Karlsruher Institut für Technologie

Algorithmen II
Übung am 29.10.2013

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Karlsruher Institut für Technologie

Organisatorisches

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Algorithmen II - Team

Vorlesung:

Prof. Dr. Dorothea Wagner

Übung:

Thomas Bläsius
(thomas.blaesius@kit.edu)

Benjamin Niedermann
(benjamin.niedermann@kit.edu)

Sprechzeiten: Termin nach Vereinbarung

Homepage und Forum

- `http://i11www.iti.kit.edu`
- Aktuelle Informationen/Termine
- Skripte, Folien, Übungsblätter
- Literaturempfehlungen
- Forum
  - Für Fragen an die Übungsleiter.
  - Für Fragen untereinander.
  - Link befindet sich auf der Homepage der Vorlesung.

Klausur

- Umfasst zwei Stunden.
- Orientierung: Vorlesung Algorithmen II des Vorjahres + Vorlesung Algorithmentechnik behandelte ähnlichen Stoff.
- Hauptklausur: voraussichtlich am 24.02.2014
- Nachklausur: noch nicht bekannt.

Genaue Klausurtermine werden rechtzeitig bekannt gegeben.

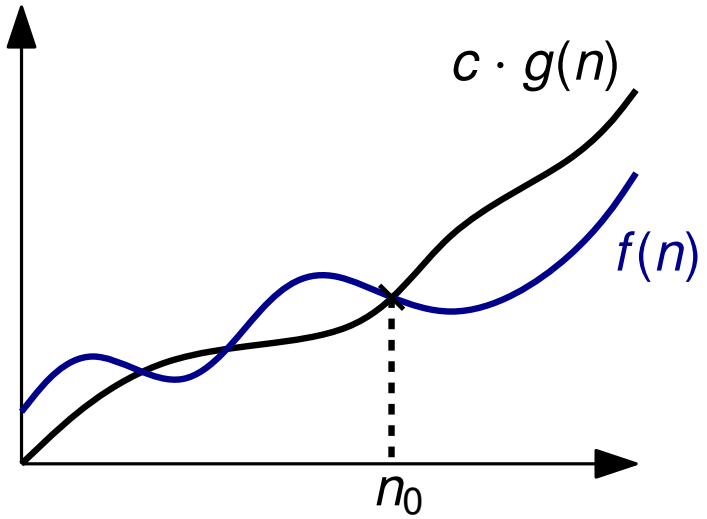
Karlsruher Institut für Technologie

Wiederholung Laufzeitanalyse

Wiederholung – Asymptotische Laufzeit

$\mathcal{O}$-Notation:

19	6	7	10	4	



$f(n) = \mathcal{O}(g(n)) :\Leftrightarrow$ es existieren positive Konstanten c und n_0, so dass

$$0 \leq f(n) \leq c \cdot g(n) \text{ für alle } n \geq n_0$$

Beispiel: Es gibt für Minimumsuche auf einem unsortieren Integer-Array der Länge n einen Algorithmus $\mathcal{A}$ mit Laufzeit $\mathcal{O}(n)$,

d.h. man kann $\mathcal{A}$ und zwei positive Konstanten c und n_0 so wählen, dass **für alle Eingaben** die Laufzeit $f(n)$ von $\mathcal{A}$ durch

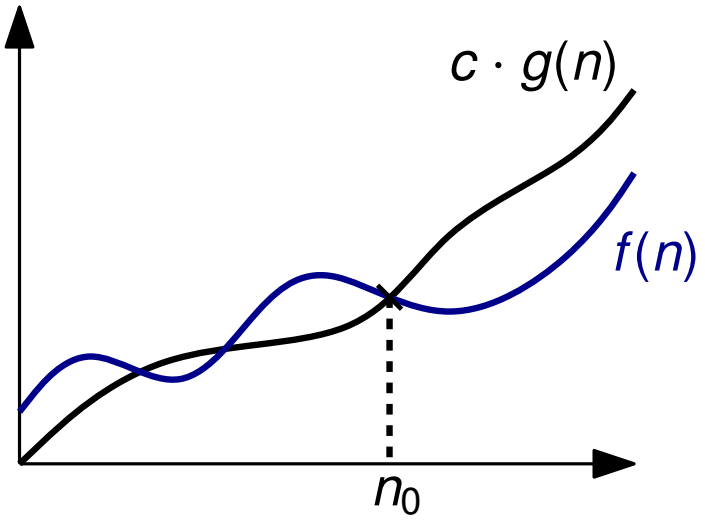
$$0 \leq f(n) \leq c \cdot n \text{ für alle } n \leq n_0$$

abgeschätzt werden kann.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Wiederholung – Asymptotische Laufzeit

$\mathcal{O}$-Notation:



19	6	7	10	4	

$f(n) = \mathcal{O}(g(n)) :\Leftrightarrow$ es existieren positive Konstanten c und n_0, so dass

$$0 \leq f(n) \leq c \cdot g(n) \text{ für alle } n \geq n_0$$

worst-case-Analyse

Beispiel: Es gibt für Minimumsuche auf einem unsortieren Integer-Array der Länge n einen Algorithmus $\mathcal{A}$ mit Laufzeit $\mathcal{O}(n)$,

d.h. man kann $\mathcal{A}$ und zwei positive Konstanten c und n_0 so wählen, dass **für alle Eingaben** die Laufzeit $f(n)$ von $\mathcal{A}$ durch

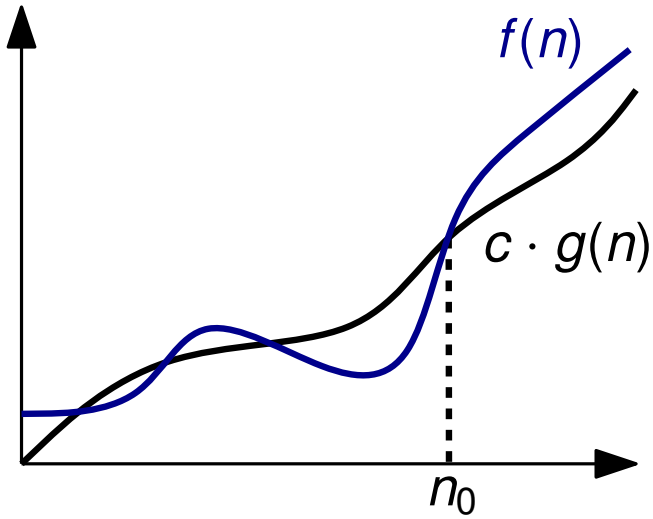
$$0 \leq f(n) \leq c \cdot n \text{ für alle } n \leq n_0$$

abgeschätzt werden kann.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Wiederholung – Asymptotische Laufzeit

Ω-Notation:



$f(n) = \Omega(g(n)) :\Leftrightarrow$ es existieren positive Konstanten c und n_0, so dass

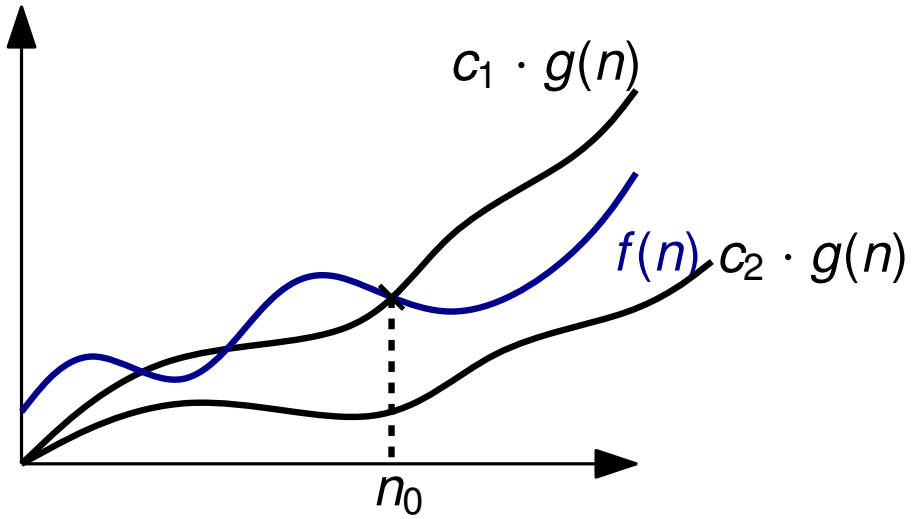
$$0 \leq c \cdot g(n) \leq f(n) \text{ für alle } n \geq n_0$$

Beispiel: Die Minimumsuche in einem unsortierten Integer-Array der Länge n besitzt Laufzeit $\Omega(n)$,
d.h. es gibt keinen Algorithmus, der dies schneller kann:
Jedes Element muss mindestens einmal betrachtet werden, damit sichergestellt ist, dass Minimum gefunden wurde.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Wiederholung – Asymptotische Laufzeit

Θ-Notation:



$f(n) = \Theta(g(n)) :\Leftrightarrow$ es existieren positive Konstanten c_1, c_2 und n_0, so dass

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \text{ für alle } n \geq n_0$$

Beispiel: Es gibt Algorithmus für Minimumssuche auf Integer-Array der Länge n mit Laufzeit $\Theta(n)$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

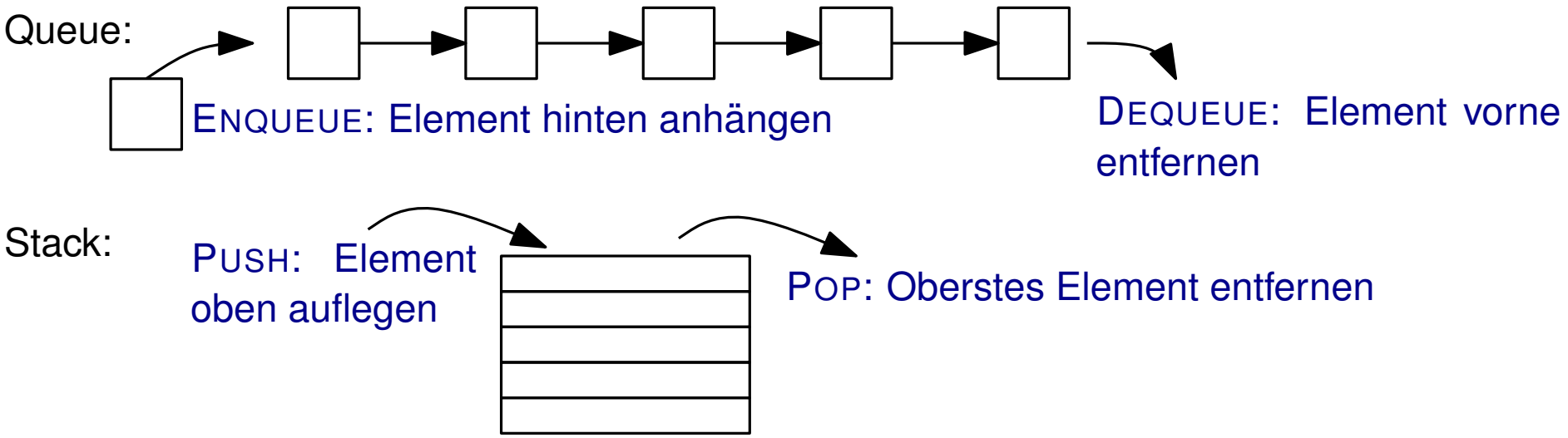
Amortisierte Analyse

Gegeben: Folge an Operationen auf einer Datenstrukturen: $o_1, \ldots, o_m$

Betrachte nicht Operation einzeln, sondern alle ausgeführten Operationen.

Beispiel:

Problem 1: Modellieren Sie eine Queue mit zwei Stacks so, dass die amortisierten Kosten von DEQUEUE und ENQUEUE jeweils in $\mathcal{O}(1)$ sind. Geben Sie die Operationen DEQUEUE und ENQUEUE in Pseudocode an und begründen Sie die amortisierten Kosten.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Lösungsidee

Führe zwei Stacks E und D ein:

ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

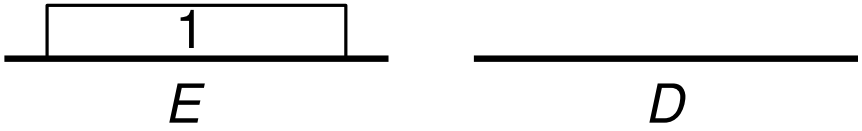
1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Lösungsidee

Führe zwei Stacks E und D ein:



ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

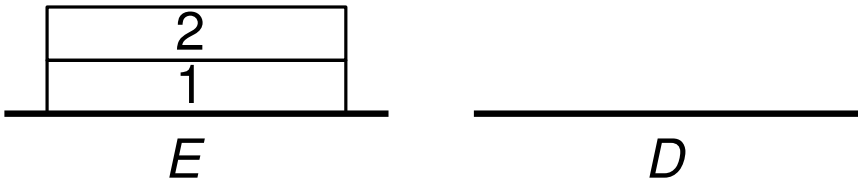
1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Lösungsidee

Führe zwei Stacks E und D ein:



ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

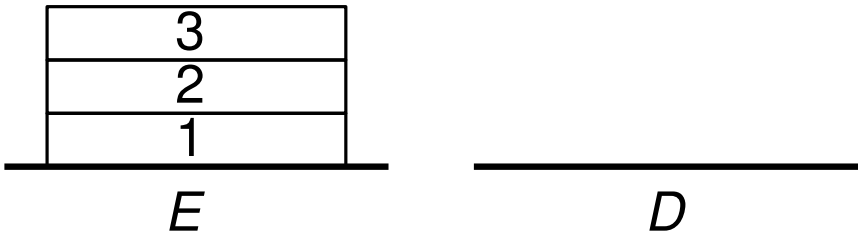
1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Lösungsidee

Führe zwei Stacks E und D ein:



ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

1. PUSH(E,x)

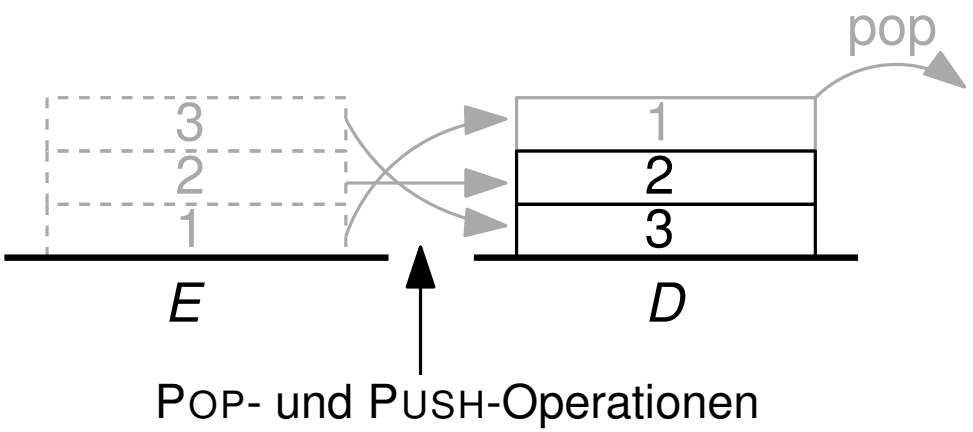
DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Lösungsidee

Führe zwei Stacks E und D ein:



POP- und PUSH-Operationen

ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

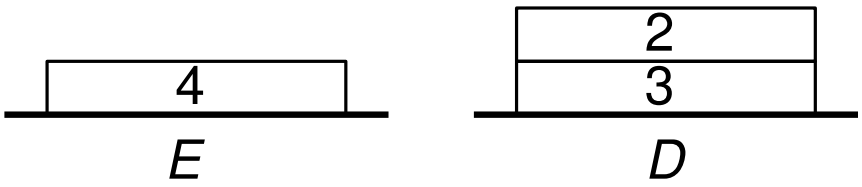
1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Lösungsidee

Führe zwei Stacks E und D ein:



ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

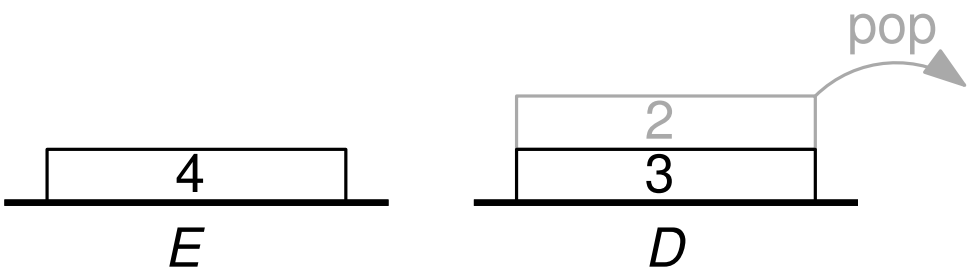
1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Lösungsidee

Führe zwei Stacks E und D ein:



ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

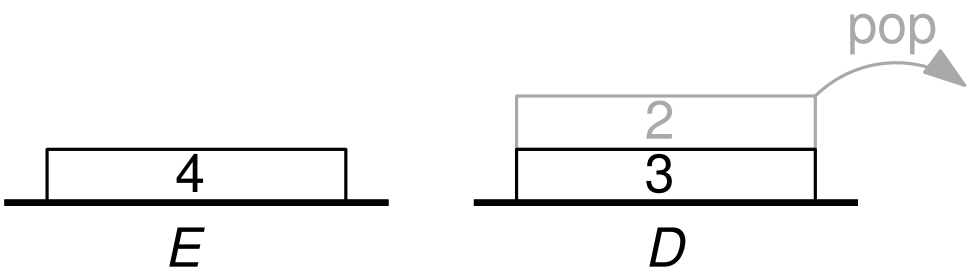
1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Lösungsidee

Führe zwei Stacks E und D ein:



ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

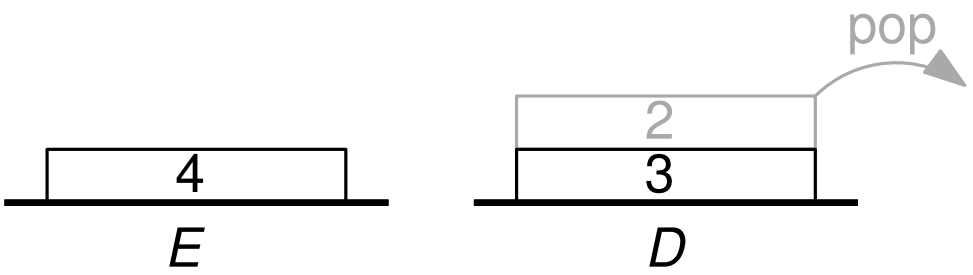
Analyse:

- ENQUEUE benötigt $\mathcal{O}(1)$ Zeit.
- DEQUEUE benötigt $\mathcal{O}(|E|)$ Zeit.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Lösungsidee

Führe zwei Stacks E und D ein:



ENQUEUE(1)
ENQUEUE(2)
ENQUEUE(3)
DEQUEUE
ENQUEUE(4)
DEQUEUE

ENQUEUE(x):

1. PUSH(E,x)

DEQUEUE:

1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Amortisierte Analyse – Buchungsmethode:

	amortisierte Kosten
ENQUEUE	3
DEQUEUE	1

Idee: Jedes Element bekommt Kredit 3, wenn es auf Stack E gelegt wird.

