Karlsruher Institut für Technologie

Algorithmen II
Übung am 25.10.2012

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

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.
  - erreichbar unter: `https://ilias.studium.kit.edu`

→ Fakultät für Informatik
→ WS 2012/2013
→ [24079] Algorithmen II
→ Anmeldung mit Rechenzentrum-account erforderlich

Schülerstudenten bitte Mail an uns.

Klausur

Karlsruher Institut für Technologie

- Umfasst zwei Stunden.
- Orientierung: Vorlesung Algorithmen II des Vorjahres + Vorlesung Algorithmentechnik behandelte ähnlichen Stoff.
- Hauptklausur: voraussichtlich am 01.03.2013
- Nachklausur: voraussichtlich am 02.10.2013

Genaue Klausurtermine werden rechtzeitig bekannt gegeben.

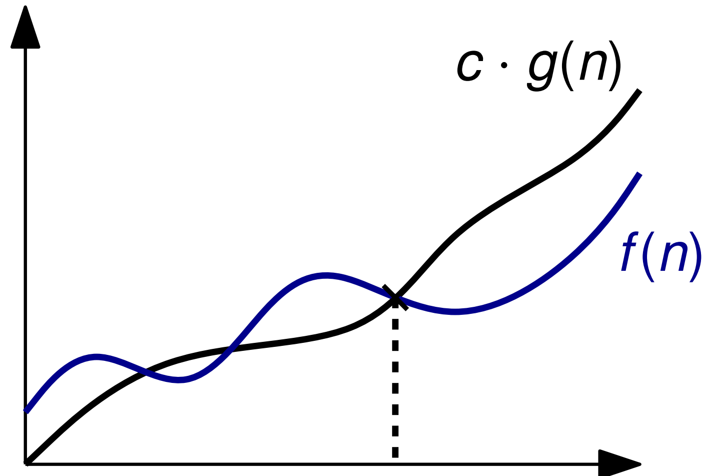
Karlsruher Institut für Technologie

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

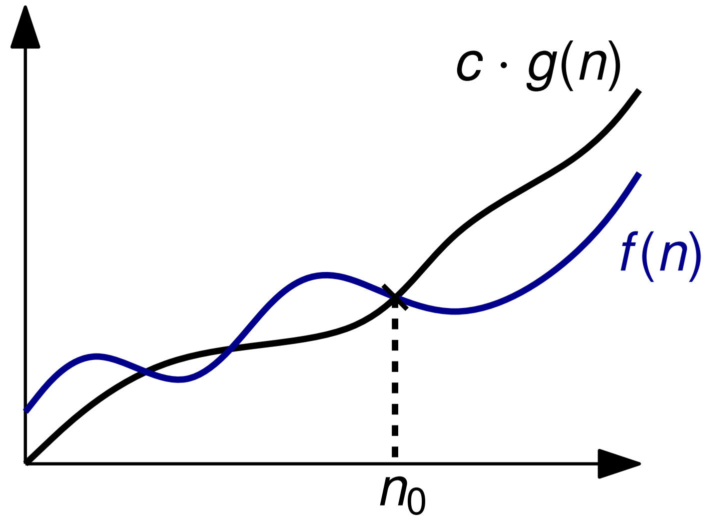
$$0 \leq f(n) \leq c \cdot n \text{ für alle } n \leq n_0$$

abgeschätzt werden kann.

Wiederholung – Asymptotische Laufzeit

$\mathcal{O}$-Notation:

19	6	7	10	4	



worst-case-Analyse

$f(n) = \mathcal{O}(g(n)) :\Leftrightarrow$ 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

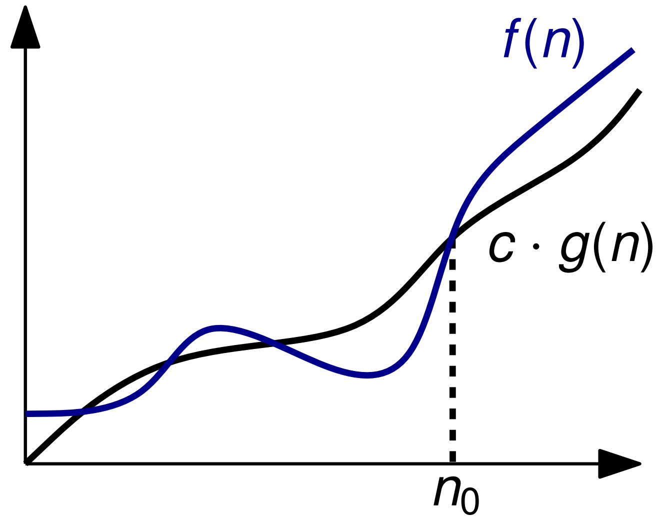
$$0 \leq f(n) \leq c \cdot n \text{ für alle } n \leq n_0$$

abgeschätzt werden kann.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Wiederholung – Asymptotische Laufzeit

Ω-Notation:



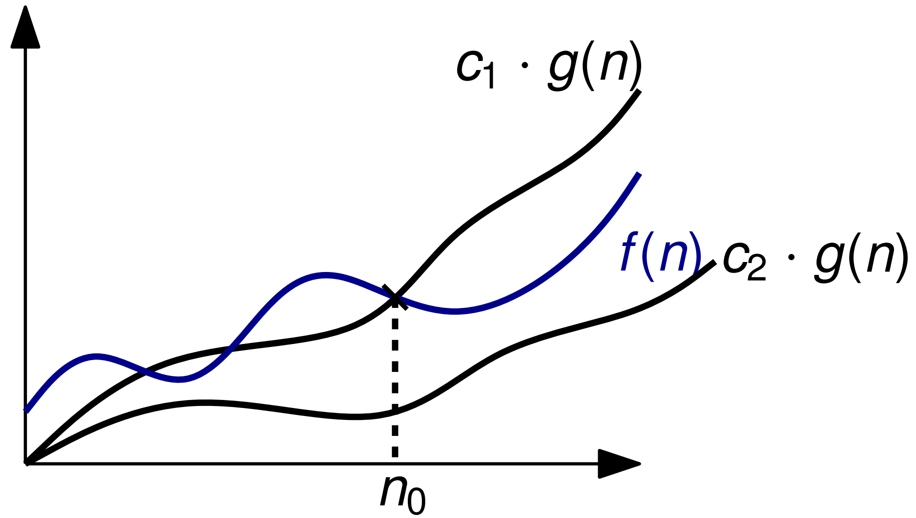
$f(n) = \Omega(g(n)) :\Leftrightarrow$ es existieren positive Konstanten c und n_0, so dass

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

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.

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: $o_1, \ldots, o_m$

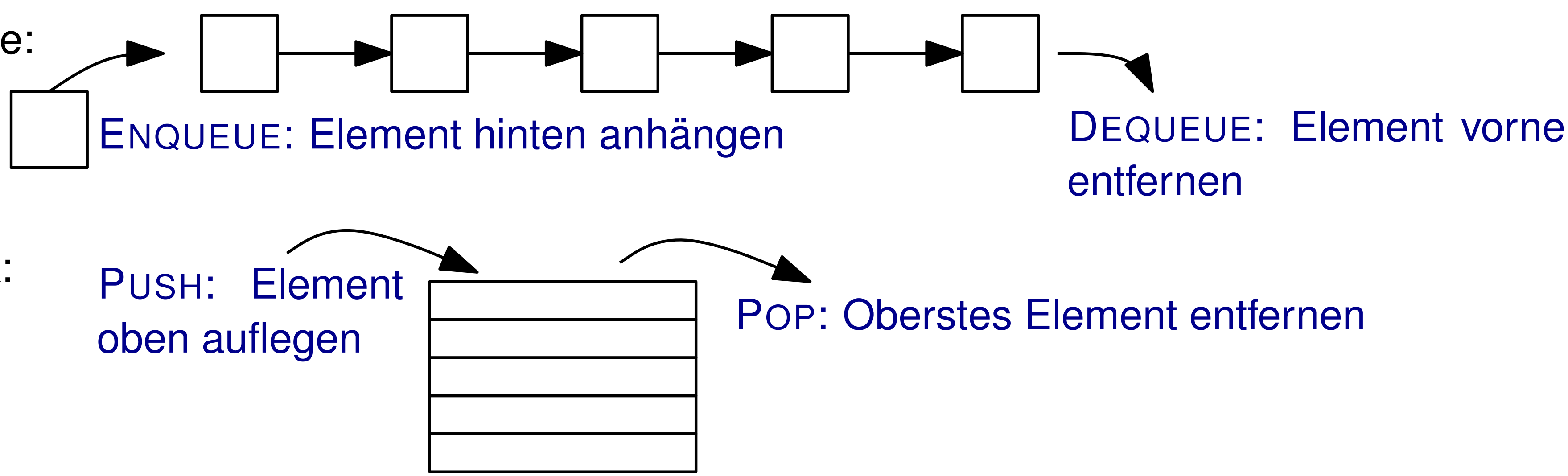
Betrachte nicht Operation einzeln, sondern alle ausgeführten Operationen.

