Algorithmen II
Vorlesung am 30.10.2012

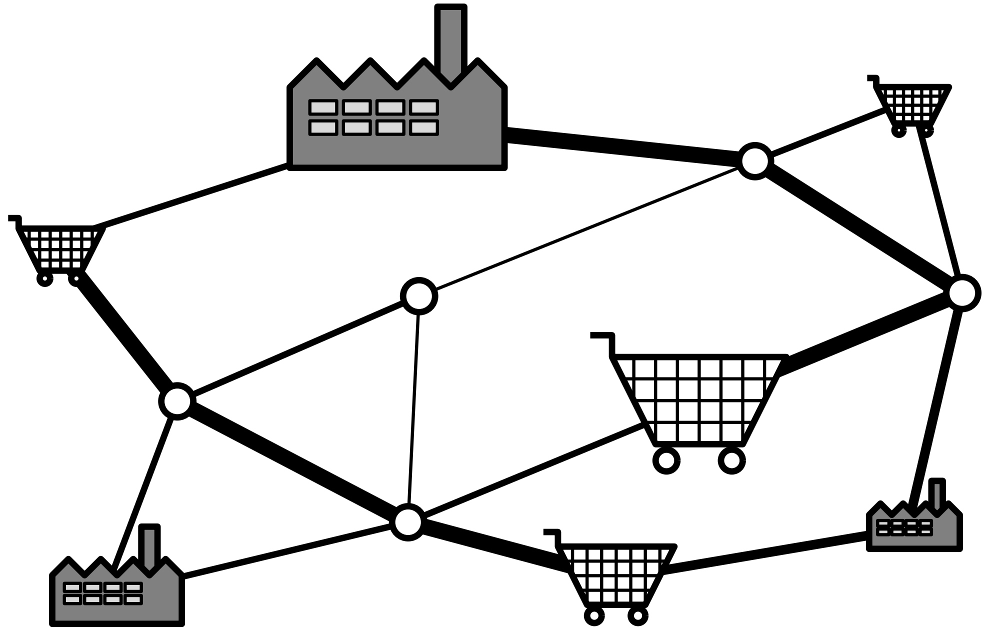
Flüsse mit Kosten · Matchings

INSTITUT FÜR THEORETISCHE INFORMATIK · PROF. DR. DOROTHEA WAGNER

Flussnetzwerke mit Kosten

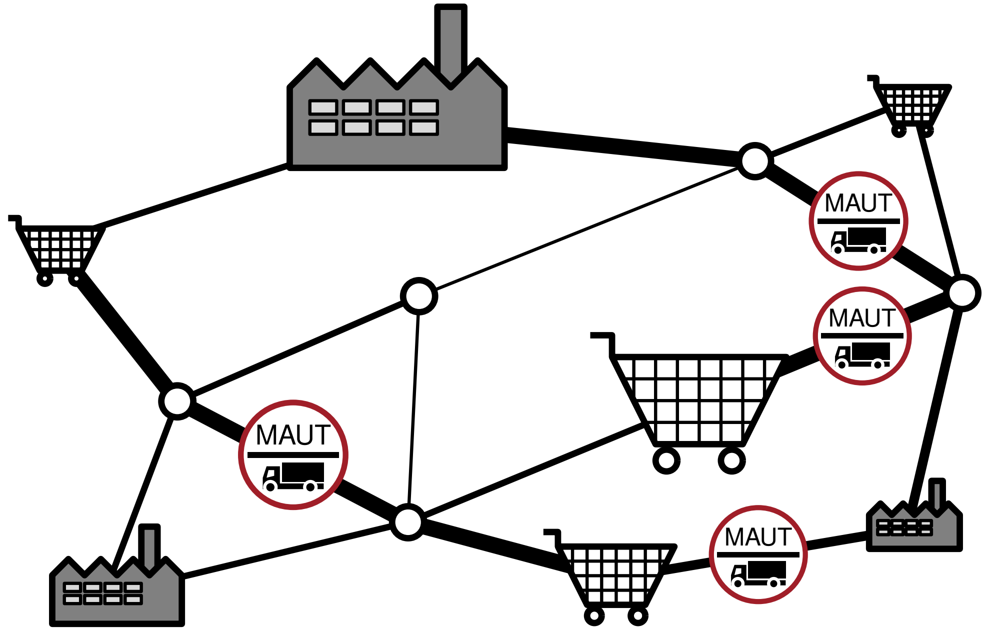
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Motivation – Transportnetzwerk



$\Rightarrow$ Flussnetzwerk mit mehreren Quellen und mehreren Senken

Motivation – Transportnetzwerk



Produzent Konsument

Zwischenstation / Transportweg

⇒ Flussnetzwerk mit mehreren Quellen und mehreren Senken

Neu: Kosten für die Nutzung eines Transportwegs

⇒ Flussnetzwerk mit mehreren Quellen/Senken und Kosten auf den Kanten

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Formale Definition

Karlsruher Institut für Technologie

Definition: Flussnetzwerk mit Kosten

Gerichteter Graph $D = (V, E)$ mit Kantenkapazitäten $c \colon E \longrightarrow \mathbb{R}_0^+$, *Kantenkosten* $\mathrm{cost} \colon E \longrightarrow \mathbb{R}$ sowie *Knotenbedarfsfunktion* $b \colon V \longrightarrow \mathbb{R}$ mit $\sum_{v \in V} b(v) = 0$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Formale Definition

Definition: Flussnetzwerk mit Kosten

Gerichteter Graph $D = (V, E)$ mit Kantenkapazitäten $c\colon E \longrightarrow \mathbb{R}_0^+$, *Kantenkosten* $\text{cost}\colon E \longrightarrow \mathbb{R}$ sowie *Knotenbedarfsfunktion* $b\colon V \longrightarrow \mathbb{R}$ mit $\sum_{v \in V} b(v) = 0$.

Senken haben positiven Bedarf, Quellen negativen.

Formale Definition

Definition: Flussnetzwerk mit Kosten

Gerichteter Graph $D = (V, E)$ mit Kantenkapazitäten $c\colon E \longrightarrow \mathbb{R}_0^+$, *Kantenkosten* $\mathrm{cost}\colon E \longrightarrow \mathbb{R}$ sowie *Knotenbedarfsfunktion* $b\colon V \longrightarrow \mathbb{R}$ mit $\sum_{v \in V} b(v) = 0$.

Senken haben positiven Bedarf, Quellen negativen.

Es wird genauso viel produziert wie konsumiert.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Formale Definition

Definition: Flussnetzwerk mit Kosten

Gerichteter Graph $D = (V, E)$ mit Kantenkapazitäten $c\colon E \longrightarrow \mathbb{R}_0^+$, *Kantenkosten* $\mathrm{cost}\colon E \longrightarrow \mathbb{R}$ sowie *Knotenbedarfsfunktion* $b\colon V \longrightarrow \mathbb{R}$ mit $\sum_{v \in V} b(v) = 0$.

Senken haben positiven Bedarf, Quellen negativen.

Es wird genauso viel produziert wie konsumiert.

Definition: Fluss & Flusskosten

Ein *Fluss* f in D ist eine Abbildung $f\colon E \longrightarrow \mathbb{R}_0^+$ mit:

- für alle $(u, v) \in E$: $0 \leq f(u, v) \leq c(u, v)$ (Kapazitätsbedingung)
- für alle $u \in V$: $\sum\limits_{v:(v,u)\in E} f(v, u) - \sum\limits_{v:(u,v)\in E} f(u, v) = b(u)$ (Flusserhaltungsbedingung)

Formale Definition

Definition: Flussnetzwerk mit Kosten

Gerichteter Graph $D = (V, E)$ mit Kantenkapazitäten $c\colon E \longrightarrow \mathbb{R}_0^+$, *Kantenkosten* $\text{cost}\colon E \longrightarrow \mathbb{R}$ sowie *Knotenbedarfsfunktion* $b\colon V \longrightarrow \mathbb{R}$ mit $\sum_{v \in V} b(v) = 0$.

Senken haben positiven Bedarf, Quellen negativen.

Es wird genauso viel produziert wie konsumiert.

Definition: Fluss & Flusskosten

Ein *Fluss* f in D ist eine Abbildung $f\colon E \longrightarrow \mathbb{R}_0^+$ mit:

- für alle $(u, v) \in E$: $0 \leq f(u, v) \leq c(u, v)$ (Kapazitätsbedingung)
- für alle $u \in V$: $\sum_{v:(v,u) \in E} f(v, u) - \sum_{v:(u,v) \in E} f(u, v) = b(u)$ (Flusserhaltungsbedingung)

Es kann sein, dass eine solche Abbildung garnicht existiert!

Formale Definition

Definition: Flussnetzwerk mit Kosten
Gerichteter Graph $D = (V, E)$ mit Kantenkapazitäten $c\colon E \longrightarrow \mathbb{R}_0^+$, *Kantenkosten* $\mathrm{cost}\colon E \longrightarrow \mathbb{R}$ sowie *Knotenbedarfsfunktion* $b\colon V \longrightarrow \mathbb{R}$ mit $\sum_{v \in V} b(v) = 0$.

Senken haben positiven Bedarf, Quellen negativen.

Es wird genauso viel produziert wie konsumiert.

