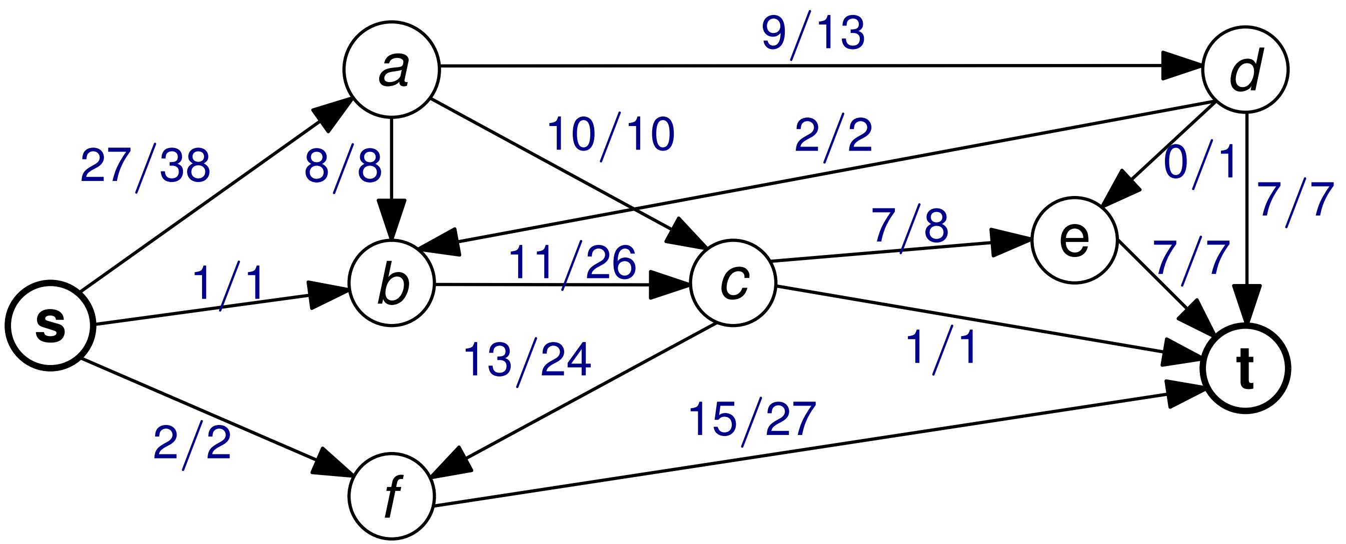
Karlsruher Institut für Technologie

Algorithmen II
Vorlesung am 24.10.2013

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Flussprobleme und Dualität



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problemdefinition

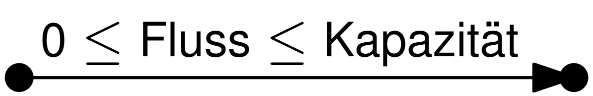
Definition:

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

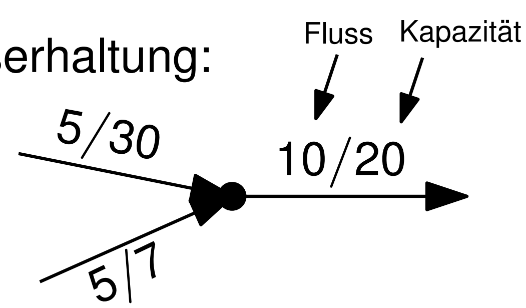
Das Tupel (D, s, t, c) heißt *Netzwerk*. Eine Abbildung $f \colon E \to \mathbb{R}_0^+$ heißt *Fluss*, wenn folgende Bedingungen gelten:

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:

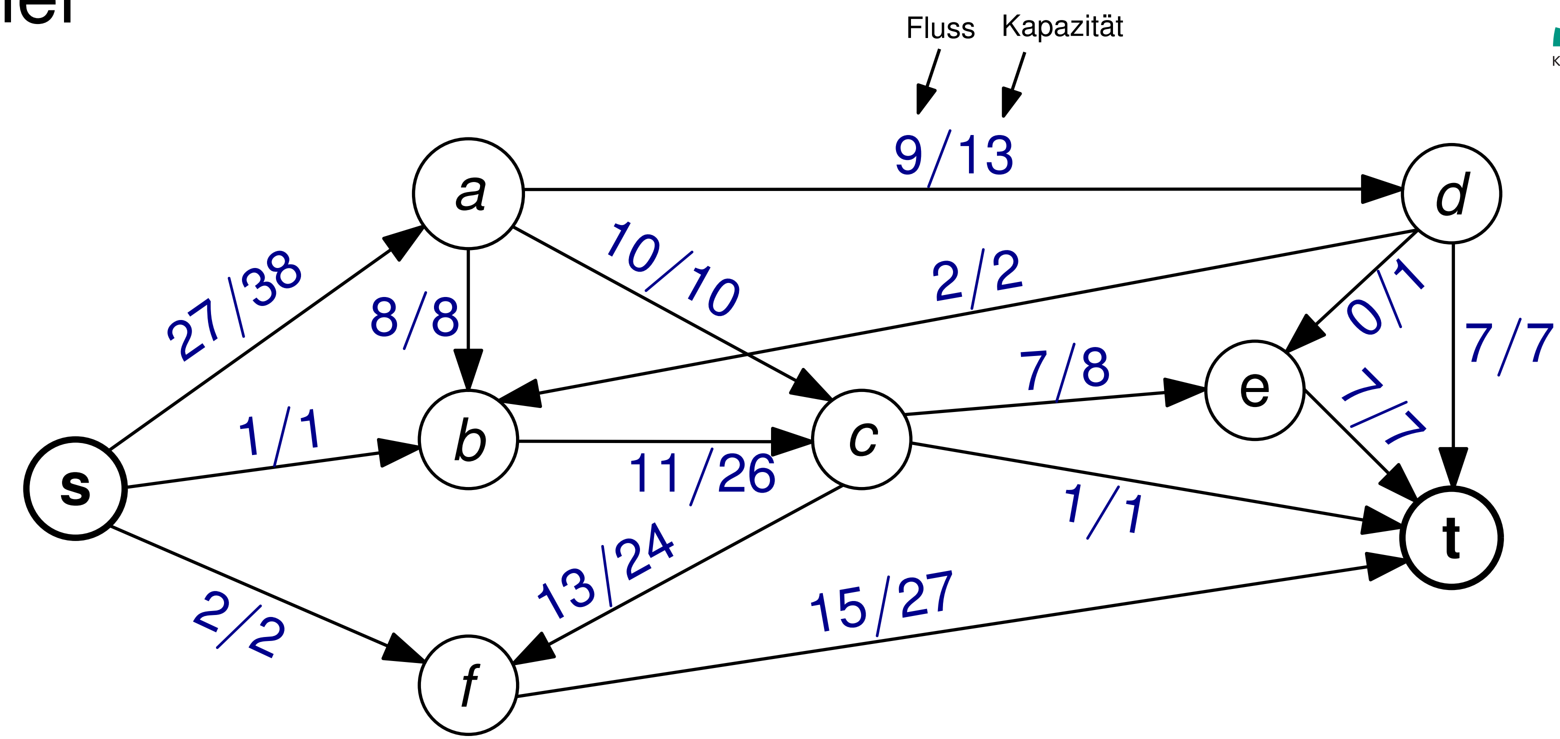


Flusserhaltung:



Zwischenknoten können Fluss weder konsumieren noch produzieren.

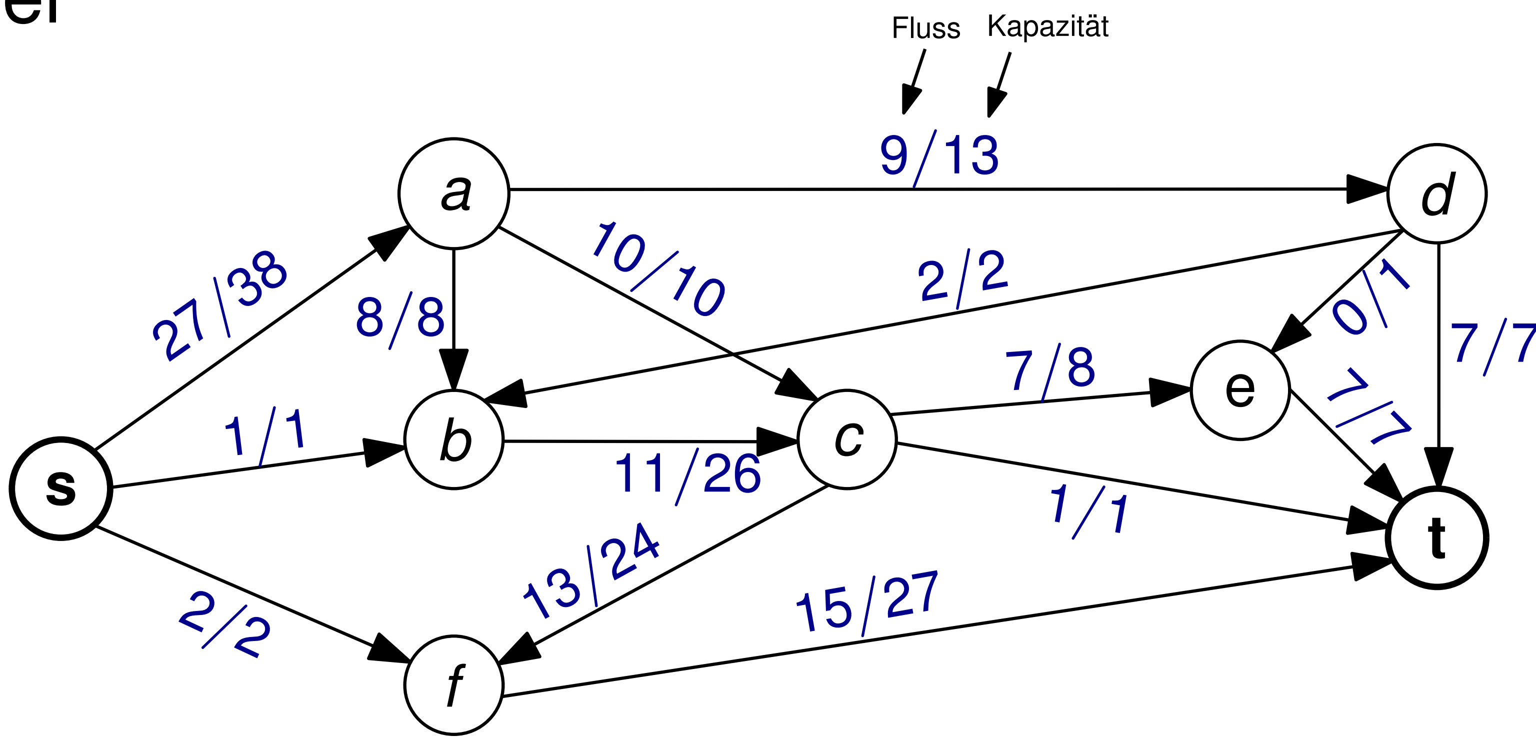
Beispiel



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 <u>besseren</u> *Fluss*?

Was heißt besser? / Wie Fluss messen?

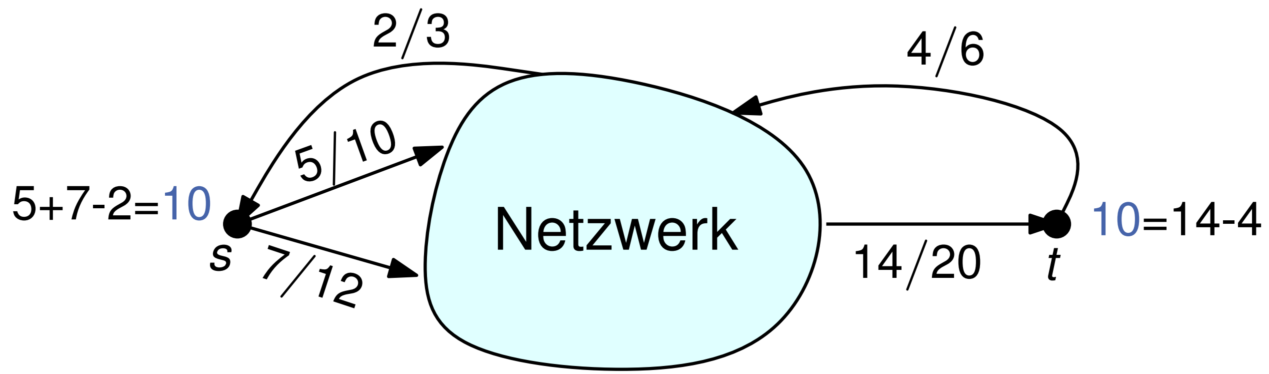
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Flusserhaltung und Wert eines Flusses

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).