- eine Einheit für PUSH(E,x)
- zwei Einheiten für PUSH(D,POP(E))

Amortisierte Kosten $\mathcal{O}(1)$ für beide Operationen.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Karlsruher Institut für Technologie

Flussnetzwerke

Problemdefinition

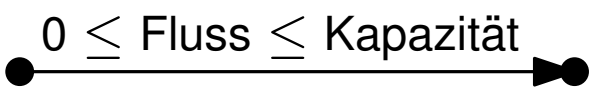
Definition:

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

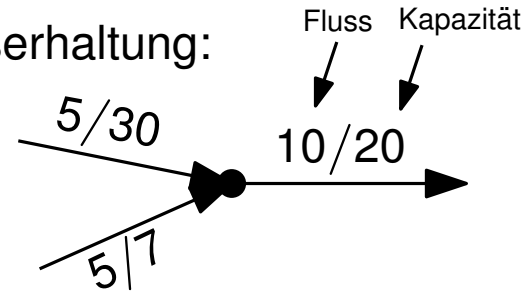
Das Tupel (D, s, t, c) heißt *Netzwerk*. Eine Abbildung $f\colon E \to \mathbb{R}_0^+$ heißt *Fluss*, wenn folgende Bedingungen gelten:

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:



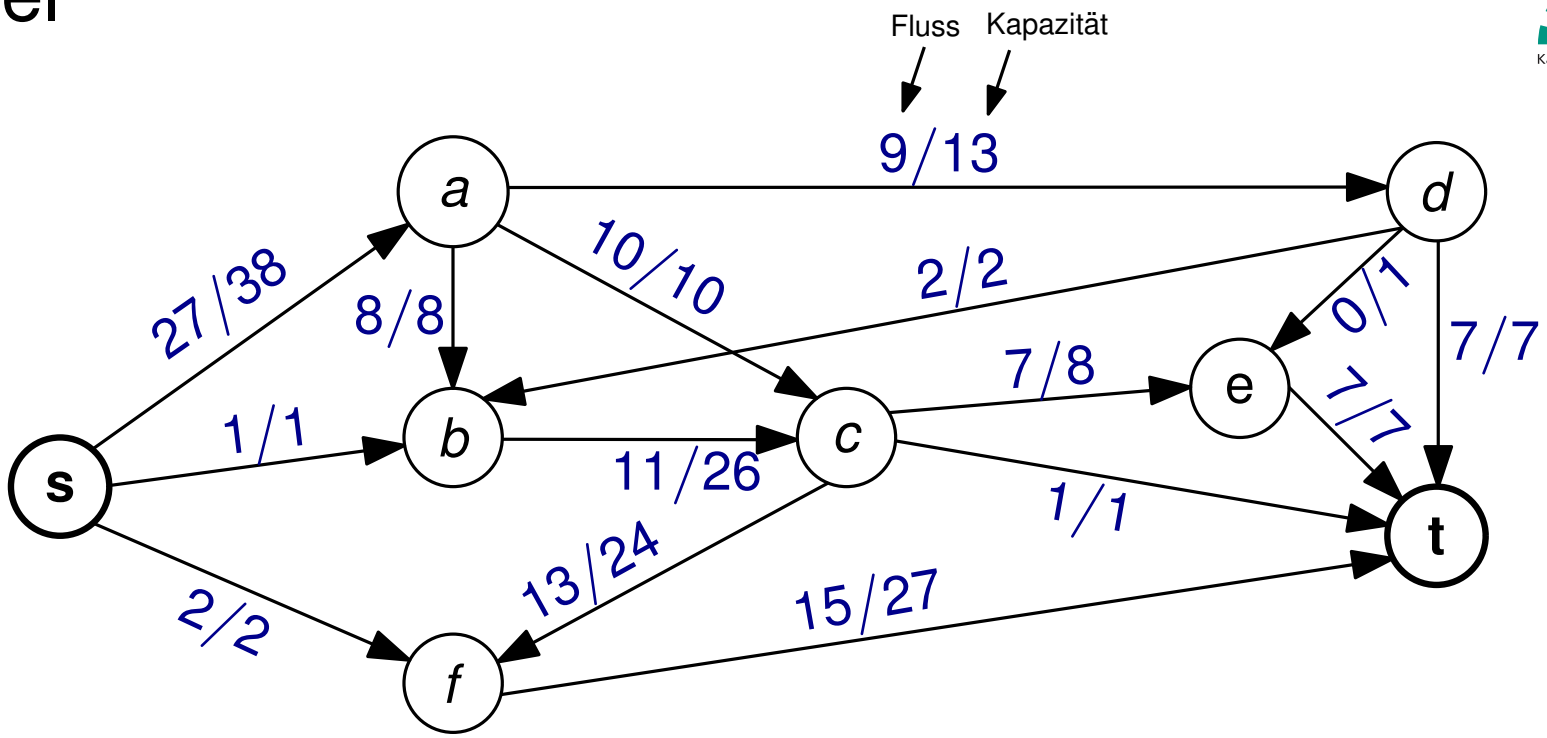
Flusserhaltung:



Zwischenknoten können Fluss weder konsumieren noch produzieren.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Beispiel

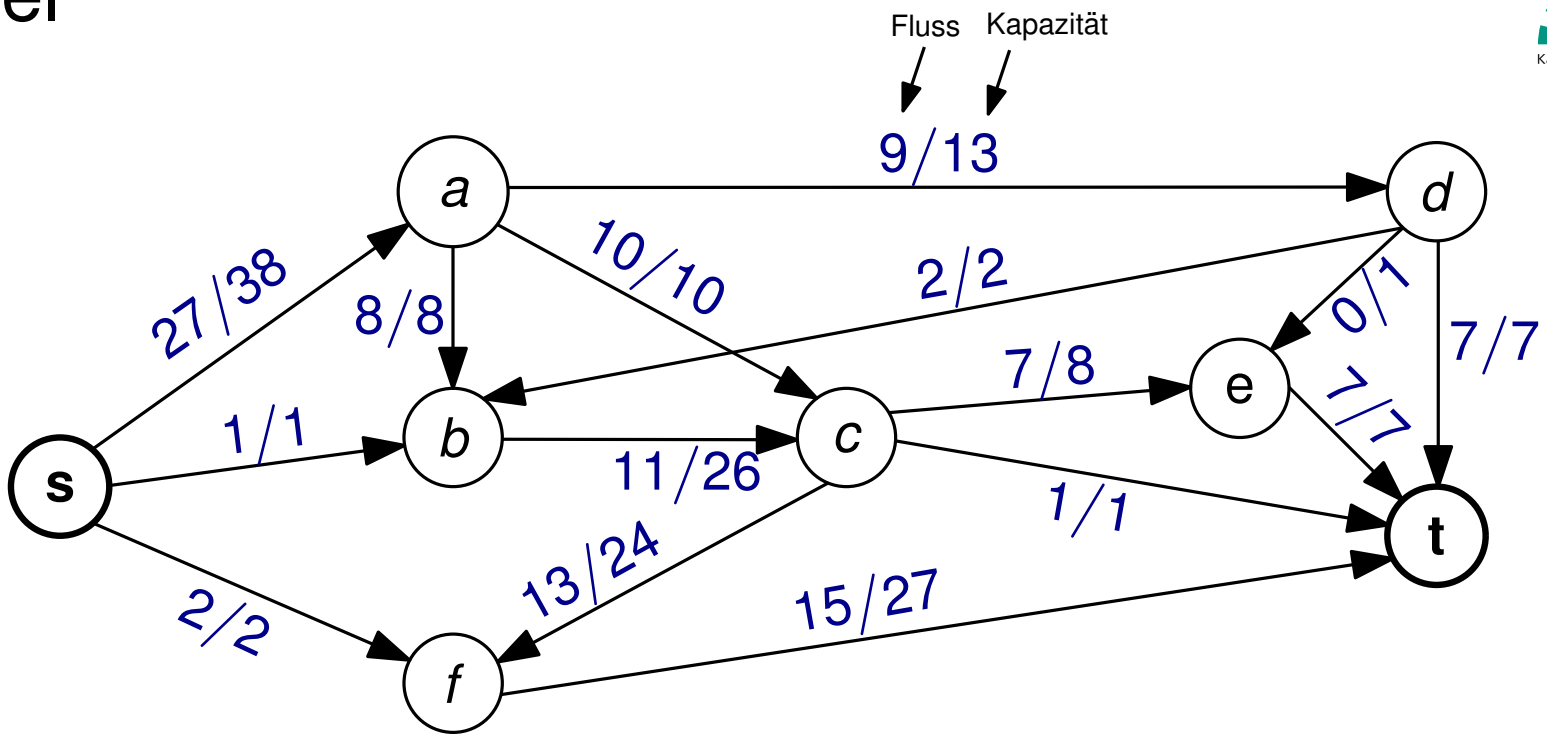


Zuweisung bildet Fluss, denn sowohl Kapazitätsbedingung als auch Flusserhaltung gelten:

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

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Beispiel



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

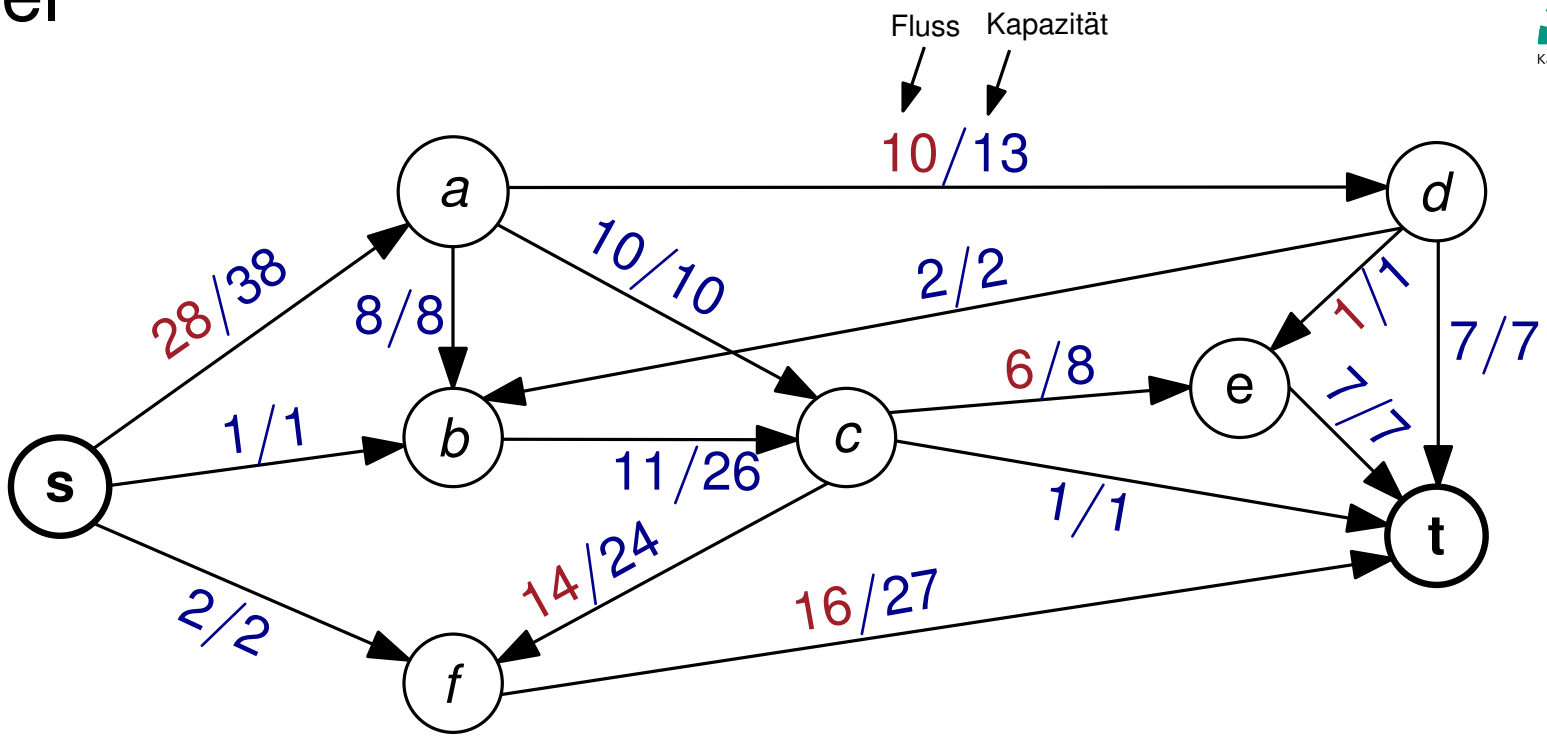
1. *Kapazitätsbedingung:* für alle $(i,j) \in E$ gilt $0 \leq f(i,j) \leq c(i,j)$
2. *Flusserhaltung:* für alle $i \in V \setminus \{s,t\}$ gilt $\sum_{(i,j)\in E} f(i,j) - \sum_{(j,i)\in E} f(j,i) = 0$

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

Was heißt besser? / Wie Fluss messen?

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Beispiel



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

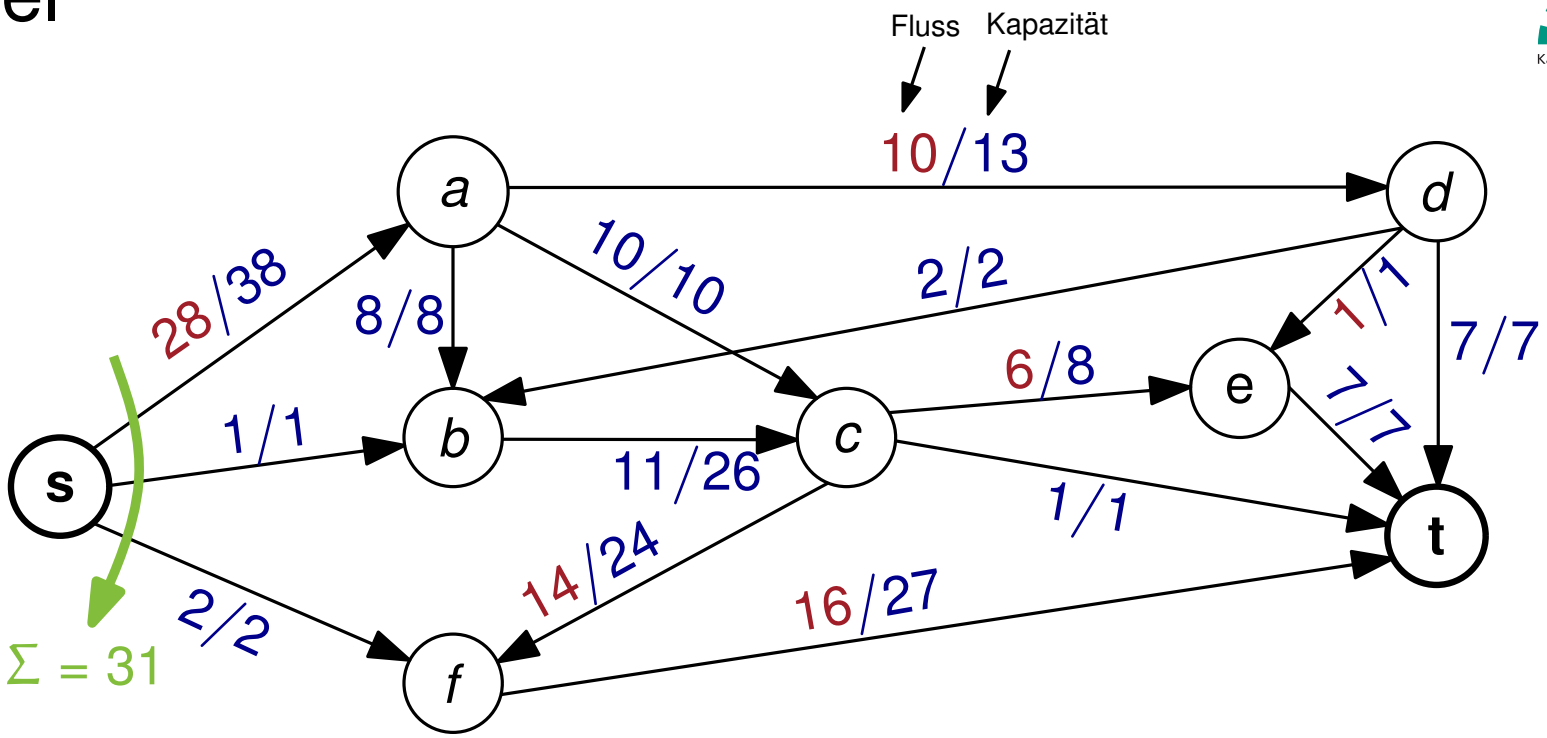
1. *Kapazitätsbedingung:* für alle $(i,j) \in E$ gilt $0 \leq f(i,j) \leq c(i,j)$
2. *Flusserhaltung:* für alle $i \in V \setminus \{s,t\}$ gilt $\sum_{(i,j)\in E} f(i,j) - \sum_{(j,i)\in E} f(j,i) = 0$

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

Was heißt besser? / Wie Fluss messen?