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.

Queue:

Stack:



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

E D

ENQUEUE(x):

1. PUSH(E,x)

DEQUEUE:

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

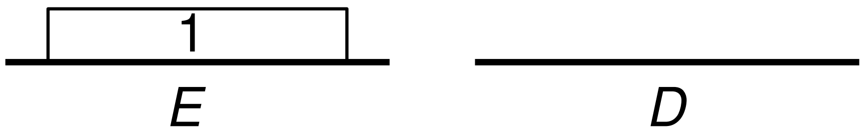
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Lösungsidee

Karlsruher Institut für Technologie

Führe zwei Stacks E und D ein:

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



ENQUEUE(x):

1. PUSH(E,x)

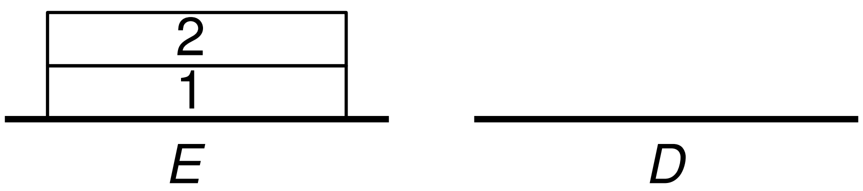
DEQUEUE:

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):

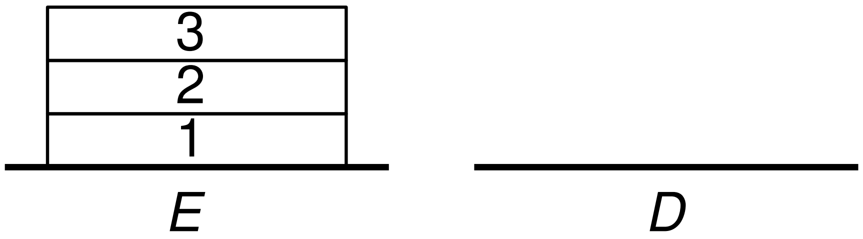
1. PUSH(E,x)

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):

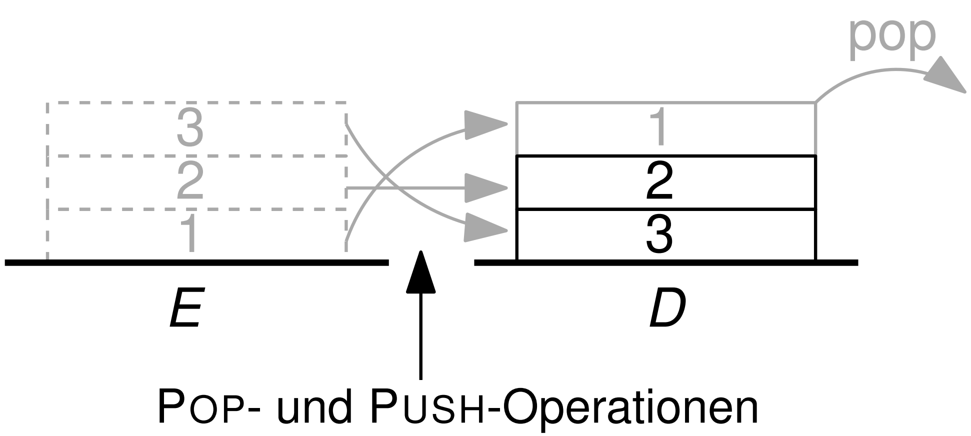
1. PUSH(E,x)

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):

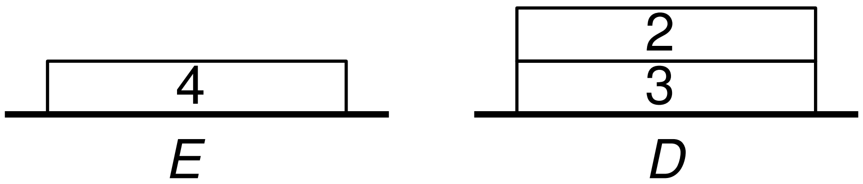
1. Push(E,x)

Dequeue:

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

Lösungsidee

Führe zwei Stacks *E* und *D* ein:



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

ENQUEUE(x):

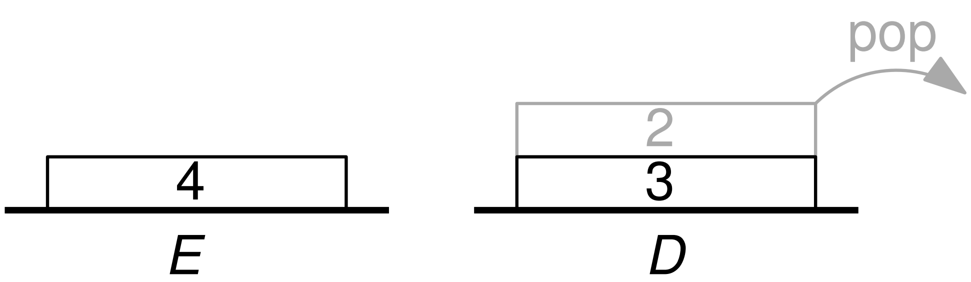
1. PUSH(E,x)

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)

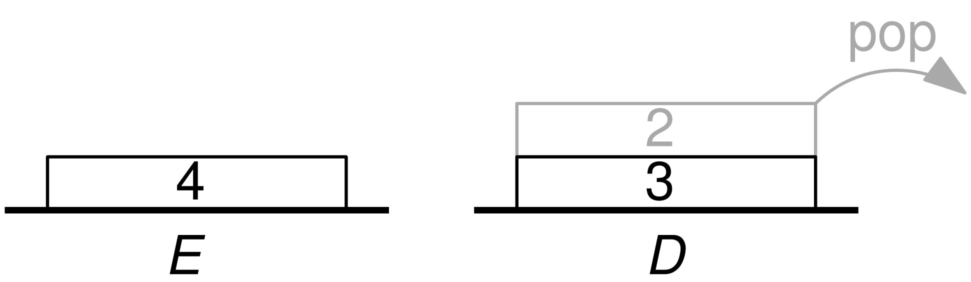
DEQUEUE:

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:



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

ENQUEUE(x):

1. PUSH(E,x)

DEQUEUE:

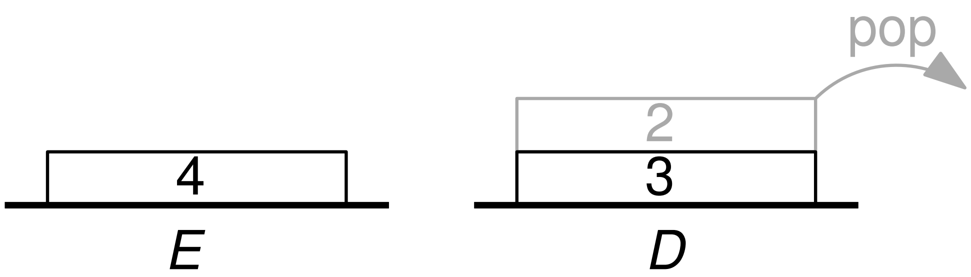
1. **Wenn** $D = \emptyset$
   (a) **Solange** $E \neq \emptyset$ **tue** PUSH(D, POP(E))
2. POP(D)

Analyse:

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

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.