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

Flusserhaltung und Wert eines Flusses

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

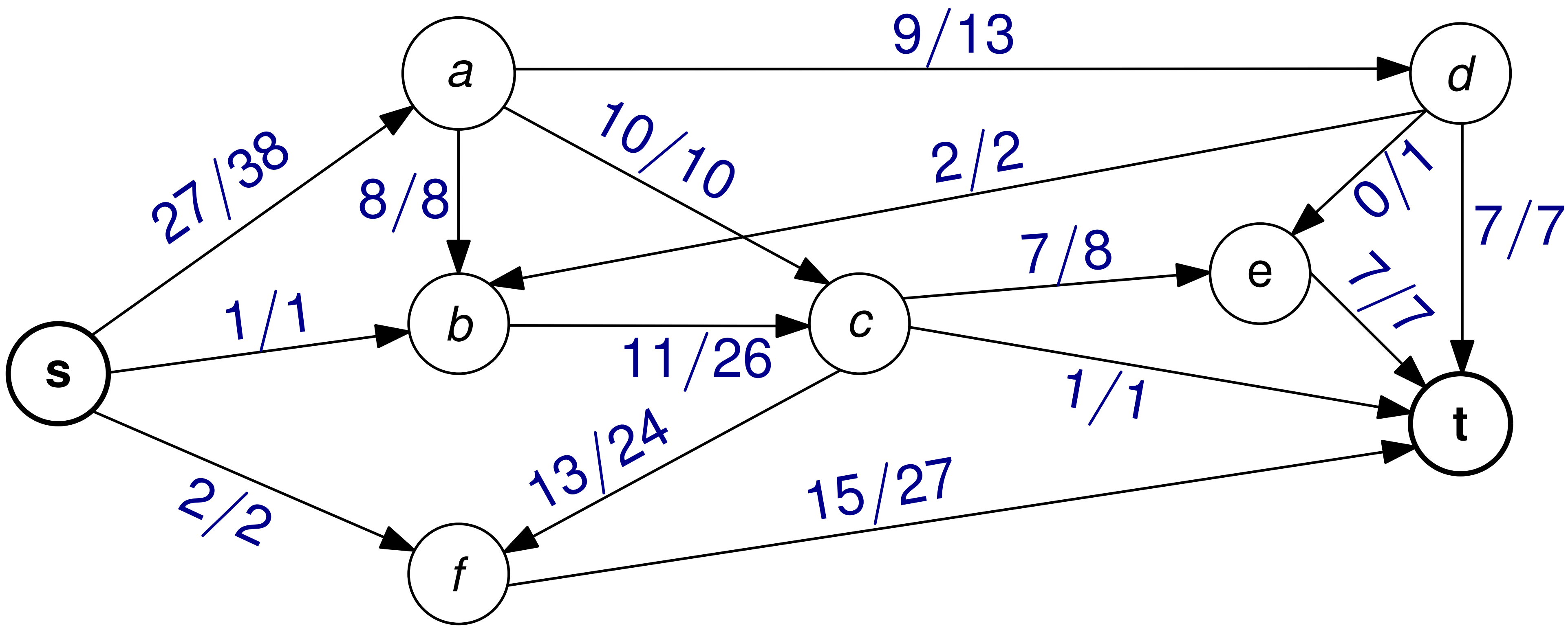
$$\sum_{(s,i)\in E} f(s,i) - \sum_{(i,s)\in E} f(i,s) = \sum_{(i,t)\in E} f(i,t) - \sum_{(t,i)\in E} f(t,i)$$

$w(f) := \sum_{(s,i)\in E} f(s,i) - \sum_{(i,s)\in E} f(i,s)$ 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 $w(f') \leq w(f)$.

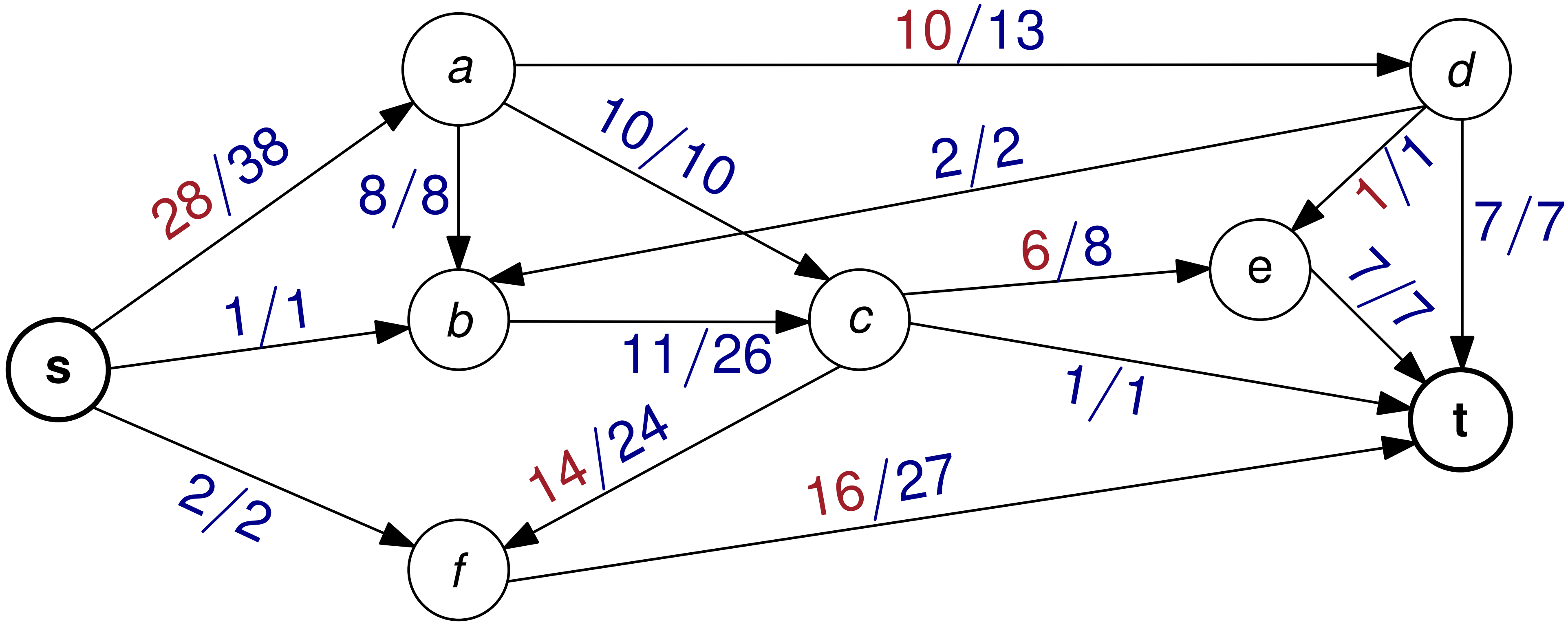
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Beispiel



Wie Fluss verbessern, dass er maximal ist?

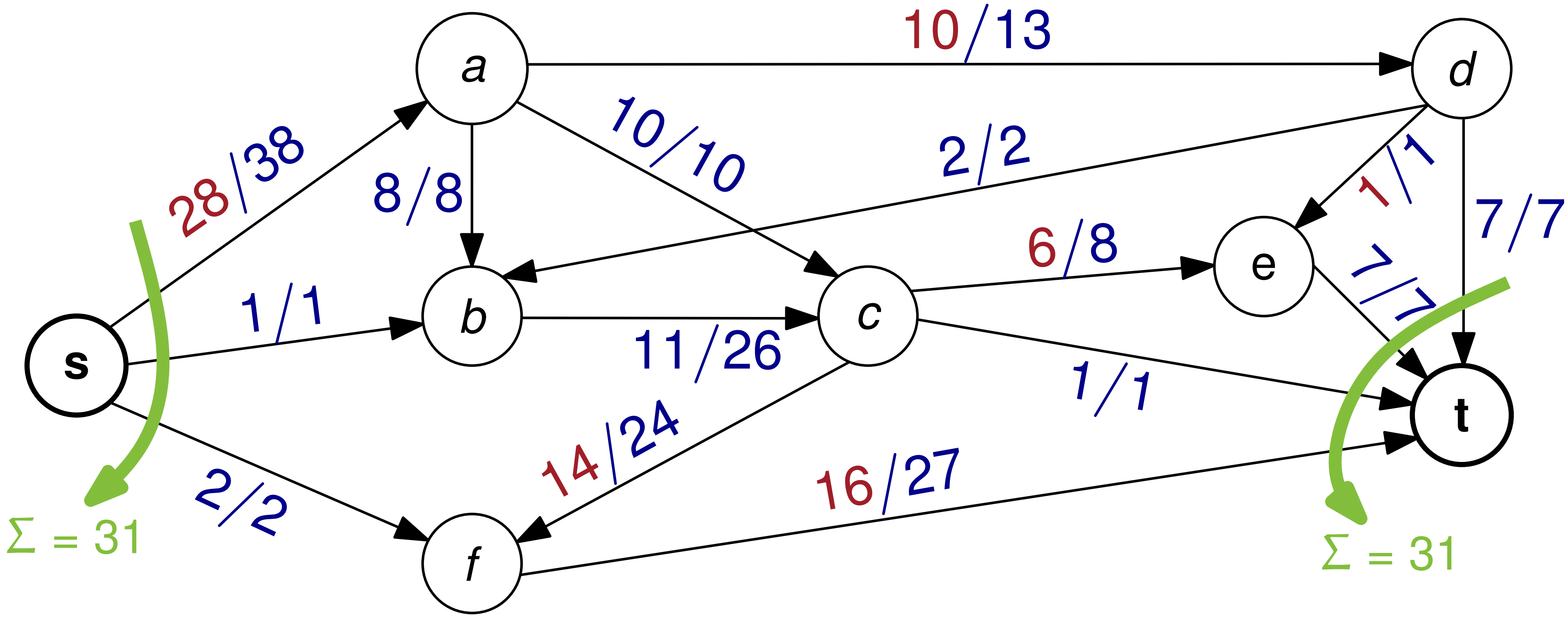
Beispiel



Welchen Wert besitzt dieser Fluss?

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Beispiel



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?

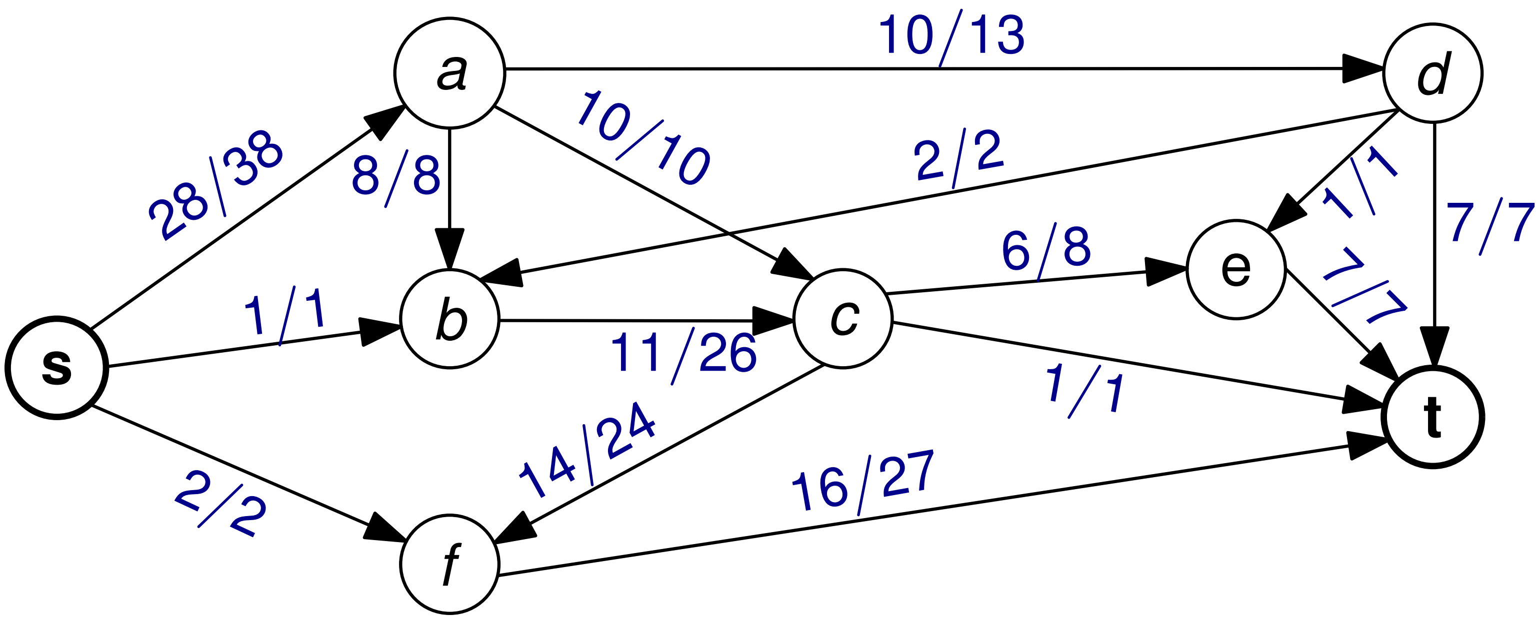
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.



