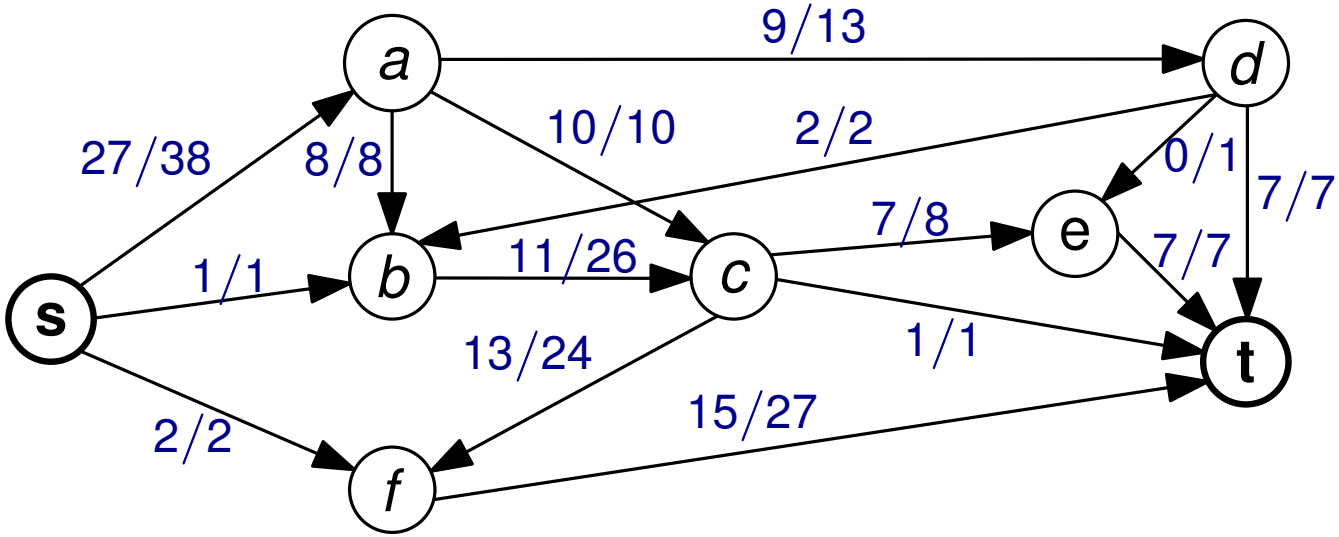
Karlsruher Institut für Technologie

Algorithmen II
Vorlesung am 18.10.2012

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Flussprobleme und Dualität



Problemdefinition

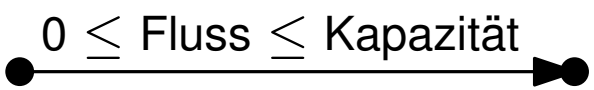
Definition:

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

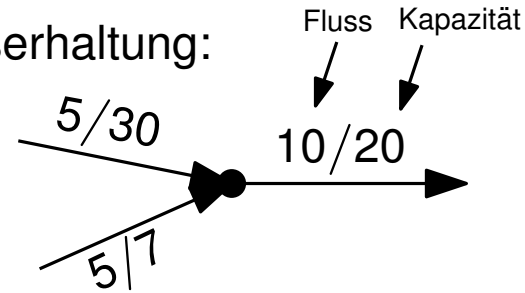
Das Tupel (D, s, t, c) heißt *Netzwerk*. Eine Abbildung $f\colon E \to \mathbb{R}_0^+$ heißt *Fluss*, wenn folgende Bedingungen gelten:

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.

