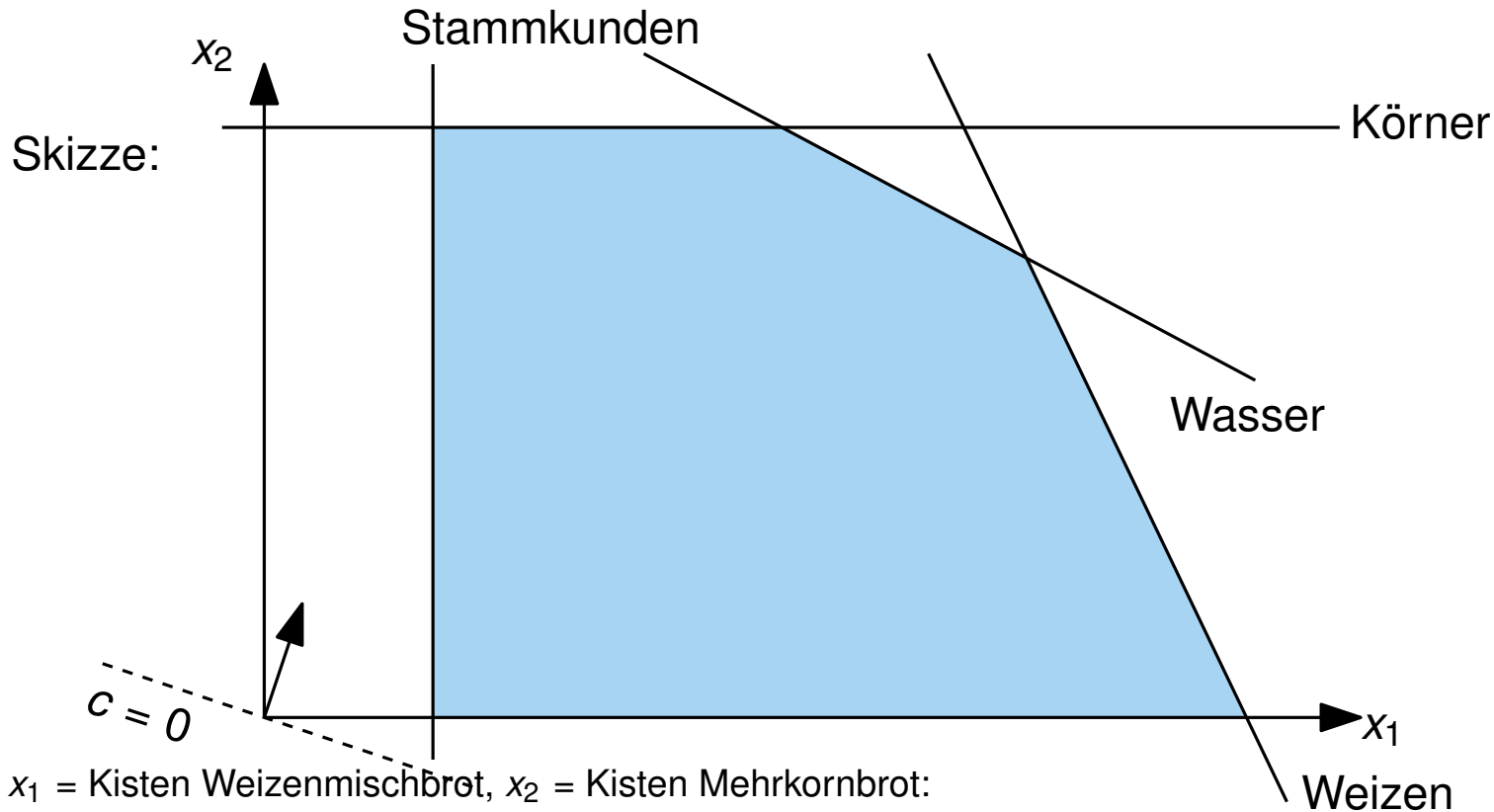
Geometrische Interpretation



x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion ZF:	$f(x_1, x_2) =$	20	x_1	+	60	x_2	= max!	
Nebenbedingungen NB:		12	x_1	+	6	x_2	≤ 630	Weizen
		8	x_1	+	12	x_2	≤ 620	Wasser
					10	x_2	≤ 350	Körner
			x_1				≥ 10	Stammkunden
			x_1				≥ 0	
			x_2				≥ 0	