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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

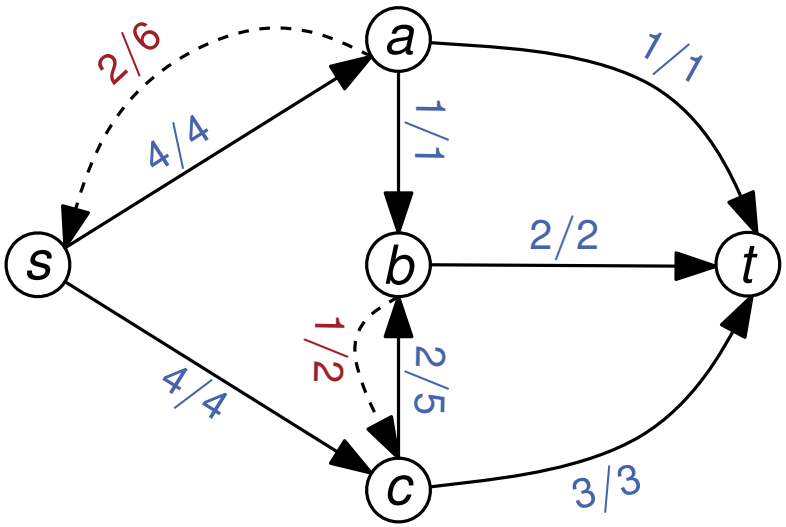
$w(f) := \sum_{(s,i)\in E} f(s,i) - \sum_{(i,s)\in E} f(i,s)$ heißt *Wert* des Flusses f.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 2: Maximale Flüsse

Gegeben:
1. Netzwerk $(D = (V, E), s, t, c)$, sodass zu einigen Kanten $(u, v) \in E$ auch $(v, u) \in E$ exisitiert.
2. Maximaler Fluss $f : E \to \mathbb{R}_0$

Zeige: Es gibt Netzwerk $(D' = (V, E'), s, t, c)$, sodass
- E' maximale Teilemenge von E ist, für die gilt $(u, v) \in E' \Rightarrow (v, u) \notin E'$, und
- für maximalen Fluss f' auf D' gilt $w(f') = w(f)$

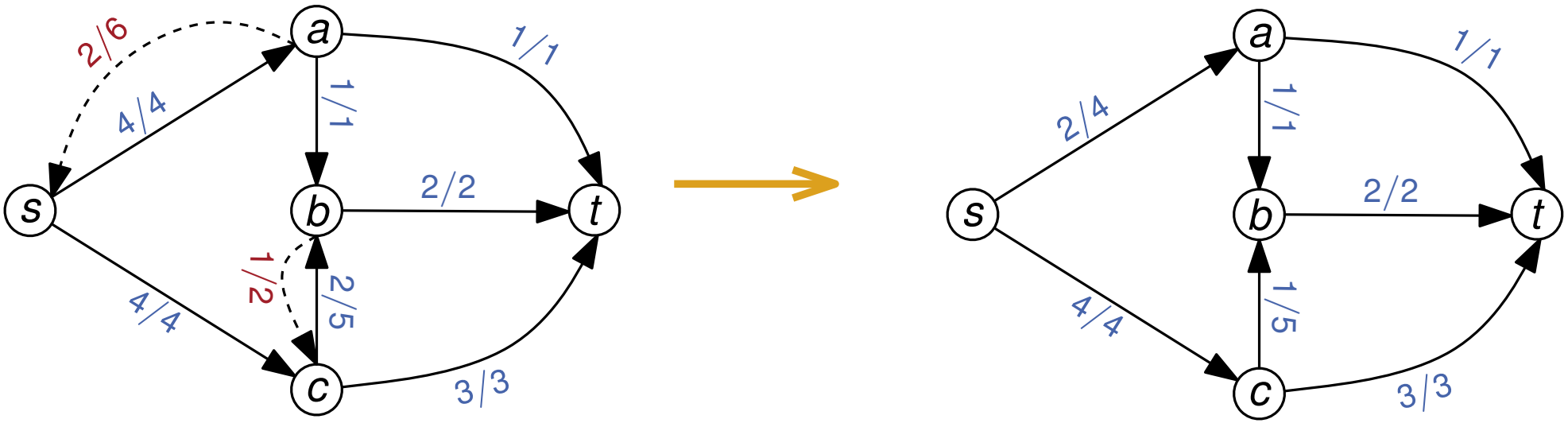


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 2: Maximale Flüsse

Gegeben:
1. Netzwerk $(D = (V, E), s, t, c)$, sodass zu einigen Kanten $(u, v) \in E$ auch $(v, u) \in E$ exisitiert.
2. Maximaler Fluss $f : E \to \mathbb{R}_0$

Zeige: Es gibt Netzwerk $(D' = (V, E'), s, t, c)$, sodass
- E' maximale Teilemenge von E ist, für die gilt $(u, v) \in E' \Rightarrow (v, u) \notin E'$, und
- für maximalen Fluss f' auf D' gilt $w(f') = w(f)$

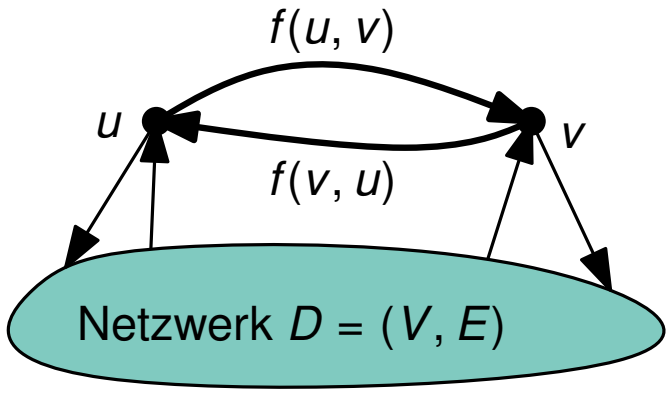


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

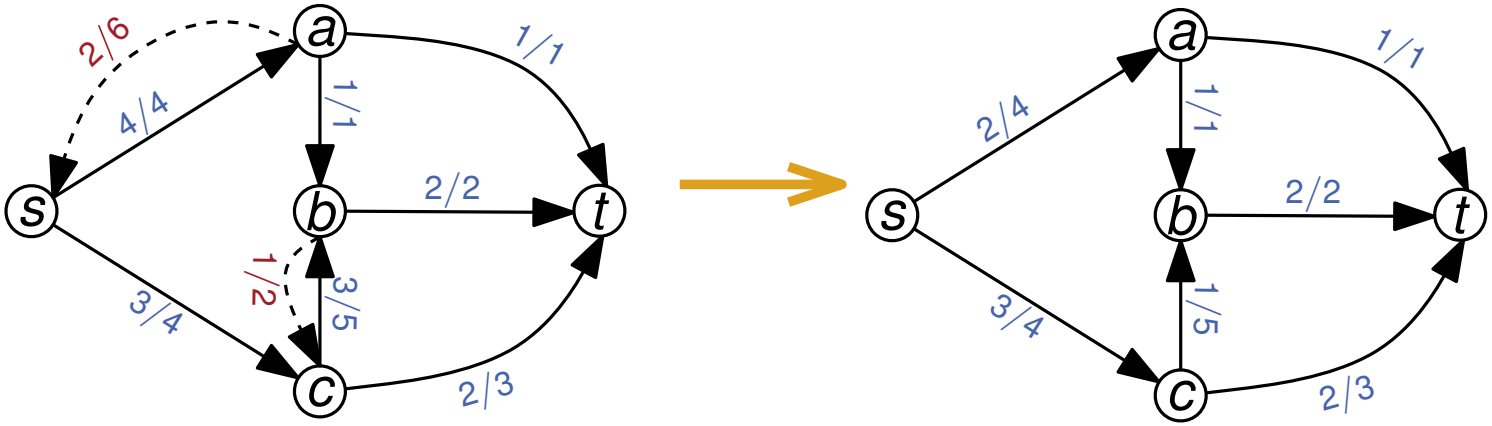
Problem 2: Maximale Flüsse

Idee: Konstruktiver Beweis

- Sei $(u, v) \in E$ mit $(v, u) \in E$
- Betrachte $D' = (V, E')$ mit $E' = E \setminus \{(v, u)\}$
- zeige D' erhält Maximalfluss



o.B.d.A.: $f(u, v) \geq f(v, u)$

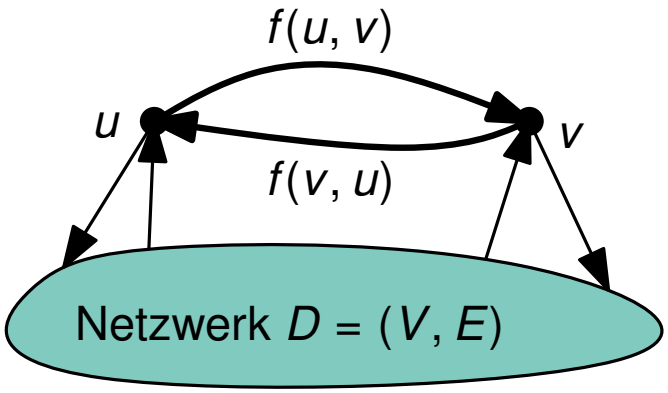


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 2: Maximale Flüsse

Idee: Konstruktiver Beweis

- Sei $(u, v) \in E$ mit $(v, u) \in E$
- Betrachte $D' = (V, E')$ mit $E' = E \setminus \{(v, u)\}$
- zeige D' erhält Maximalfluss

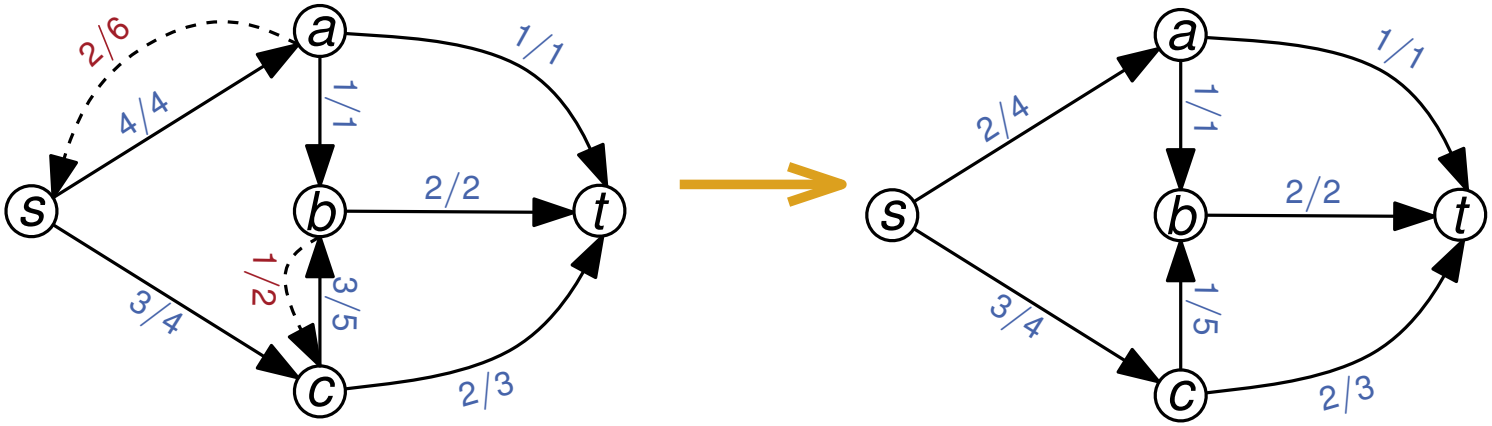


Definiere Fluss f' für alle $(i, j) \in E'$:

$$f'(i,j) := \begin{cases} f(i,j) & \text{falls } (i,j) \neq (u,v) \\ f(i,j) - f(j,i) & \text{falls } (i,j) = (u,v) \end{cases}$$

o.B.d.A.: $f(u, v) \geq f(v, u)$

- Ist f' wirklich ein Fluss?
- Ist f' Maximalfluss in D'?
- Gilt $w(f) = w(f')$?



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

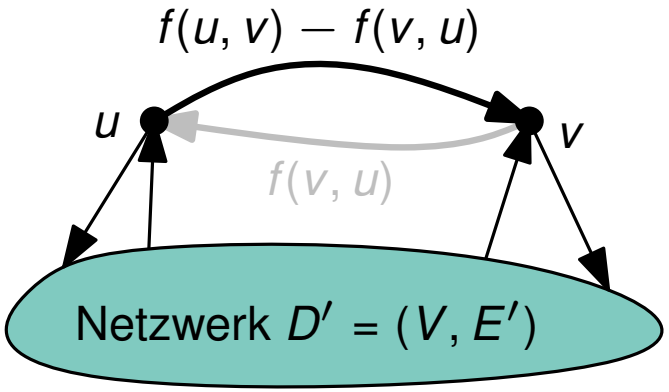
Kapazitätsbedingung

1. Kapazitätsbedingung:

für alle $(i, j) \in E'$ gilt $0 \leq f'(i, j) \leq c(i, j)$

- Für alle Kanten $(i, j) \neq (u, v)$ erfüllt, da bereits für D und f erfüllt.
- Wegen $f(u, v) \geq f(v, u)$ gilt

$$0 \leq f'(u, v) = f(u, v) - f(v, u) \leq c(u, v)$$



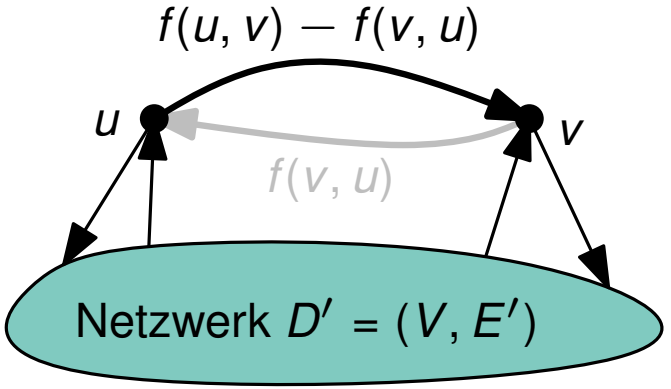
o.B.d.A.: $f(u, v) \geq f(v, u)$

Flusserhaltung

2. Flusserhaltung: für alle $i \in V \setminus \{s, t\}$ gilt $\sum_{(i,j) \in E'} f'(i,j) - \sum_{(j,i) \in E'} f'(j,i) = 0$

Für alle $i \in V \setminus \{s, t, u, v\}$ erfüllt. Für den Knoten u betrachte Wert $w'(u)$:

$$w'(u) = \sum_{(w,u) \in E'} f'(w,u) - \sum_{(u,w) \in E'} f'(u,w)$$



o.B.d.A.: $f(u,v) \geq f(v,u)$

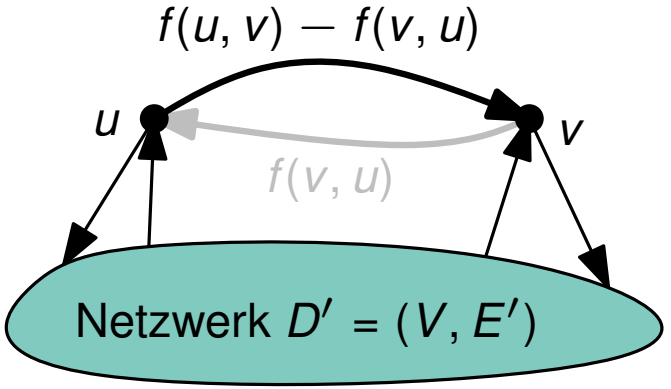
Flusserhaltung

2. Flusserhaltung: für alle $i \in V \setminus \{s,t\}$ gilt $\sum_{(i,j)\in E'} f'(i,j) - \sum_{(j,i)\in E'} f'(j,i) = 0$