Definition: Fluss & Flusskosten
Ein *Fluss* f in D ist eine Abbildung $f\colon E \longrightarrow \mathbb{R}_0^+$ mit:

- für alle $(u, v) \in E$: $0 \leq f(u, v) \leq c(u, v)$ (Kapazitätsbedingung)
- für alle $u \in V$: $\sum_{v:(v,u) \in E} f(v, u) - \sum_{v:(u,v) \in E} f(u, v) = b(u)$ (Flusserhaltungsbedingung)

Die *Kosten* eines Flusses f berechnen sich durch $\mathrm{cost}(f) = \sum_{e \in E} f(e) \cdot \mathrm{cost}(e)$.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Formale Definition

Definition: Flussnetzwerk mit Kosten
Gerichteter Graph $D = (V, E)$ mit Kantenkapazitäten $c\colon E \longrightarrow \mathbb{R}_0^+$, *Kantenkosten* $\mathrm{cost}\colon E \longrightarrow \mathbb{R}$ sowie *Knotenbedarfsfunktion* $b\colon V \longrightarrow \mathbb{R}$ mit $\sum_{v \in V} b(v) = 0$.

Senken haben positiven Bedarf, Quellen negativen.

Es wird genauso viel produziert wie konsumiert.

Definition: Fluss & Flusskosten
Ein *Fluss* f in D ist eine Abbildung $f\colon E \longrightarrow \mathbb{R}_0^+$ mit:

- für alle $(u, v) \in E$: $0 \leq f(u, v) \leq c(u, v)$ (Kapazitätsbedingung)
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v, u) - \sum_{v:(u,v)\in E} f(u, v) = b(u)$ (Flusserhaltungsbedingung)

Die *Kosten* eines Flusses f berechnen sich durch $\mathrm{cost}(f) = \sum_{e\in E} f(e) \cdot \mathrm{cost}(e)$.

Problem: MinCostFlow
Finde einen *kostenminimalen* Fluss f in D. (d. h. $\mathrm{cost}(f) \leq \mathrm{cost}(f')$ für alle Flüsse f')

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MinCostFlow – Ein Lösungsansatz

Ein Algorithmus in zwei Schritten:

(1) Finde einen gültigen Fluss f im Flussnetzwerk D (falls es einen gibt).

Instanz (D, c, cost, b) von MinCostFlow

erstelle „Superquelle“ s & „Supersenke“ t

Instanz (D', c', s, t) von MaxFlow

benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MINCOSTFLOW – Ein Lösungsansatz

Ein Algorithmus in zwei Schritten:

(1) Finde einen gültigen Fluss f im Flussnetzwerk D (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ s & „Supersenke“ t

Instanz (D', c', s, t) von MAXFLOW

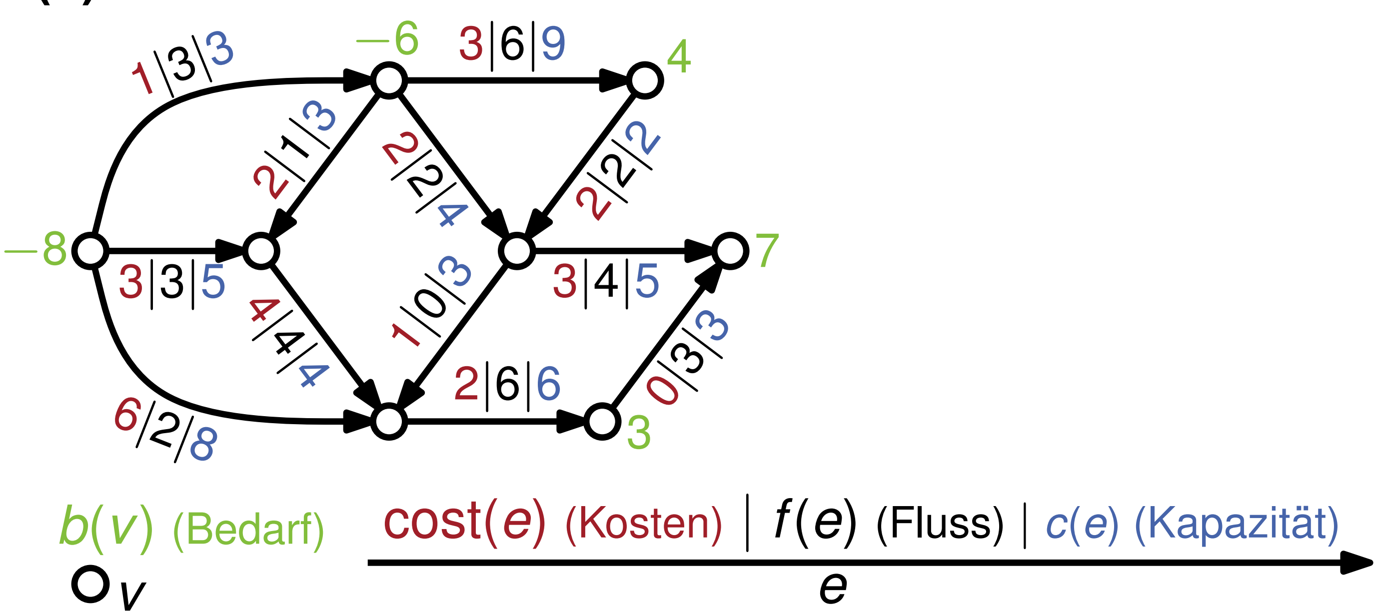
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D

(2) Verbessere f schrittweise.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MINCOSTFLOW – Ein Lösungsansatz

Karlsruher Institut für Technologie

Ein Algorithmus in zwei Schritten:

(1) Finde einen gültigen Fluss *f* im Flussnetzwerk *D* (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ s & „Supersenke“ t

Instanz (D', c', s, t) von MAXFLOW

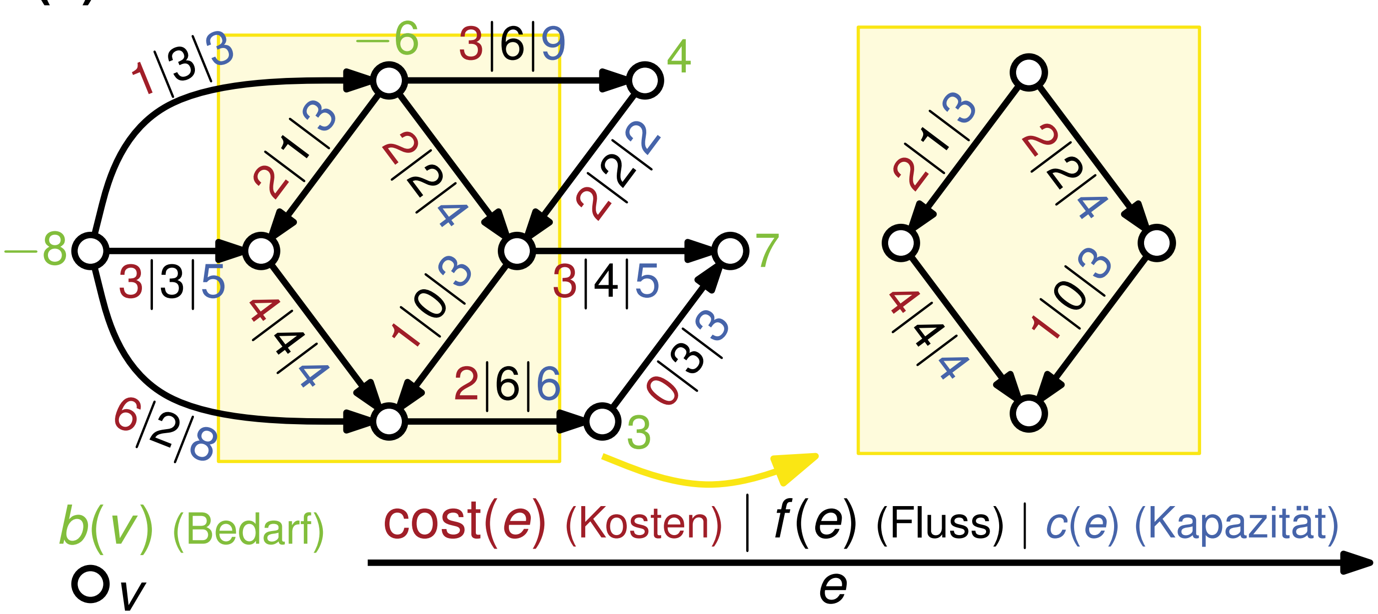
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D

(2) Verbessere *f* schrittweise.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

MINCOSTFLOW – Ein Lösungsansatz

Ein Algorithmus in zwei Schritten:

(1) Finde einen gültigen Fluss *f* im Flussnetzwerk *D* (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ *s* & „Supersenke“ *t*