Geometrische Interpretation



Zielfunktion **ZF:** $f(x_1, x_2) = 20 x_1 + 60 x_2 = \max!$

Nebenbedingungen **NB:**

$$12 x_1 + 6 x_2 \leq 630 \quad \text{Weizen}$$

$$8 x_1 + 12 x_2 \leq 620 \quad \text{Wasser}$$

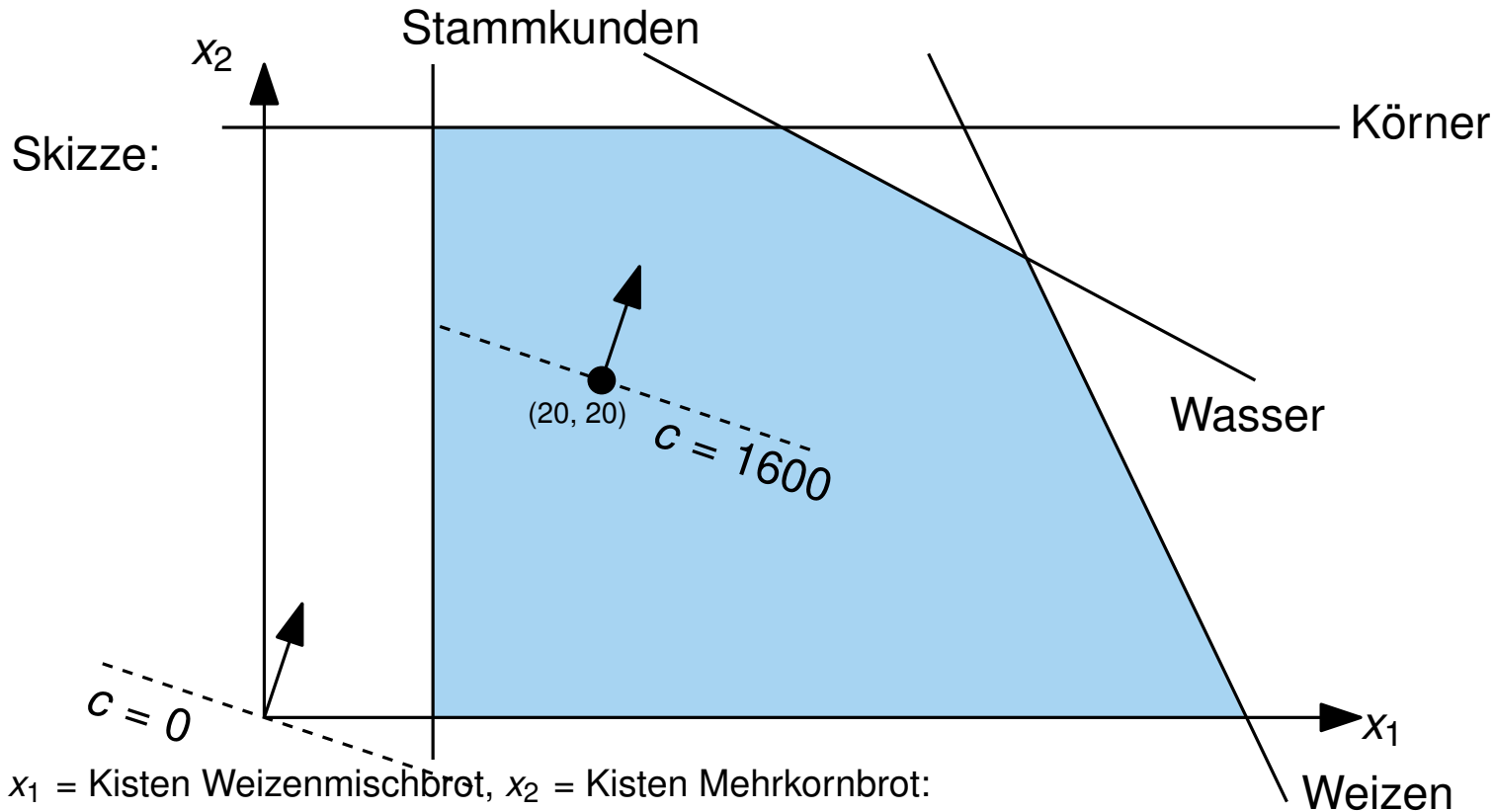
$$10 x_2 \leq 350 \quad \text{Körner}$$

$$x_1 \geq 10 \quad \text{Stammkunden}$$

$$x_1 \geq 0$$

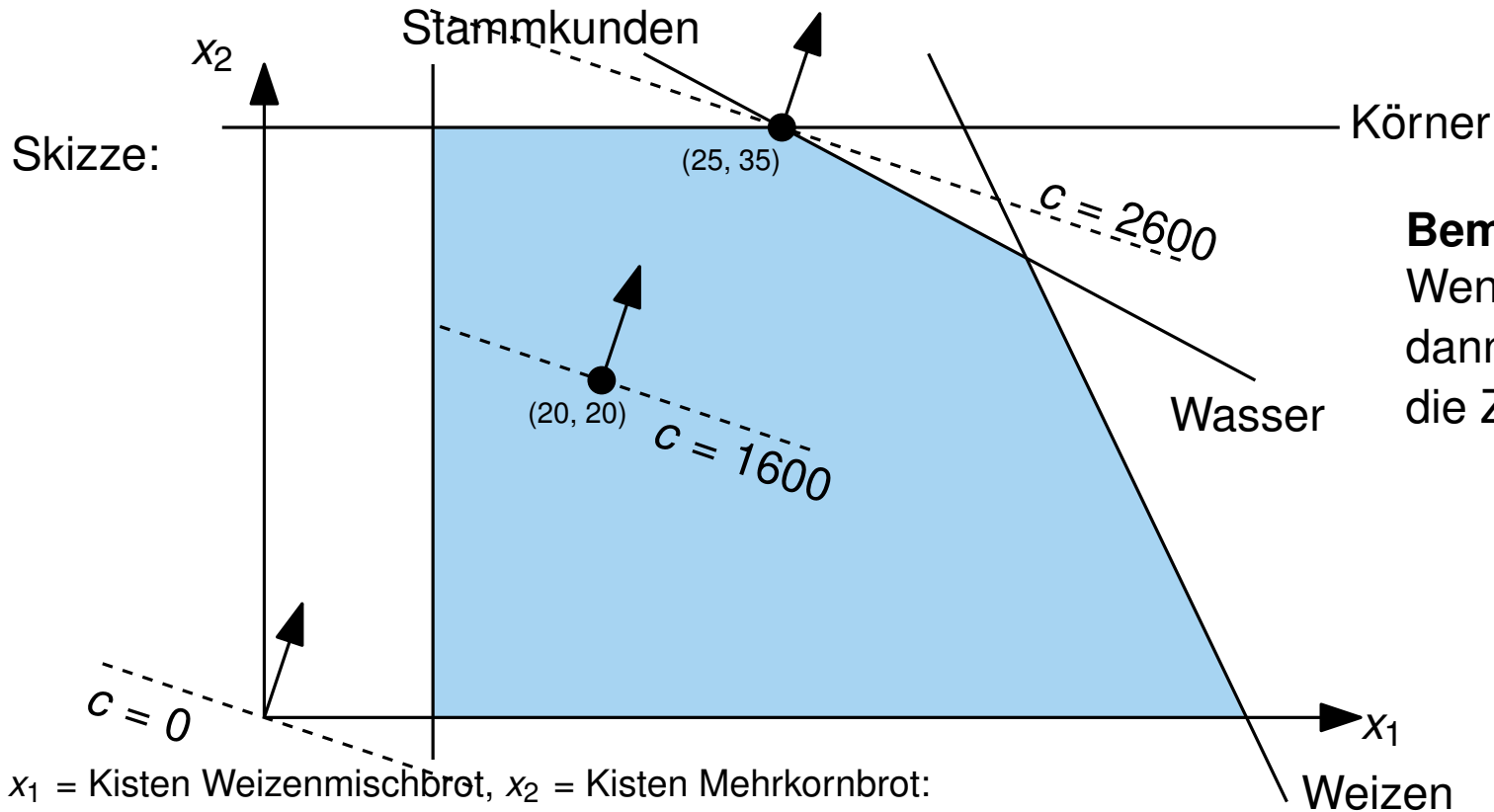
$$x_2 \geq 0$$

Geometrische Interpretation



Zielfunktion ZF:	$f(x_1, x_2) =$	20	x_1	+	60	x_2	= max!	
Nebenbedingungen NB:		12	x_1	+	6	x_2	≤ 630	Weizen
		8	x_1	+	12	x_2	≤ 620	Wasser
					10	x_2	≤ 350	Körner
			x_1				≥ 10	Stammkunden
			x_1				≥ 0	
			x_2				≥ 0	

Geometrische Interpretation



Bemerkung:

Wenn es eine Lösung gibt, dann gibt es eine *Ecke*, die die Zielfunktion maximiert.

x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion ZF:	$f(x_1, x_2) =$	20	x_1	+	60	x_2	= max!	
Nebenbedingungen NB:		12	x_1	+	6	x_2	≤ 630	Weizen
		8	x_1	+	12	x_2	≤ 620	Wasser
					10	x_2	≤ 350	Körner
			x_1				≥ 10	Stammkunden
			x_1				≥ 0	
			x_2				≥ 0	

Flussproblem als Lineares Programm

Betrachte das Netzwerk (D, s, t, c) :

Führe für jede Kante (i, j) eine **Variable** $x_{i,j}$ ein.