Für alle $i \in V \setminus \{s,t,u,v\}$ erfüllt. Für den Knoten u betrachte Wert $w'(u)$:

$$w'(u) = \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'} f'(u,w)$$

$$= \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'\setminus\{(u,v)\}} f'(u,w) - f'(u,v)$$



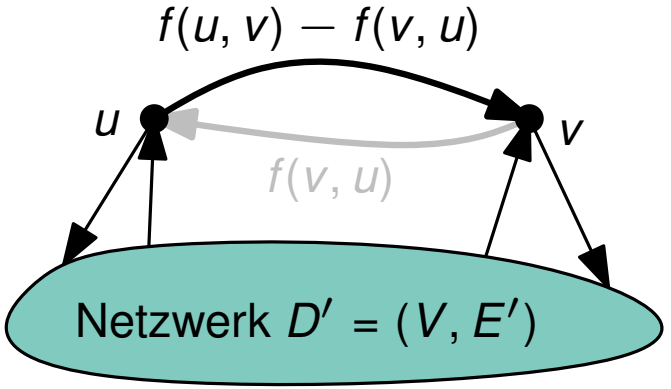
o.B.d.A.: $f(u,v) \geq f(v,u)$

Flusserhaltung

2. Flusserhaltung: für alle $i \in V \setminus \{s,t\}$ gilt $\sum_{(i,j)\in E'} f'(i,j) - \sum_{(j,i)\in E'} f'(j,i) = 0$

Für alle $i \in V \setminus \{s,t,u,v\}$ erfüllt. Für den Knoten u betrachte Wert $w'(u)$:

$$
\begin{aligned}
w'(u) &= \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'} f'(u,w) \\
&= \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'\setminus\{(u,v)\}} f'(u,w) - f'(u,v) \\
&= \sum_{(w,u)\in E'} f(w,u) - \sum_{(u,w)\in E'\setminus\{(u,v)\}} f(u,w) - (f(u,v) - f(v,u))
\end{aligned}
$$



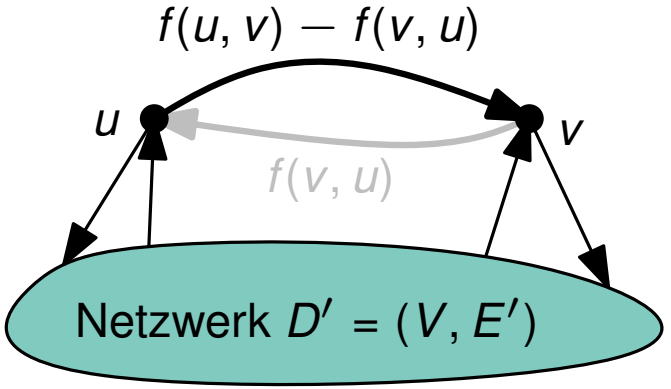
o.B.d.A.: $f(u,v) \geq f(v,u)$

Flusserhaltung

2. Flusserhaltung: für alle $i \in V \setminus \{s,t\}$ gilt $\sum_{(i,j)\in E'} f'(i,j) - \sum_{(j,i)\in E'} f'(j,i) = 0$

Für alle $i \in V \setminus \{s,t,u,v\}$ erfüllt. Für den Knoten u betrachte Wert $w'(u)$:

$$
\begin{aligned}
w'(u) &= \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'} f'(u,w) \\
&= \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'\setminus\{(u,v)\}} f'(u,w) - f'(u,v) \\
&= \sum_{(w,u)\in E'} f(w,u) - \sum_{(u,w)\in E'\setminus\{(u,v)\}} f(u,w) - (f(u,v) - f(v,u)) \\
&= \sum_{(w,u)\in E} f(w,u) - \sum_{(u,w)\in E} f(u,w)
\end{aligned}
$$



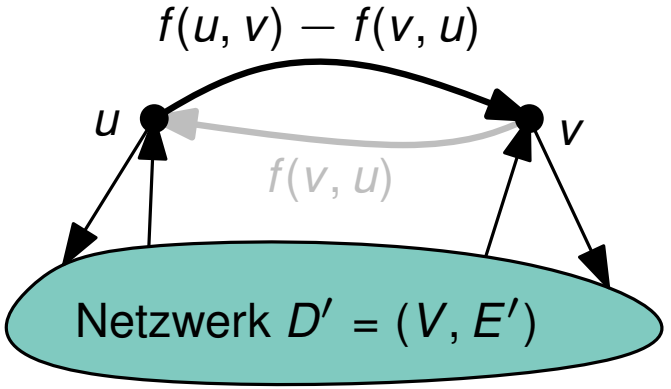
o.B.d.A.: $f(u,v) \geq f(v,u)$

Flusserhaltung

2. Flusserhaltung: für alle $i \in V \setminus \{s,t\}$ gilt $\sum_{(i,j)\in E'} f'(i,j) - \sum_{(j,i)\in E'} f'(j,i) = 0$

Für alle $i \in V \setminus \{s,t,u,v\}$ erfüllt. Für den Knoten u betrachte Wert $w'(u)$:

$$
\begin{aligned}
w'(u) &= \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'} f'(u,w) \\
&= \sum_{(w,u)\in E'} f'(w,u) - \sum_{(u,w)\in E'\setminus\{(u,v)\}} f'(u,w) - f'(u,v) \\
&= \sum_{(w,u)\in E'} f(w,u) - \sum_{(u,w)\in E'\setminus\{(u,v)\}} f(u,w) - (f(u,v) - f(v,u)) \\
&= \sum_{(w,u)\in E} f(w,u) - \sum_{(u,w)\in E} f(u,w) \\
&= w(u)
\end{aligned}
$$



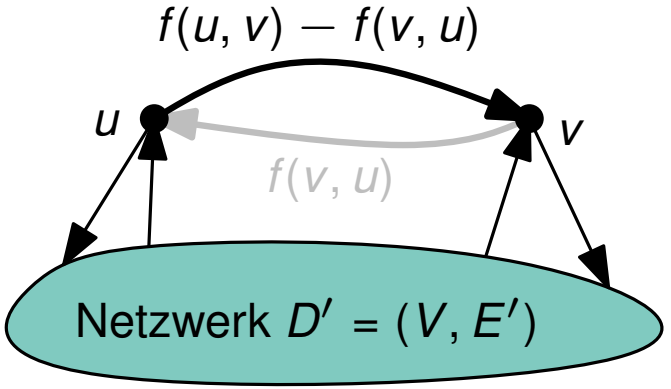
o.B.d.A.: $f(u,v) \geq f(v,u)$

Flusserhaltung

2. Flusserhaltung: für alle $i \in V \setminus \{s, t\}$ gilt $\sum_{(i,j) \in E'} f'(i,j) - \sum_{(j,i) \in E'} f'(j,i) = 0$

Für alle $i \in V \setminus \{s, t, u, v\}$ erfüllt. Für den Knoten u betrachte Wert $w'(u)$:

$$w'(u) = \sum_{(w,u) \in E'} f'(w,u) - \sum_{(u,w) \in E'} f'(u,w) = w(u)$$



o.B.d.A.: $f(u,v) \geq f(v,u)$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Flusserhaltung

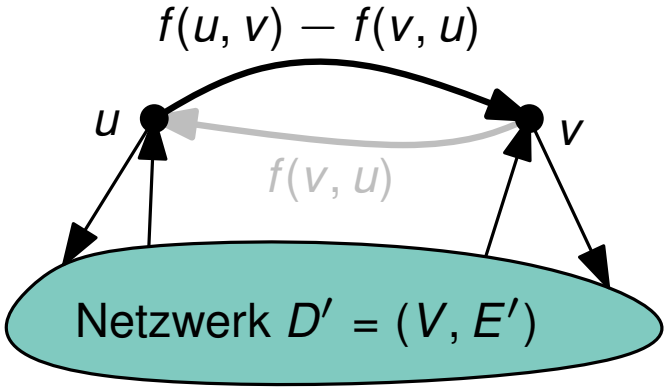
2. Flusserhaltung: für alle $i \in V \setminus \{s, t\}$ gilt $\sum_{(i,j) \in E'} f'(i,j) - \sum_{(j,i) \in E'} f'(j,i) = 0$

Für alle $i \in V \setminus \{s, t, u, v\}$ erfüllt. Für den Knoten u betrachte Wert $w'(u)$:

$$w'(u) = \sum_{(w,u) \in E'} f'(w,u) - \sum_{(u,w) \in E'} f'(u,w) = w(u)$$

- Falls $u \neq s$ und $u \neq t$, dann $w(u) = 0 \rightarrow$ Flusserhaltung gilt für u.
- Falls $u = s$ oder $u = t$, dann $\mathbf{w}(\mathbf{f}') = w'(u) = w(u) = \mathbf{w}(\mathbf{f})$

Analoges Vorgehen für Knoten v.



o.B.d.A.: $f(u,v) \geq f(v,u)$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

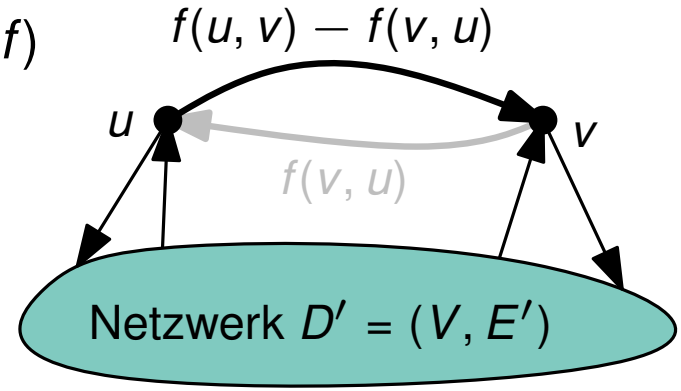
Maximalfluss

Maximalfluss wird erhalten:

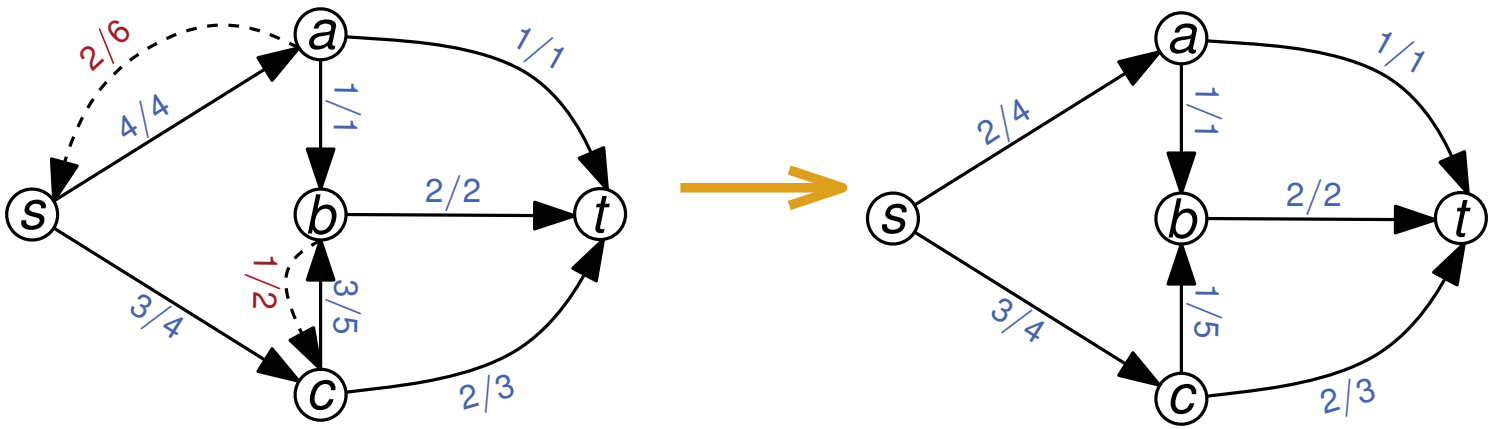
Sei $\hat{f}$ Maximalfluss in D':

- $w'(\hat{f}) \geq w'(f') = w(f)$
- Jeder Fluss in D' ist auch ein Fluss in D: $w(\hat{f}) \leq w(f)$

$\rightarrow w(f) = w'(f') = w'(\hat{f})$



o.B.d.A.: $f(u, v) \geq f(v, u)$

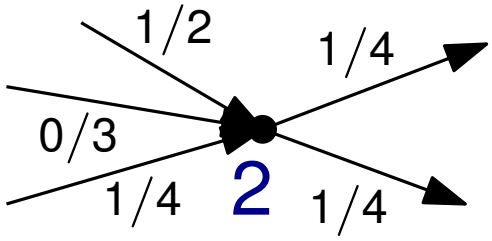


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

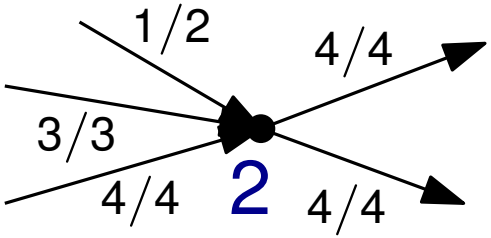
Problem 3: Flüsse mit Knotenkapazitäten

Erweitere Netzwerk um Knotenkapazitäten $\gamma\colon V \to \mathbb{R}_0^+$: (D, s, t, c, γ)

Abbildung $f\colon E \to \mathbb{R}_0^+$ heißt *Fluss* in (D, s, t, c, γ), wenn:



gültig



nicht gültig

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

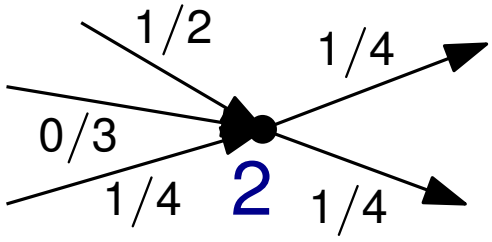
Problem 3: Flüsse mit Knotenkapazitäten

Erweitere Netzwerk um Knotenkapazitäten $\gamma\colon V \to \mathbb{R}_0^+$: (D, s, t, c, γ)

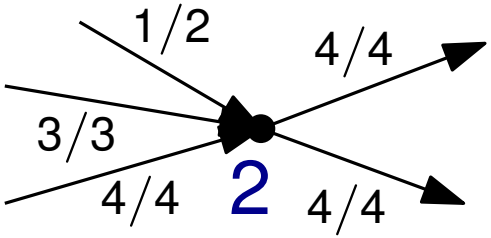
Abbildung $f\colon E \to \mathbb{R}_0^+$ heißt *Fluss* in (D, s, t, c, γ), wenn:

1. Kapazitätsbedingung:

$0 \leq f(i,j) \leq c(i,j)$ für alle $(i,j) \in E$



gültig



nicht gültig

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

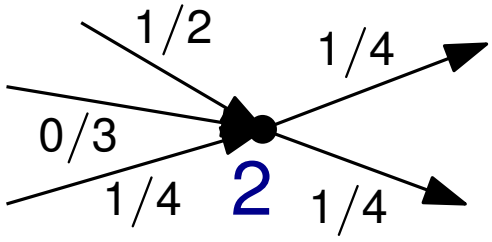
Problem 3: Flüsse mit Knotenkapazitäten

Erweitere Netzwerk um Knotenkapazitäten $\gamma\colon V \to \mathbb{R}_0^+$: (D, s, t, c, γ)

Abbildung $f\colon E \to \mathbb{R}_0^+$ heißt *Fluss* in (D, s, t, c, γ), wenn:

1. Kapazitätsbedingung:

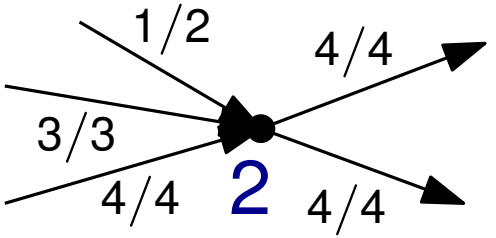
$$0 \leq f(i,j) \leq c(i,j) \text{ für alle } (i,j) \in E$$



gültig

2. Flusserhaltung:

$$\sum_{(i,j)\in E} f(i,j) - \sum_{(j,i)\in E} f(j,i) = 0 \text{ für alle } i \in V \setminus \{s,t\}$$



nicht gültig

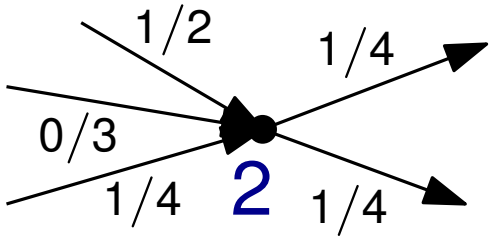
Problem 3: Flüsse mit Knotenkapazitäten

Erweitere Netzwerk um Knotenkapazitäten $\gamma\colon V \to \mathbb{R}_0^+$: (D, s, t, c, γ)