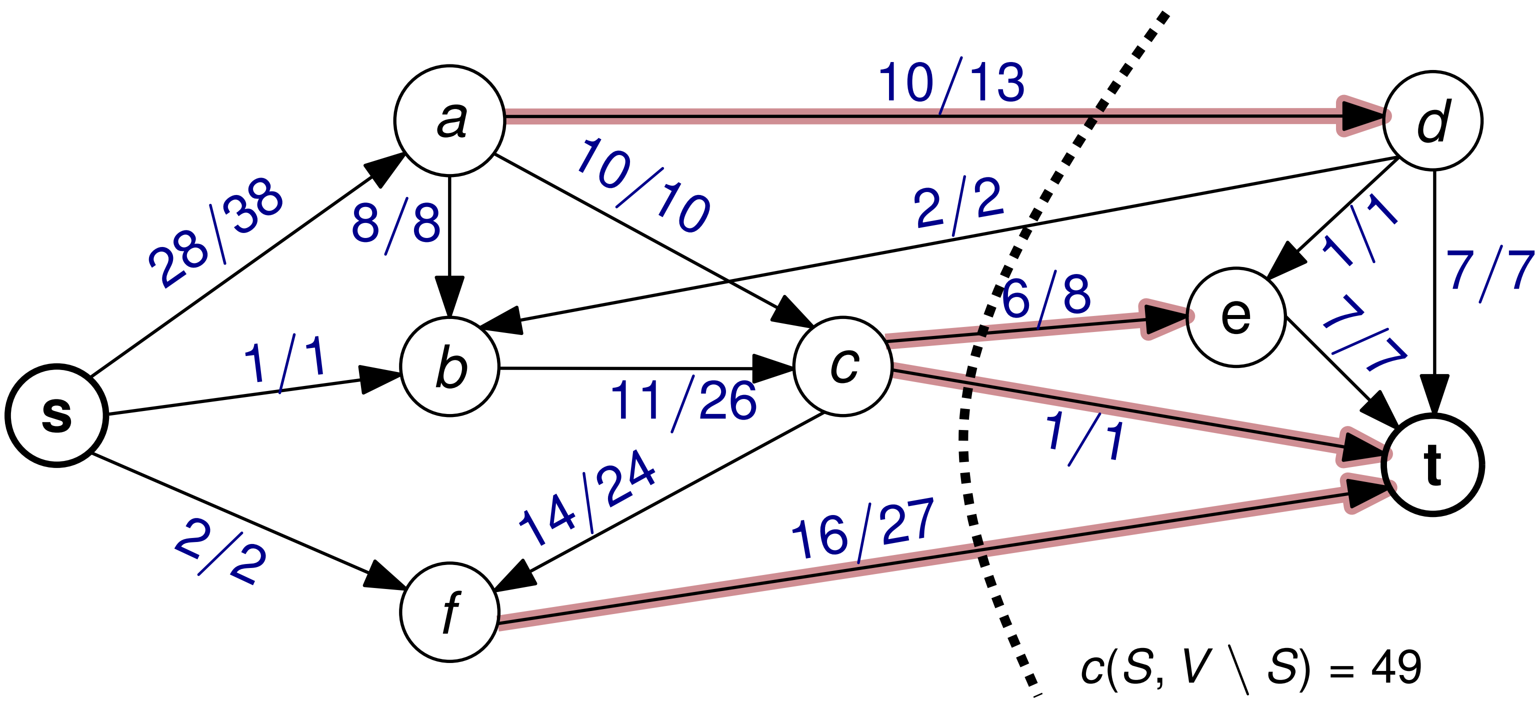
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.



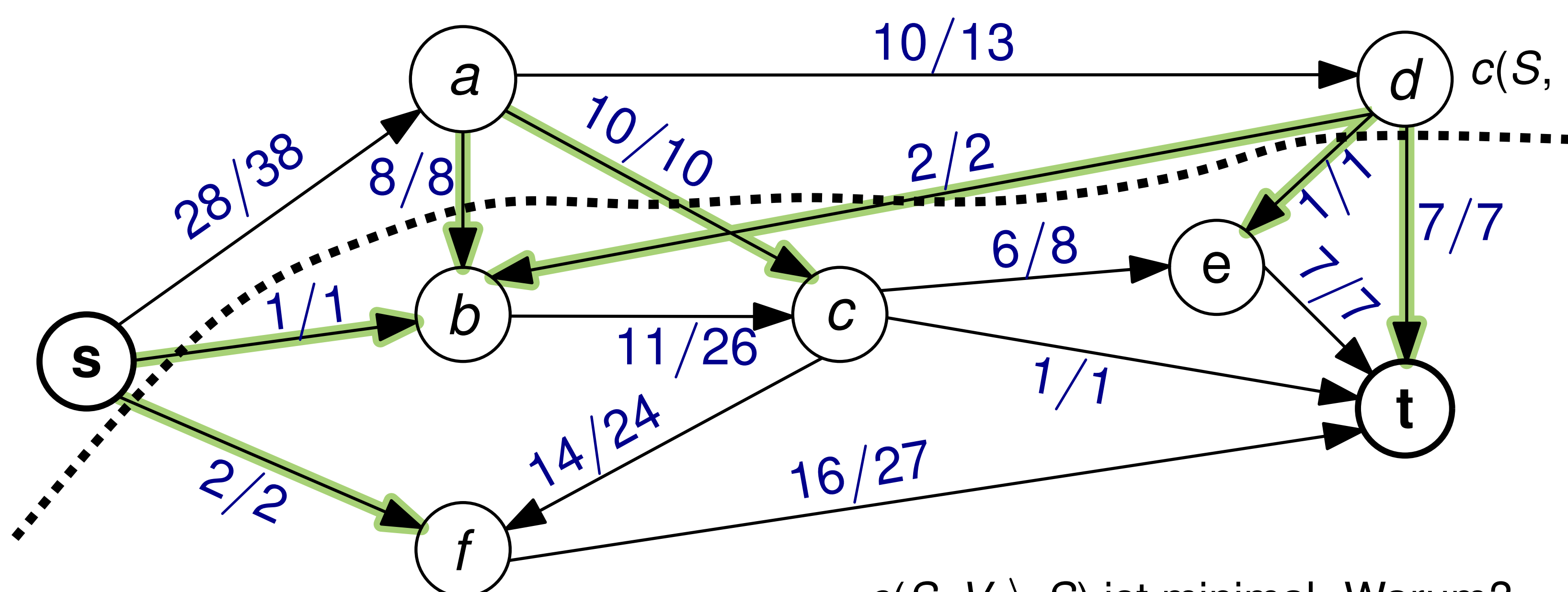
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.



$c(S, V \setminus S) = 31$

$c(S, V \setminus S)$ ist minimal. Warum?

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.

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

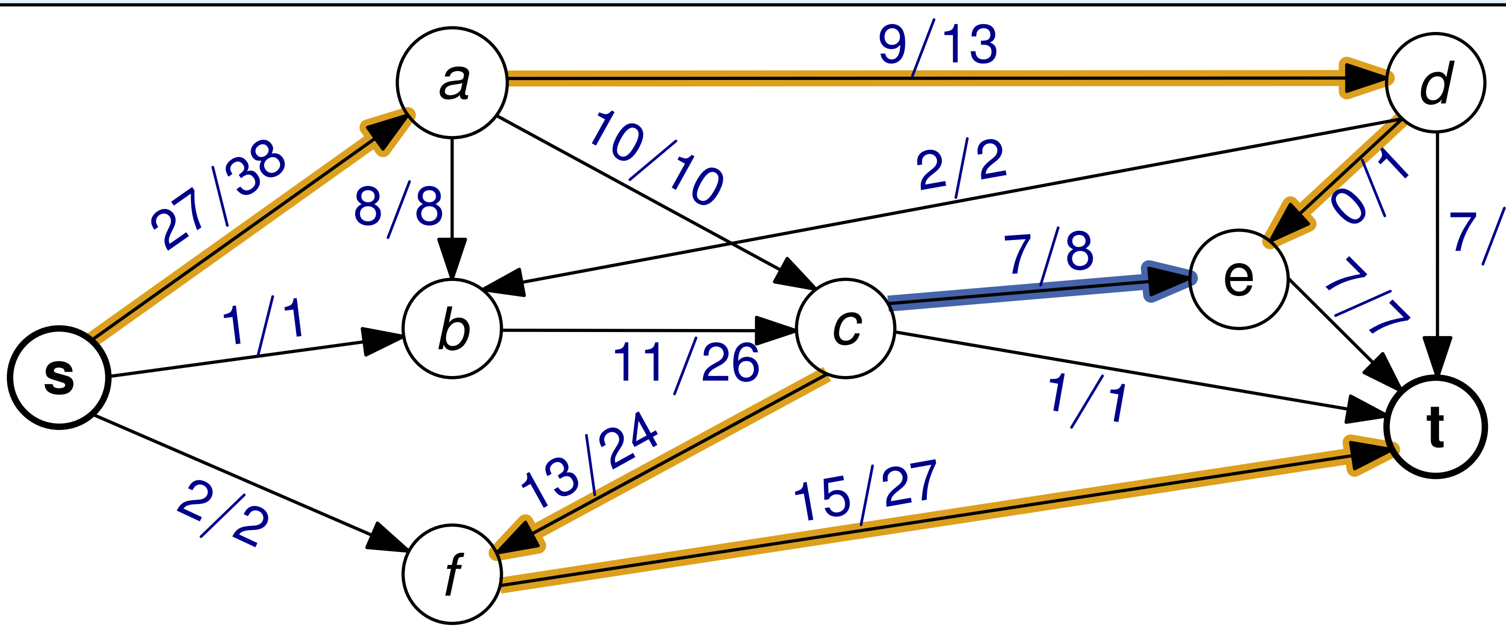
Erhöhende Wege

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*.

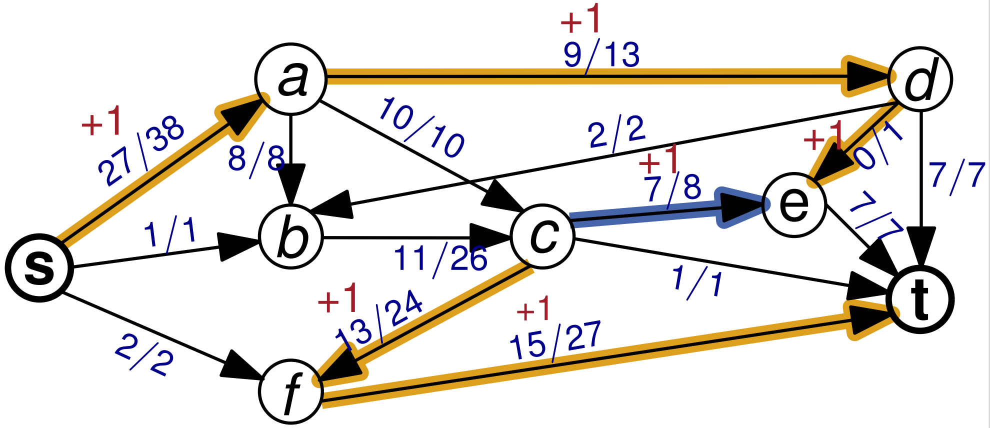
Der Weg P heißt *erhöhender Weg* bezüglich f, wenn

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.

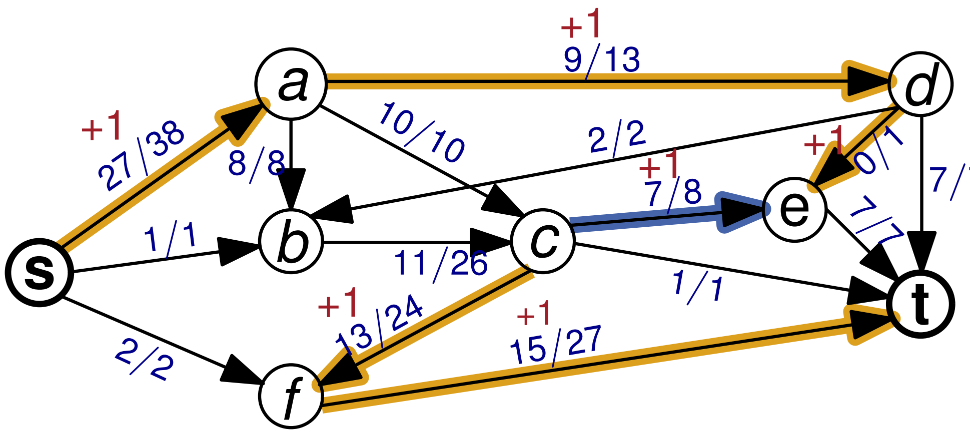
Beweis: "$\Rightarrow$" Annahme f ist maximal und es gibt erhöhenden Weg W.

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:

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

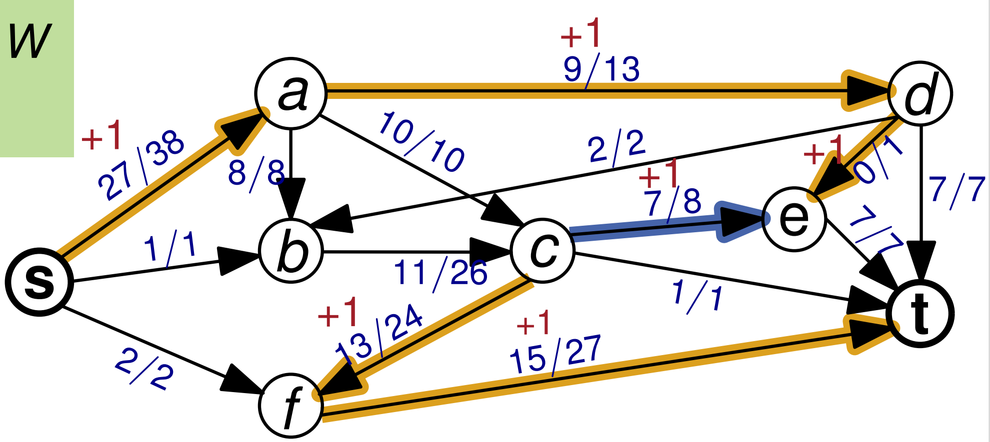
$$\Delta := \min\{\Delta(i,j) \mid (i,j) \in W\} \ .$$

Sei nun $f' : E \to \mathbb{R}_0^+$ 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.