Instanz (D', c', s, t) von MAXFLOW

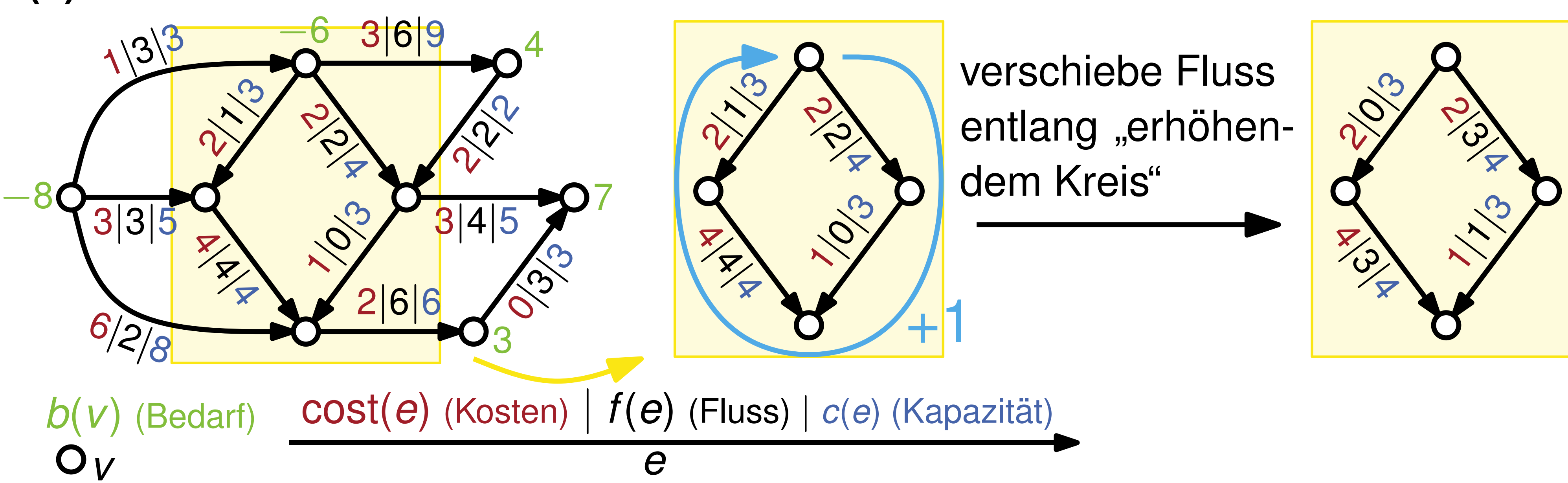
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss *f* in *D*

(2) Verbessere *f* schrittweise.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

(1) Finde einen gültigen Fluss *f* im Flussnetzwerk *D* (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ & „Supersenke“

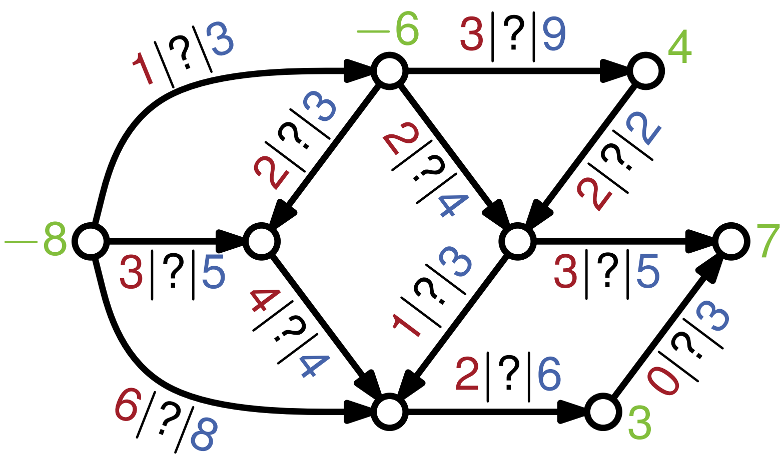
Instanz (D', c', s, t) von MAXFLOW

benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

(1) Finde einen gültigen Fluss f im Flussnetzwerk D (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ & „Supersenke“

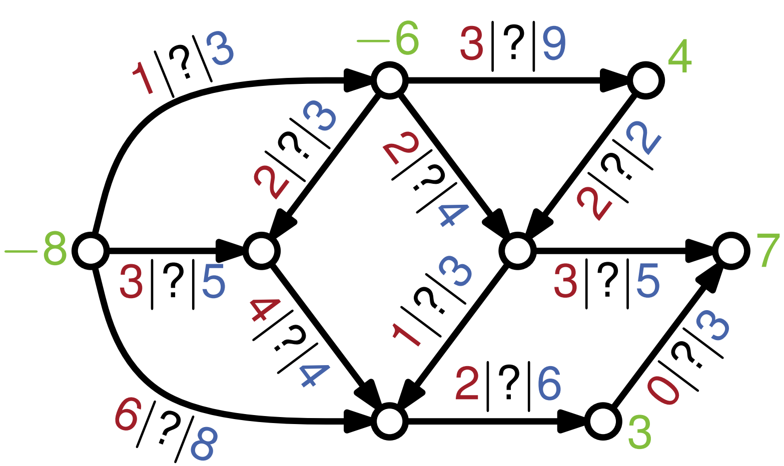
Instanz (D', c', s, t) von MAXFLOW

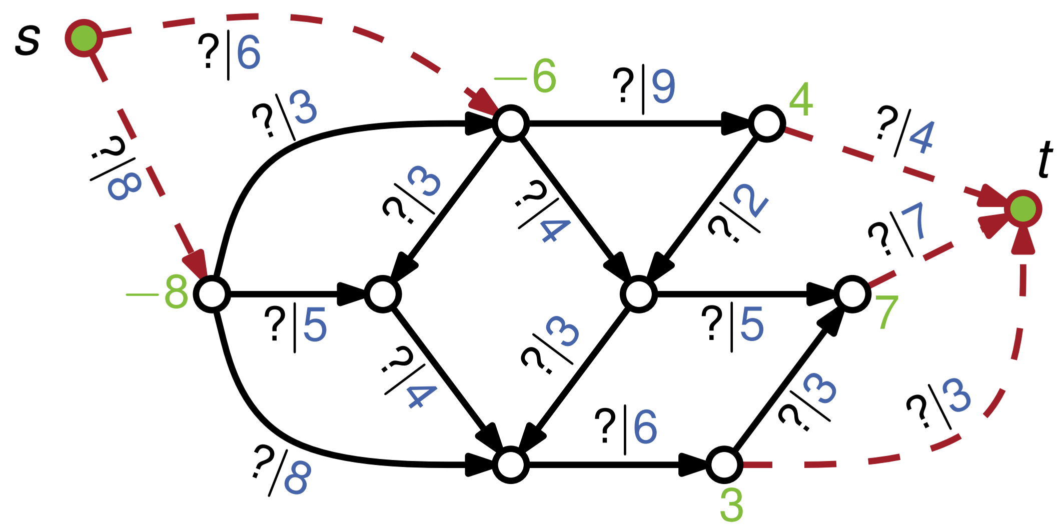
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D





- Erstelle *Supersenke* t. Erstelle Kante (v, t) mit $c(v, t) = b(v)$ falls $b(v) > 0$.
- Erstelle *Superquelle* s. Erstelle Kante (s, v) mit $c(s, v) = -b(v)$ falls $b(v) < 0$.

Berechnung eines gültigen Flusses

(1) Finde einen gültigen Fluss f im Flussnetzwerk D (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ & „Supersenke“

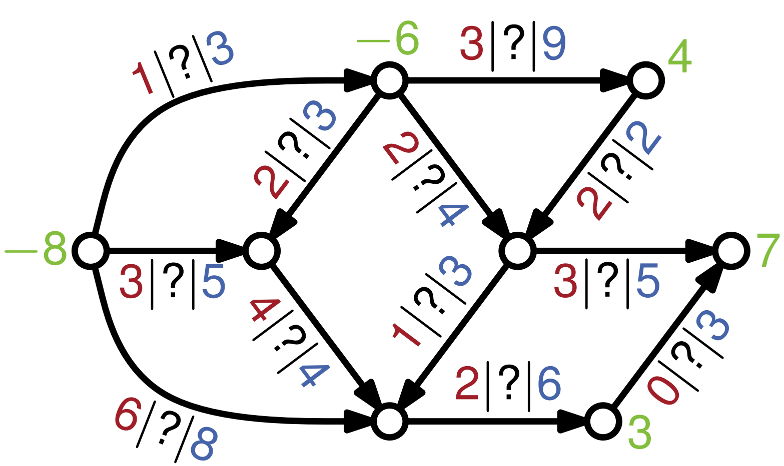
Instanz (D', c', s, t) von MAXFLOW

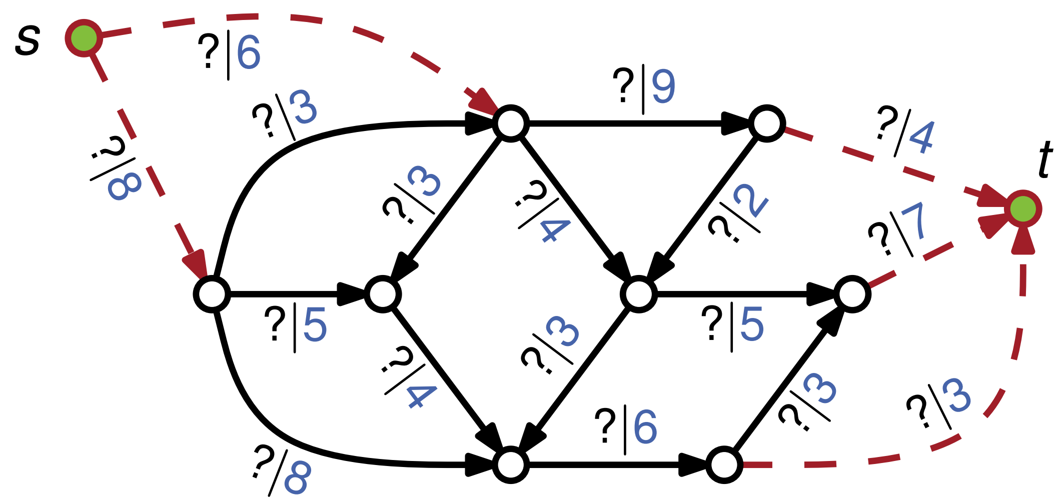
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D





- Erstelle *Supersenke* t. Erstelle Kante (v, t) mit $c(v, t) = b(v)$ falls $b(v) > 0$.
- Erstelle *Superquelle* s. Erstelle Kante (s, v) mit $c(s, v) = -b(v)$ falls $b(v) < 0$.