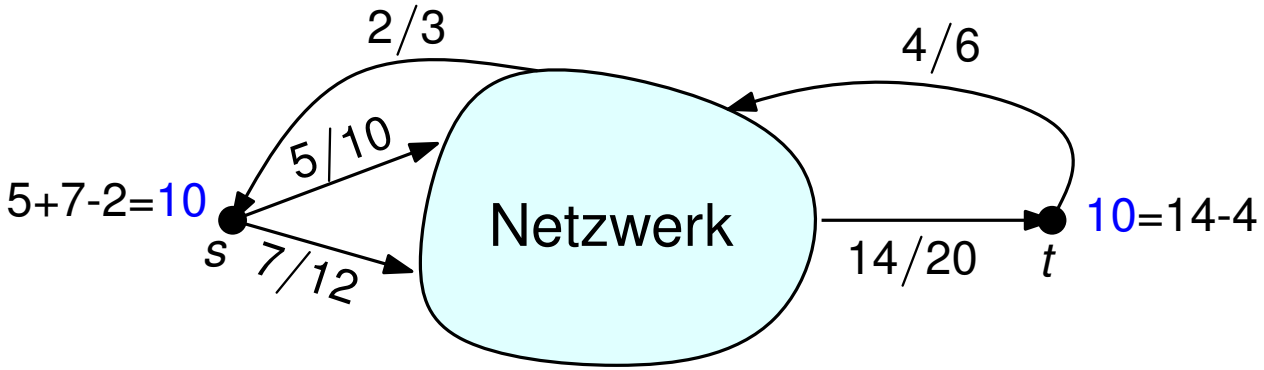
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Flusserhaltung und Wert eines Flusses

Lemma: 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

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Flusserhaltung und Wert eines Flusses

Lemma: 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 $w(f') \leq w(f)$.

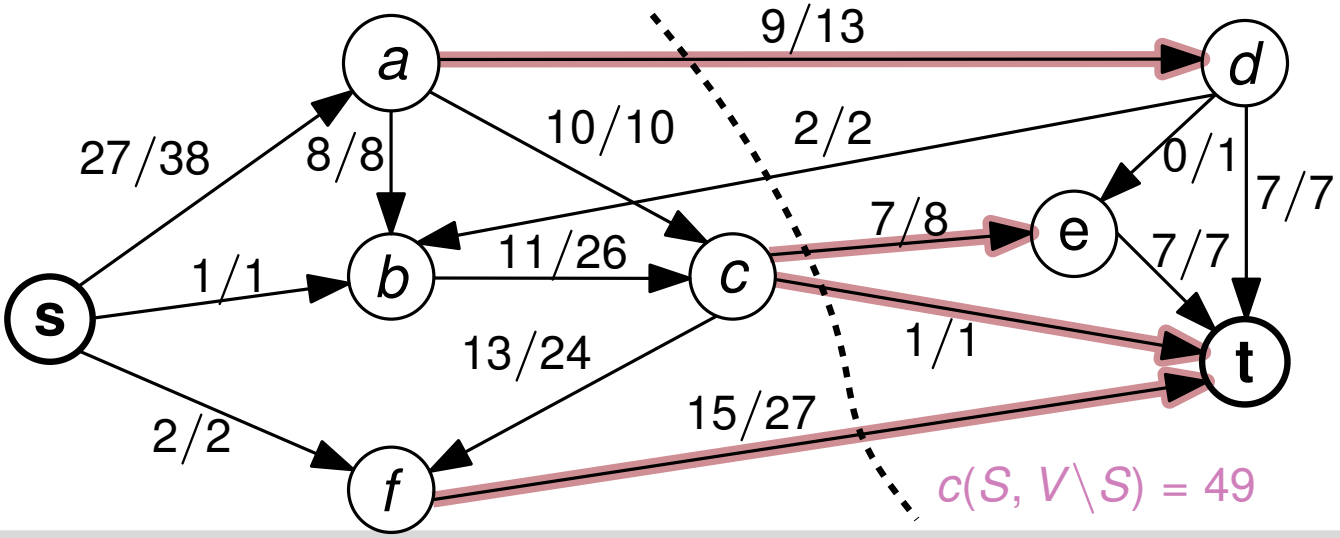
Schnitte in Netzwerken

Definition: Die von einer Menge $S \subset V$ induzierte 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: Die von einer Menge $S \subset V$ induzierte 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: 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 $w(f) \leq c(S, V \setminus S)$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