Flussnetzwerke

Wiederholung

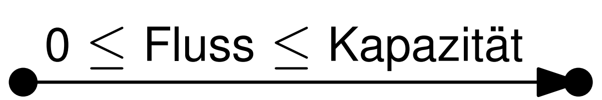
Definition:

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

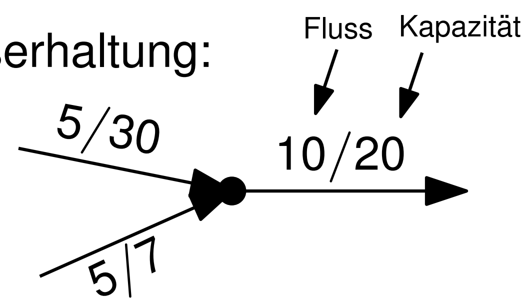
Das Tupel (D, s, t, c) heißt *Netzwerk*. Eine Abbildung $f \colon E \to \mathbb{R}_0^+$ heißt *Fluss*, wenn folgende Bedingungen gelten:

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:



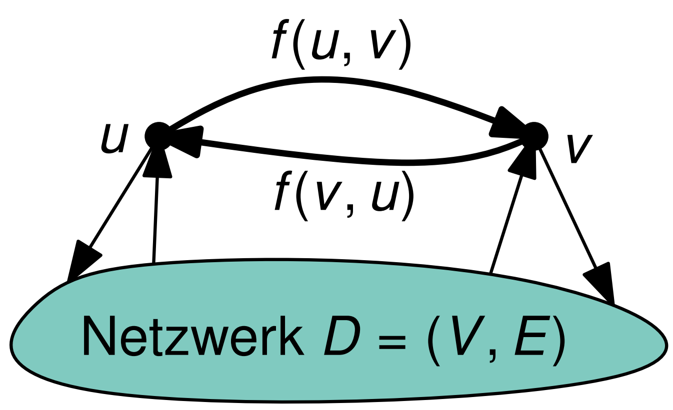
Zwischenknoten können Fluss weder konsumieren noch produzieren.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 2: Maximale Flüsse

Problem 2: Gegeben sei ein Netzwerk $D = (V, E)$, in dem es zu einigen Kanten $(u, v) \in E$ auch Kanten $(v, u) \in E$ gibt. Weiterhin sei ein maximaler Fluss $f : E \to \mathbb{R}_0^+$ gegeben.

Zeigen Sie, dass man dieses Flussproblem auf ein Flussproblem auf dem Netzwerk $D' = (V, E')$ überführen kann, wobei gilt: E' ist maximale Teilmenge von E mit $(u, v) \in E' \Rightarrow (v, u) \notin E'$, so dass der Wert des Maximalflusses nicht verändert wird.



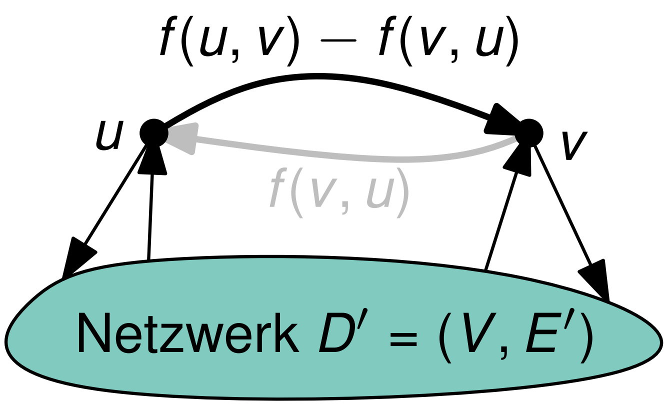
o.B.d.A.: $f(u, v) \geq f(v, u)$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 2: Maximale Flüsse

Problem 2: Gegeben sei ein Netzwerk $D = (V, E)$, in dem es zu einigen Kanten $(u, v) \in E$ auch Kanten $(v, u) \in E$ gibt. Weiterhin sei ein maximaler Fluss $f : E \to \mathbb{R}_0^+$ gegeben.

Zeigen Sie, dass man dieses Flussproblem auf ein Flussproblem auf dem Netzwerk $D' = (V, E')$ überführen kann, wobei gilt: E' ist maximale Teilmenge von E mit $(u, v) \in E' \Rightarrow (v, u) \notin E'$, so dass der Wert des Maximalflusses nicht verändert wird.



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

Idee: Konstruktiver Beweis

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

Definiere Fluss f':

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

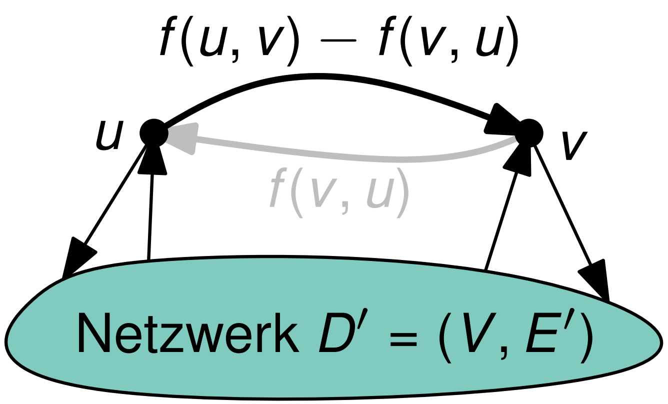
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Problem 2: Maximale Flüsse

Karlsruher Institut für Technologie

Problem 2: Gegeben sei ein Netzwerk $D = (V, E)$, in dem es zu einigen Kanten $(u, v) \in E$ auch Kanten $(v, u) \in E$ gibt. Weiterhin sei ein maximaler Fluss $f : E \to \mathbb{R}_0^+$ gegeben.

Zeigen Sie, dass man dieses Flussproblem auf ein Flussproblem auf dem Netzwerk $D' = (V, E')$ überführen kann, wobei gilt: E' ist maximale Teilmenge von E mit $(u, v) \in E' \Rightarrow (v, u) \notin E'$, so dass der Wert des Maximalflusses nicht verändert wird.



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

Idee: Konstruktiver Beweis

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

Definiere Fluss f':

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

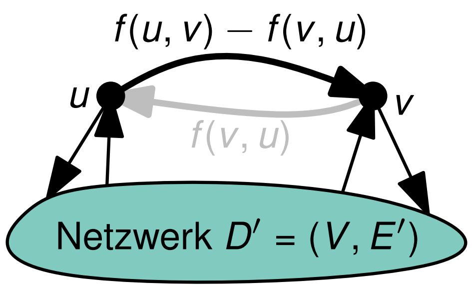
- Ist f' wirklich ein Fluss?
- Ist f' Maximalfluss in D'
- Gilt $w(f) = w(f')$?

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Kapazitätsbedingung

Problem 2: Gegeben sei ein Netzwerk $D = (V, E)$, in dem es zu einigen Kanten $(u, v) \in E$ auch Kanten $(v, u) \in E$ gibt. Weiterhin sei ein maximaler Fluss $f : E \to \mathbb{R}_0^+$ gegeben.

Zeigen Sie, dass man dieses Flussproblem auf ein Flussproblem auf dem Netzwerk $D' = (V, E')$ überführen kann, wobei gilt: E' ist maximale Teilmenge von E mit $(u, v) \in E' \Rightarrow (v, u) \notin E'$, so dass der Wert des Maximalflusses nicht verändert wird.



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

1. Zeige: 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

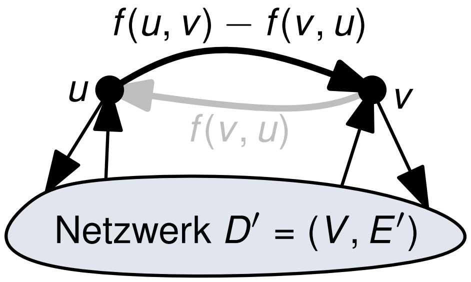
$$0 \leq f'(u, v) = f(u, v) - f(v, u) \leq c(u, v)$$