Berechnung eines gültigen Flusses

(1) Finde einen gültigen Fluss f im Flussnetzwerk D (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ & „Supersenke“

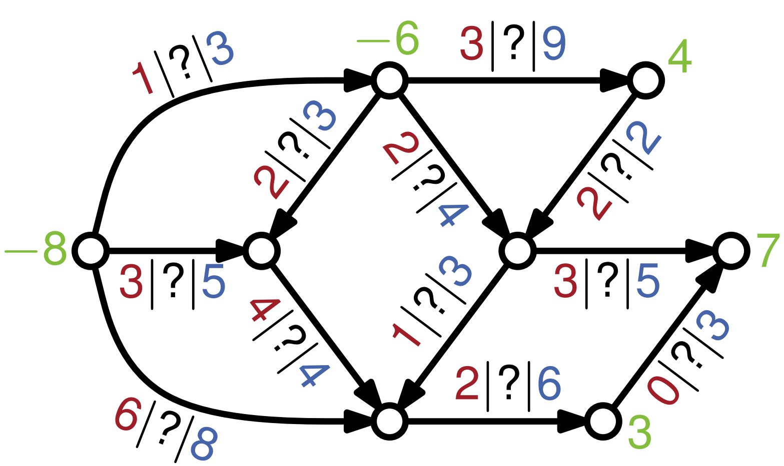
Instanz (D', c', s, t) von MAXFLOW

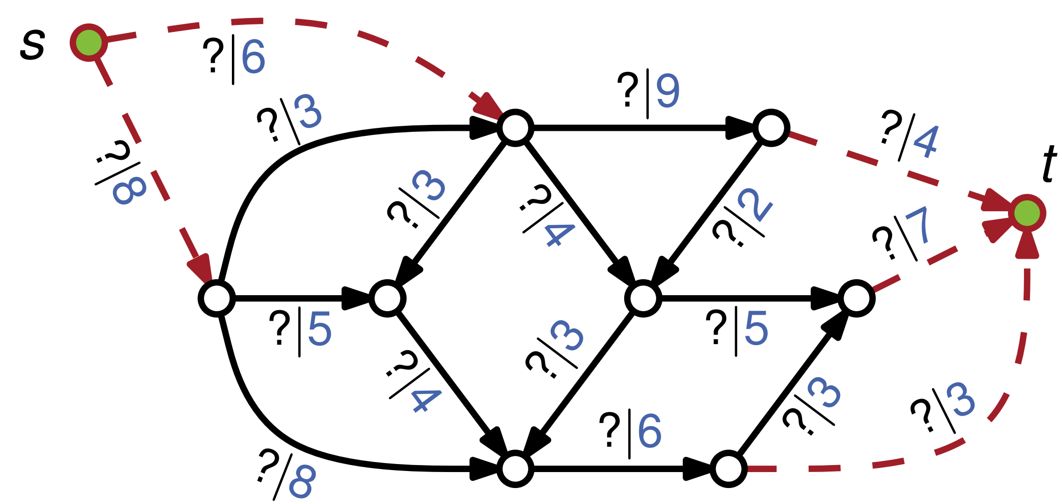
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D





- Berechne einen maximalen Fluss zwischen s und t in D'.

Berechnung eines gültigen Flusses

(1) Finde einen gültigen Fluss f im Flussnetzwerk D (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ & „Supersenke“

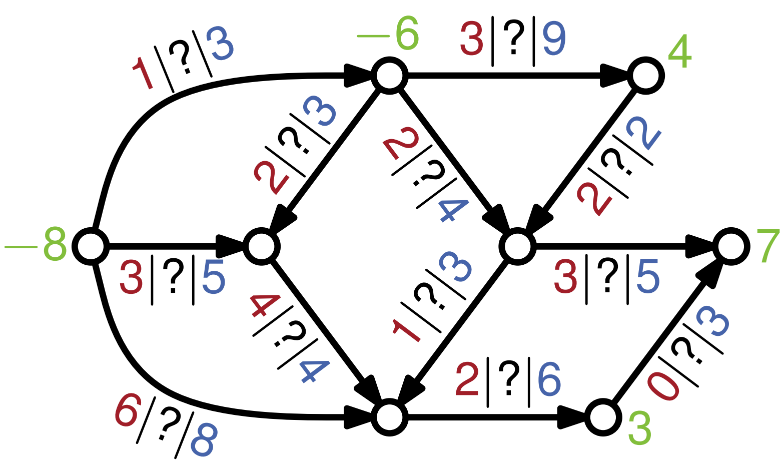
Instanz (D', c', s, t) von MAXFLOW

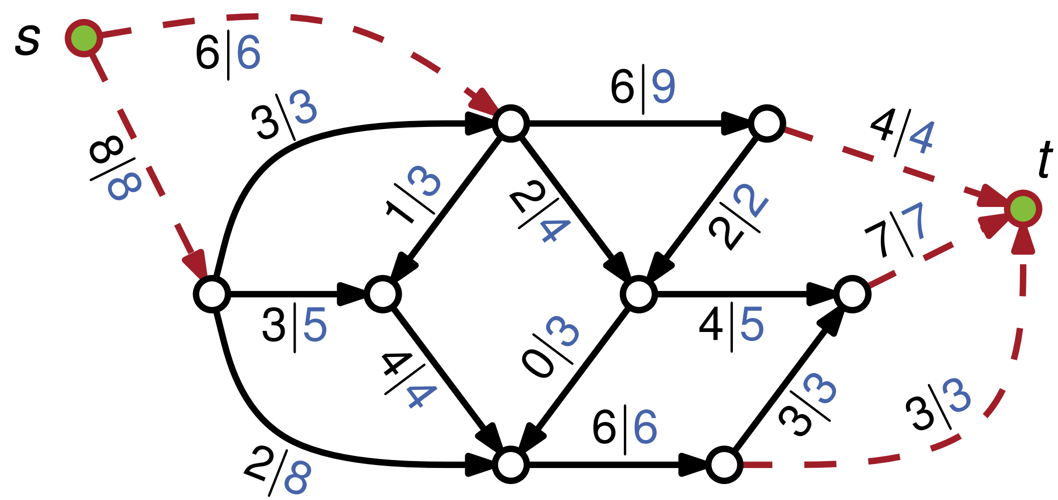
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D





- Berechne einen maximalen Fluss zwischen s und t in D'.

Berechnung eines gültigen Flusses

(1) Finde einen gültigen Fluss f im Flussnetzwerk D (falls es einen gibt).

Instanz (D, c, cost, b) von MINCOSTFLOW

erstelle „Superquelle“ & „Supersenke“

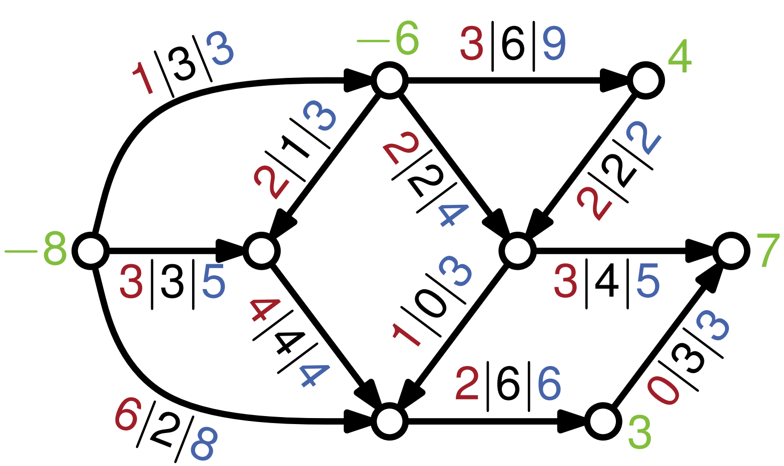
Instanz (D', c', s, t) von MAXFLOW

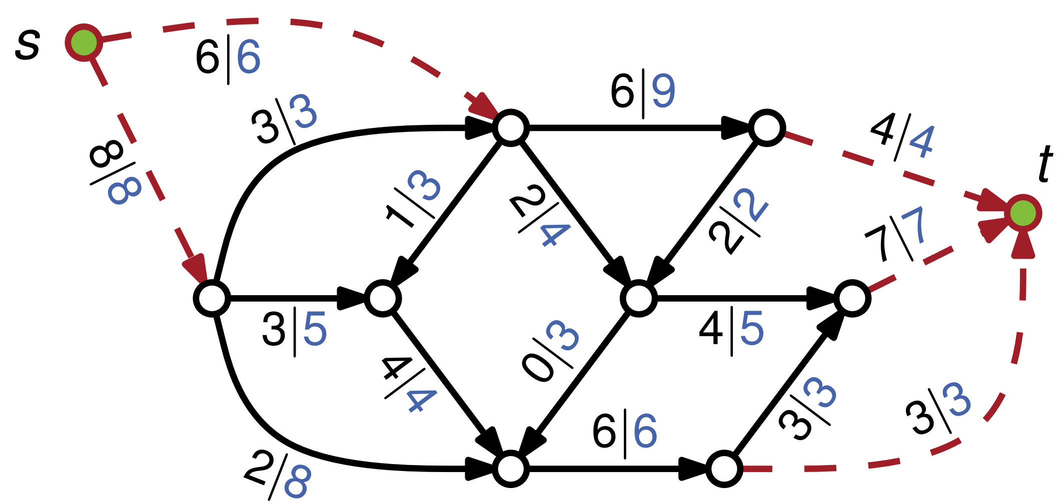
benutze Algorithmus für maximale Flüsse

maximaler Fluss f' in D'