Abbildung $f\colon E \to \mathbb{R}_0^+$ heißt *Fluss* in (D, s, t, c, γ), wenn:

1. Kapazitätsbedingung:

$$0 \leq f(i,j) \leq c(i,j) \text{ für alle } (i,j) \in E$$



gültig

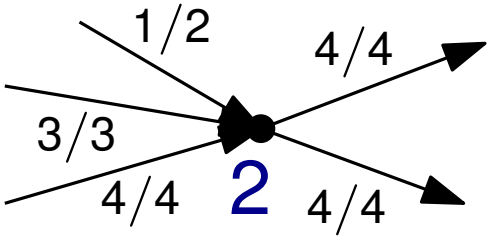
2. Flusserhaltung:

$$\sum_{(i,j)\in E} f(i,j) - \sum_{(j,i)\in E} f(j,i) = 0 \text{ für alle } i \in V \setminus \{s,t\}$$

3. Kontenkapazitätsbedinung:

$$\sum_{(v,w)\in E} f(v,w) \leq \gamma(v) \qquad \text{wenn } v \in V \setminus \{t\}$$

$$\sum_{(u,v)\in E} f(u,v) \leq \gamma(v) \qquad \text{wenn } v = t$$



nicht gültig

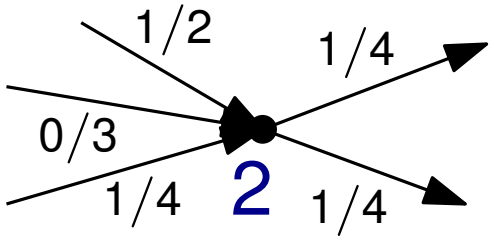
Problem 3: Flüsse mit Knotenkapazitäten

Erweitere Netzwerk um Knotenkapazitäten $\gamma : V \to \mathbb{R}_0^+$: (D, s, t, c, γ)

Abbildung $f : E \to \mathbb{R}_0^+$ heißt *Fluss* in (D, s, t, c, γ), wenn:

1. Kapazitätsbedingung:

$$0 \leq f(i,j) \leq c(i,j) \text{ für alle } (i,j) \in E$$



gültig

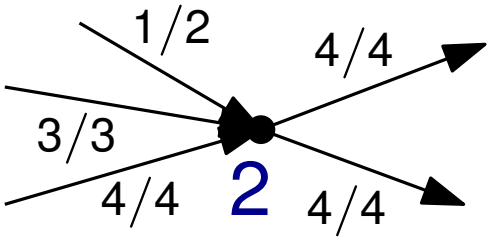
2. Flusserhaltung:

$$\sum_{(i,j)\in E} f(i,j) - \sum_{(j,i)\in E} f(j,i) = 0 \text{ für alle } i \in V \setminus \{s,t\}$$

3. Kontenkapazitätsbedinung:

$$\sum_{(v,w)\in E} f(v,w) \leq \gamma(v) \qquad \text{wenn } v \in V \setminus \{t\}$$

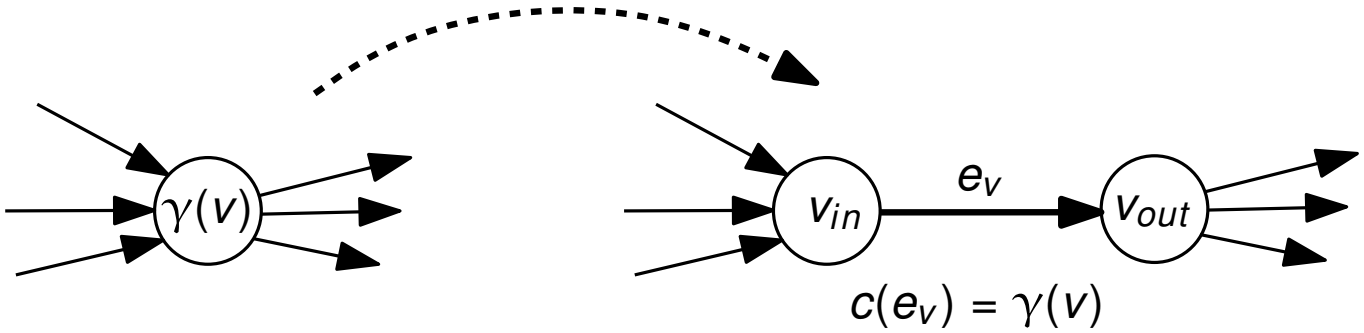
$$\sum_{(u,v)\in E} f(u,v) \leq \gamma(v) \qquad \text{wenn } v = t$$



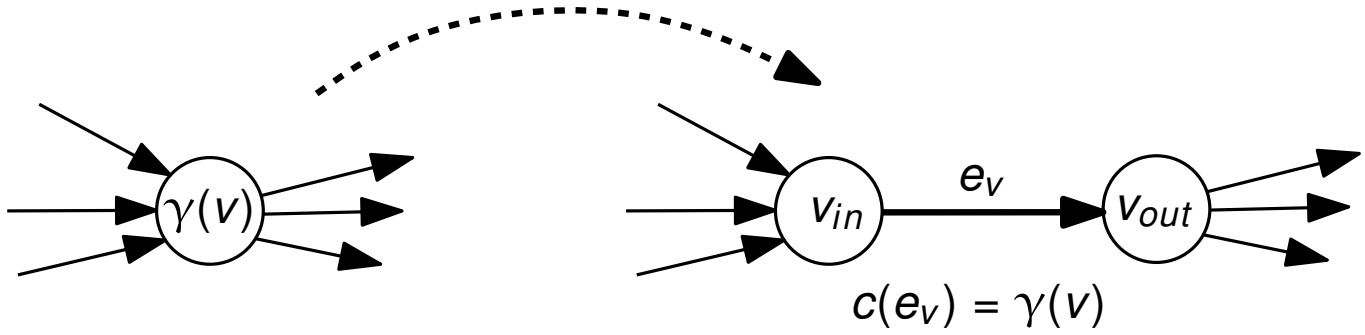
nicht gültig

Zeige: Keine wirkliche Erweiterung zur klassischen Formulierung.

Lösung



Lösung



Netzwerk $N' = (D' = (V', E'); s'; t'; c')$ aus Netzwerk $N = (D = (V, E); s; t; c; \gamma)$:

- Knoten: $V' = \bigcup_{v \in V} \{v_{in}, v_{out}\}$
- Quelle und Senke: $s' = s_{\text{in}}$, und $t' = t_{\text{out}}$
- Kanten $E' = \{(v_{\text{out}}, w_{\text{in}}) \mid (v, w) \in E\} \quad \cup \quad \{(v_{\text{in}}, v_{\text{out}}) \mid v \in V\}$
- Kapazitäten: $c' : E' \to \mathbb{R}_0^+$ mit $c'(v_{\text{out}}, w_{\text{in}}) = c(v, w)$ und $c'(v_{\text{in}}, v_{\text{out}}) = \gamma(v)$

Begründung:

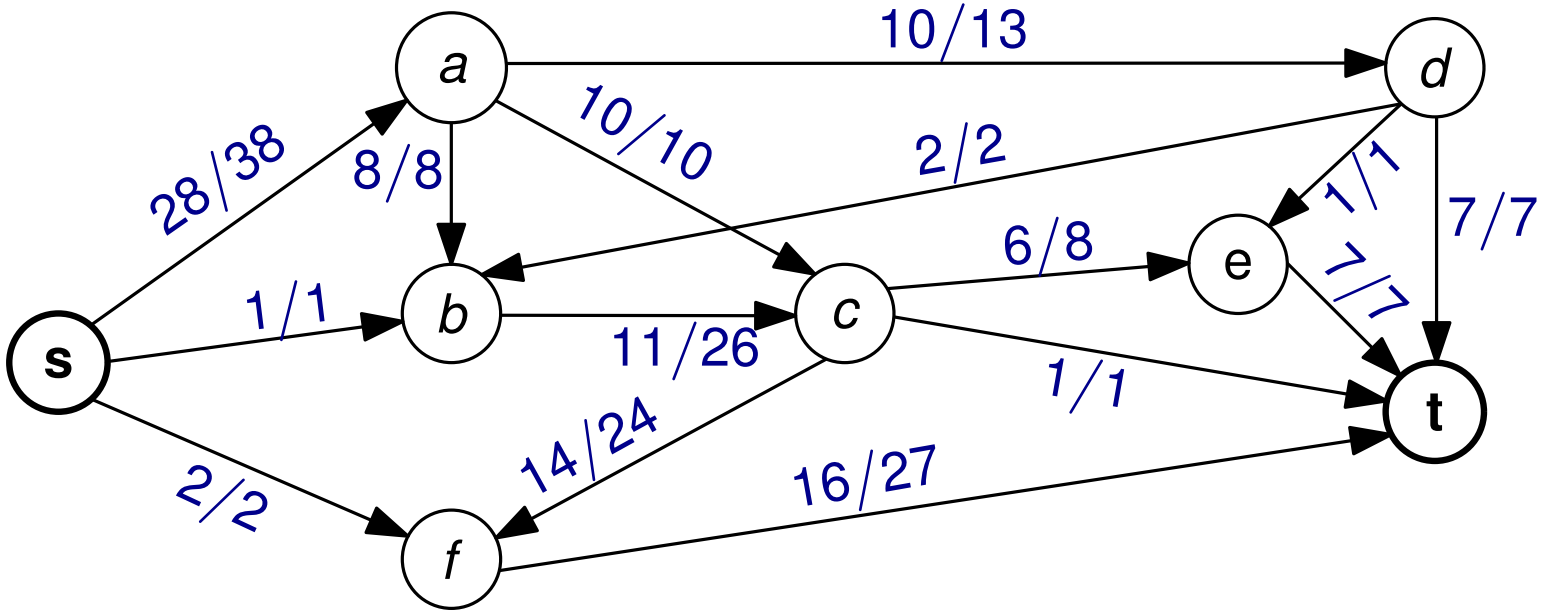
- Knotenbedingung wird durch Kapazität der neuen Kante realisiert.
- Nachbarschaften und weitere Kapazitätsbedinung bleiben erhalten.

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.



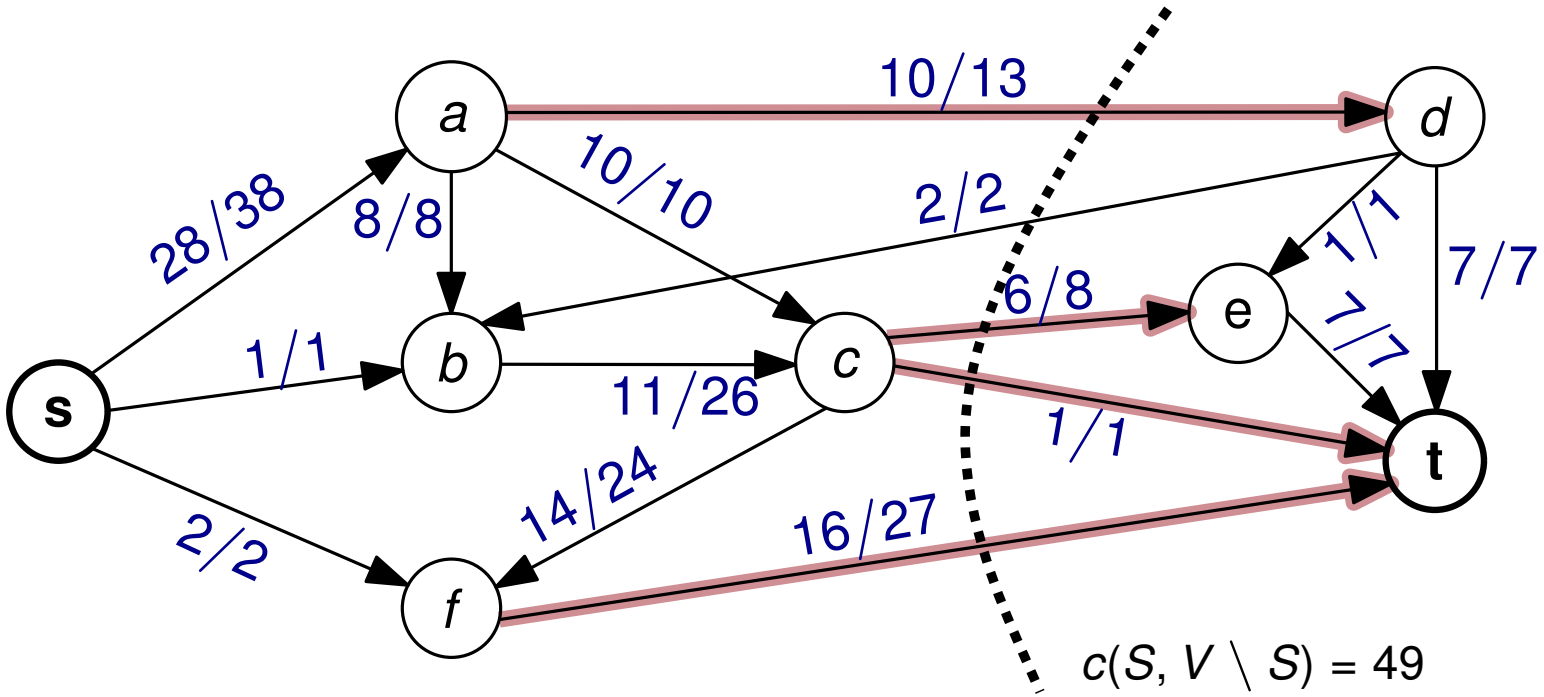
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.



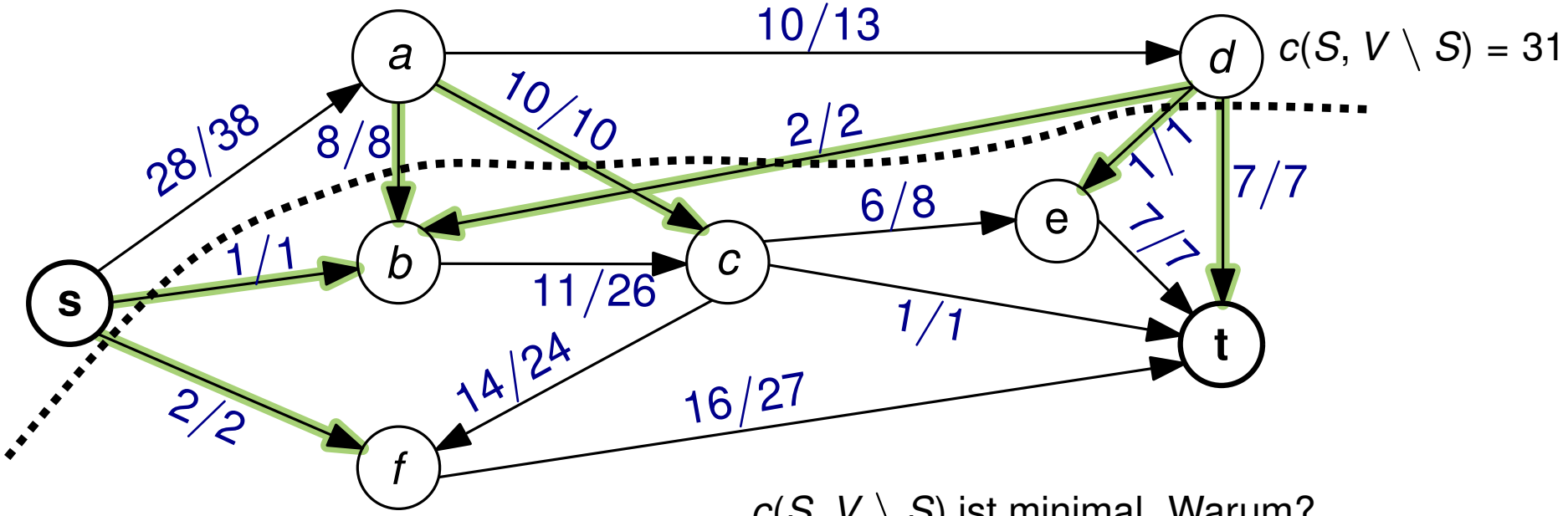
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.



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

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.

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

Karlsruher Institut für Technologie

Lineare Programme

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**:

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

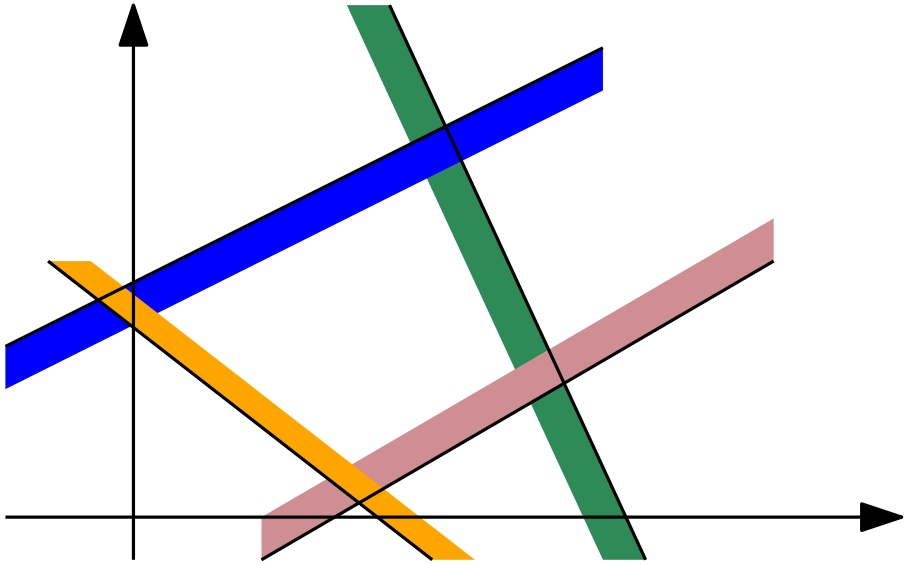
Ziel: bestimme $x_1, \ldots, x_n$ so, dass $f(\overline{x})$ maximal/minimal ist.