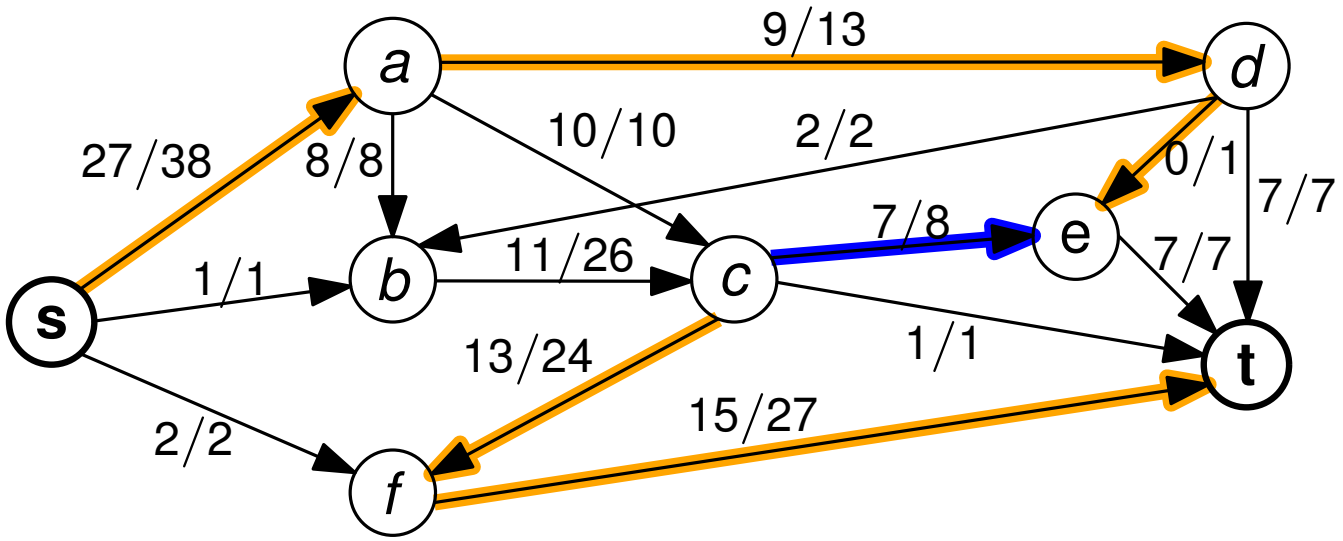
Erhöhende Wege

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.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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

Max-Flow Min-Cut Theorem von Ford & Fulkerson

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

Kann mithilfe des Satzes vom erhöhenden Weg gezeigt werden:

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.

Max-Flow Min-Cut Theorem von Ford & Fulkerson

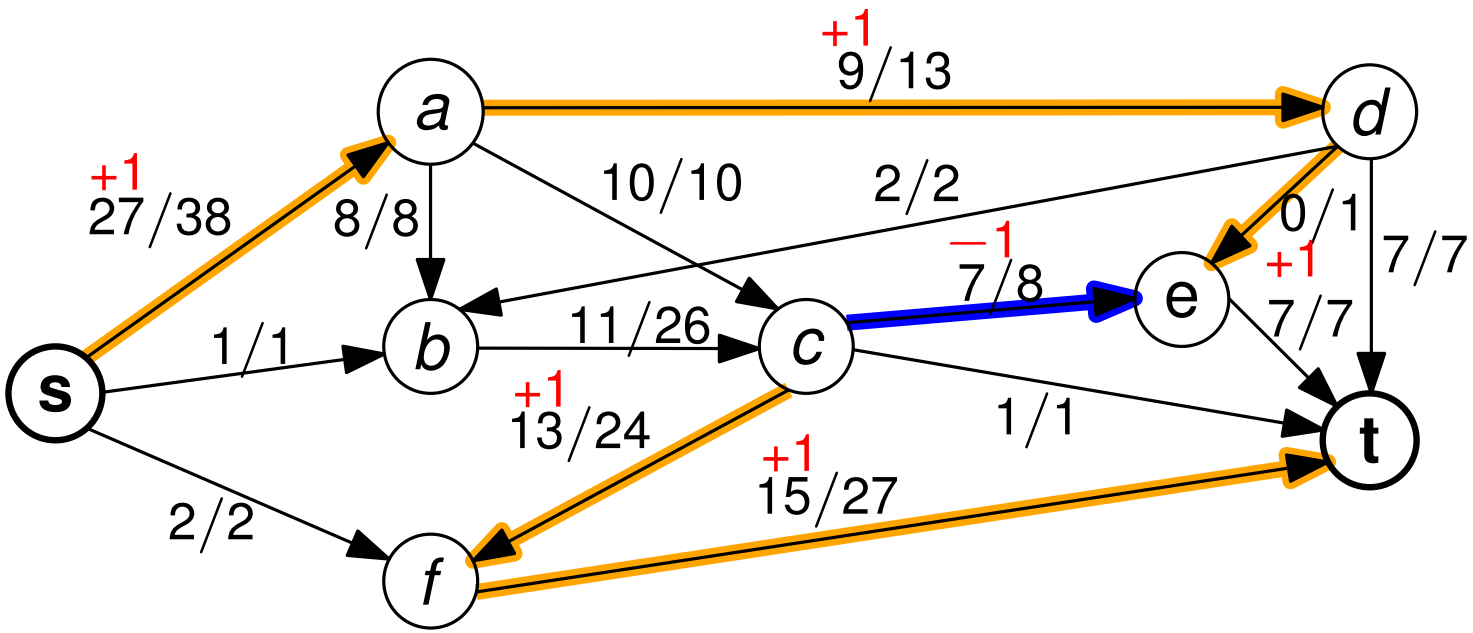
Satz: In einem Netzwerk (D, s, t, c) ist der Wert eines Maximalflusses gleich der minimalen Kapazität eines s-t-Schnittes.