lösche „Superquelle“ & „Supersenke“

gültiger Fluss f in D





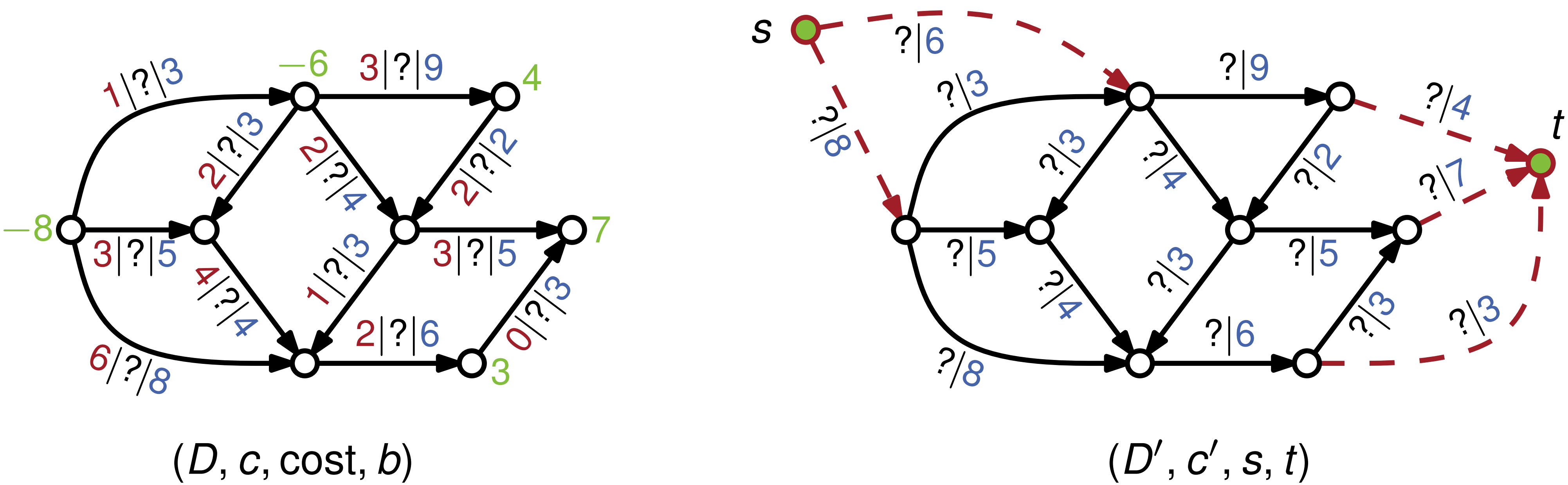
- Lösche s und t (und inzidente Kanten). Setze $f(e) = f'(e)$ für jede Kante $e \in E$.
- **Beh.:** Wenn $f'(s, v) = c(s, v)$ (für alle $v \in V$), dann ist f gültig. Sonst gibt es keinen gültigen Fluss in D.

Berechnung eines gültigen Flusses

Definition: Konstruktion von D'

Für eine Instanz ($D = (V, E), c, \text{cost}, b$) von MinCostFlow ist die Instanz ($D' = (V', E'), c', s, t$) von MaxFlow wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$



Berechnung eines gültigen Flusses

Definition: Konstruktion von D'

Für eine Instanz ($D = (V, E), c, \text{cost}, b$) von MINCOSTFLOW ist die Instanz ($D' = (V', E'), c', s, t$) von MAXFLOW wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Satz: D und D' sind äquivalent

Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D. Andernfalls gibt es keinen gültigen Fluss in D.

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D. Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

Definition: Konstruktion von D'
Für eine Instanz $(D = (V, E), c, \text{cost}, b)$ von MINCOSTFLOW ist die Instanz $(D' = (V', E'), c', s, t)$ von MAXFLOW wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D. Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u, v) \in E$: $0 \leq f(u, v) \leq c(u, v)$ (Kapazitätsbedingung) ✓

Definition: Konstruktion von D'
Für eine Instanz $(D = (V, E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V', E'), c', s, t)$ von MaxFlow wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u, v) \in E$: $0 \leq f(u, v) \leq c(u, v)$ (Kapazitätsbedingung) ✔
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v, u) - \sum_{v:(u,v)\in E} f(u, v) = b(u)$ (Flusserhaltungsbedingung)

Definition: Konstruktion von D'
Für eine Instanz $(D = (V, E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V', E'), c', s, t)$ von MaxFlow wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u, v) \in E$: $0 \leq f(u, v) \leq c(u, v)$ (Kapazitätsbedingung) ✓
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v, u) - \sum_{v:(u,v)\in E} f(u, v) = b(u)$ (Flusserhaltungsbedingung)

Sei $u \in V$ beliebig.

Fall 1: $b(u) = 0$
Folgt aus Flusserhaltung in D'.

Definition: Konstruktion von D'
Für eine Instanz $(D = (V, E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V', E'), c', s, t)$ von MaxFlow wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s,v) = c(s,v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D. Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u,v) \in E$: $0 \leq f(u,v) \leq c(u,v)$ (Kapazitätsbedingung) ✓
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$ (Flusserhaltungsbedingung)

Sei $u \in V$ beliebig.

Fall 1: $b(u) = 0$
Folgt aus Flusserhaltung in D'.

Fall 2: $b(u) < 0$ (u ist Quelle)

$$\sum_{v:(v,u)\in E'} f'(v,u) - \sum_{v:(u,v)\in E'} f'(u,v) = 0$$

(eingehender − ausgehender Fluss in D')

Definition: Konstruktion von D'
Für eine Instanz $(D = (V,E), c, \mathrm{cost}, b)$ von MINCOSTFLOW ist die Instanz $(D' = (V',E'), c', s, t)$ von MAXFLOW wie folgt definiert.

- $V' = V \cup \{s,t\}$
- $E_s = \{(s,v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s,v) = -b(v)$
- $E_t = \{(v,t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v,t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Berechnung eines gültigen Flusses

Karlsruher Institut für Technologie

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s,v) = c(s,v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u,v) \in E$: $0 \leq f(u,v) \leq c(u,v)$ (Kapazitätsbedingung) ✓
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$ (Flusserhaltungsbedingung)

Sei $u \in V$ beliebig.

Fall 1: $b(u) = 0$
Folgt aus Flusserhaltung in D'.

Fall 2: $b(u) < 0$ (u ist Quelle)

$$\sum_{v:(v,u)\in E'} f'(v,u) - \sum_{v:(u,v)\in E'} f'(u,v) = 0$$

(eingehender − ausgehender Fluss in D')

$$\Leftrightarrow \sum_{v:(v,u)\in E} f(v,u) + f'(s,u) - \sum_{v:(u,v)\in E} f(u,v) = 0$$

Definition: Konstruktion von D'
Für eine Instanz $(D = (V,E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V',E'), c', s, t)$ von MaxFlow wie folgt definiert.

- $V' = V \cup \{s,t\}$
- $E_s = \{(s,v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s,v) = -b(v)$
- $E_t = \{(v,t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v,t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

Karlsruher Institut für Technologie

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s,v) = c(s,v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u,v) \in E$: $0 \leq f(u,v) \leq c(u,v)$ (Kapazitätsbedingung) ✔
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$ (Flusserhaltungsbedingung)

Sei $u \in V$ beliebig.

Fall 1: $b(u) = 0$
Folgt aus Flusserhaltung in D'.

Definition: Konstruktion von D'
Für eine Instanz ($D = (V,E), c, \text{cost}, b$) von MinCostFlow ist die Instanz ($D' = (V', E'), c', s, t$) von MaxFlow wie folgt definiert.

- $V' = V \cup \{s,t\}$
- $E_s = \{(s,v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s,v) = -b(v)$
- $E_t = \{(v,t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v,t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Fall 2: $b(u) < 0$ (u ist Quelle)

$$\boxed{\sum_{v:(v,u)\in E'} f'(v,u)} - \sum_{v:(u,v)\in E'} f'(u,v) = 0$$

(eingehender $-$ ausgehender Fluss in D')

$$\Leftrightarrow \boxed{\sum_{v:(v,u)\in E} f(v,u) + f'(s,u)} - \sum_{v:(u,v)\in E} f(u,v) = 0$$

u hat in D die gleichen eingehenden Kanten wie in D', abgesehen von (s,u)

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

Karlsruher Institut für Technologie

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s,v) = c(s,v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u,v) \in E$: $0 \leq f(u,v) \leq c(u,v)$ (Kapazitätsbedingung)
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$ (Flusserhaltungsbedingung)

Sei $u \in V$ beliebig.

Fall 1: $b(u) = 0$
Folgt aus Flusserhaltung in D'.