Idee: $x_{i,j}$ gibt den Fluss an, der über die Kante (i, j) fließt.

Maximiere den Wert des Flusses:

$$f(\bar{x}) = \sum_{(s,i) \in E} x_{s,i} - \sum_{(i,s) \in E} x_{i,s}$$

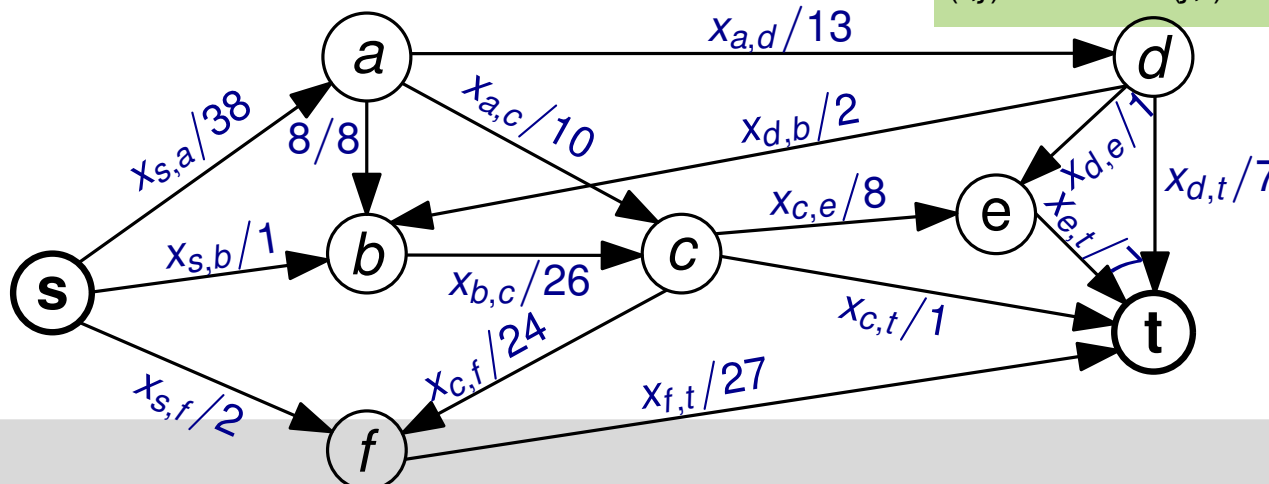
Unter den Bedingungen:

1. *Kapazitätsbedingung:* für alle $(i, j) \in E$

- $0 \leq x_{i,j}$
- $x_{i,j} \leq c(i, j)$

2. *Flusserhaltung:* für alle $i \in V \setminus \{s, t\}$

$$\sum_{(i,j) \in E} x_{i,j} - \sum_{(j,i) \in E} x_{j,i} = 0$$



Primales Programm:

$$f(\bar{x}) = \bar{x}^T \bar{c} = \max!$$

$$A\bar{x} \leq \bar{b}$$

$$\bar{x} \geq 0$$

$$N = \{\bar{x} \in \mathbb{R}^n \mid A\bar{x} \leq \bar{b}, \bar{x} \geq 0\}$$

Duales Programm:

$$g(\bar{y}) = \bar{y}^T \bar{b} = \min!$$

$$\bar{y}^T A \geq \bar{c}$$

$$\bar{y} \geq 0$$

$$M = \{\bar{y} \in \mathbb{R}^m \mid \bar{y}^T A \geq \bar{c}, \bar{y} \geq 0\}$$

Schwacher Dualitätssatz: Für alle zulässigen Lösungen $\bar{x} \in N$ und $\bar{y} \in M$ des primalen bzw. dualen Programms gilt

$$\bar{x}^T \bar{c} \leq \bar{y}^T \bar{b}$$

Starker Dualitätssatz:

Primales Programm lösbar \Leftrightarrow zugehöriges duales Programm lösbar

und wenn lösbar, dann $\max_{\bar{x} \in N} f(\bar{x}) = \min_{\bar{y} \in M} g(\bar{y})$