Matrixschreibweise des LP:

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

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

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



Lösungsraum 2-dimensionales LP

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

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.

x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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

Nebenbedingungen **NB:**

$$12\,x_1 + 6\,x_2 \leq 630$$
$$8\,x_1 + 12\,x_2 \leq 620$$
$$10\,x_2 \leq 350$$
$$x_1 \geq 10$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

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	

Geometrische Interpretation

Skizze:

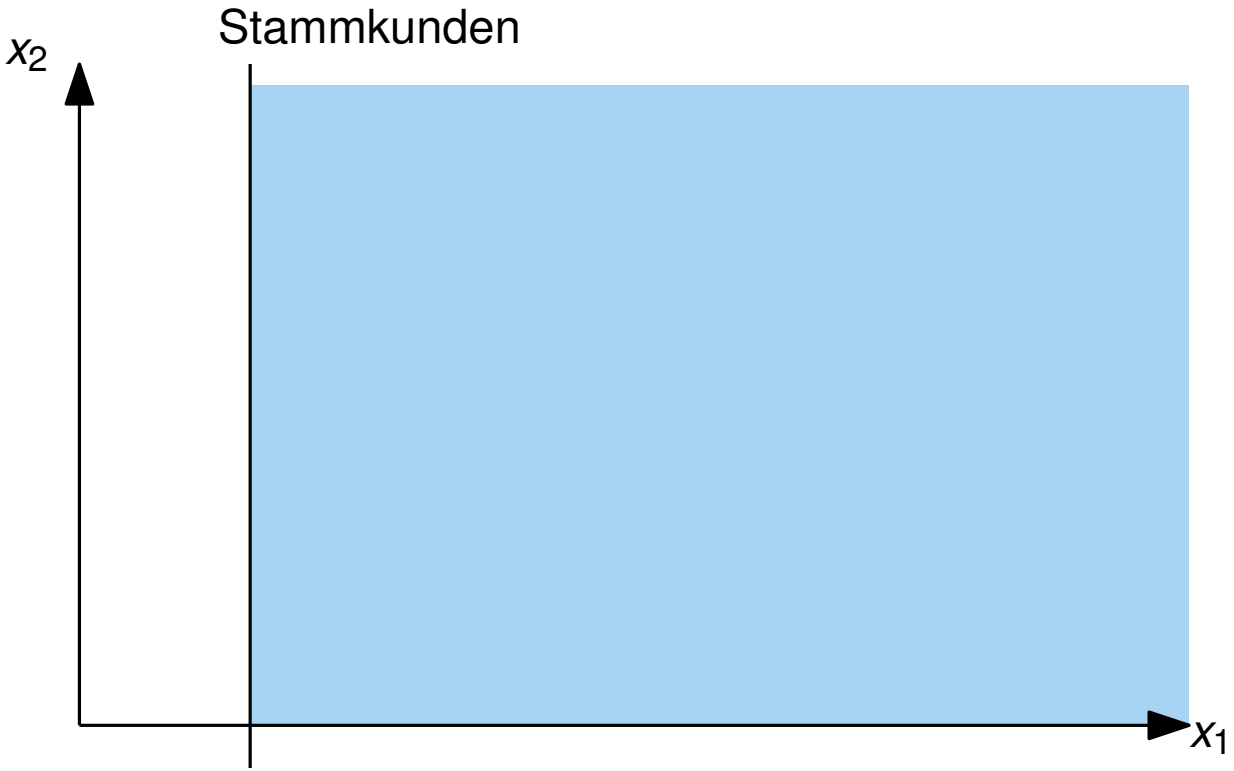
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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:

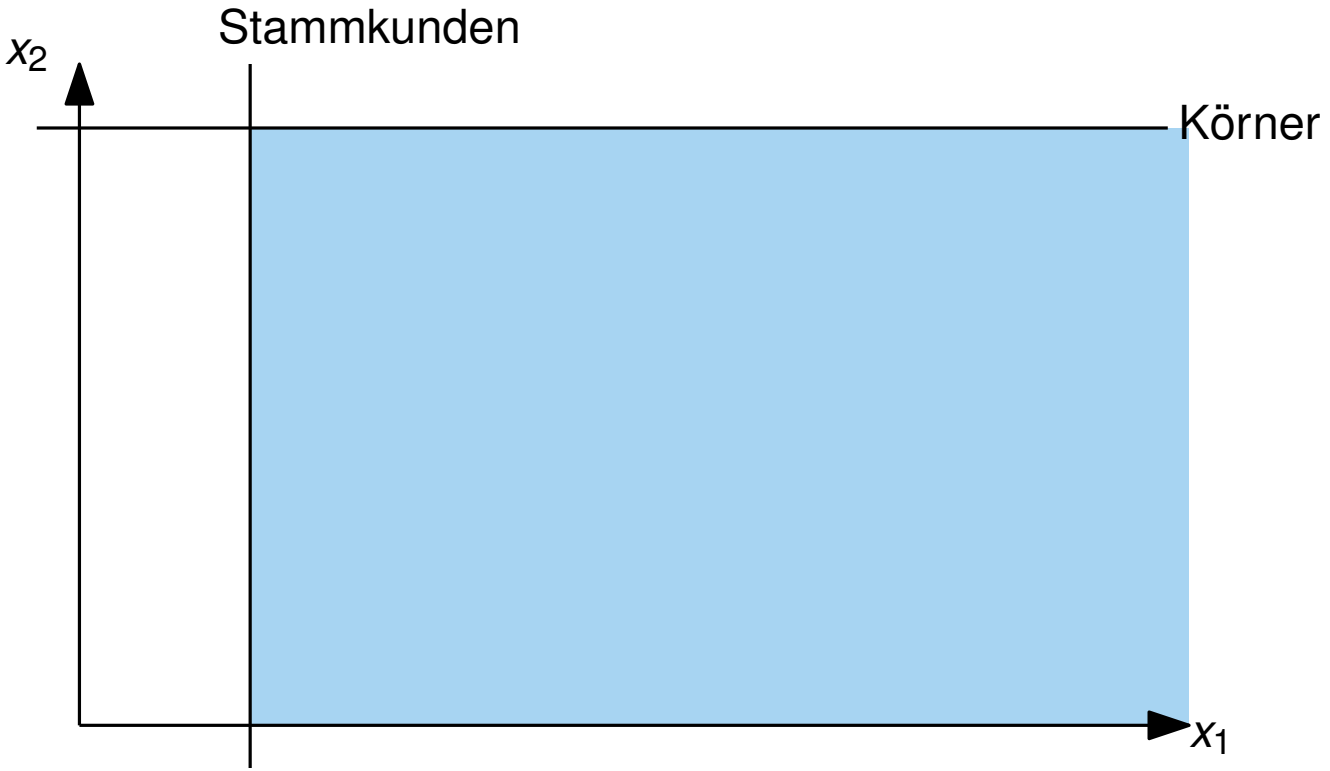


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

Zielfunktion **ZF:**	$f(x_1, x_2) =$	20	x_1	+	60	x_2	= max!		
Nebenbedingungen **NB:**		12	x_1	+	6	x_2	$\leq$ 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		

Geometrische Interpretation

Skizze:

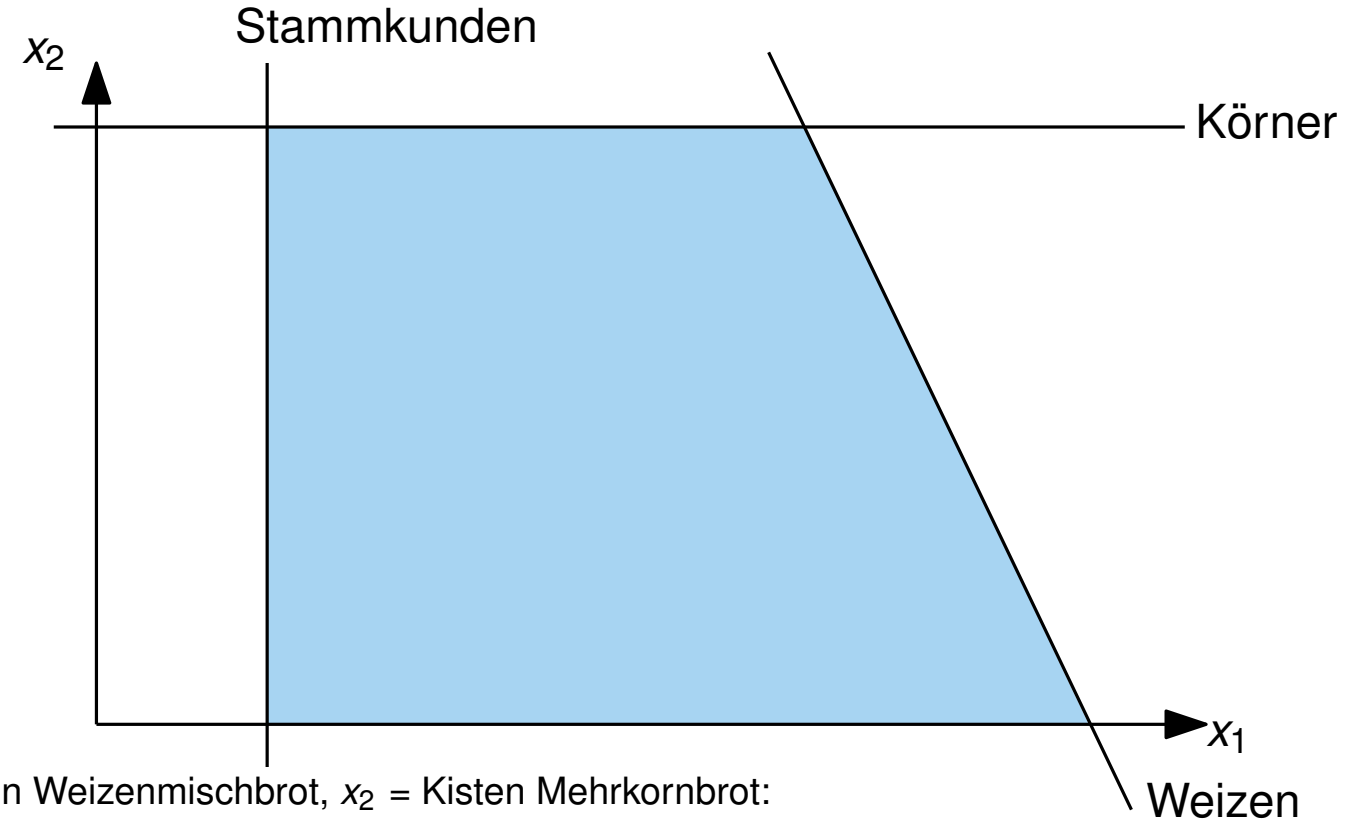


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Geometrische Interpretation

Skizze:

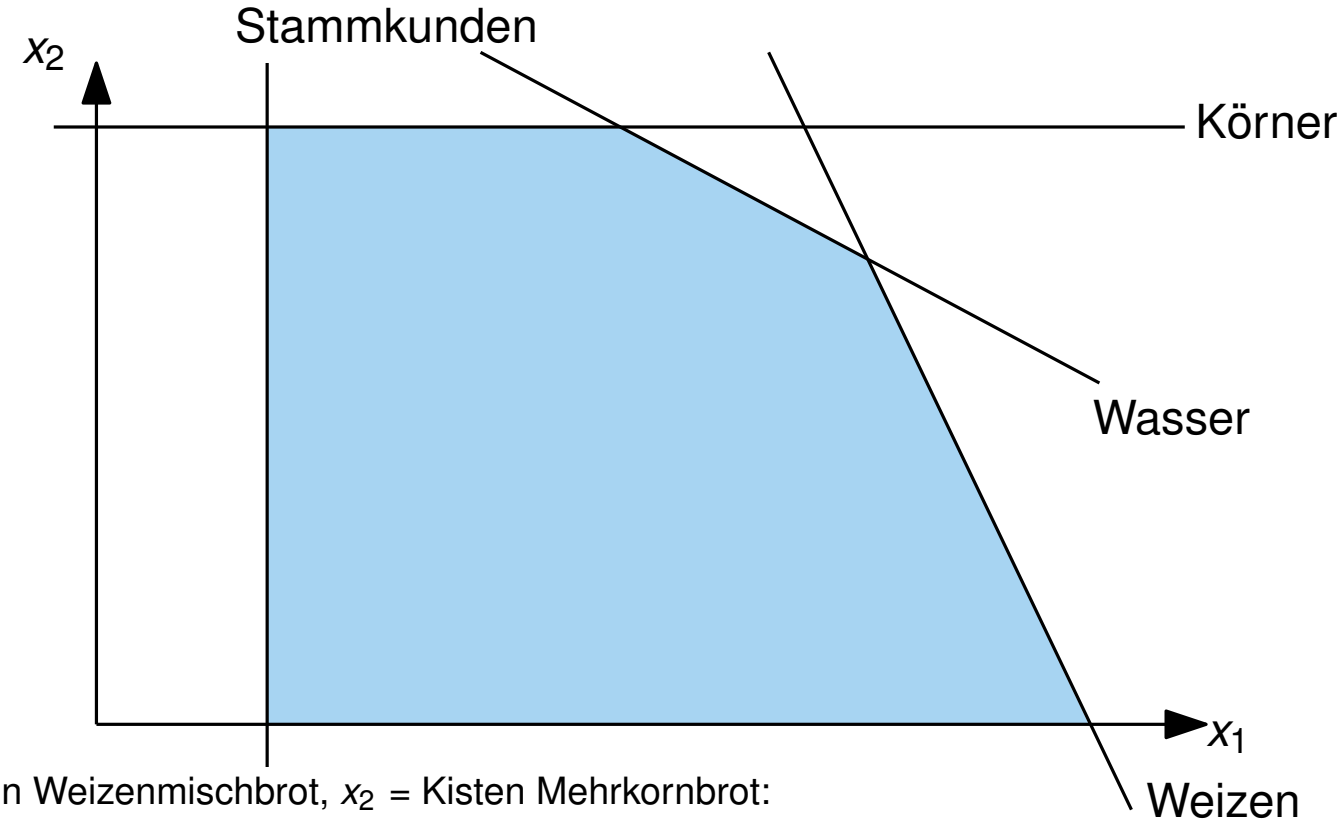


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Geometrische Interpretation

Skizze:

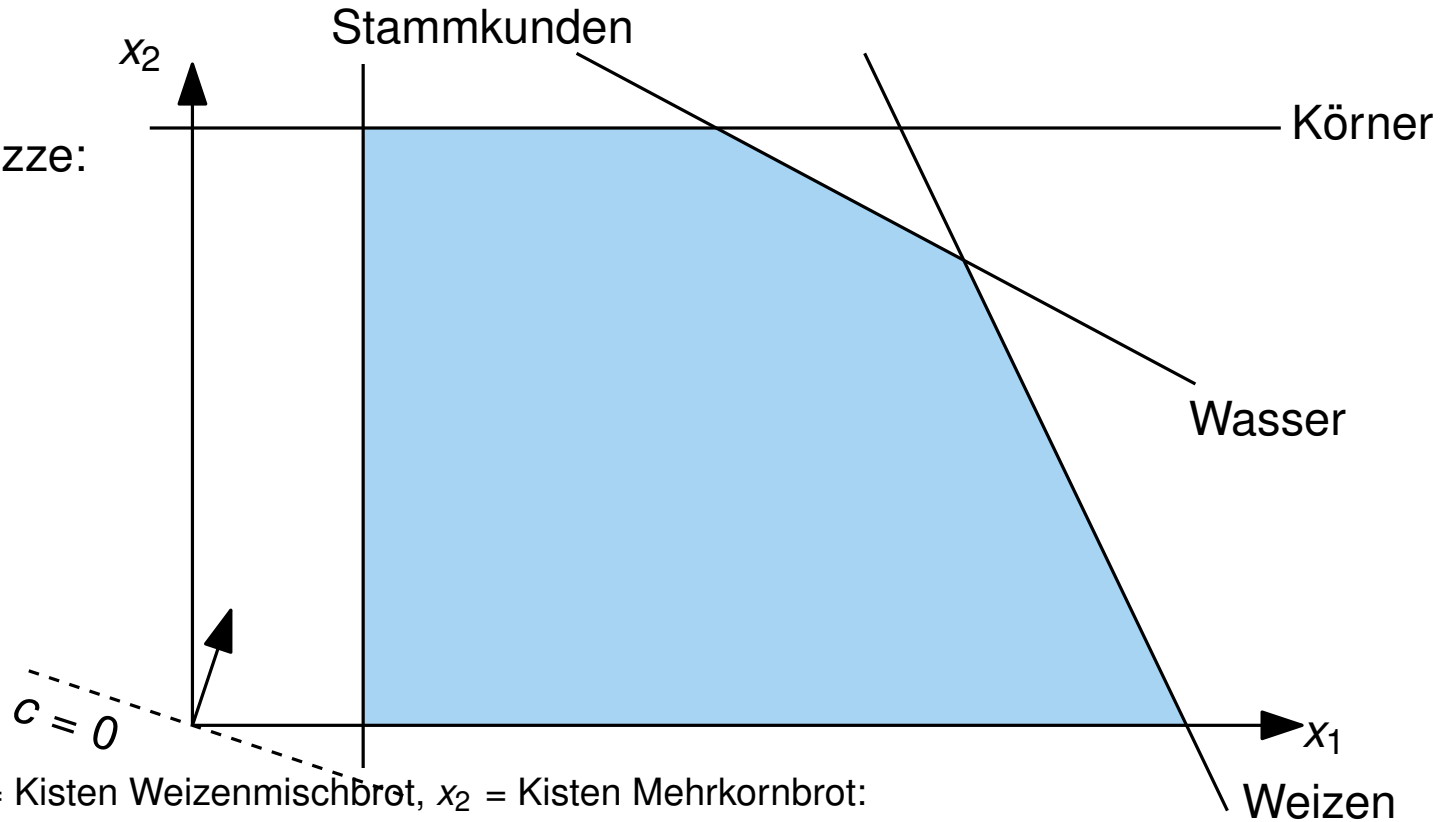


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Geometrische Interpretation

Skizze:

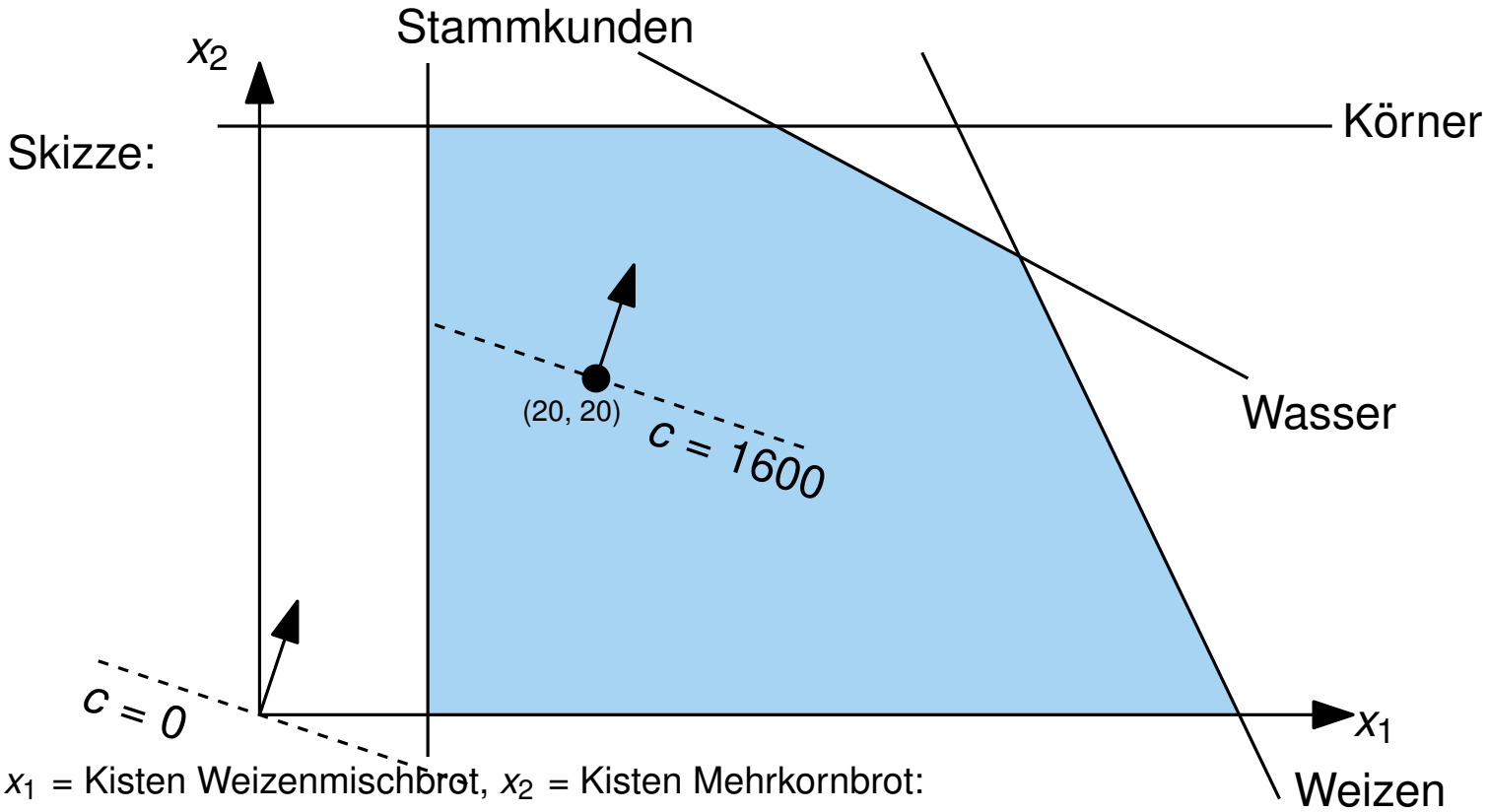


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Geometrische Interpretation

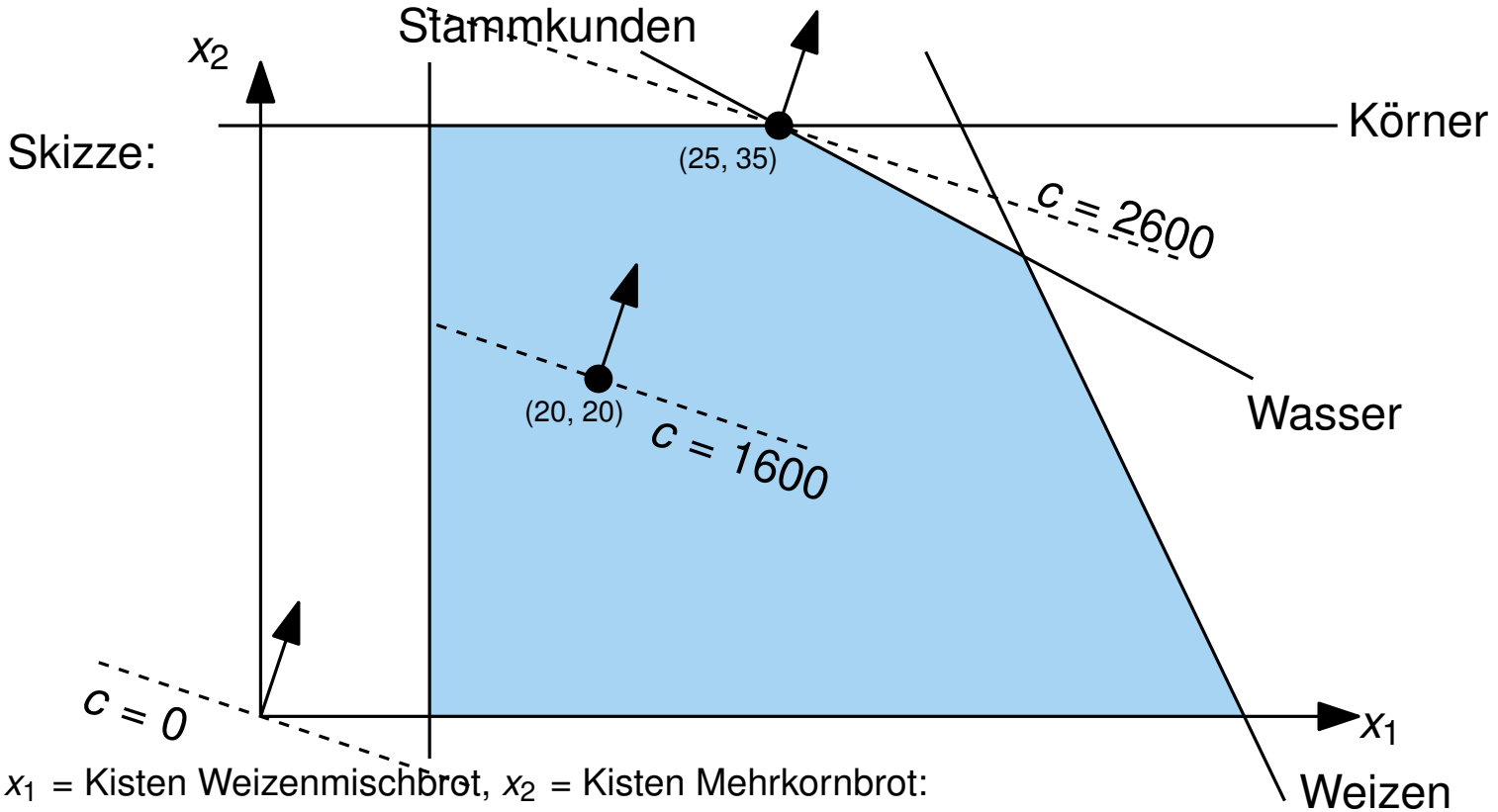
Skizze:



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	

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	$\leq$ 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	

Karlsruher Institut für Technologie

Dualität von Linearen Programmen

Obere Schranke

Betrachte folgendes Programm:

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 0

Schätze eine obere Schranke für f mithilfe der Nebenbedingungen ab:

Obere Schranke

Betrachte folgendes Programm:

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 0

Schätze eine obere Schranke für f mithilfe der Nebenbedingungen ab:

$2x_1 + 3x_2 \leq 4x_1 + 8x_2 \leq 12$ 12 ist obere Schranke!

Obere Schranke

Betrachte folgendes Programm:

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 0

Schätze eine obere Schranke für f mithilfe der Nebenbedingungen ab:

$2x_1 + 3x_2 \leq 4x_1 + 8x_2 \leq 12$ 12 ist obere Schranke!

besser:

$2x_1+3x_2 \leq \frac{4x_1 + 8x_2}{2} \leq \frac{12}{2} = 6$ 6 ist obere Schranke

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Obere Schranke

Betrachte folgendes Programm:

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 0

Schätze eine obere Schranke für f mithilfe der Nebenbedingungen ab:

$$2x_1+3x_2 = \frac{4x_1 + 8x_2 + 2x_1 + x_2}{3} \leq \frac{12 + 3}{3} = 5$$

5 ist obere Schranke!

Verallgemeinerung

Versuche Ungleichung der Form

$$d_1 x_1 + d_2 x_2 \leq h,$$

zu finden, so dass $d_1 \geq 2$, $d_2 \geq 3$ und h ist möglichst klein.

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	$\leq$ 12
		2	x_1	+		x_2	$\leq$ 3
		3	x_1	+	2	x_2	$\leq$ 4
			x_1				$\geq$ 0
						x_2	$\geq$ 0

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Verallgemeinerung

Versuche Ungleichung der Form

$$d_1 x_1 + d_2 x_2 \leq h,$$

zu finden, so dass $d_1 \geq 2$, $d_2 \geq 3$ und h ist möglichst klein.

Für $x_1, x_2 \geq 0$ gilt dann nämlich:

$$2x_1 + 3x_2 \leq d_1 x_1 + d_2 x_2 \leq h$$

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	$\leq$ 12
		2	x_1	+		x_2	$\leq$ 3
		3	x_1	+	2	x_2	$\leq$ 4
			x_1				$\geq$ 0
						x_2	$\geq$ 0

Verallgemeinerung

Versuche Ungleichung der Form

$$d_1 x_1 + d_2 x_2 \leq h,$$

zu finden, so dass $d_1 \geq 2$, $d_2 \geq 3$ und h ist möglichst klein.

Setze:

$h = 12y_1 + 3y_2 + 4y_3$

$d_1 = 4y_1 + 2y_2 + 3y_3$,

$d_2 = 8y_1 + 1y_2 + 2y_3$,

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	$\leq$ 12
		2	x_1	+		x_2	$\leq$ 3
		3	x_1	+	2	x_2	$\leq$ 4
			x_1				$\geq$ 0
						x_2	$\geq$ 0

Verallgemeinerung

Versuche Ungleichung der Form

$$d_1 x_1 + d_2 x_2 \leq h,$$

zu finden, so dass $d_1 \geq 2$, $d_2 \geq 3$ und h ist möglichst klein.

Setze:

$h = 12y_1 + 3y_2 + 4y_3$
$d_1 = 4y_1 + 2y_2 + 3y_3,$
$d_2 = 8y_1 + 1y_2 + 2y_3,$

Wie y_1, y_2 und y_3 bestimmen, so dass $y_1, y_2, y_3 \geq 0$ und $d_1 \geq 2$, $d_2 \geq 3$ und h möglichst klein?

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	$\leq$ 12
		2	x_1	+		x_2	$\leq$ 3
		3	x_1	+	2	x_2	$\leq$ 4
			x_1				$\geq$ 0
						x_2	$\geq$ 0

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Verallgemeinerung

Versuche Ungleichung der Form

$$d_1 x_1 + d_2 x_2 \leq h,$$

zu finden, so dass $d_1 \geq 2$, $d_2 \geq 3$ und h ist möglichst klein.

Setze:

$h = 12y_1 + 3y_2 + 4y_3$

$d_1 = 4y_1 + 2y_2 + 3y_3,$

$d_2 = 8y_1 + 1y_2 + 2y_3,$

ZF: $g(y_1, y_2, y_3) = 12y_1 + 3y_2 + 4y_3 = \min!$

NB:

$4y_1 + 2y_2 + 3y_3 \geq 2$

$8y_1 + 1y_2 + 2y_3 \geq 3$

$y_1, y_2, y_3 \geq 0$

Duales Programm *DP*

Zielfunktion **ZF:** $f(x_1, x_2) = 2x_1 + 3x_2 = \max!$

Nebenbedingungen **NB:**

$4x_1 + 8x_2 \leq 12$

$2x_1 + x_2 \leq 3$

$3x_1 + 2x_2 \leq 4$

$x_1 \geq 0$

$x_2 \geq 0$

Primales Programm *PP*

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Dualität

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 0

Primales Programm *PP*

ZF:	$g(y_1, y_2, y_3) =$	12	y_1	+	3	y_2	+	4	y_3	= min!
NB:		4	y_1	+	2	y_2	+	3	y_3	≥ 2
		8	y_1	+	1	y_2	+	2	y_3	≥ 3
									y_1, y_2, y_3	≥ 0

Duales Programm *DP*

Schwacher Dualitätssatz besagt: Für alle zulässigen Lösungen (x_1, x_2) von *PP* und alle zulässigen Lösungen (y_1, y_2, y_3) von *DP* gilt:

$$12y_1 + 3y_2 + 4y_3 \geq 2x_1 + 3x_2$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Dualität

Zielfunktion **ZF:**	$f(x_1, x_2) =$	2	x_1	+	3	x_2	= max!
Nebenbedingungen **NB:**		4	x_1	+	8	x_2	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 0

Primales Programm *PP*

ZF:	$g(y_1, y_2, y_3) =$	12	y_1	+	3	y_2	+	4	y_3	= min!
NB:		4	y_1	+	2	y_2	+	3	y_3	≥ 2
		8	y_1	+	1	y_2	+	2	y_3	≥ 3
									y_1, y_2, y_3	≥ 0

Duales Programm *DP*

Starker Dualitätssatz besagt:

PP lösbar $\Leftrightarrow$ *DP* lösbar

und wenn lösbar, dann $\max f(x_1, x_2) = \min g(y_1, y_2, y_3)$ unter Nebenbedingungen.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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

Problem 4: Lineare Programmierung

Marco Stanley Fogg ist ein Student, dem nach dem Tod seines Onkels lediglich dessen antiquarische Buchsammlung als finanzielle Rücklage bleibt. Um sein Studium möglichst lange durch den Verkauf der Bücher finanzieren zu können, versucht er, seine Ernährung auf ein Minimum zu beschränken. Nachdem er für eine Menge von m wichtigen Nährstoffen $1, \ldots, m$ jeweils den minimalen täglichen Bedarf b_i ($i = 1, \ldots, m$) für einen Mann seines Alters und Gewichts in Erfahrung gebracht hat, sucht er den lokalen Supermarkt auf und ermittelt für eine Menge von n Produkten $1, \ldots, n$ jeweils den Preis pro Einheit c_j ($j = 1, \ldots, n$) sowie den Anteil a_{ij} des Nährstoffes i ($i = 1, \ldots, m$) am Produkt j ($j = 1, \ldots, n$).

Problem 4: Lineare Programmierung

Marco Stanley Fogg ist ein Student, dem nach dem Tod seines Onkels lediglich dessen antiquarische Buchsammlung als finanzielle Rücklage bleibt. Um sein Studium möglichst lange durch den Verkauf der Bücher finanzieren zu können, versucht er, seine Ernährung auf ein Minimum zu beschränken. Nachdem er für eine Menge von m wichtigen Nährstoffen $1, \ldots, m$ jeweils den minimalen täglichen Bedarf b_i ($i = 1, \ldots, m$) für einen Mann seines Alters und Gewichts in Erfahrung gebracht hat, sucht er den lokalen Supermarkt auf und ermittelt für eine Menge von n Produkten $1, \ldots, n$ jeweils den Preis pro Einheit c_j ($j = 1, \ldots, n$) sowie den Anteil a_{ij} des Nährstoffes i ($i = 1, \ldots, m$) am Produkt j ($j = 1, \ldots, n$).