Fall 2: $b(u) < 0$ (u ist Quelle)

$$\sum_{v:(v,u)\in E'} f'(v,u) \boxed{- \sum_{v:(u,v)\in E'} f'(u,v)} = 0$$

(eingehender − ausgehender Fluss in D')

$$\Leftrightarrow \sum_{v:(v,u)\in E} f(v,u) + f'(s,u) \boxed{- \sum_{v:(u,v)\in E} f(u,v)} = 0$$

u hat in D die gleichen ausgehenden Kanten wie in D'

Definition: Konstruktion von D'
Für eine Instanz $(D = (V,E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V',E'), c', s, t)$ von MaxFlow wie folgt definiert.

- $V' = V \cup \{s,t\}$
- $E_s = \{(s,v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s,v) = -b(v)$
- $E_t = \{(v,t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v,t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

Karlsruher Institut für Technologie

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s,v) = c(s,v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u,v) \in E$: $0 \leq f(u,v) \leq c(u,v)$ (Kapazitätsbedingung) ✓
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$ (Flusserhaltungsbedingung)

Sei $u \in V$ beliebig.

Fall 1: $b(u) = 0$
Folgt aus Flusserhaltung in D'.

Definition: Konstruktion von D'
Für eine Instanz ($D = (V, E), c, \text{cost}, b$) von MINCOSTFLOW ist die Instanz ($D' = (V', E'), c', s, t$) von MAXFLOW wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Fall 2: $b(u) < 0$ (u ist Quelle)

$$\sum_{v:(v,u)\in E'} f'(v,u) - \sum_{v:(u,v)\in E'} f'(u,v) = 0$$

(eingehender − ausgehender Fluss in D')

$$\Leftrightarrow \sum_{v:(v,u)\in E} f(v,u) + f'(s,u) - \sum_{v:(u,v)\in E} f(u,v) = 0 \Leftrightarrow \sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

Karlsruher Institut für Technologie

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s,v) = c(s,v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

- für alle $(u,v) \in E$: $0 \leq f(u,v) \leq c(u,v)$ (Kapazitätsbedingung)
- für alle $u \in V$: $\sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$ (Flusserhaltungsbedingung)

Sei $u \in V$ beliebig.

Fall 1: $b(u) = 0$
Folgt aus Flusserhaltung in D'.

Fall 2: $b(u) < 0$ (u ist Quelle)

$$\sum_{v:(v,u)\in E'} f'(v,u) - \sum_{v:(u,v)\in E'} f'(u,v) = 0$$

(eingehender $-$ ausgehender Fluss in D')

$$\Leftrightarrow \sum_{v:(v,u)\in E} f(v,u) + f'(s,u) - \sum_{v:(u,v)\in E} f(u,v) = 0 \Leftrightarrow \sum_{v:(v,u)\in E} f(v,u) - \sum_{v:(u,v)\in E} f(u,v) = b(u)$$

Fall 3: $b(u) > 0$
analog

Definition: Konstruktion von D'
Für eine Instanz $(D = (V,E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V', E'), c', s, t)$ von MaxFlow wie folgt definiert.

- $V' = V \cup \{s,t\}$
- $E_s = \{(s,v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s,v) = -b(v)$
- $E_t = \{(v,t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v,t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

Zeige: Gegeben ein gültiger Fluss f in D, dann gibt es einen Fluss f' in D' mit $f'(s, v) = c(s, v)$ für alle $v \in V$.

Definition: Konstruktion von D'
Für eine Instanz $(D = (V, E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V', E'), c', s, t)$ von MaxFlow wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

Zeige: Gegeben ein gültiger Fluss f in D, dann gibt es einen Fluss f' in D' mit $f'(s, v) = c(s, v)$ für alle $v \in V$.

- $f'(e) = f(e)$ für $e \in E$
- $f'(s, v) = -b(v)$ für alle Quellen v
- $f'(v, t) = b(v)$ für alle Senken v

Definition: Konstruktion von D'
Für eine Instanz ($D = (V, E), c, \text{cost}, b$) von MinCostFlow ist die Instanz ($D' = (V', E'), c', s, t$) von MaxFlow wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Berechnung eines gültigen Flusses

Karlsruher Institut für Technologie

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

Zeige: Gegeben ein gültiger Fluss f in D, dann gibt es einen Fluss f' in D' mit $f'(s, v) = c(s, v)$ für alle $v \in V$.

- $f'(e) = f(e)$ für $e \in E$
- $f'(s, v) = -b(v)$ für alle Quellen v
- $f'(v, t) = b(v)$ für alle Senken v

Definition: Konstruktion von D'
Für eine Instanz $(D = (V, E), c, \text{cost}, b)$ von MINCOSTFLOW ist die Instanz $(D' = (V', E'), c', s, t)$ von MAXFLOW wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Die Abbildung f' ist ein Fluss in D'.

- Kapazitätsbedingung ✓
- Flusserhaltungsbedingung ✓

nachrechnen!!

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Berechnung eines gültigen Flusses

Satz: D und D' sind äquivalent
Sei f' ein maximaler Fluss in D'. Wenn $f'(s, v) = c(s, v)$ für alle $v \in V$ gilt, dann ist die Abbildung f mit $f(e) = f'(e)$ für alle Kanten $e \in E$ ein gültiger Fluss in D.
Andernfalls gibt es keinen gültigen Fluss in D.

Beweis:

Zeige: Gegeben ein gültiger Fluss f in D, dann gibt es einen Fluss f' in D' mit $f'(s, v) = c(s, v)$ für alle $v \in V$.

- $f'(e) = f(e)$ für $e \in E$
- $f'(s, v) = -b(v)$ für alle Quellen v
- $f'(v, t) = b(v)$ für alle Senken v

Definition: Konstruktion von D'
Für eine Instanz $(D = (V, E), c, \text{cost}, b)$ von MinCostFlow ist die Instanz $(D' = (V', E'), c', s, t)$ von MaxFlow wie folgt definiert.
- $V' = V \cup \{s, t\}$
- $E_s = \{(s, v) \mid b(v) < 0\}$, mit Kapazitäten $c'(s, v) = -b(v)$
- $E_t = \{(v, t) \mid b(v) > 0\}$, mit Kapazitäten $c'(v, t) = b(v)$
- $E' = E \cup E_s \cup E_t$, mit Kapazitäten $c'(e) = c(e)$ für $e \in E$

Die Abbildung f' ist ein Fluss in D'.
- Kapazitätsbedingung ✓
- Flusserhaltungsbedingung ✓

nachrechnen!!

Außerdem gilt: $f'(s, v) = c(s, v)$ für alle $v \in V$.

$\square$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

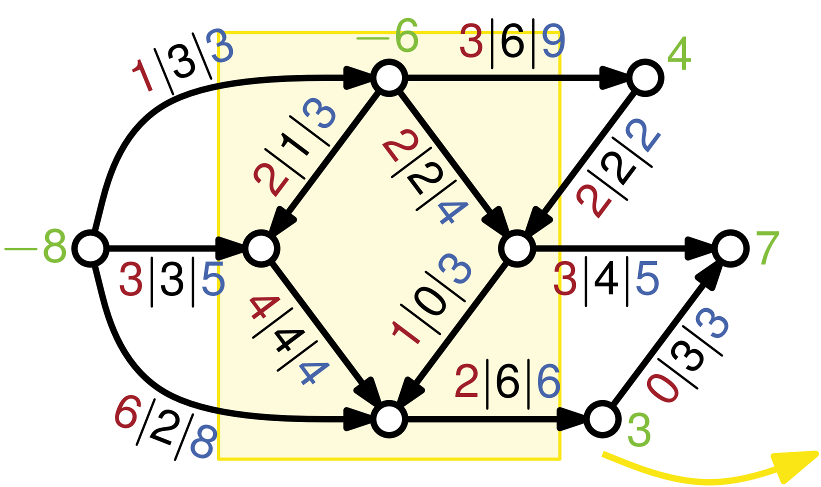
Schrittweise Verbesserung eines gültigen Flusses

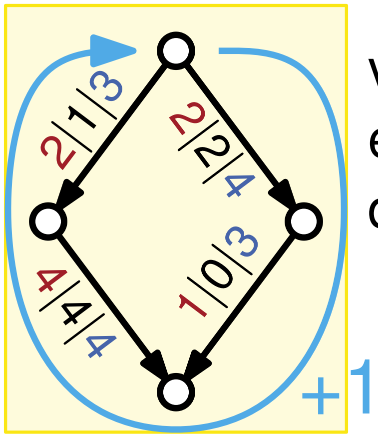
(1) Finde einen gültigen Fluss *f* im Flussnetzwerk *D* (falls es einen gibt).

fertig

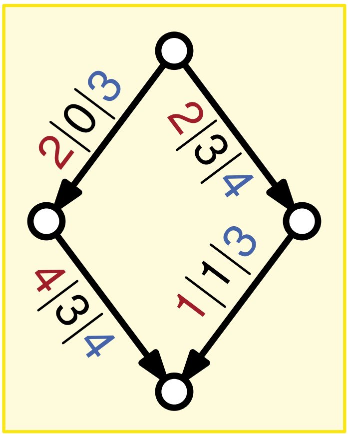
das kommt jetzt

(2) Verbessere *f* schrittweise.





verschiebe Fluss entlang „erhöhendem Kreis“



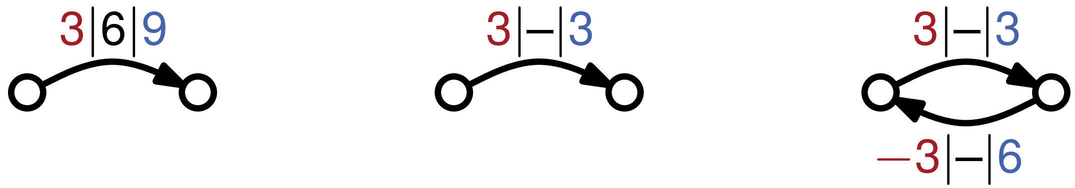
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Residualnetzwerk & Zirkulation

Definition: Residualnetzwerk

Sei $(D = (V, E), c, \mathrm{cost}, b)$ eine Instanz von MINCOSTFLOW und sei f ein Fluss in D. Das *Residualnetzwerk* $(D_f = (V, E_f), r_f, \mathrm{cost}_f, b_f)$ ist wie folgt definiert.