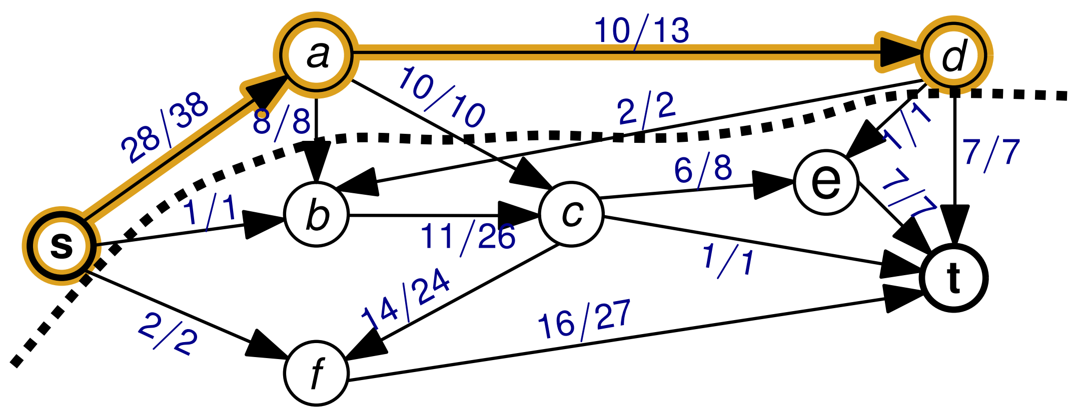


Satz vom erhöhenden Weg

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 $s \in S$ $\quad$ $S \neq V$, 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.

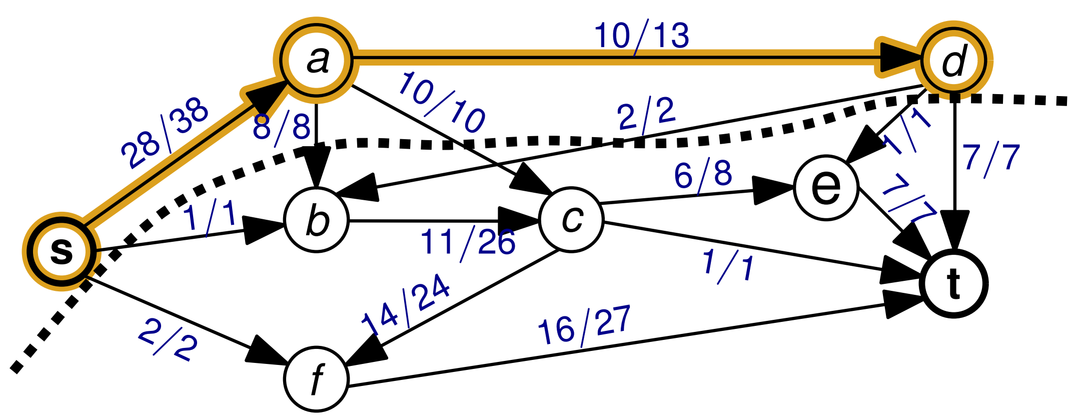
Beweis: "$\Leftarrow$" Enthalte S alle Knoten, die bzgl. f auf einem erhöhenden Weg von s aus erreichbar sind.

Es gilt: $S \neq \emptyset$, weil $s \in S$ $\quad$ $S \neq V$, 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*.

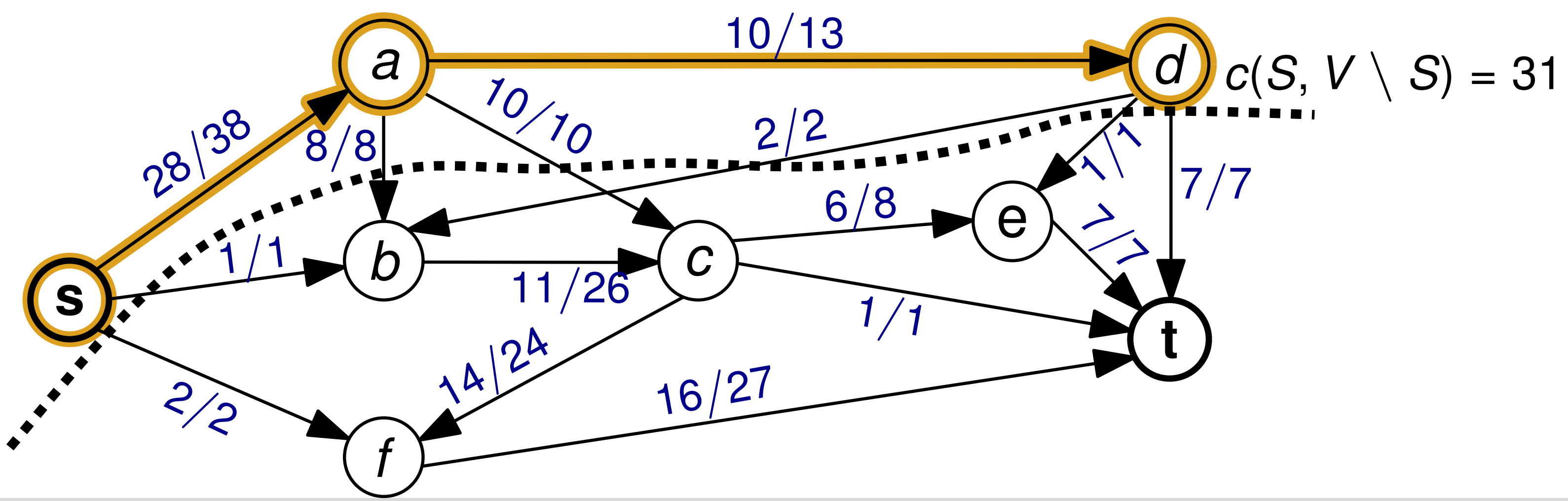
Nach **Lemma 4.5** gilt $w(f) = c(S, V \setminus S)$ und somit ist f maximal.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.



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:

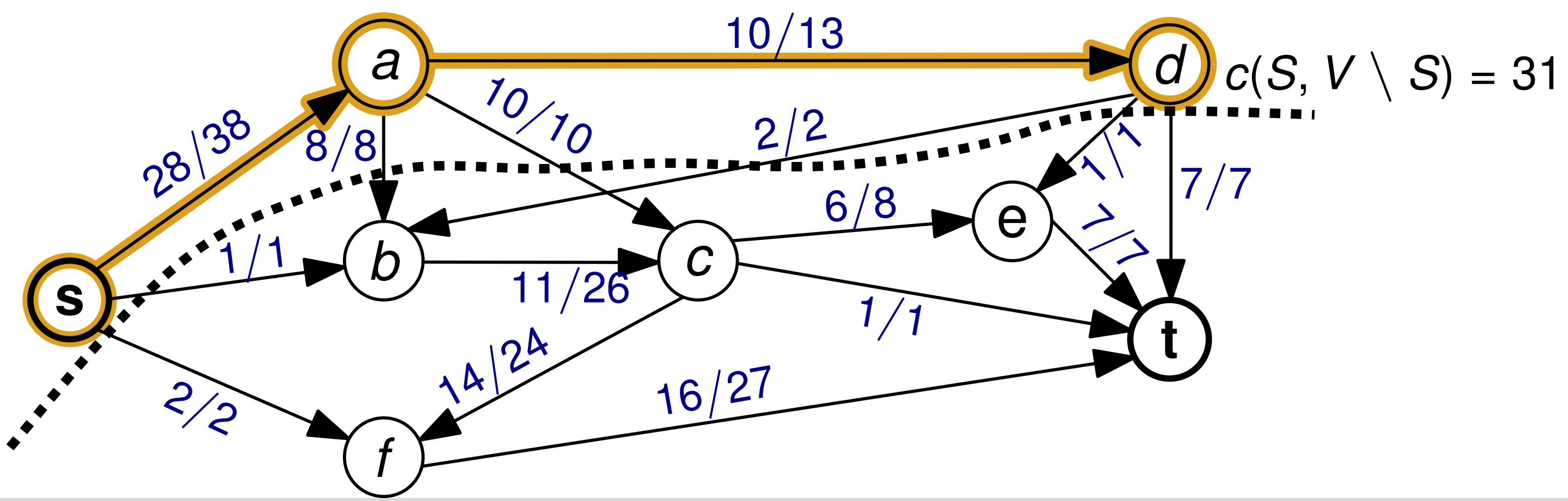
Da f Maximalfluss ist, gilt:

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:

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



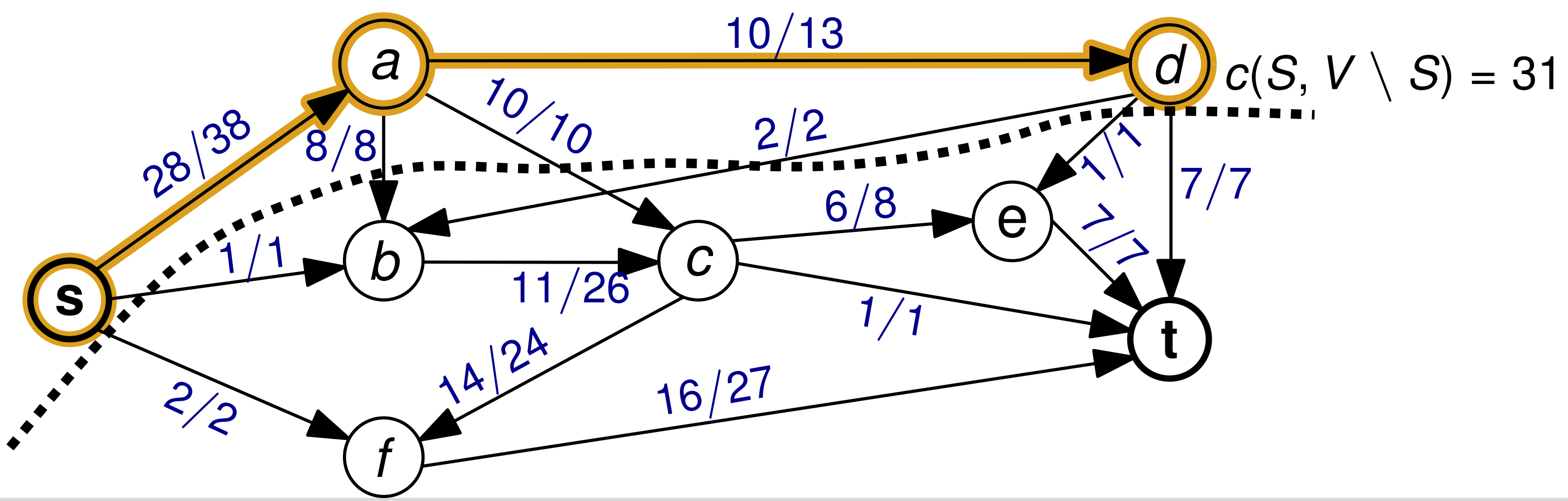
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.

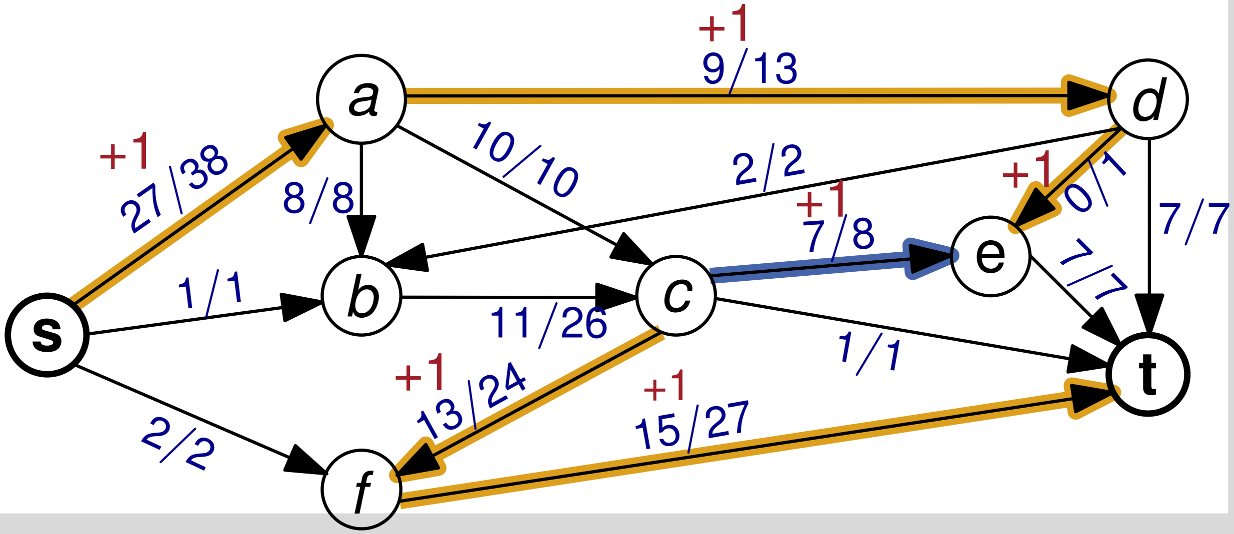
Bemerkungen: Für einen Fluss f in einem Netzwerk (D, s, t, c) sind die folgenden Aussagen äquivalent:

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



Ganzzahligkeitssatz

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



Ganzzahligkeitssatz

Satz 4.11: Sei (D, s, t, c) ein Netzwerk mit $c\colon E \to \mathbb{N}_0$. 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 $(i,j) \in E$

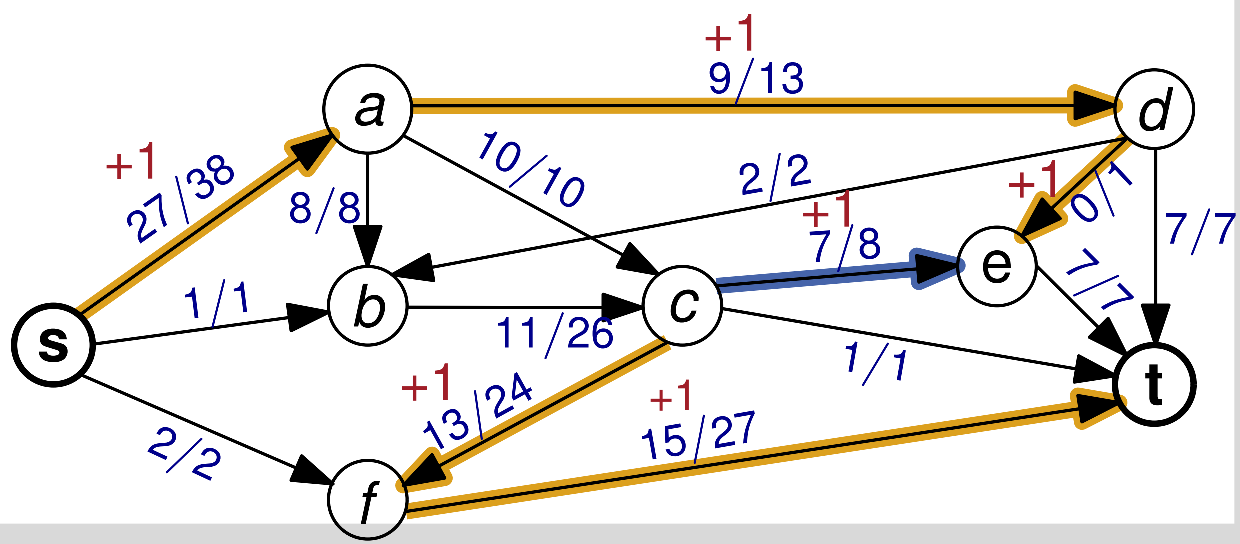
Falls f_0 nicht maximal, dann gibt es erhöhenden Weg W.