Gesucht: lineares Programm L.

Problem 4: Lineare Programmierung

Marco Stanley Fogg ist ein Student, dem nach dem Tod seines Onkels lediglich dessen antiquarische Buchsammlung als finanzielle Rücklage bleibt. Um sein Studium möglichst lange durch den Verkauf der Bücher finanzieren zu können, versucht er, seine Ernährung auf ein Minimum zu beschränken. Nachdem er für eine Menge von m wichtigen Nährstoffen $1, \ldots, m$ jeweils den minimalen täglichen Bedarf b_i ($i = 1, \ldots, m$) für einen Mann seines Alters und Gewichts in Erfahrung gebracht hat, sucht er den lokalen Supermarkt auf und ermittelt für eine Menge von n Produkten $1, \ldots, n$ jeweils den Preis pro Einheit c_j ($j = 1, \ldots, n$) sowie den Anteil a_{ij} des Nährstoffes i ($i = 1, \ldots, m$) am Produkt j ($j = 1, \ldots, n$).

Gesucht: lineares Programm L.

Zielfunktion: $\text{minimiere} \sum_{j=1}^{n} c_j x_j$ (Kosten den Einkaufs)

Nebenbedingungen: $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$ für $i = 1, \ldots, m$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 4: Lineare Programmierung

Marco Stanley Fogg ist ein Student, dem nach dem Tod seines Onkels lediglich dessen antiquarische Buchsammlung als finanzielle Rücklage bleibt. Um sein Studium möglichst lange durch den Verkauf der Bücher finanzieren zu können, versucht er, seine Ernährung auf ein Minimum zu beschränken. Nachdem er für eine Menge von m wichtigen Nährstoffen $1, \ldots, m$ jeweils den minimalen täglichen Bedarf b_i ($i = 1, \ldots, m$) für einen Mann seines Alters und Gewichts in Erfahrung gebracht hat, sucht er den lokalen Supermarkt auf und ermittelt für eine Menge von n Produkten $1, \ldots, n$ jeweils den Preis pro Einheit c_j ($j = 1, \ldots, n$) sowie den Anteil a_{ij} des Nährstoffes i ($i = 1, \ldots, m$) am Produkt j ($j = 1, \ldots, n$).

Gesucht: Duales Programm zu L und sinnvolle Interpretation.

Problem 4: Lineare Programmierung

Marco Stanley Fogg ist ein Student, dem nach dem Tod seines Onkels lediglich dessen antiquarische Buchsammlung als finanzielle Rücklage bleibt. Um sein Studium möglichst lange durch den Verkauf der Bücher finanzieren zu können, versucht er, seine Ernährung auf ein Minimum zu beschränken. Nachdem er für eine Menge von m wichtigen Nährstoffen $1, \ldots, m$ jeweils den minimalen täglichen Bedarf b_i ($i = 1, \ldots, m$) für einen Mann seines Alters und Gewichts in Erfahrung gebracht hat, sucht er den lokalen Supermarkt auf und ermittelt für eine Menge von n Produkten $1, \ldots, n$ jeweils den Preis pro Einheit c_j ($j = 1, \ldots, n$) sowie den Anteil a_{ij} des Nährstoffes i ($i = 1, \ldots, m$) am Produkt j ($j = 1, \ldots, n$).

Gesucht: Duales Programm zu L und sinnvolle Interpretation.

Zielfunktion: $\text{maximiere} \ \sum_{i=1}^{m} y_i b_i$

Nebenbedingungen: $\sum_{i=1}^{m} y_i a_{ij} \leq c_j$ für $j = 1, \ldots, n$

Problem 4: Lineare Programmierung

Marco Stanley Fogg ist ein Student, dem nach dem Tod seines Onkels lediglich dessen antiquarische Buchsammlung als finanzielle Rücklage bleibt. Um sein Studium möglichst lange durch den Verkauf der Bücher finanzieren zu können, versucht er, seine Ernährung auf ein Minimum zu beschränken. Nachdem er für eine Menge von m wichtigen Nährstoffen $1, \ldots, m$ jeweils den minimalen täglichen Bedarf b_i ($i = 1, \ldots, m$) für einen Mann seines Alters und Gewichts in Erfahrung gebracht hat, sucht er den lokalen Supermarkt auf und ermittelt für eine Menge von n Produkten $1, \ldots, n$ jeweils den Preis pro Einheit c_j ($j = 1, \ldots, n$) sowie den Anteil a_{ij} des Nährstoffes i ($i = 1, \ldots, m$) am Produkt j ($j = 1, \ldots, n$).

- Der Vektor y enthält zu jedem Nährstoff i einen Eintrag y_i, der dem festzulegenden Preis einer Einheit des Nährstoffes i entspricht.
- Das duale Programm maximiert dann die Kosten für einen Einkauf.
- Die Kosten der für eine Einheit eines Produktes benötigten Nährstoffe, darf einen maximalen Preis c_i nicht überschreiten.
- Der Besitzer des lokalen Supermarktes möchte den Preis eines solchen Einkaufs maximieren.

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.

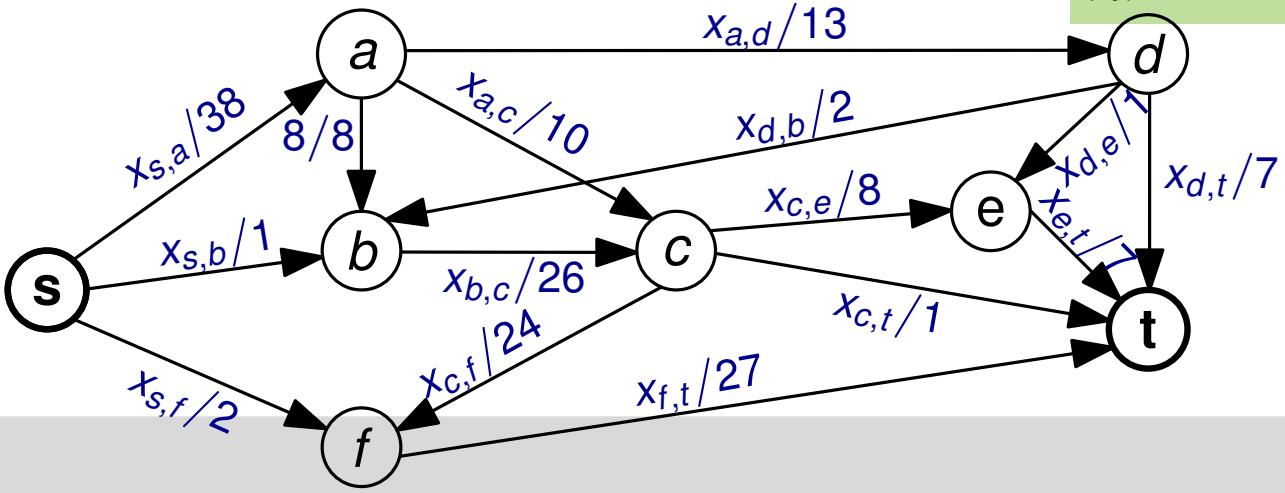
Maximiere den Wert des Flusses:

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

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 $i \in V \setminus \{s, t\}$

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



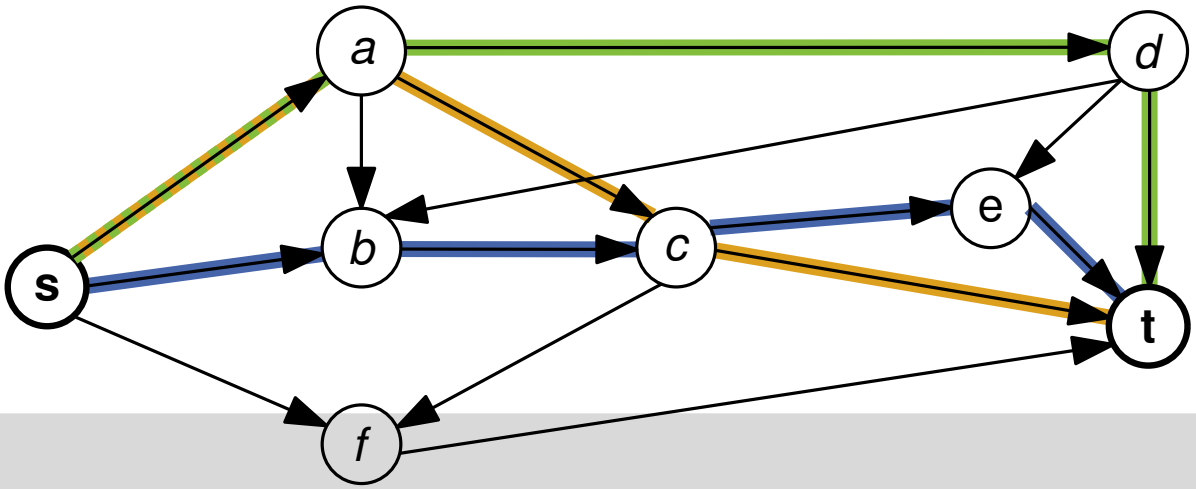
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Alternative Formulierung

Betrachte das Netzwerk (D, s, t, c) und sei $\mathcal{P}$ die Menge aller einfachen Pfade von s nach t.

Führe für jeden einfachen Pfad P von s nach t eine **Variable** x_P ein.

Idee: x_P gibt den Fluss an, der über den Pfad P fließt.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Alternative Formulierung

Betrachte das Netzwerk (D, s, t, c) und sei $\mathcal{P}$ die Menge aller einfachen Pfade von s nach t.

Führe für jeden einfachen Pfad P von s nach t eine **Variable** x_P ein.

Idee: x_P gibt den Fluss an, der über den Pfad P fließt.

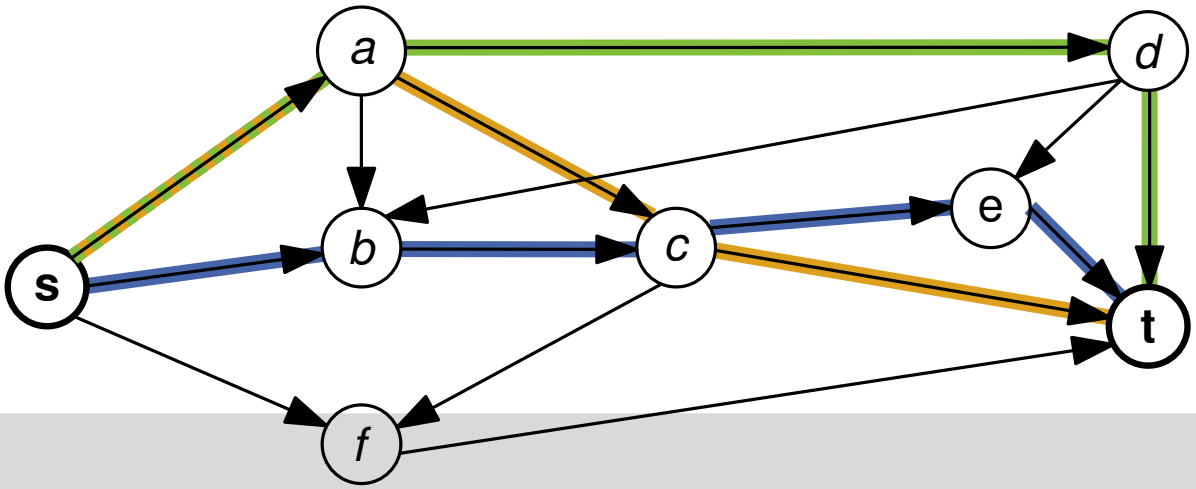
Maximiere:

$$f(\overline{x}) = \sum_{P \in P} x_P$$

Unter den Bedingungen:

$$\sum_{P \in \mathcal{P}:(u,v) \in P} x_P \leq c(u,v) \quad \text{für alle } (u,v) \in E$$

$$x_P \geq 0 \quad \text{für alle } P \in \mathcal{P}$$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Alternative Formulierung

Betrachte das Netzwerk (D, s, t, c) und sei $\mathcal{P}$ die Menge aller einfachen Pfade von s nach t.

Führe für jeden einfachen Pfad P von s nach t eine **Variable** x_P ein.

Idee: x_P gibt den Fluss an, der über den Pfad P fließt.

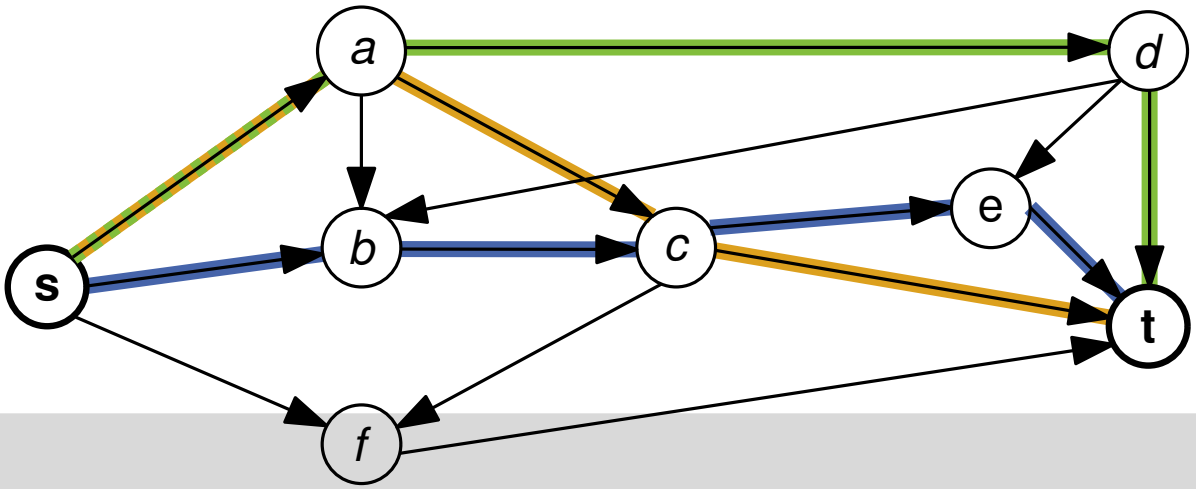
Maximiere:

$$f(\overline{x}) = \sum_{P \in P} x_P$$

Unter den Bedingungen:

$$\sum_{P \in \mathcal{P}:(u,v) \in P} x_P \leq c(u,v) \quad \text{für alle } (u,v) \in E$$

$$x_P \geq 0 \quad \text{für alle } P \in \mathcal{P}$$



Problem dieser Formulierung:
$\mathcal{P}$ kann exponentiell viele Pfade enthalten.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Alternative Formulierung

Betrachte das Netzwerk (D, s, t, c) und sei $\mathcal{P}$ die Menge aller einfachen Pfade von s nach t.

Führe für jeden einfachen Pfad P von s nach t eine **Variable** x_P ein.

Idee: x_P gibt den Fluss an, der über den Pfad P fließt.

Maximiere:

$$f(\overline{x}) = \sum_{P \in P} x_P$$

Unter den Bedingungen:

$$\sum_{P \in \mathcal{P}:(u,v) \in P} x_P \leq c(u,v) \quad \text{für alle } (u,v) \in E$$

$$x_P \geq 0 \quad \text{für alle } P \in \mathcal{P}$$

Duales Programm: Führe für jede Kante $(u, v) \in E$ **Variable** $y_{u,v}$ ein.

Minimiere:

$$\sum_{(u,v) \in E} y_{u,v} \cdot c(u,v)$$

Unter den Bedingungen:

$$\sum_{(u,v) \in P} y_{u,v} \geq 1 \quad \text{für alle } P \in \mathcal{P}$$

$$y_{u,v} \geq 0 \quad \text{für alle } (u,v) \in E$$

Alternative Formulierung

Betrachte das Netzwerk (D, s, t, c) und sei $\mathcal{P}$ die Menge aller einfachen Pfade von s nach t.

Führe für jeden einfachen Pfad P von s nach t eine **Variable** x_P ein.

Idee: x_P gibt den Fluss an, der über den Pfad P fließt.

Maximiere:

$$f(\overline{x}) = \sum_{P \in P} x_P$$

Unter den Bedingungen:

$$\sum_{P \in \mathcal{P}:(u,v) \in P} x_P \leq c(u,v) \quad \text{für alle } (u,v) \in E$$

$$x_P \geq 0 \quad \text{für alle } P \in \mathcal{P}$$

Duales Programm: Führe für jede Kante $(u, v) \in E$ **Variable** $y_{u,v}$ ein.

Minimiere:

$$\sum_{(u,v) \in E} y_{u,v} \cdot c(u,v)$$

Unter den Bedingungen:

$$\sum_{(u,v) \in P} y_{u,v} \geq 1 \quad \text{für alle } P \in \mathcal{P}$$

$$y_{u,v} \geq 0 \quad \text{für alle } (u,v) \in E$$

Für $y_{u,v} \in \{0, 1\}$ beschreiben Variablen einen s, t-Schnitt.