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: 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 $w(f) \in \mathbb{N}_0$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Lösungsverfahren für Flussprobleme und minimale Schnitte

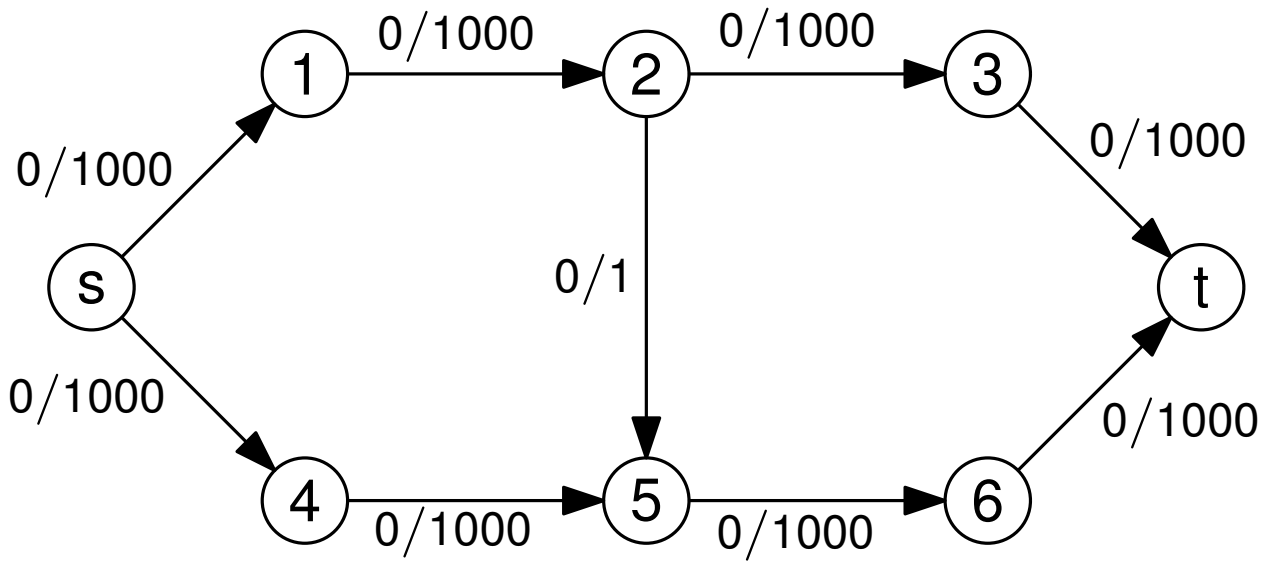
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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:

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



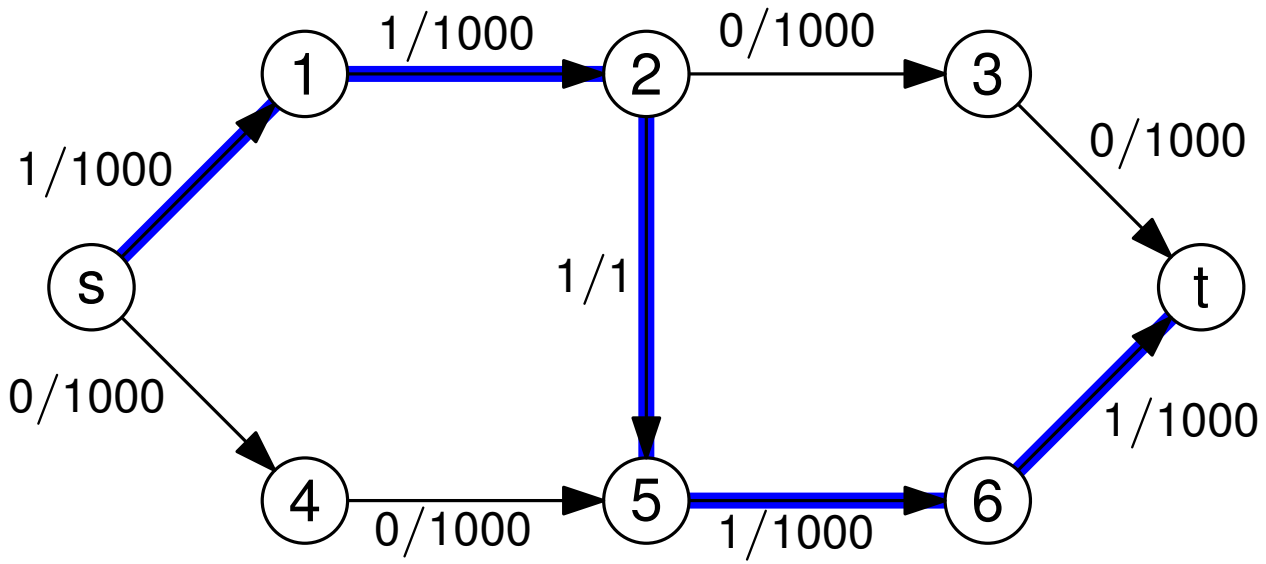
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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:

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



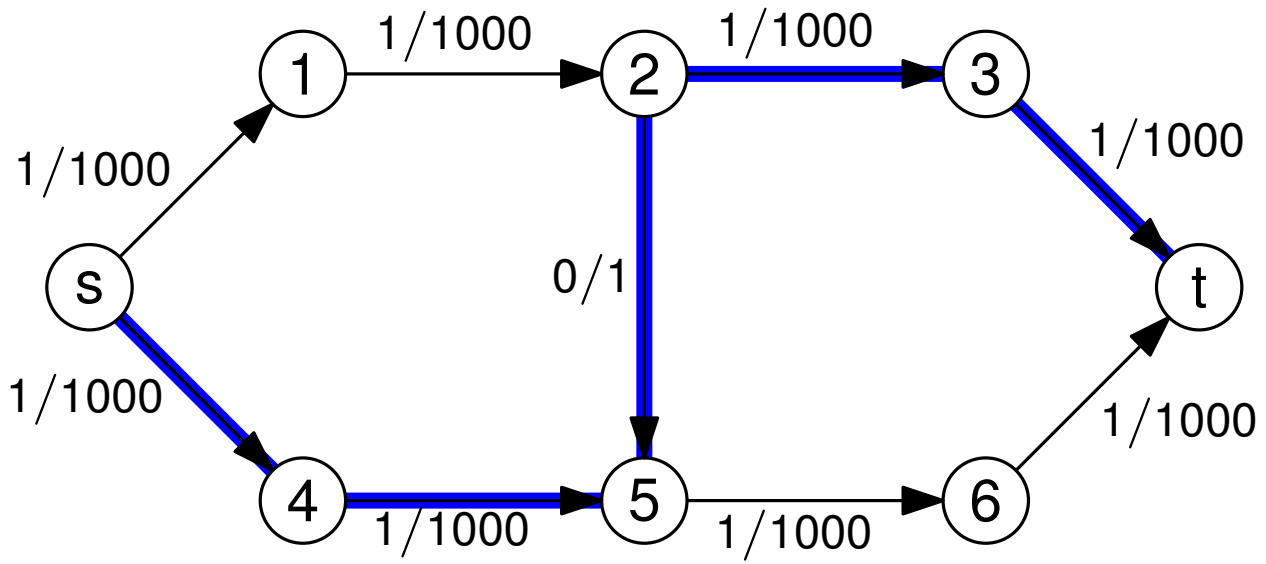
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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:

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



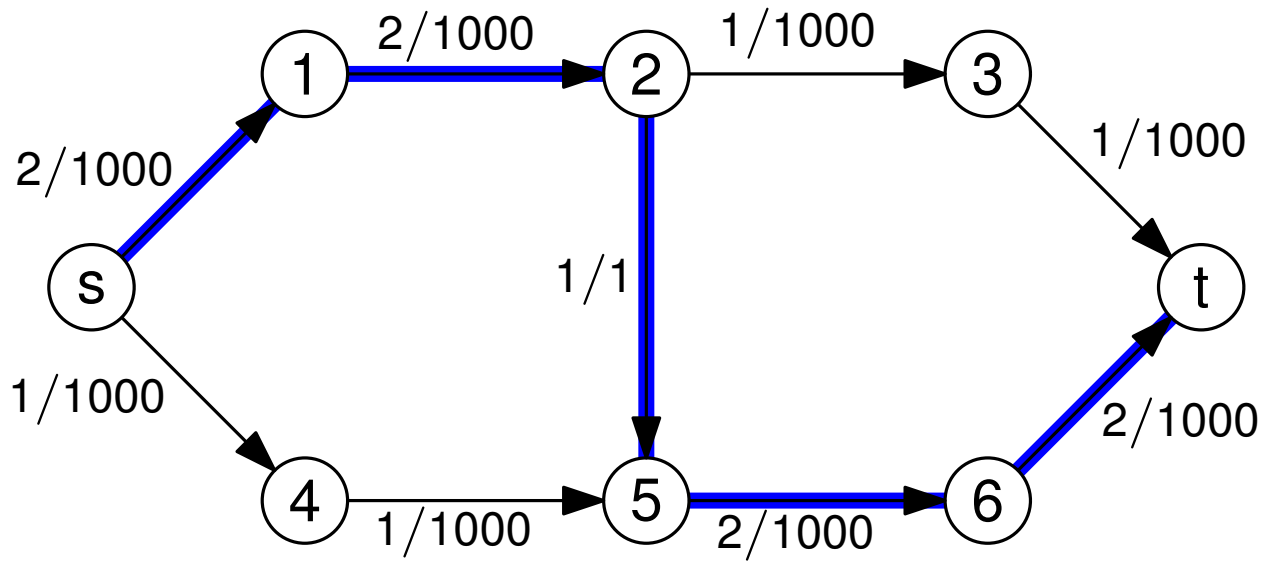
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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:

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



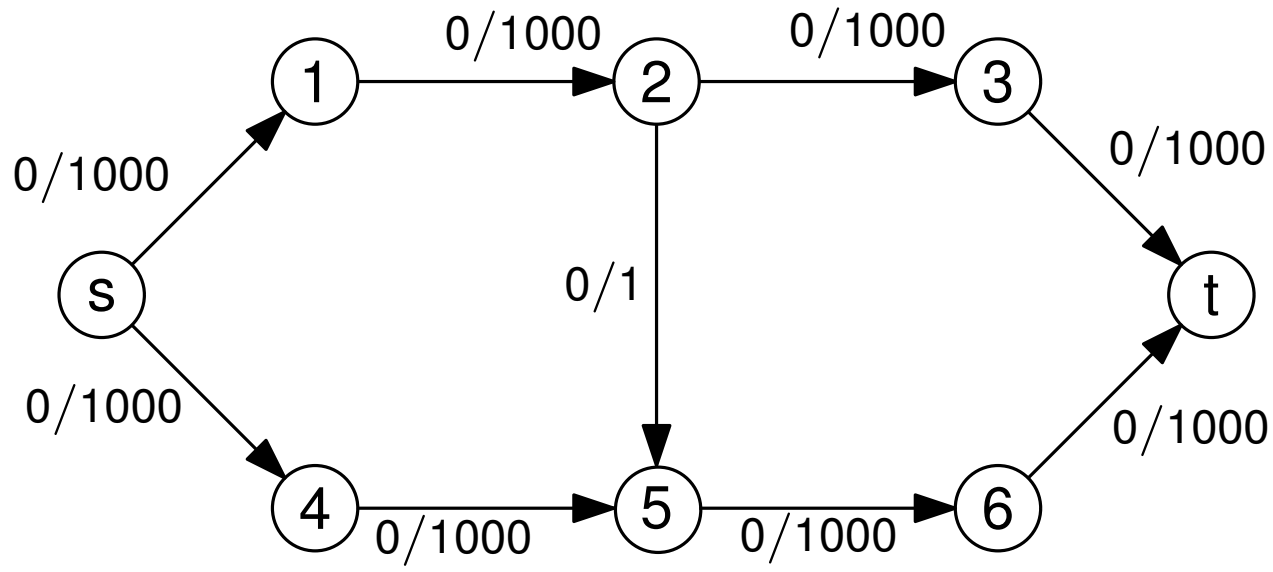
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.