Flusserhaltung

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

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

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



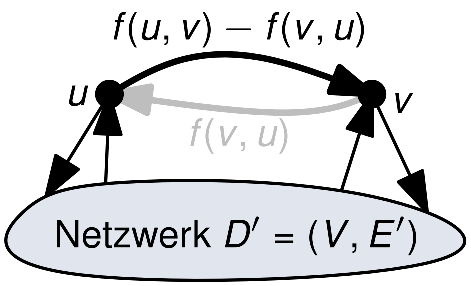
o.B.d.A.: $f(u, v) \geq f(v, u)$

Flusserhaltung

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

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

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



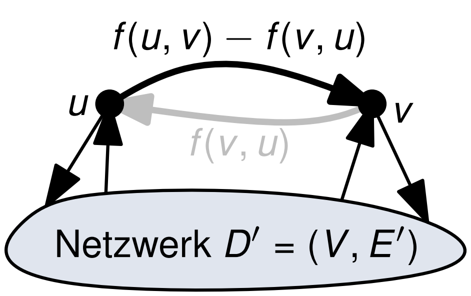
o.B.d.A.: $f(u, v) \geq f(v, u)$

Flusserhaltung

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

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

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



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

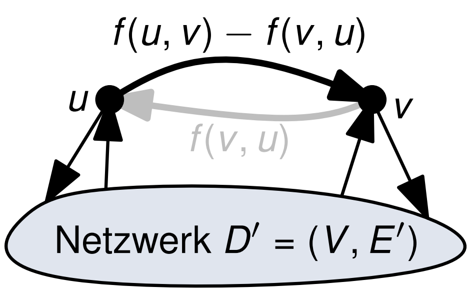
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Flusserhaltung

2. Zeige 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.: $f(u,v) \geq f(v,u)$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

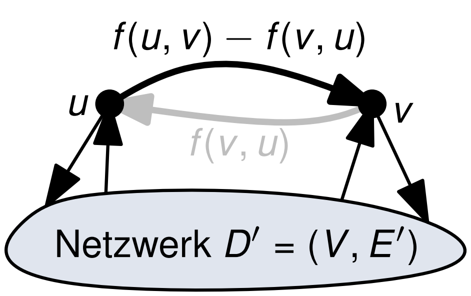
Flusserhaltung

Karlsruher Institut für Technologie

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

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

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



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

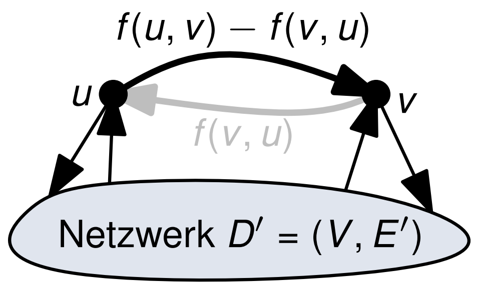
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Flusserhaltung

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

Für alle $i \in V \backslash \{s, t, u, v\}$ 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

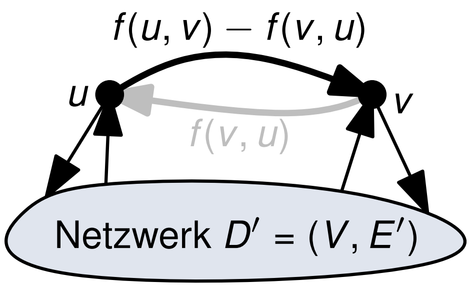
2. Zeige für alle $i \in V \backslash \{s, t\}$ gilt $\sum_{(i,j) \in E'} f'(i, j) - \sum_{(j,i) \in E'} f'(j, i) = 0$

Für alle $i \in V \backslash \{s, t, u, v\}$ 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.: $f(u, v) \geq f(v, u)$

Flusserhaltung

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

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

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

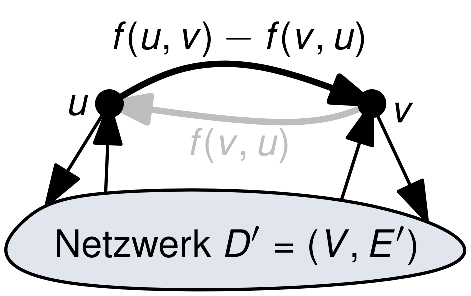
- Falls $u \neq s$ und $u \neq t$, dann $w(u) = 0 \rightarrow$ Flusserhaltung gilt für u.
- Falls $u = s$ oder $u = t$, dann $\mathbf{w}(\mathbf{f}') = w'(u) = w(u) = \mathbf{w}(\mathbf{f})$

Analoges Vorgehen für Knoten v.

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

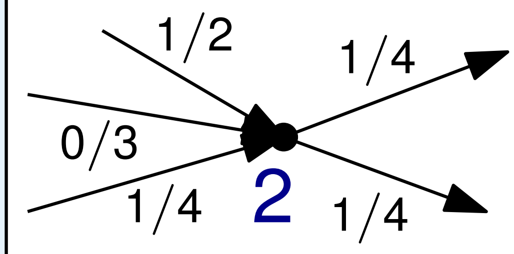
Das maximale Flussproblem aus der Vorlesung kann folgendermaßen erweitert werden: Neben den Kantenkapazitäten c seien noch Knotenkapazitäten $\gamma : V \to \mathbb{R}_0^+$ gegeben. In einem solchen Netzwerk (D, s, t, c, γ) heißt eine Abbildung f *Fluss*, wenn sie neben den bekannten Eigenschaften (Flusserhaltungsbedingung und (Kanten)-Kapazitätsbedingung) auch die folgende erfüllt: Für alle $v \in V$ ist die *Knotenkapazitätsbedingung*

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

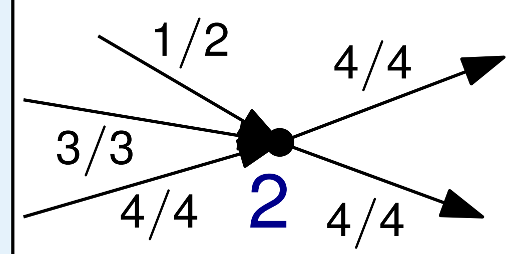
$$\sum_{(u,v)\in E} f(u,v) \leq \gamma(v) \qquad \text{wenn } v = t$$

erfüllt.

Zeigen Sie, dass die Bestimmung eines maximalen Flusses in einem Netzwerk mit Kanten- **und** Knotenkapazitäten auf ein maximales Flussproblem in einem normalen (d. h. ohne Knotenkapazitäten) Netzwerk mit vergleichbarer Größe zurückgeführt werden kann.

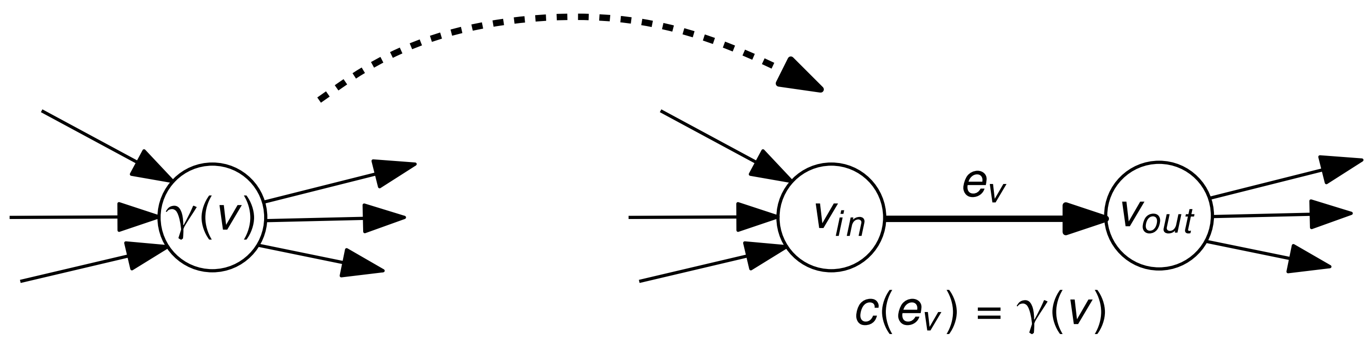


gültig



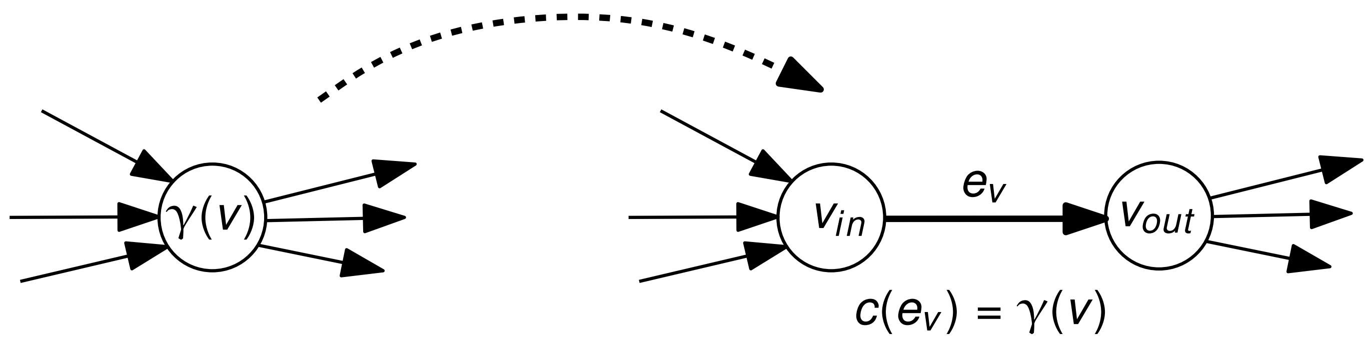
nicht gültig

Lösung



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

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.