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

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



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Ganzzahligkeitssatz

Satz 4.11: Sei (D, s, t, c) ein Netzwerk mit $c\colon E \to \mathbb{N}_0$. 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

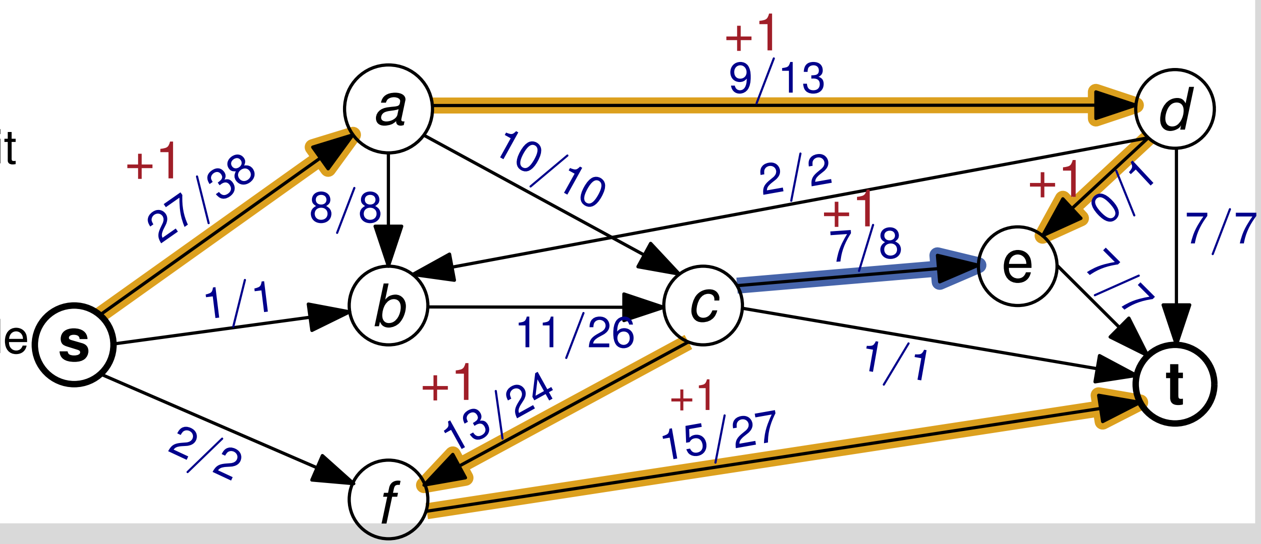
$$\Delta_0 := \min\{\Delta(i,j) \mid (i,j) \text{ auf erhöhendem Weg } W\}\ .$$

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.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Lösungsverfahren für Flussprobleme und minimale Schnitte

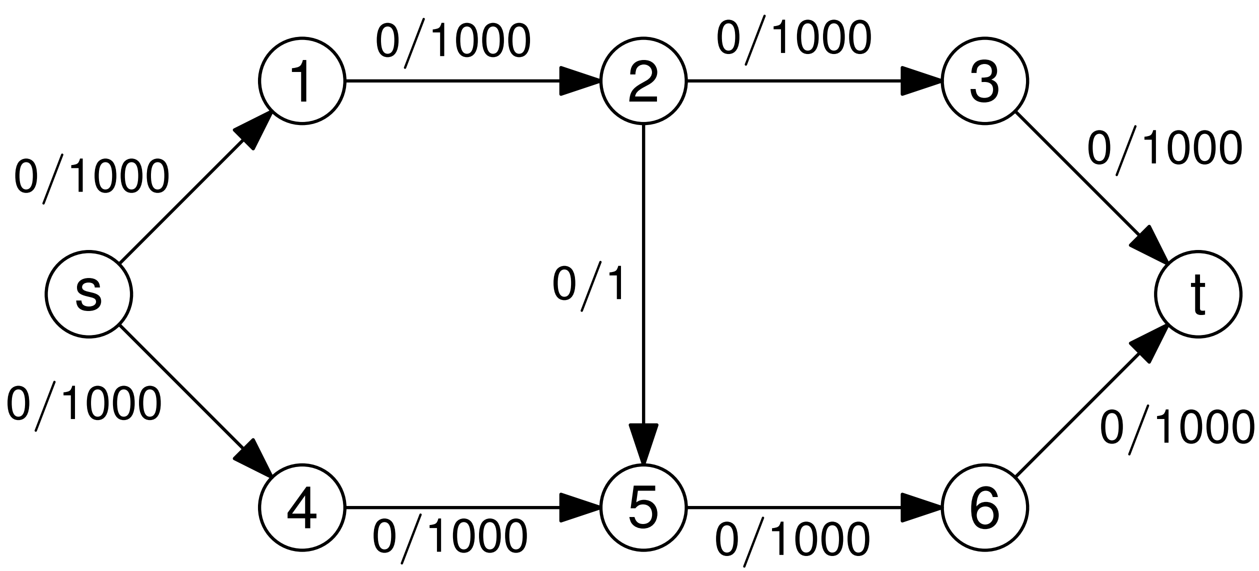
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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 $\langle e_1, e_2, \ldots, e_k \rangle$ 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:

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



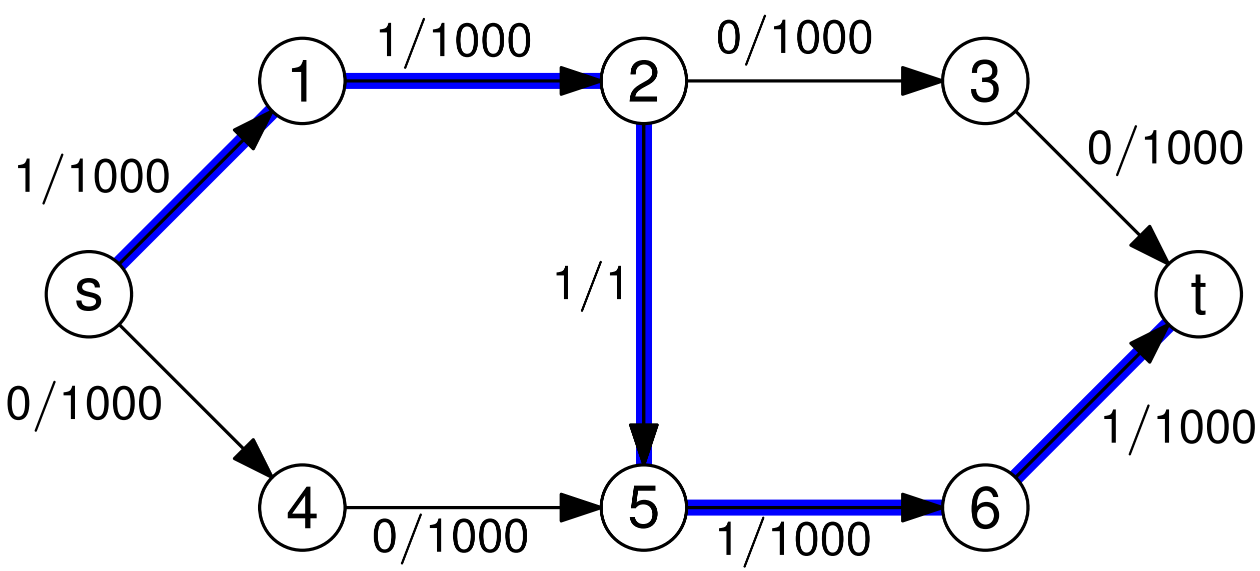
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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 $\langle e_1, e_2, \ldots, e_k \rangle$ 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:

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



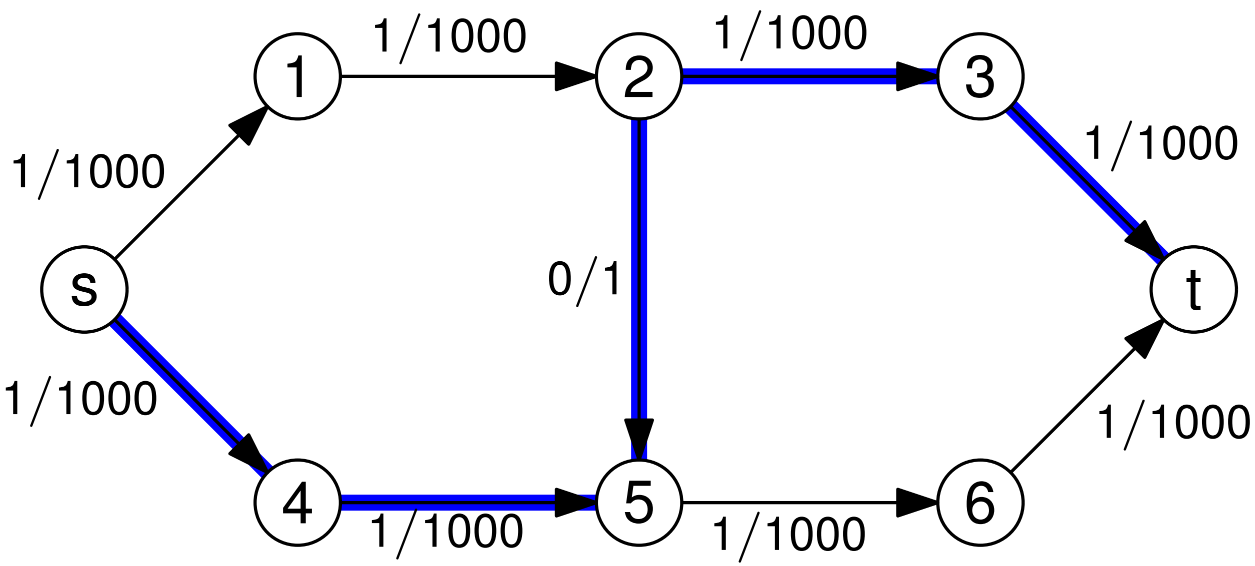
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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 $\langle e_1, e_2, \ldots, e_k \rangle$ 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:

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



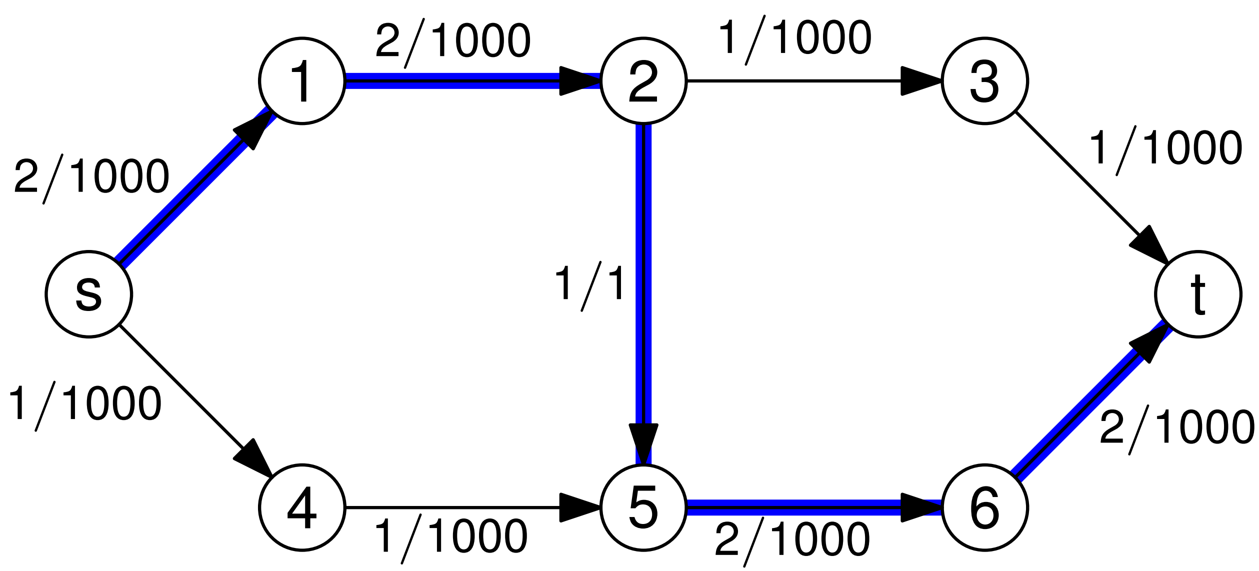
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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 $\langle e_1, e_2, \ldots, e_k \rangle$ 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:

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



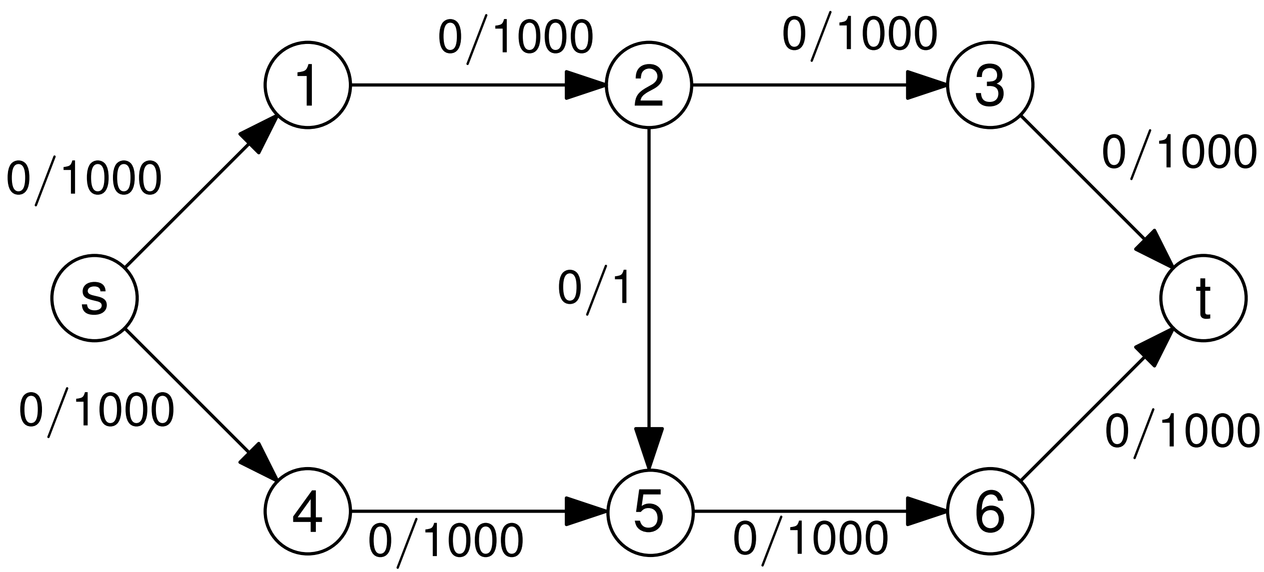
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.

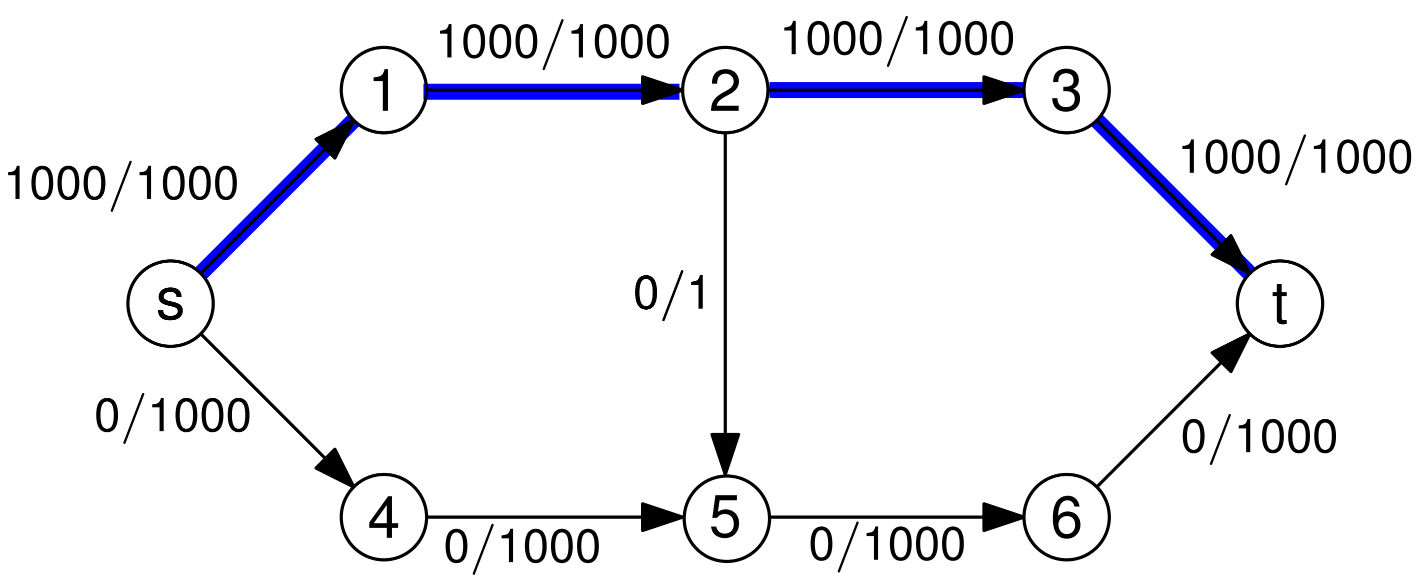
Algorithmus von Edmonds und Karp

- 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.)



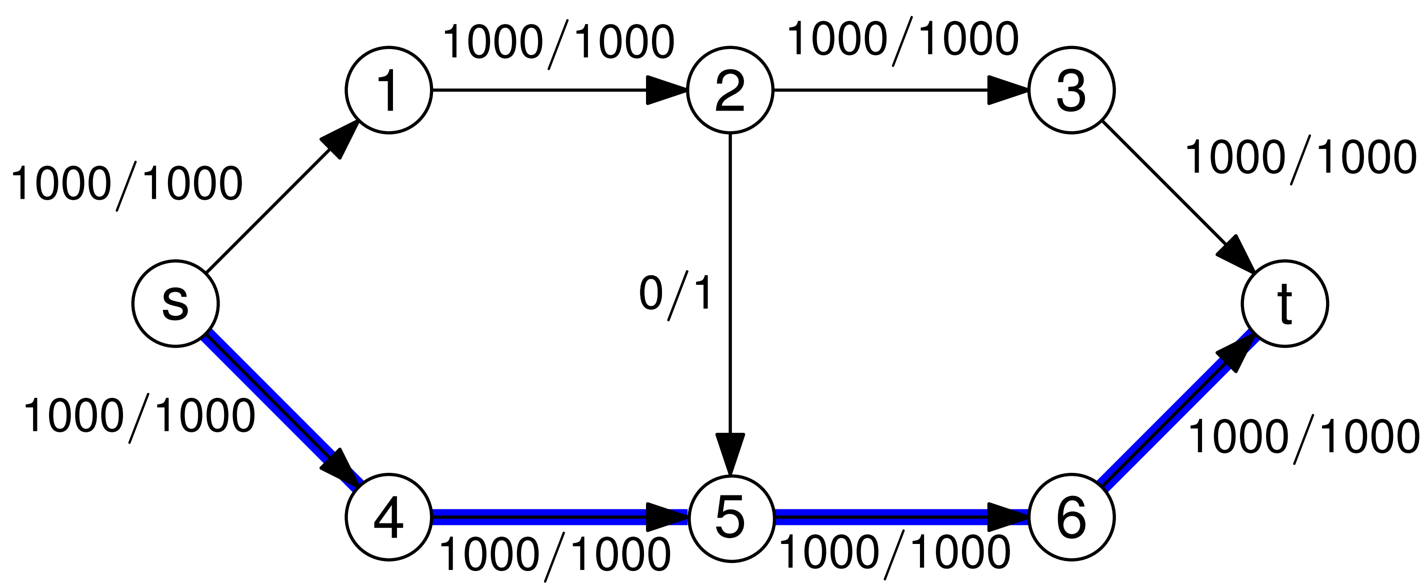
Algorithmus von Edmonds und Karp

- 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**: $\overline{x} = (x_1, \dots, x_n)^T$
2. einer **linearen Zielfunktion**: $f(\overline{x}) = c_1 \cdot x_1 + \cdots + c_n \cdot x_n$
3. **Nebenbedingungen**:
$$a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 + \cdots + a_{1,n} \cdot x_n \leq b_1$$
$$\dots$$
$$a_{m,1} \cdot x_1 + a_{m,2} \cdot x_2 + \cdots + a_{m,n} \cdot x_n \leq b_m$$

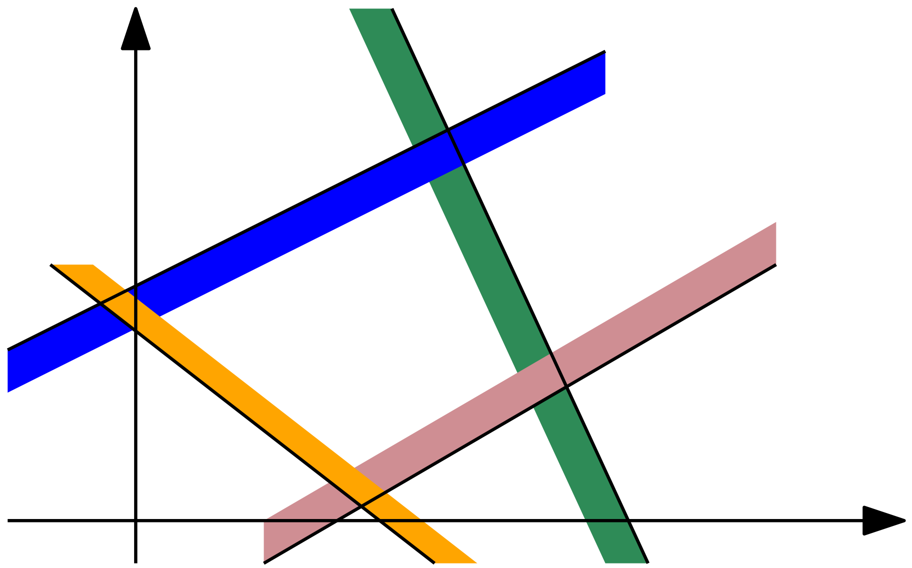
Ziel: bestimme $x_1, \dots, x_n$ so, dass $f(\overline{x})$ maximal/minimal ist.

Matrixschreibweise des LP:

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

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

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



Lösungsraum 2-dimensionales LP

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Beispiel: Bäckerei

	Weizenmehl	*Wasser*	*Mischkornschrot*
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

	Weizenmehl	*Wasser*	*Mischkornschrot*
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:

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

Nebenbedingungen **NB:**

$$
\begin{aligned}
12\,x_1 + 6\,x_2 &\leq 630 \\
8\,x_1 + 12\,x_2 &\leq 620 \\
10\,x_2 &\leq 350 \\
x_1 &\geq 10 \\
x_1 &\geq 0 \\
x_2 &\geq 0
\end{aligned}
$$

Geometrische Interpretation

Skizze:

x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

$$
\begin{array}{llrlcrll}
\text{Zielfunktion } \textbf{ZF:} & f(x_1, x_2) = & 20 & x_1 & + & 60 & x_2 & = \max! & \\
\text{Nebenbedingungen } \textbf{NB:} & & 12 & x_1 & + & 6 & x_2 & \leq 630 & \text{Weizen} \\
& & 8 & x_1 & + & 12 & x_2 & \leq 620 & \text{Wasser} \\
& & & & & 10 & x_2 & \leq 350 & \text{Körner} \\
& & & x_1 & & & & \geq 10 & \text{Stammkunden} \\
& & & x_1 & & & & \geq 0 & \\
& & & x_2 & & & & \geq 0 &
\end{array}
$$

Geometrische Interpretation

Skizze:

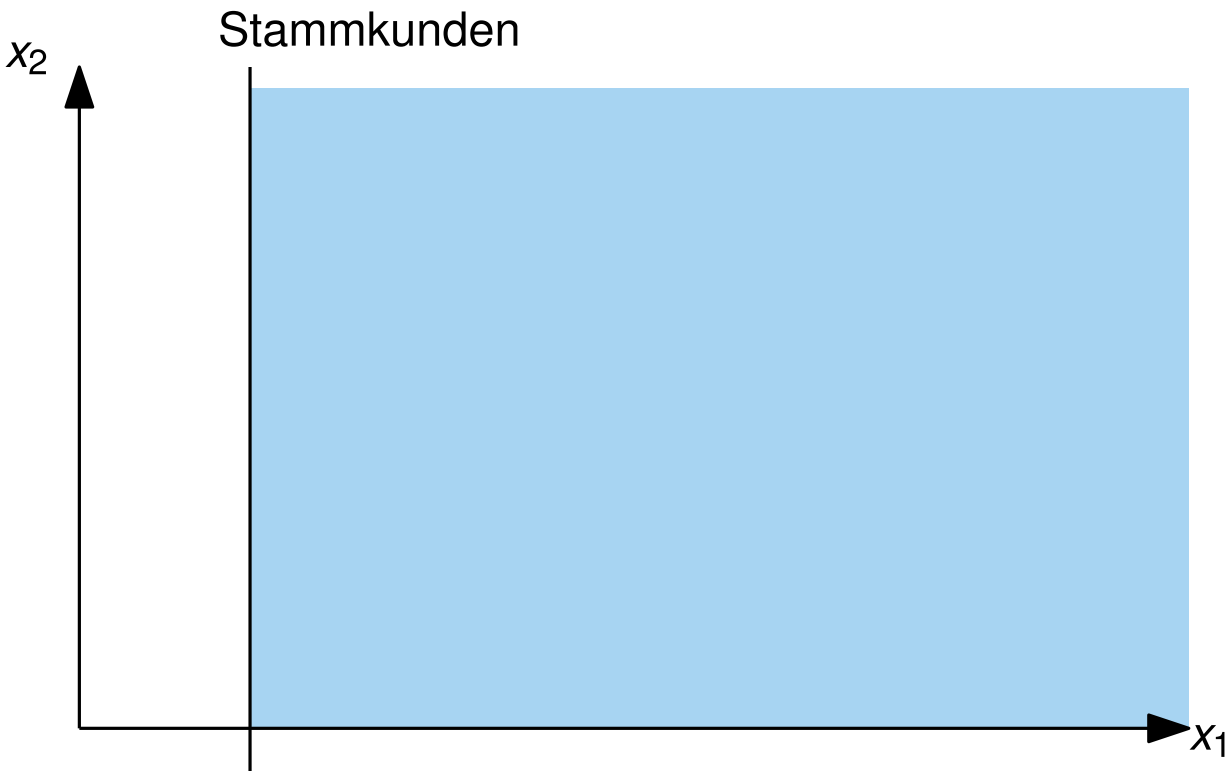
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

$$
\begin{array}{llrlcrll}
\text{Zielfunktion } \textbf{ZF:} & f(x_1, x_2) = & 20 & x_1 & + & 60 & x_2 & = \max! & \\
\text{Nebenbedingungen } \textbf{NB:} & & 12 & x_1 & + & 6 & x_2 & \leq 630 & \text{Weizen} \\
 & & 8 & x_1 & + & 12 & x_2 & \leq 620 & \text{Wasser} \\
 & & & & & 10 & x_2 & \leq 350 & \text{Körner} \\
 & & & x_1 & & & & \geq 10 & \text{Stammkunden} \\
 & & & x_1 & & & & \geq 0 & \\
 & & & x_2 & & & & \geq 0 &
\end{array}
$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Geometrische Interpretation

Skizze:



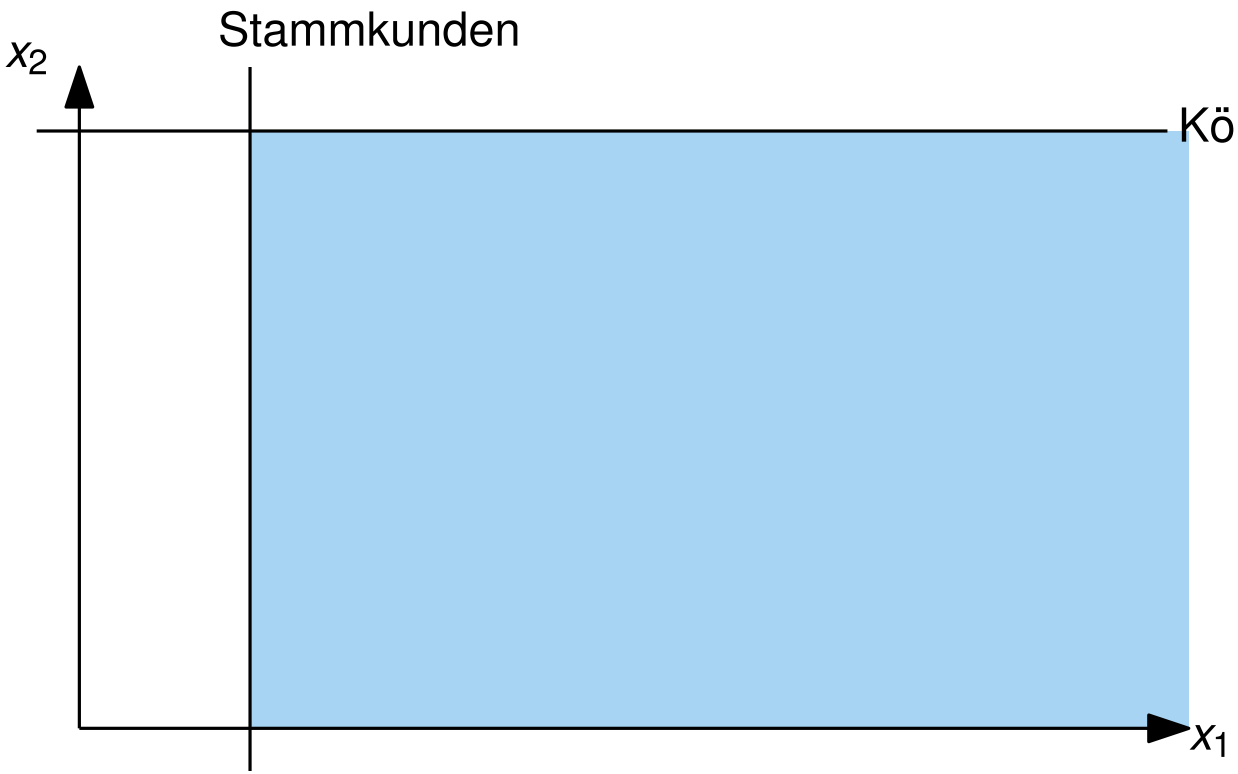
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Geometrische Interpretation

Skizze:



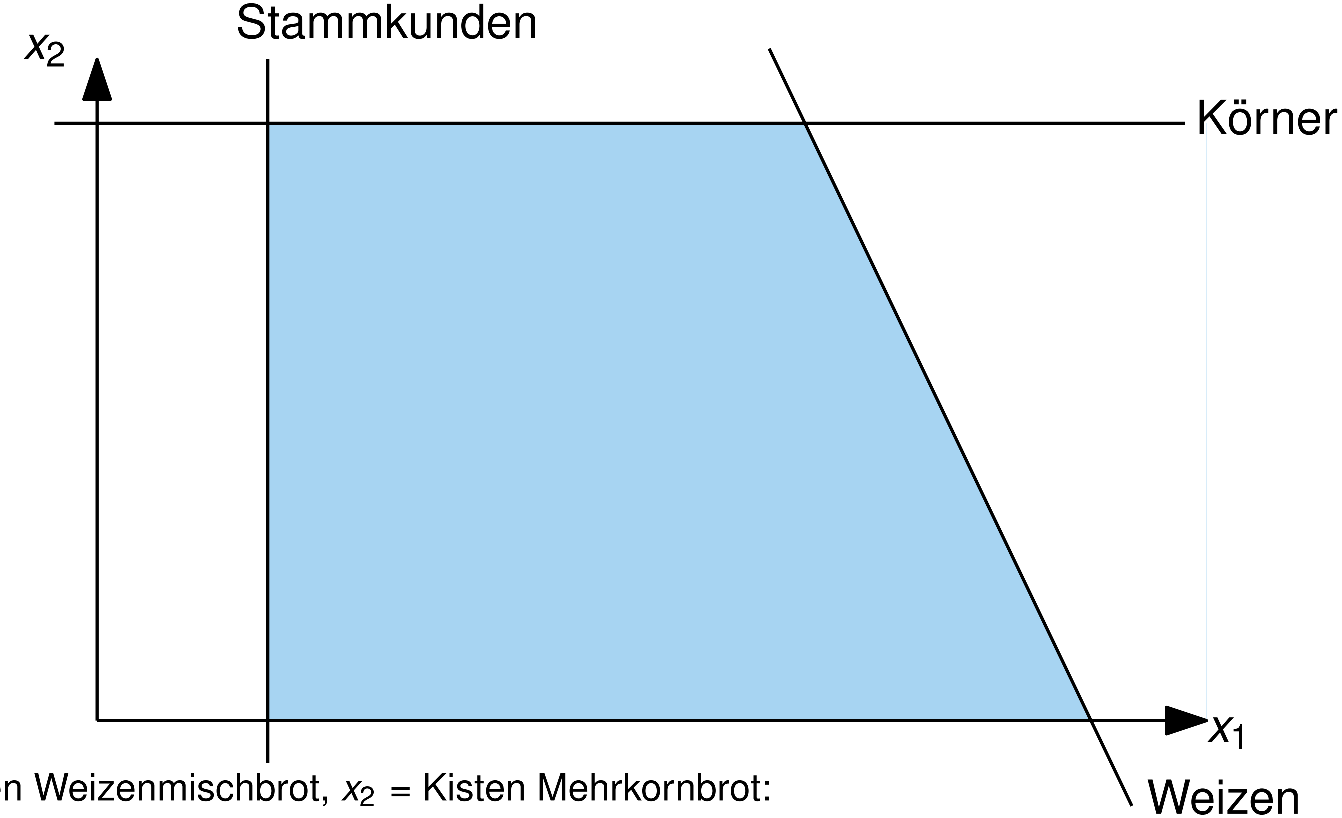
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Geometrische Interpretation

Skizze:



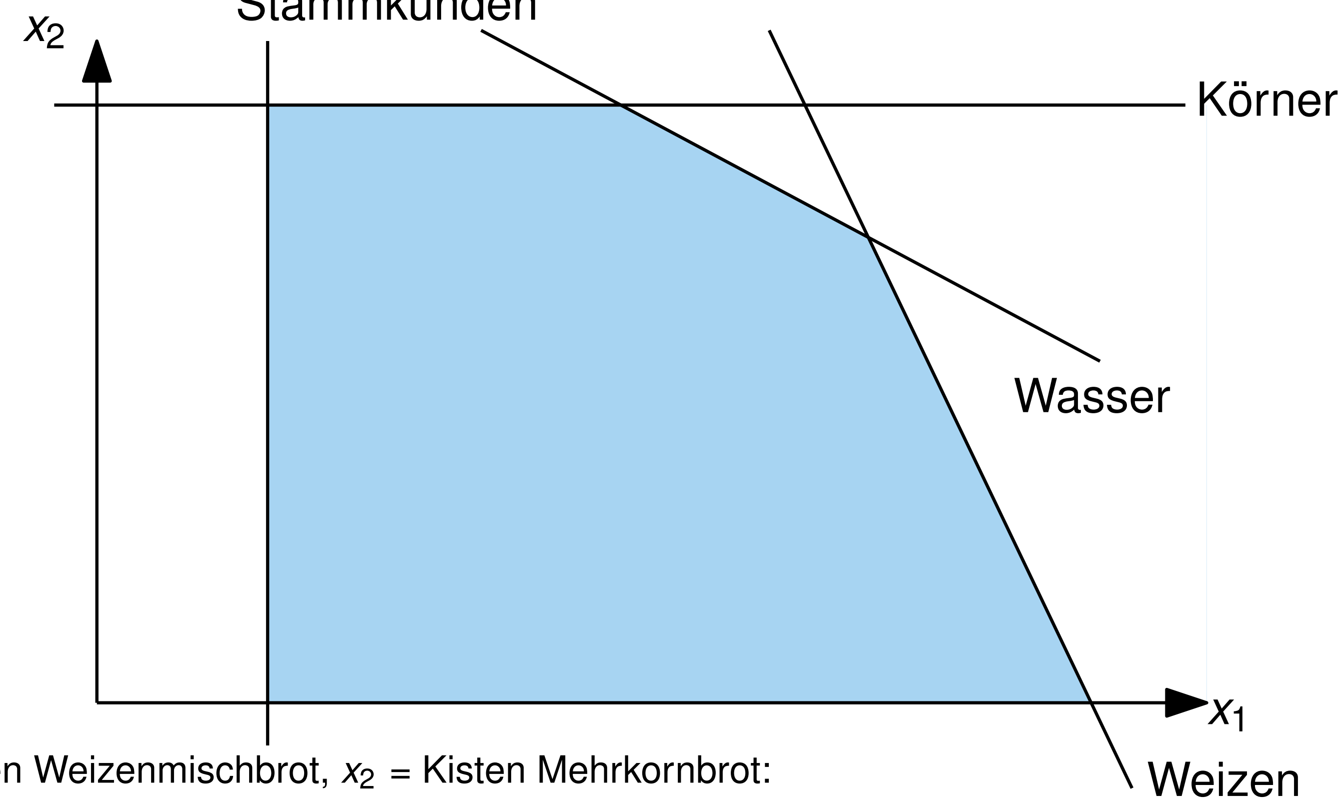
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Geometrische Interpretation

Skizze:



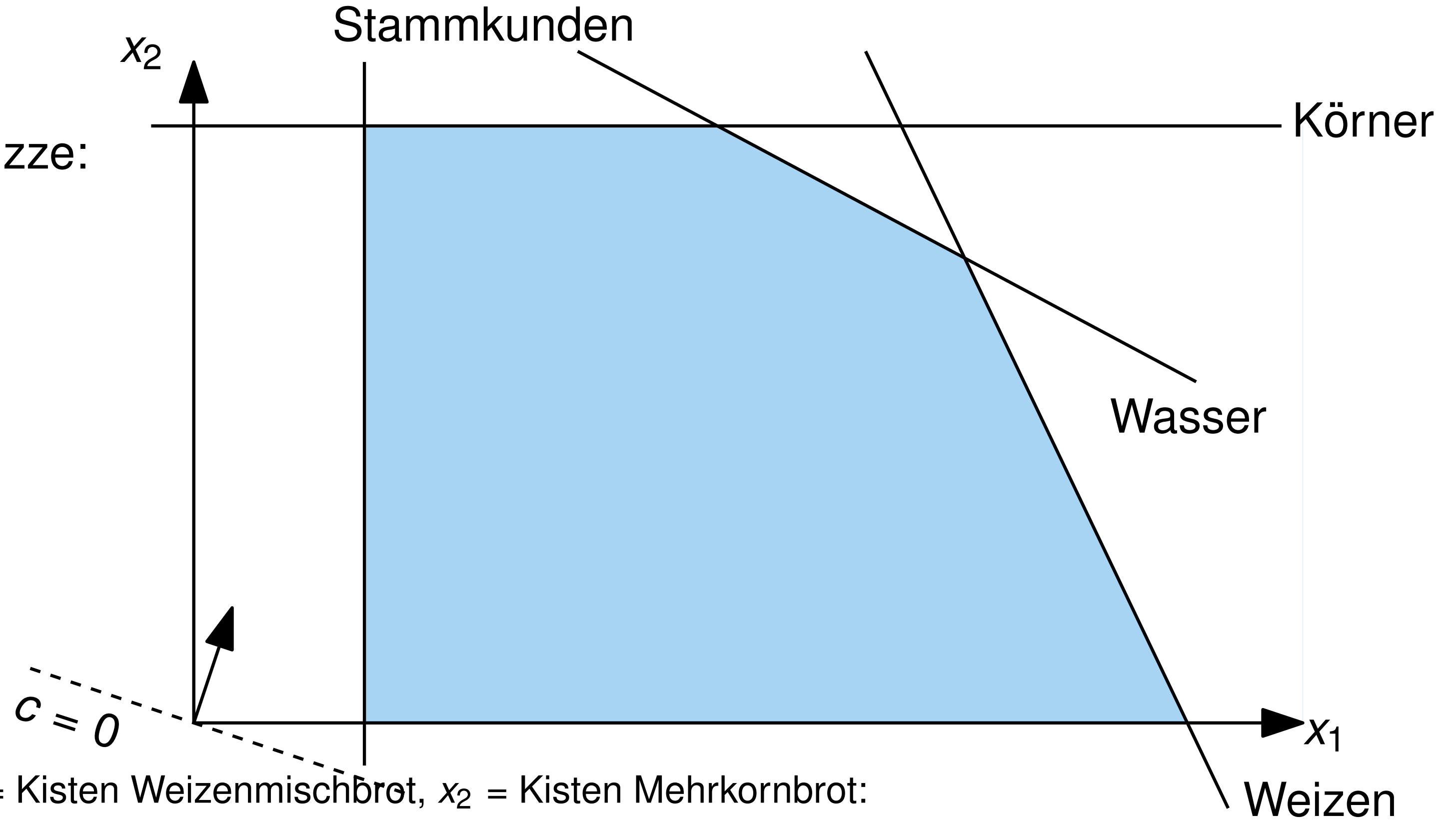
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Geometrische Interpretation

Skizze:

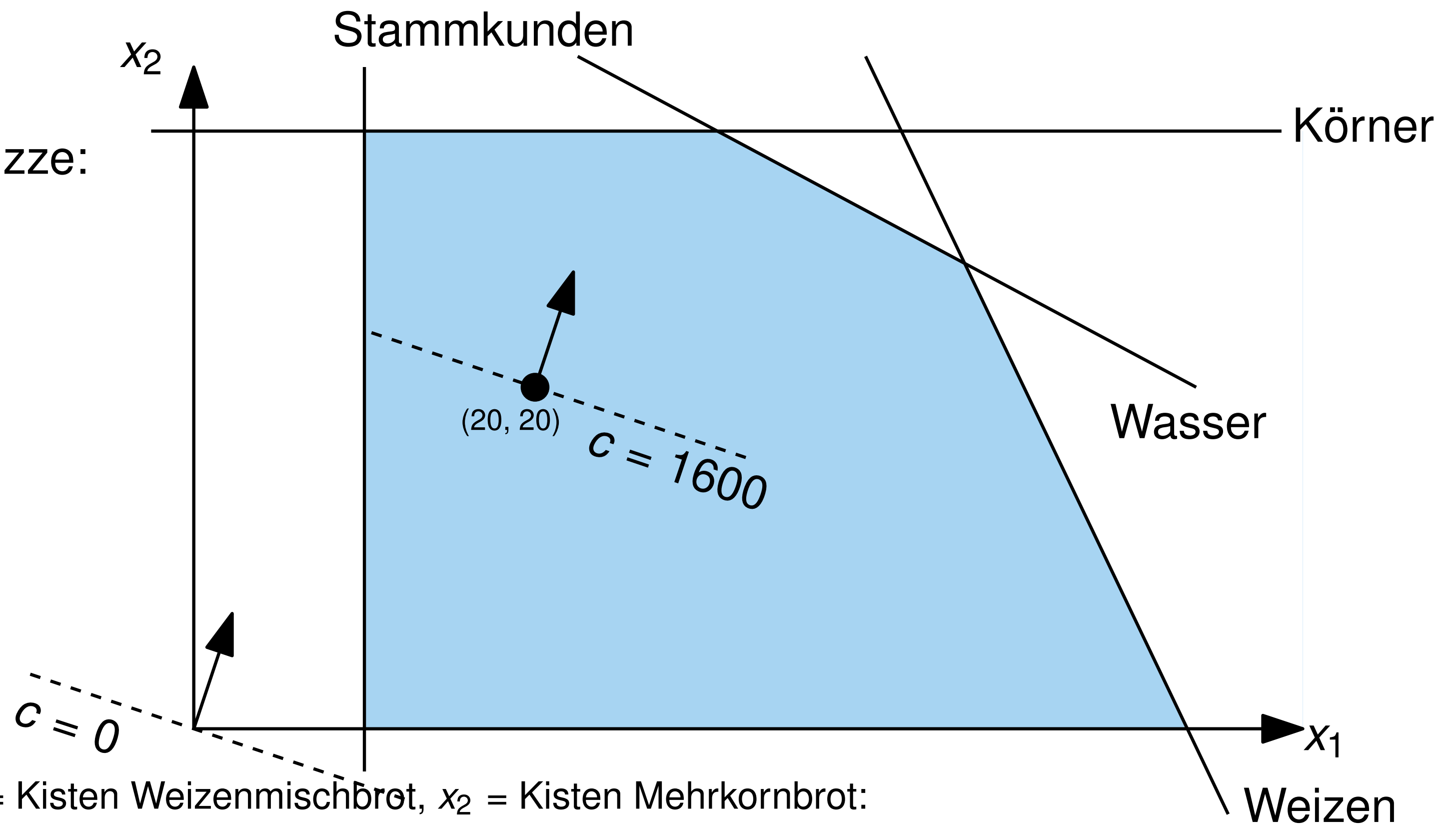


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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

Skizze:



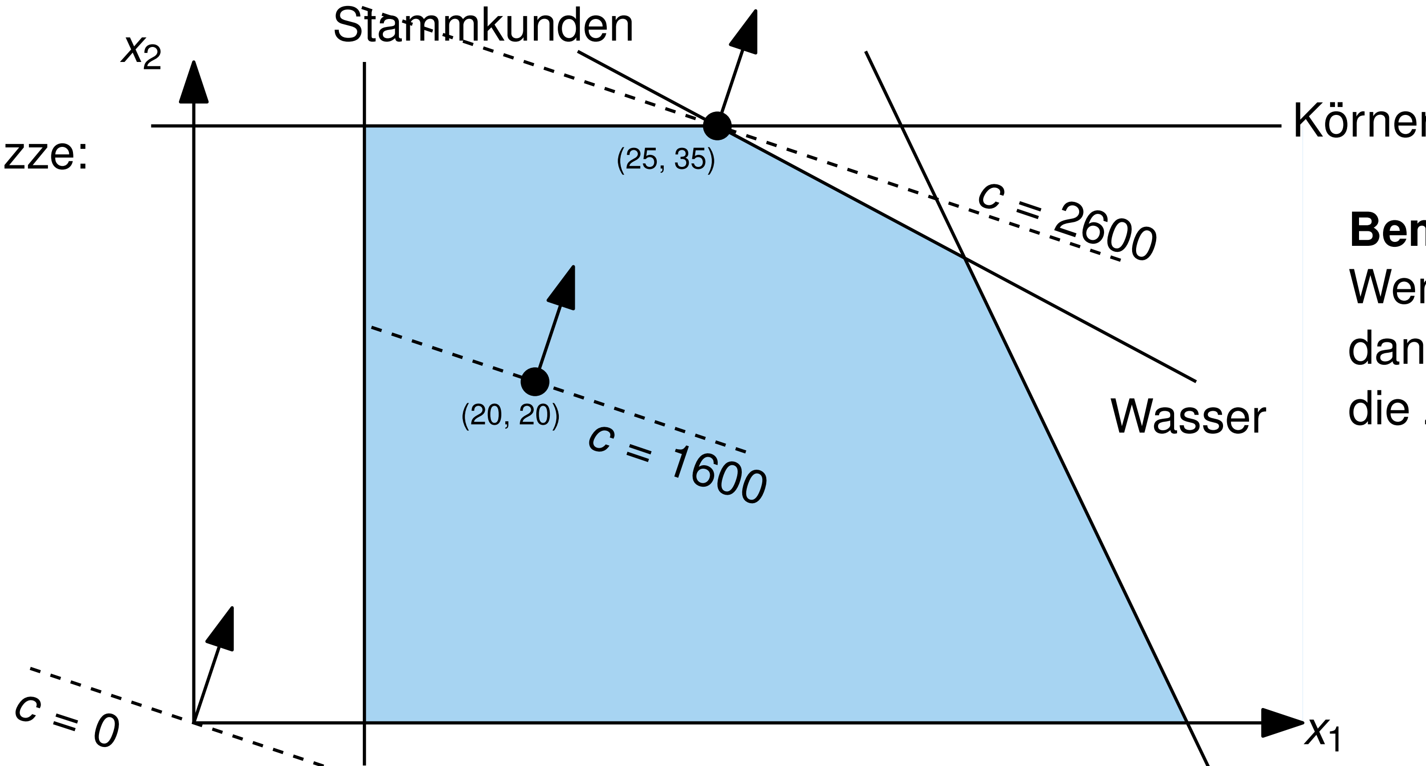
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

$$
\begin{array}{llrlcrll}
\text{Zielfunktion } \textbf{ZF:} & f(x_1, x_2) = & 20 & x_1 & + & 60 & x_2 & = \max! & \\
\text{Nebenbedingungen } \textbf{NB:} & & 12 & x_1 & + & 6 & x_2 & \leq 630 & \text{Weizen} \\
& & 8 & x_1 & + & 12 & x_2 & \leq 620 & \text{Wasser} \\
& & & & & 10 & x_2 & \leq 350 & \text{Körner} \\
& & & x_1 & & & & \geq 10 & \text{Stammkunden} \\
& & & x_1 & & & & \geq 0 & \\
& & & x_2 & & & & \geq 0 &
\end{array}
$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Geometrische Interpretation

Skizze:



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	

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Flussproblem als Lineares Programm

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

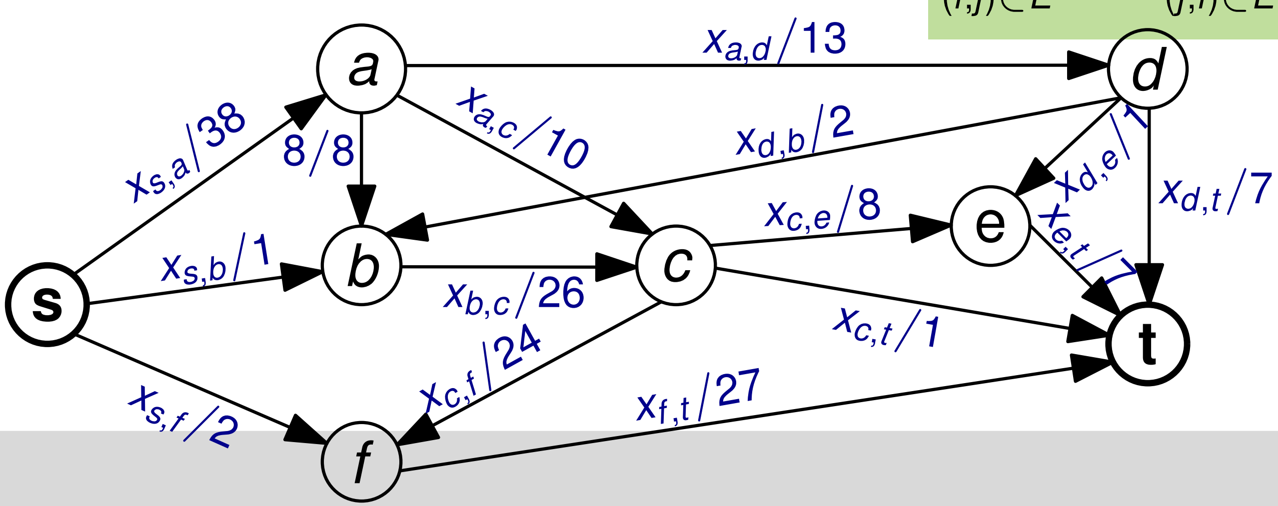
Führe für jede Kante (i, j) eine **Variable** $x_{i,j}$ ein.

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

Maximiere den Wert des Flusses: $f(\overline{x}) = \sum_{(s,i)\in E} x_{s,i} - \sum_{(i,s)\in E} x_{i,s}$

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



Dualität

Primales Programm:

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

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

$\overline{x} \geq 0$

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

Duales Programm:

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

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

$\overline{y} \geq 0$

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

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

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

Starker Dualitätssatz:

Primales Programm lösbar $\Leftrightarrow$ zugehöriges duales Programm lösbar

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