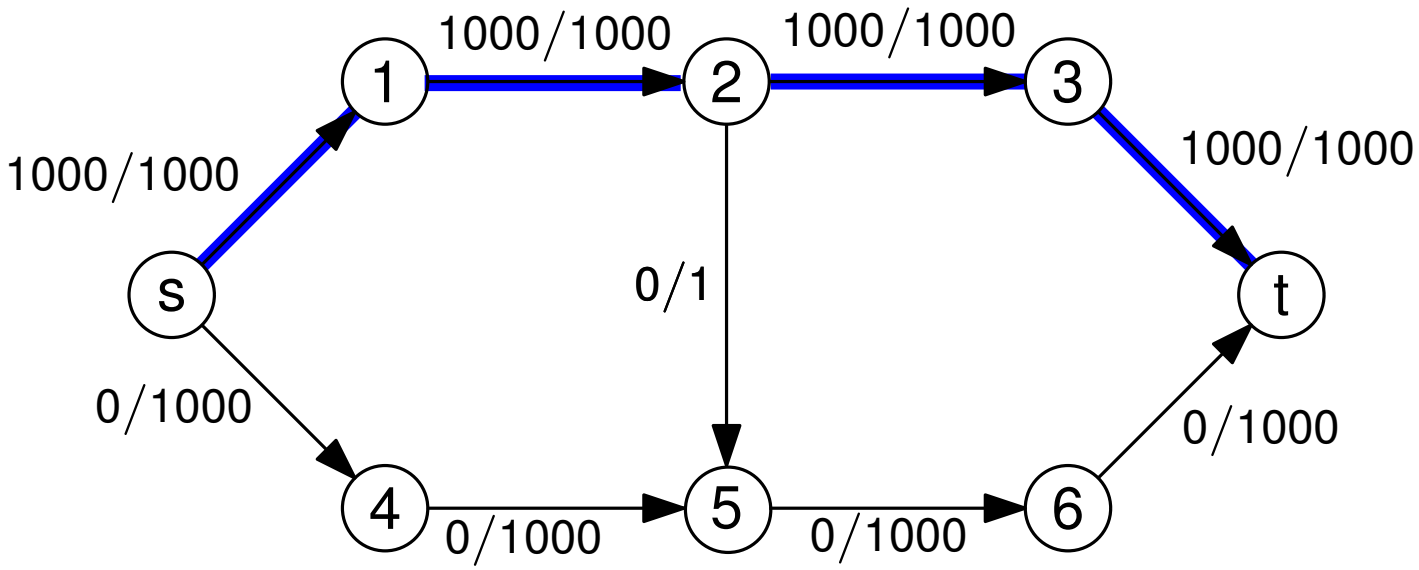
Algorithmus von Edmonds und Karp

- 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.)