Karlsruher Institut für Technologie

Flussalgorithmus von Goldberg und Tarjan

Algorithmus von Goldberg und Tarjan

Für alle $(v, w) \in V \times V$ mit $(v, w) \notin E$ setze:

- $c(v, w) \leftarrow 0$

Für alle $(v, w) \in V \times V$ setze:

- $f(v, w) \leftarrow 0$
- $r_f(v, w) \leftarrow c(v, w)$

Setze $dist(s) \leftarrow |V|$

Für alle $v \in V \setminus \{s\}$ setze:

- $f(s, v) \leftarrow c(s, v)$, $r_f(s, v) \leftarrow 0$
- $f(v, s) \leftarrow -c(s, v)$, $r_f(v, s) \leftarrow c(v, s) - f(v, s)$
- $dist(v) \leftarrow 0$
- $e(v) \leftarrow c(s, v)$

Eingabe: Netzwerk (D, s, t, c) mit $D = (V, E)$ und $c : E \rightarrow \mathbb{R}_0^+$
Ausgabe: Maximaler Fluss f.

$e(v)$: Flussüberschuss an Knoten v
$dist(v)$: Markierung des Knoten v

Solange es aktiven Knoten gibt:

- Wähle beliebigen aktiven Knoten v.
- Führe für v eine zulässige Operation PUSH oder RELABEL aus.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

PUSH und RELABEL

Operation PUSH(D, f, v, w)
Vorbedingung: v ist aktiv, $r_f(v,w) > 0$ und $dist(v) = dist(w) + 1$.
Effekt: Flussüberschuss wird von v nach w über Kante (v, w) geschoben.

- $\Delta \leftarrow \min\{e(v), r_f(v,w)\}$
- $f(v,w) \leftarrow f(v,w) + \Delta$, $f(w,v) \leftarrow f(w,v) - \Delta$
- $r_f(v,w) \leftarrow r_f(v,w) - \Delta$, $r_f(w,v) \leftarrow r_f(w,v) + \Delta$
- $e(v) \leftarrow e(v) - \Delta$, $e(w) \leftarrow e(w) + \Delta$

Operation RELABEL(D, f, v, $dist$)
Vorbedingung: v ist aktiv und für alle w mit $r_f(v,w) > 0$ gilt $dist(v) \leq dist(w)$
Effekt: $dist(v)$ wird erhöht.

$$dist(v) = \begin{cases} \infty\,, & \text{falls } \{w \mid r_f(v,w) > 0\} = \emptyset\,, \\ \min\{dist(w) + 1 \mid r_f(v,w) > 0\} & \text{sonst.} \end{cases}$$

Ein Knoten $v \in V \setminus \{t\}$ heißt *aktiv* im Laufe des Algorithmus, wenn $e(v) > 0$ und $dist(v) < \infty$.

Lineare Programme

Lineare Programme

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:

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

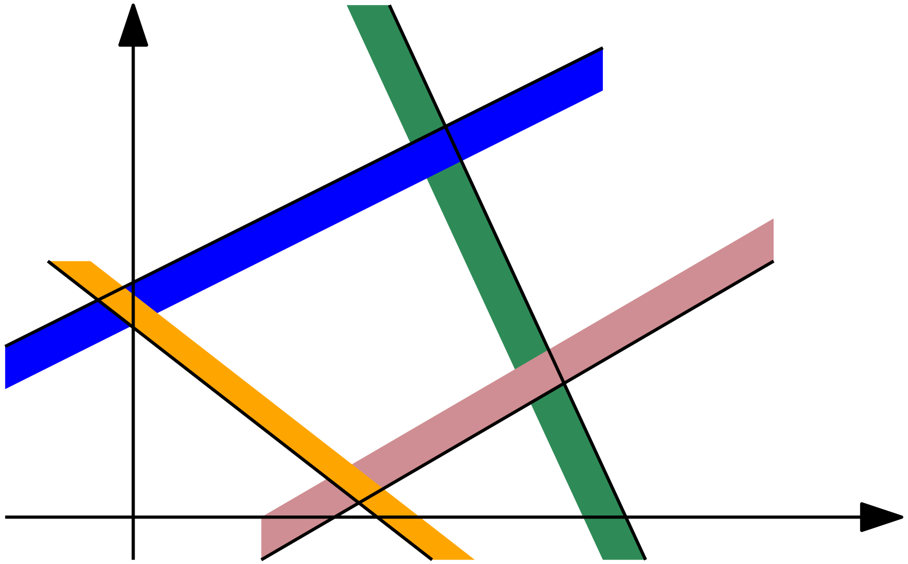
Ziel: bestimme $x_1, \dots, x_n$ so, dass $f(\overline{x})$ maximal/minimal ist.

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:**

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

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

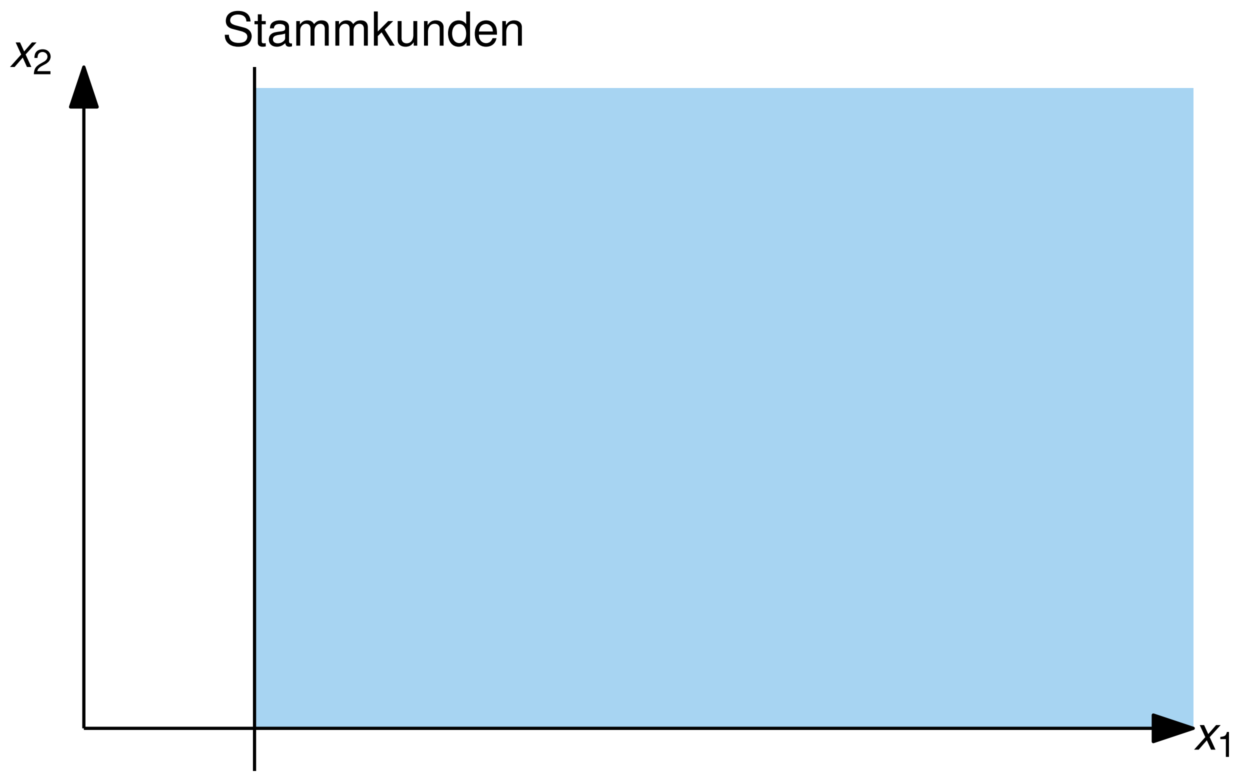
Skizze:

x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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

Geometrische Interpretation

Skizze:

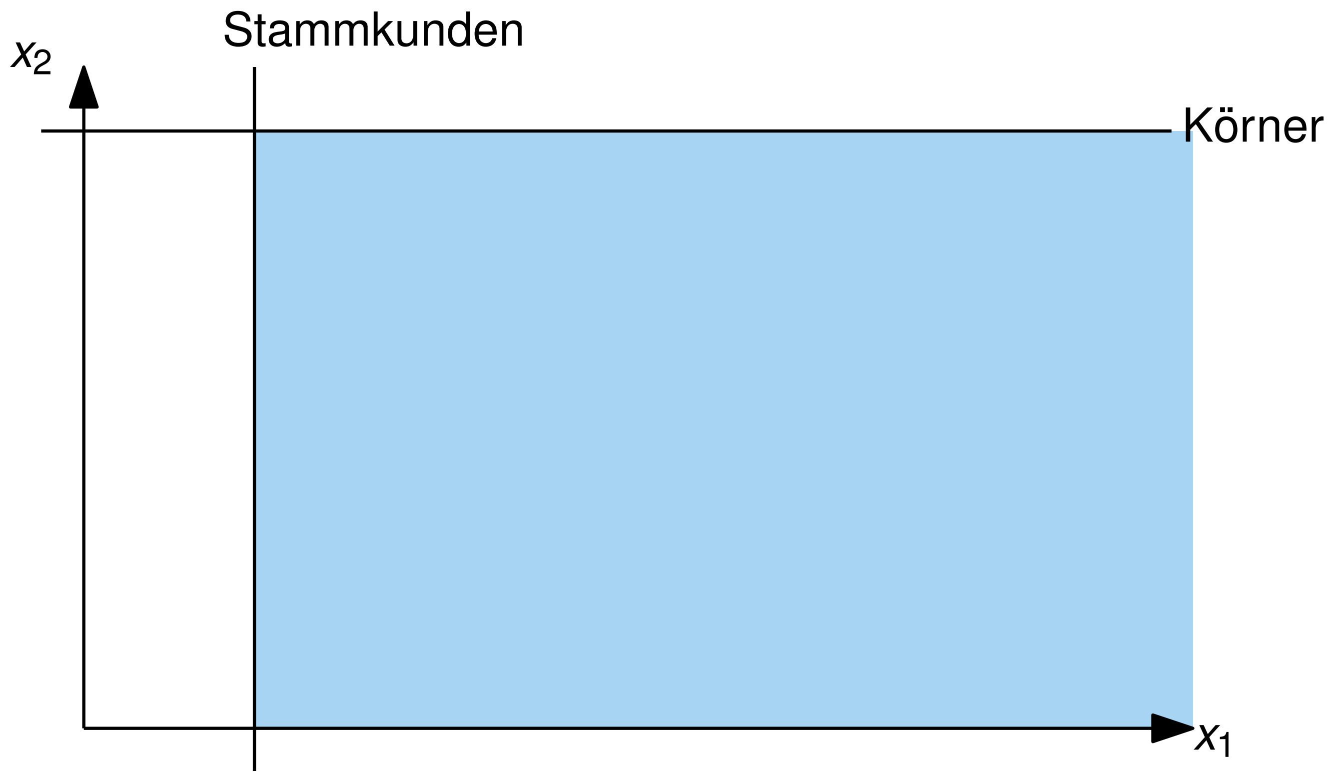


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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

Geometrische Interpretation

Skizze:

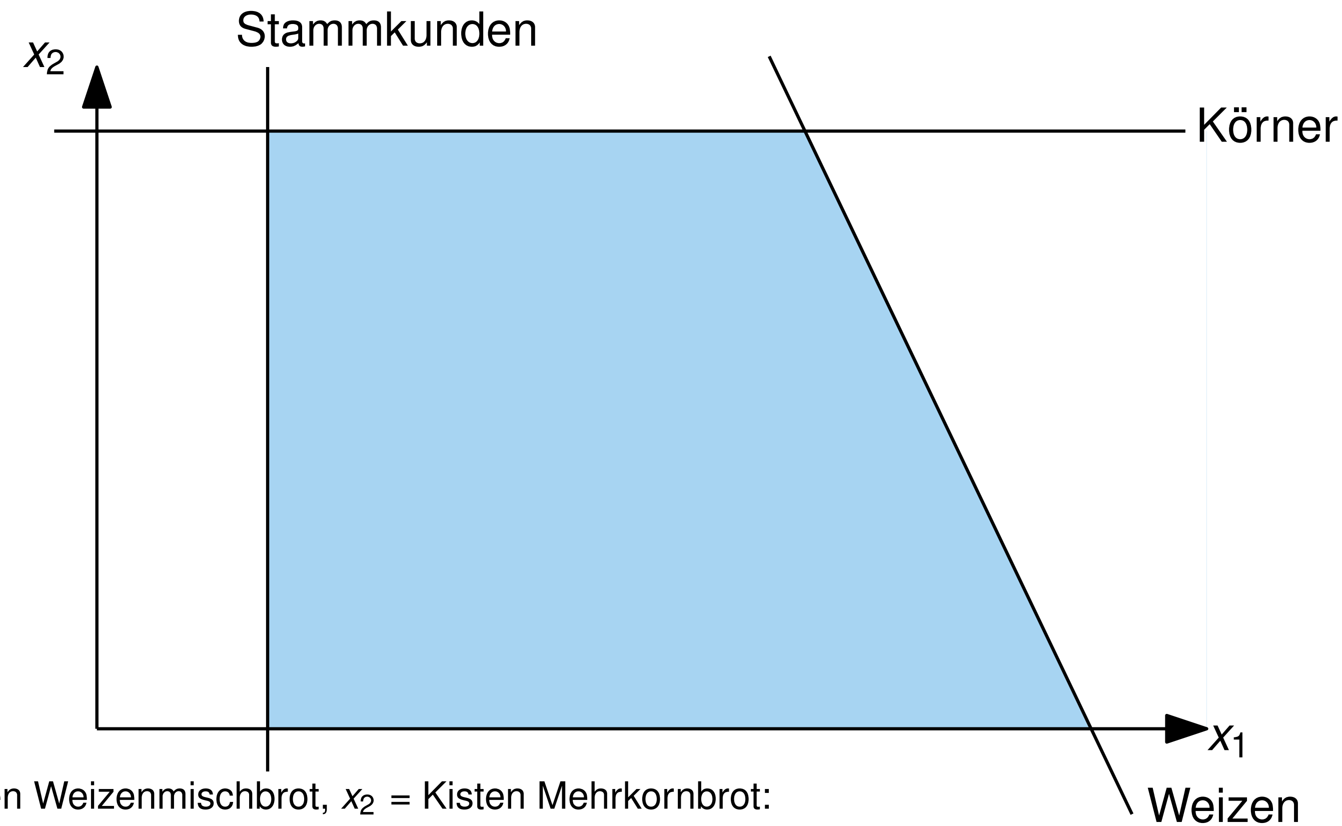


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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:

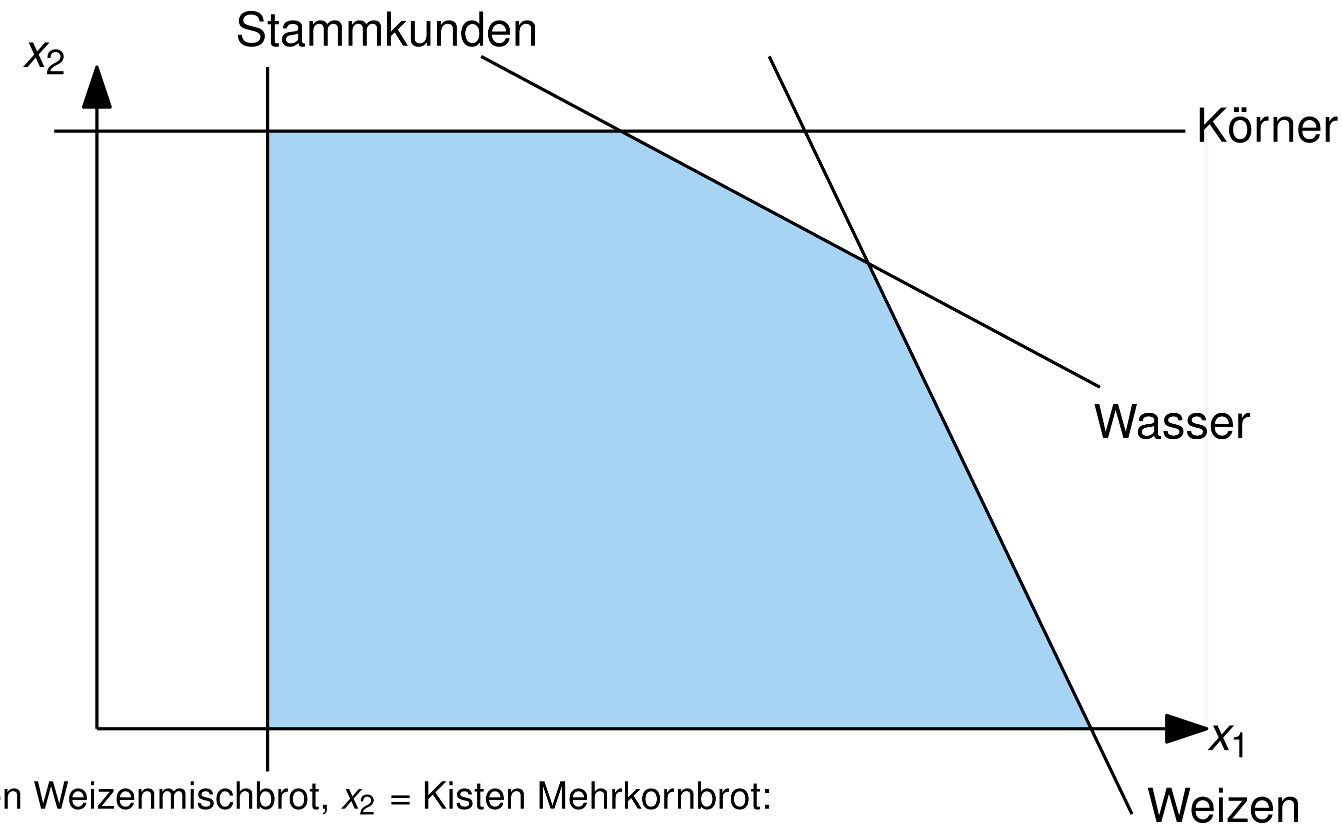


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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:



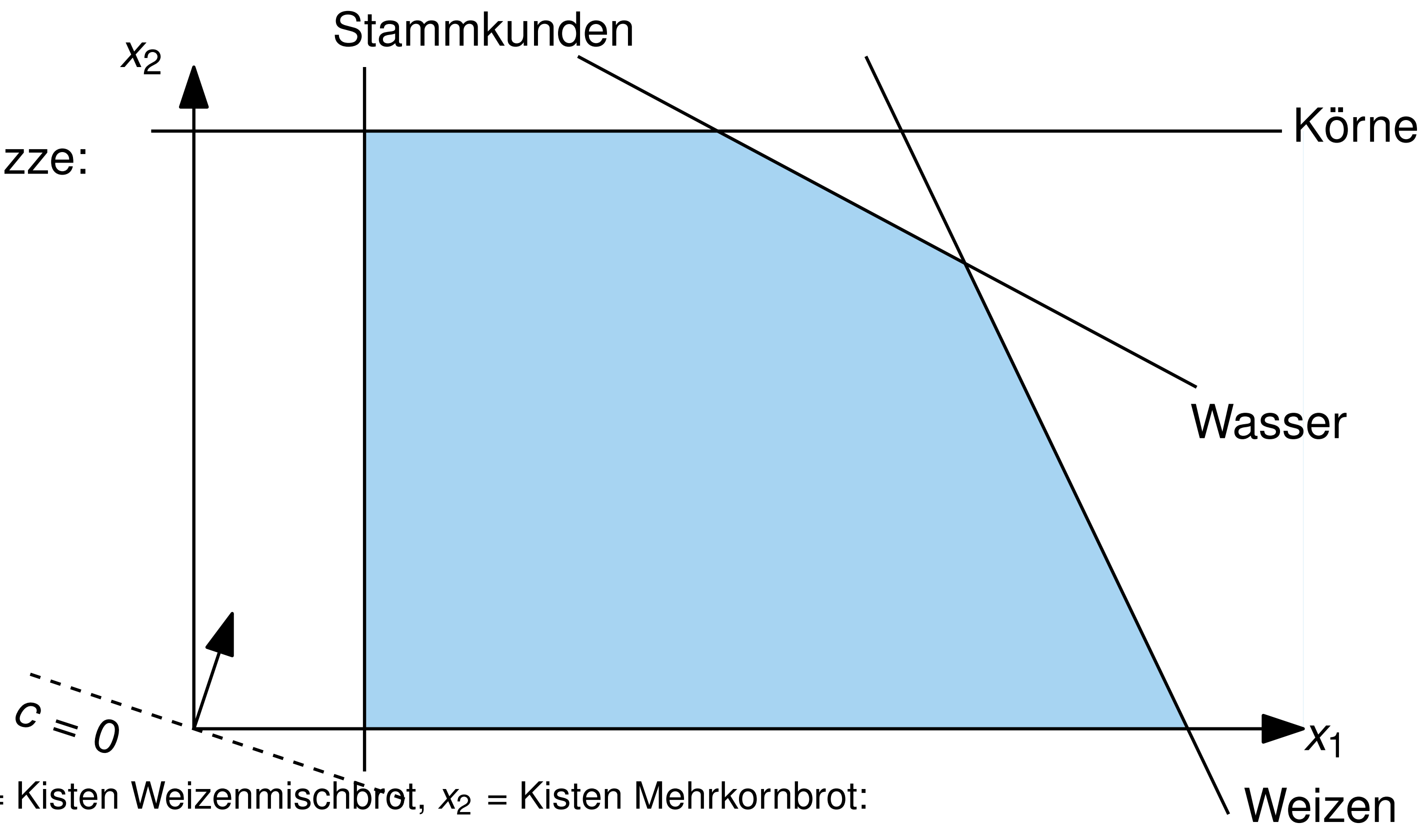
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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:

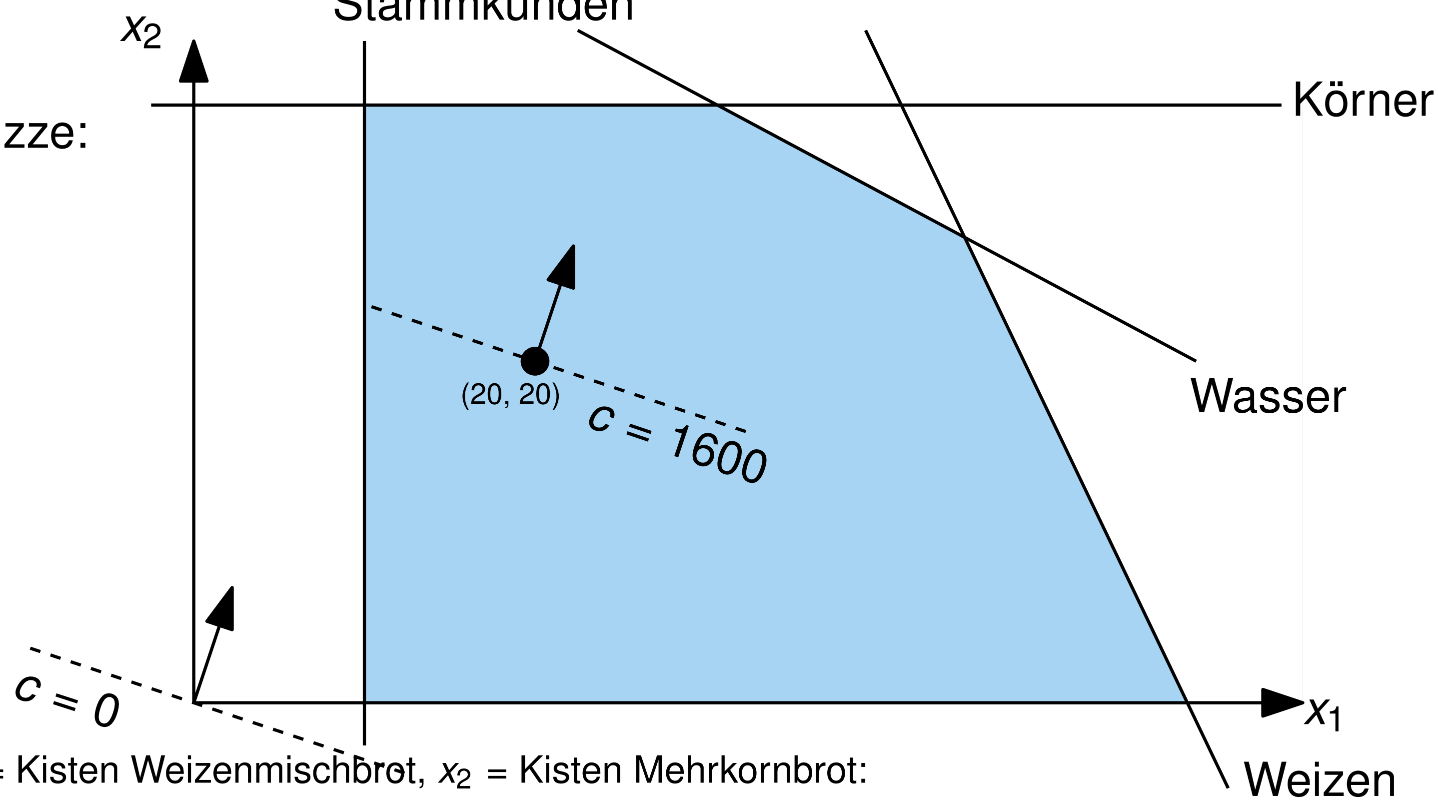


x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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	

Geometrische Interpretation

Skizze:



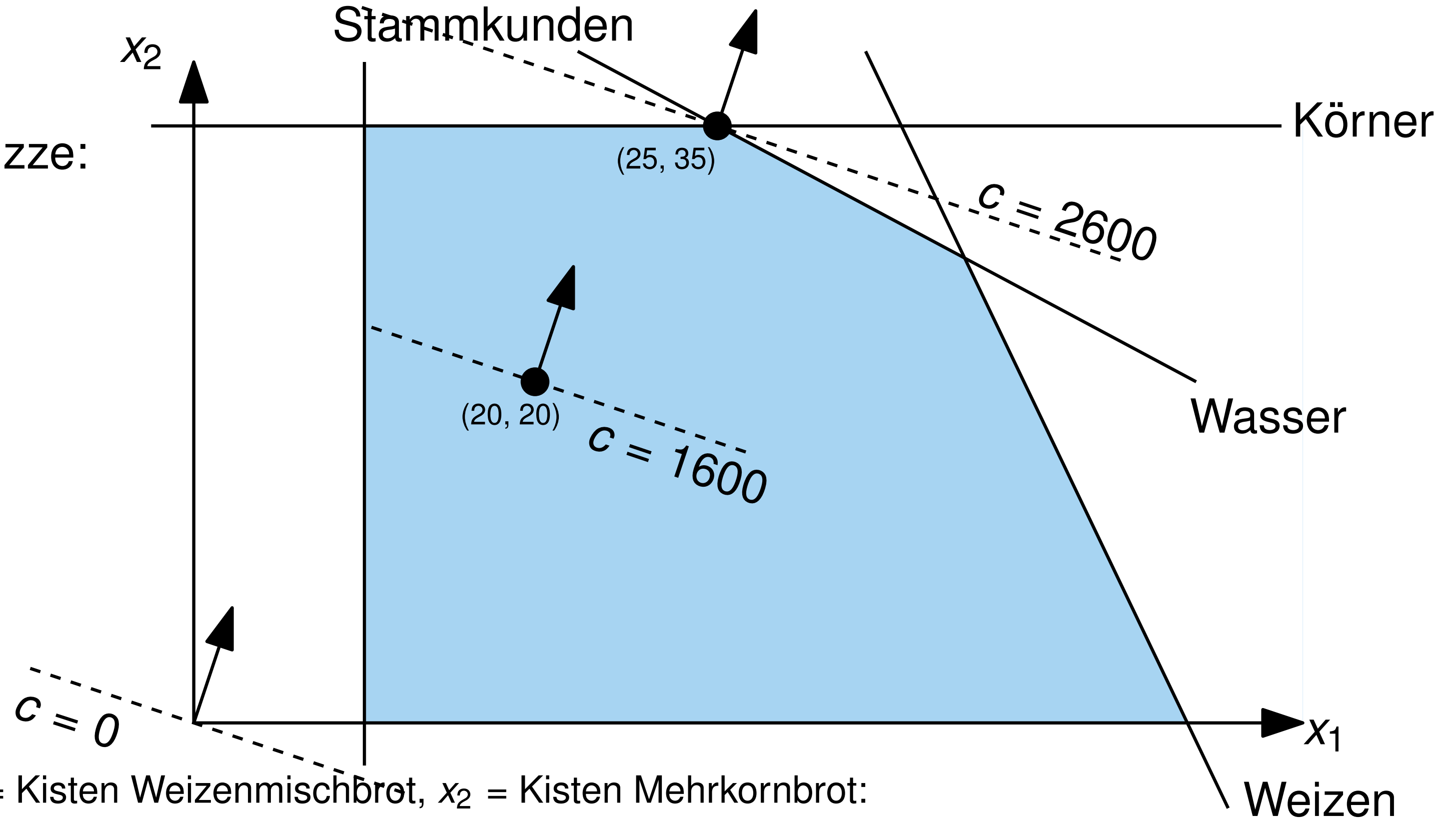
x_1 = Kisten Weizenmischbrot, x_2 = Kisten Mehrkornbrot:

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

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	

Dualität von Linearen Programmen

Obere Schranke

Betrachte folgendes Programm:

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

Nebenbedingungen **NB:**

$$
\begin{aligned}
4x_1 + 8x_2 &\leq 12\\
2x_1 + x_2 &\leq 3\\
3x_1 + 2x_2 &\leq 4\\
x_1 &\geq 0\\
x_2 &\geq 0
\end{aligned}
$$

Schätze eine obere Schranke für f mithilfe der Nebenbedingungen ab:

Folgendes Beispiel aus
Understanding and Using Linear Programming
von Jiří Matoušek und Bernd Gärtner.
Im Uni-Netz erhältlich als PDF unter springerlink.com.

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

Obere Schranke

Betrachte folgendes Programm:

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

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	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 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.

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	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 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	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 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	≤ 12
		2	x_1	+		x_2	≤ 3
		3	x_1	+	2	x_2	≤ 4
			x_1				≥ 0
						x_2	≥ 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:	$g(y_1, y_2, y_3) =$	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
										$y_1, y_2, y_3 \geq 0$

Duales Programm *DP*

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*

Dualität

Karlsruher Institut für Technologie

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
Primales Programm PP						x_2	≥ 0

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
Duales Programm DP								$y_1, y_2, y_3 \geq 0$		

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

Karlsruher Institut für Technologie

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
Primales Programm *PP*						x_2	≥ 0

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
Duales Programm *DP*										$y_1, y_2, y_3 \geq 0$

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.

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 $\max\limits_{\overline{x} \in N} f(\overline{x}) = \min\limits_{\overline{y} \in M} g(\overline{y})$

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 $1, \dots, m$ jeweils den minimalen täglichen Bedarf b_i $(i = 1, \dots, m)$ 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 $1, \dots, n$ jeweils den Preis pro Einheit c_j $(j = 1, \dots, n)$ sowie den Anteil a_{ij} des Nährstoffes i $(i = 1, \dots, m)$ am Produkt j $(j = 1, \dots, n)$.

Formulieren Sie das beschriebene Optimierungsproblem als lineares Programm L.

Zielfunktion: 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 $i = 1, \dots, m$

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 $(j = 1, \ldots, n)$.

Formulieren Sie das zu L duale lineare Programm D. Geben Sie eine sinnvolle Interpretation des dualen Programms an und überlegen Sie sich dabei, wer ein Interesse daran haben könnte, das duale Programm zu optimieren.

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

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

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_j nicht überschreiten.
- Der Besitzer des lokalen Supermarktes möchte den Preis eines solchen Einkaufs maximieren.