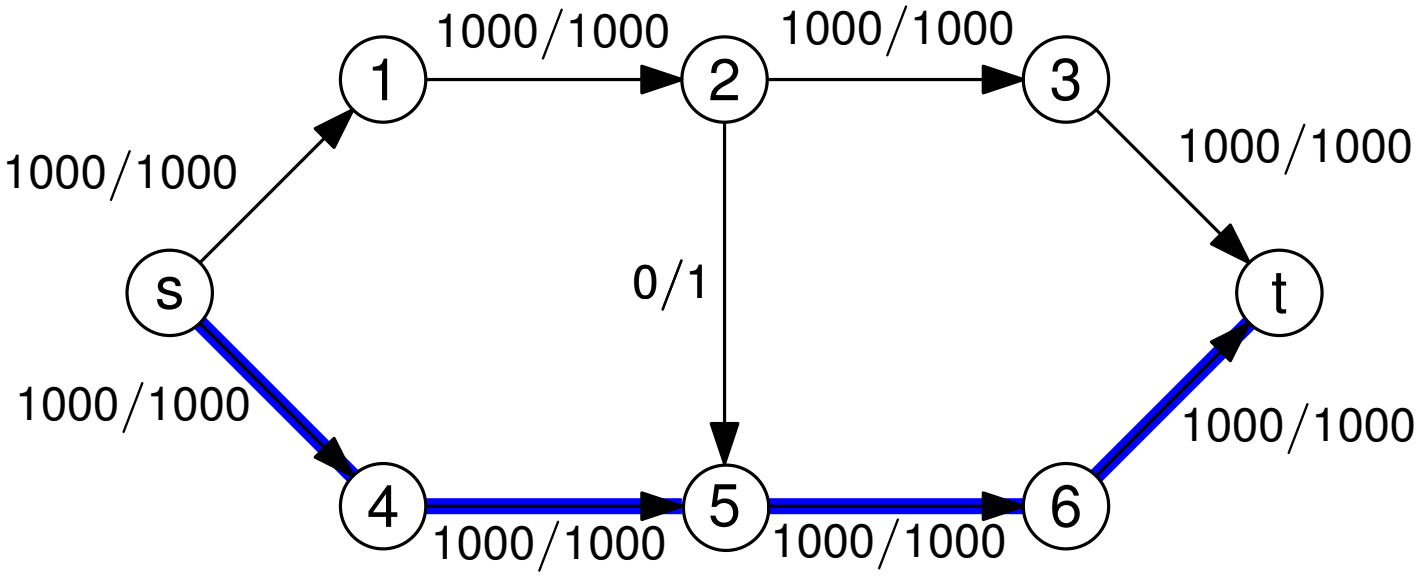
Algorithmus von Edmonds und Karp

- 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: $\overline{x} = (x_1, \ldots, x_n)^T$
2. einer linearen Zielfunktion: $f(\overline{x}) = c_1 \cdot x_1 + \cdots + c_n \cdot x_n$
3. Nebenbedingungen:

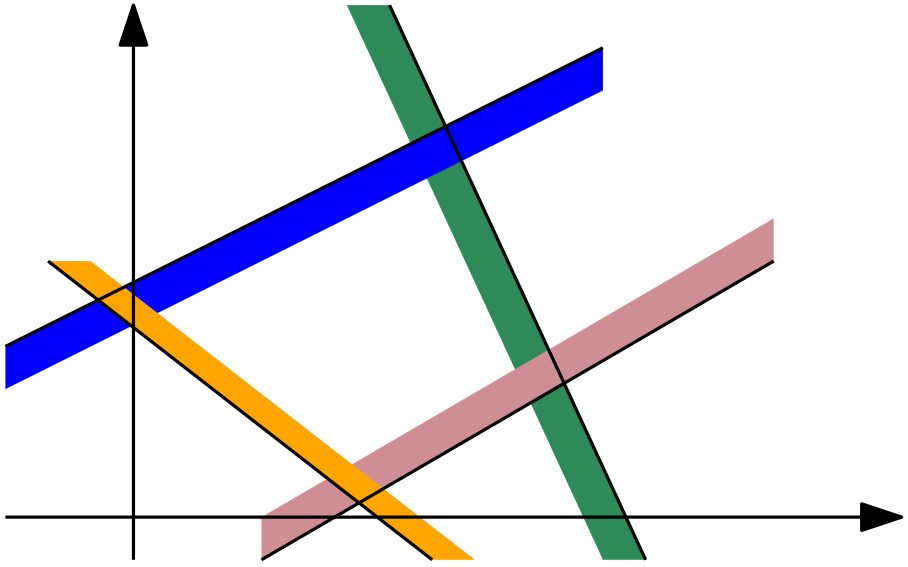
$$a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 + \cdots + a_{1,n} \cdot x_n \leq b_1$$
$$\ldots$$
$$a_{m,1} \cdot x_1 + a_{m,2} \cdot x_2 + \cdots + a_{m,n} \cdot x_n \leq b_m$$

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

Matrixschreibweise des LP:

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

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



Lösungsraum 2-dimensionales LP

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 $\max_{\overline{x} \in N} f(\overline{x}) = \min_{\overline{y} \in M} g(\overline{y})$