- Starte mit $D_f = D$ und setze die Kapazitäten auf $r_f(e) = c(e) - f(e)$.
- Für $e = (u, v) \in E$ füge Gegenkante $\bar{e} = (v, u)$ mit Kapazität $r_f(e') = f(e)$ ein.
- Für $e \in E$ setze $\mathrm{cost}_f(e) = \mathrm{cost}(e)$ und $\mathrm{cost}_f(\bar{e}) = -\mathrm{cost}(e)$
- Für $v \in V$ setze $b_f(v) = 0$



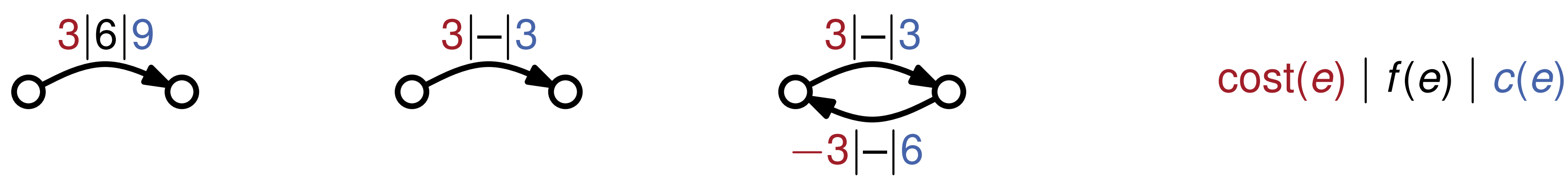
$\mathrm{cost}(e) \mid f(e) \mid c(e)$

Residualnetzwerk & Zirkulation

Definition: Residualnetzwerk

Sei $(D = (V, E), c, \mathrm{cost}, b)$ eine Instanz von MinCostFlow und sei f ein Fluss in D. Das *Residualnetzwerk* $(D_f = (V, E_f), r_f, \mathrm{cost}_f, b_f)$ ist wie folgt definiert.

- Starte mit $D_f = D$ und setze die Kapazitäten auf $r_f(e) = c(e) - f(e)$.
- Für $e = (u, v) \in E$ füge Gegenkante $\bar{e} = (v, u)$ mit Kapazität $r_f(e') = f(e)$ ein.
- Für $e \in E$ setze $\mathrm{cost}_f(e) = \mathrm{cost}(e)$ und $\mathrm{cost}_f(\bar{e}) = -\mathrm{cost}(e)$
- Für $v \in V$ setze $b_f(v) = 0$



Definition: Zirkulation

Ein Fluss im Residualnetzwerk D_f wird auch *Zirkulation* genannt.

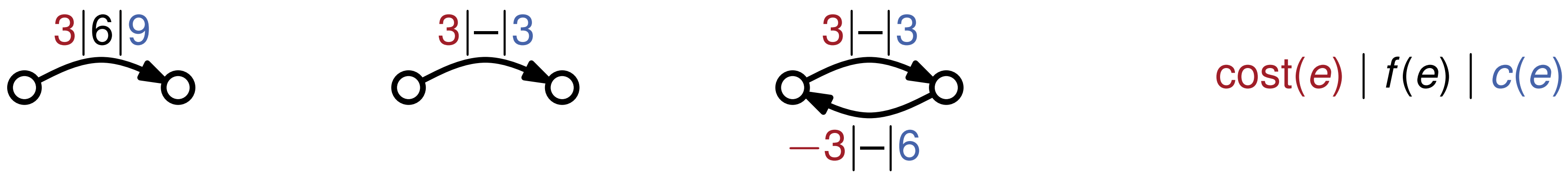
(Für jeden Knoten v gilt: eingehender Fluss = ausgehender Fluss, da $b_f(v) = 0$)

Residualnetzwerk & Zirkulation

Definition: Residualnetzwerk

Sei $(D = (V, E), c, \text{cost}, b)$ eine Instanz von MinCostFlow und sei f ein Fluss in D. Das *Residualnetzwerk* $(D_f = (V, E_f), r_f, \text{cost}_f, b_f)$ ist wie folgt definiert.

- Starte mit $D_f = D$ und setze die Kapazitäten auf $r_f(e) = c(e) - f(e)$.
- Für $e = (u, v) \in E$ füge Gegenkante $\bar{e} = (v, u)$ mit Kapazität $r_f(e') = f(e)$ ein.
- Für $e \in E$ setze $\text{cost}_f(e) = \text{cost}(e)$ und $\text{cost}_f(\bar{e}) = -\text{cost}(e)$
- Für $v \in V$ setze $b_f(v) = 0$



Definition: Zirkulation

Ein Fluss im Residualnetzwerk D_f wird auch *Zirkulation* genannt.

(Für jeden Knoten v gilt: eingehender Fluss = ausgehender Fluss, da $b_f(v) = 0$)

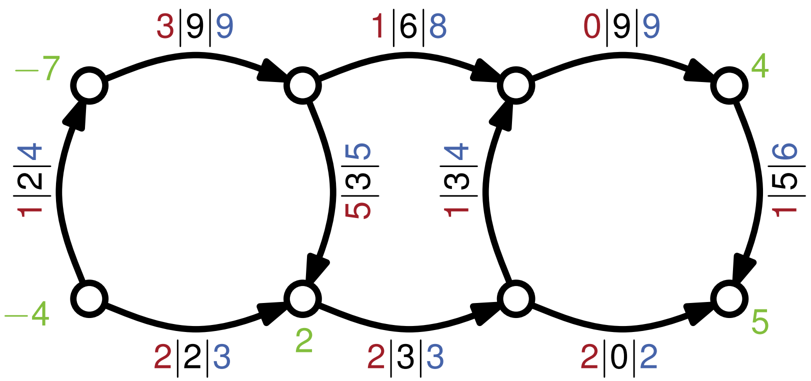
Lemma: erhöhende Zirkulation

Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D.

Erhöhende Zirkulation

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis" durch Beispiel:

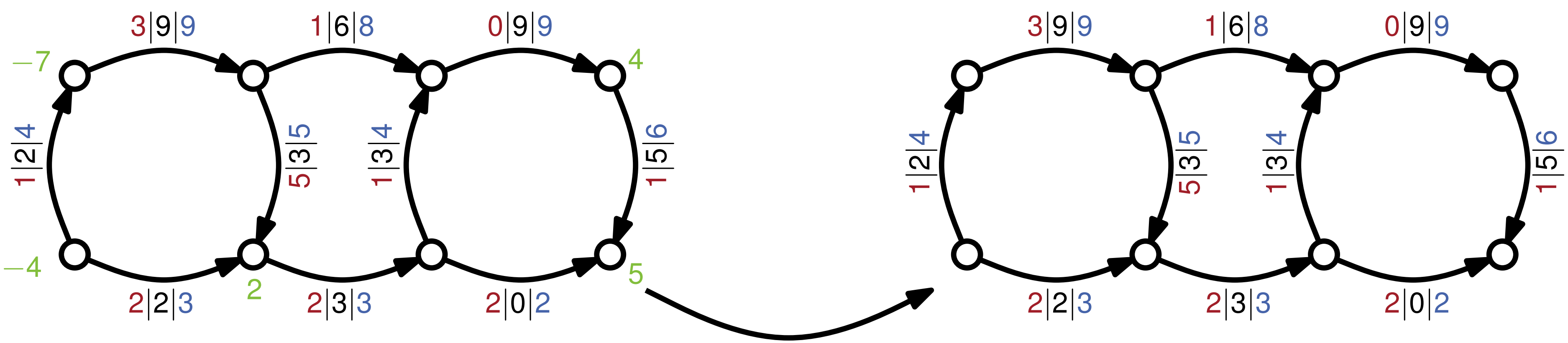


D mit Fluss f
$\text{cost}(f) = 68$

Erhöhende Zirkulation

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis“ durch Beispiel:



D mit Fluss f
$\text{cost}(f) = 68$

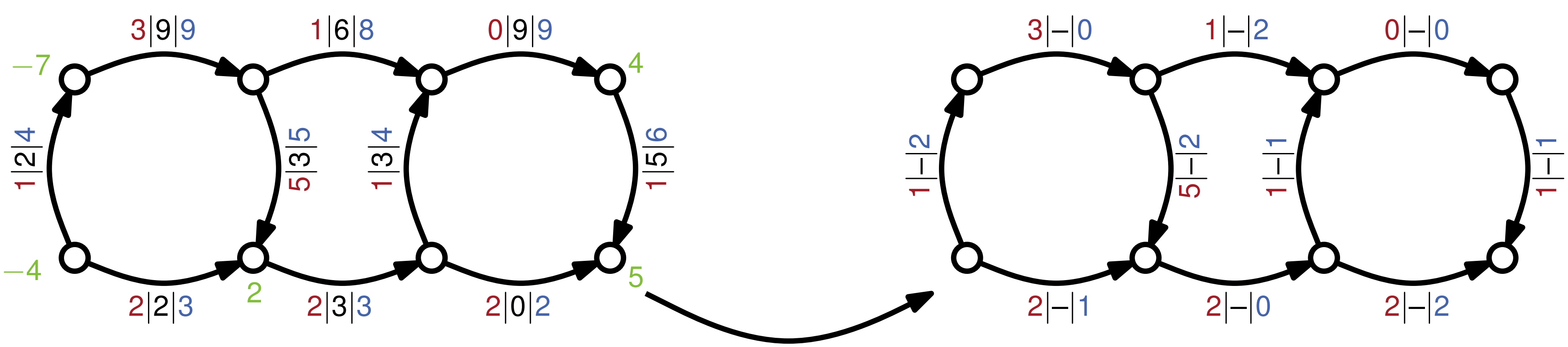
D_f erstellen:
1. D Kopieren

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Erhöhende Zirkulation

> **Lemma: erhöhende Zirkulation**
> Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis" durch Beispiel:



D mit Fluss f

$\text{cost}(f) = 68$

D_f erstellen:

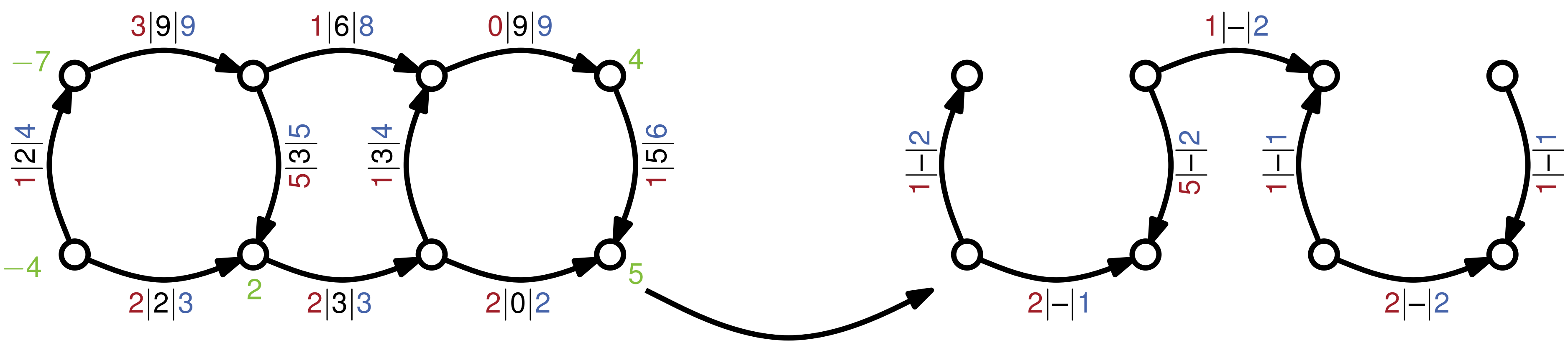
1. D Kopieren
2. Kapazitäten anpassen

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Erhöhende Zirkulation

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis“ durch Beispiel:



D mit Fluss f
$\text{cost}(f) = 68$

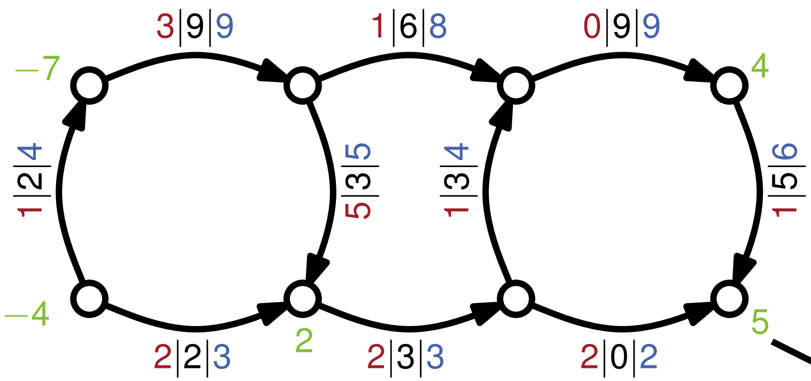
D_f erstellen:
1. D Kopieren
2. Kapazitäten anpassen

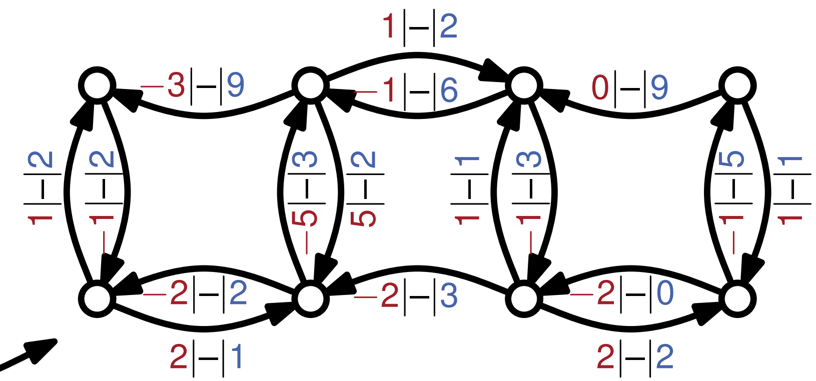
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Erhöhende Zirkulation

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis“ durch Beispiel:





D mit Fluss f
$\text{cost}(f) = 68$

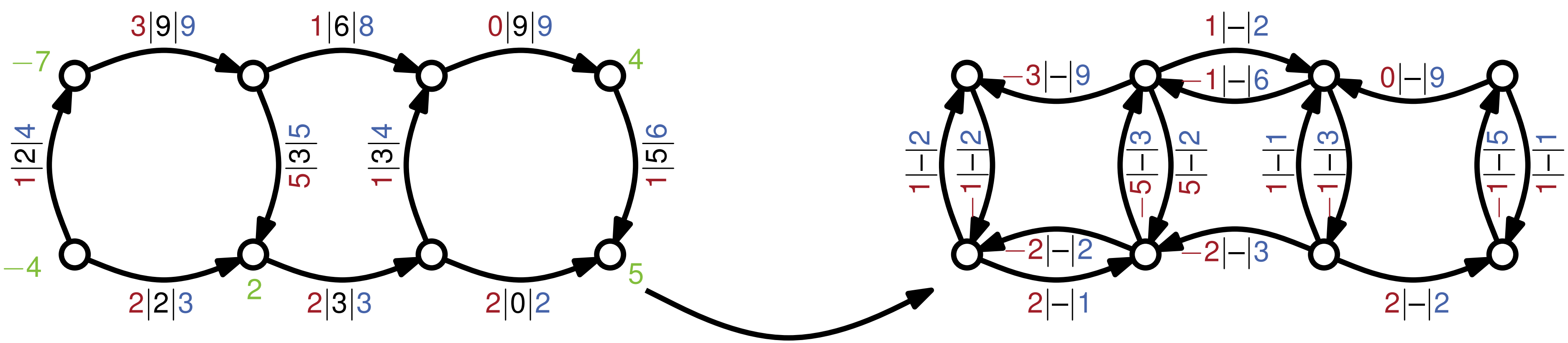
D_f erstellen:
1. D Kopieren
2. Kapazitäten anpassen
3. Gegenkanten einfügen

Residualnetzwerk D_f

Erhöhende Zirkulation

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis“ durch Beispiel:



D mit Fluss *f*
$\text{cost}(f) = 68$

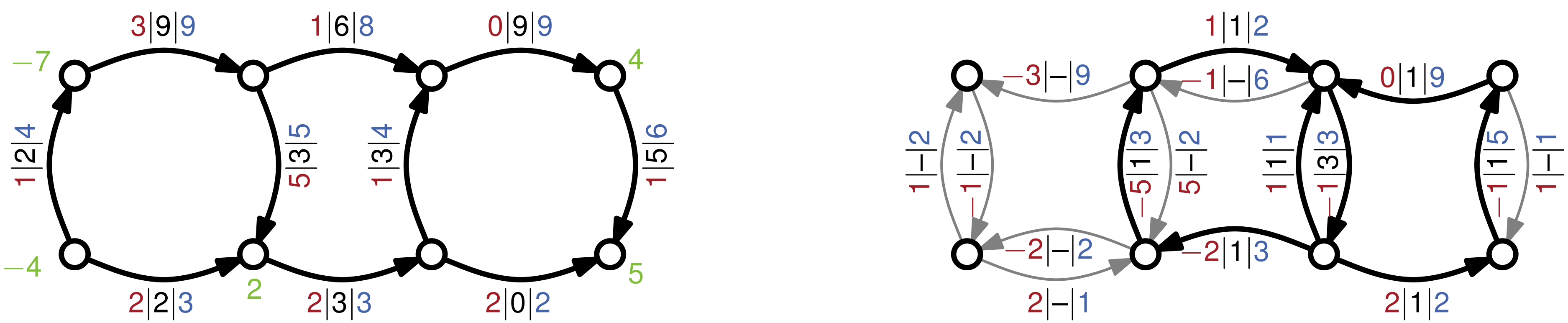
D_f erstellen:
1. *D* Kopieren
2. Kapazitäten anpassen
3. Gegenkanten einfügen

Residualnetzwerk D_f

Erhöhende Zirkulation

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis" durch Beispiel:



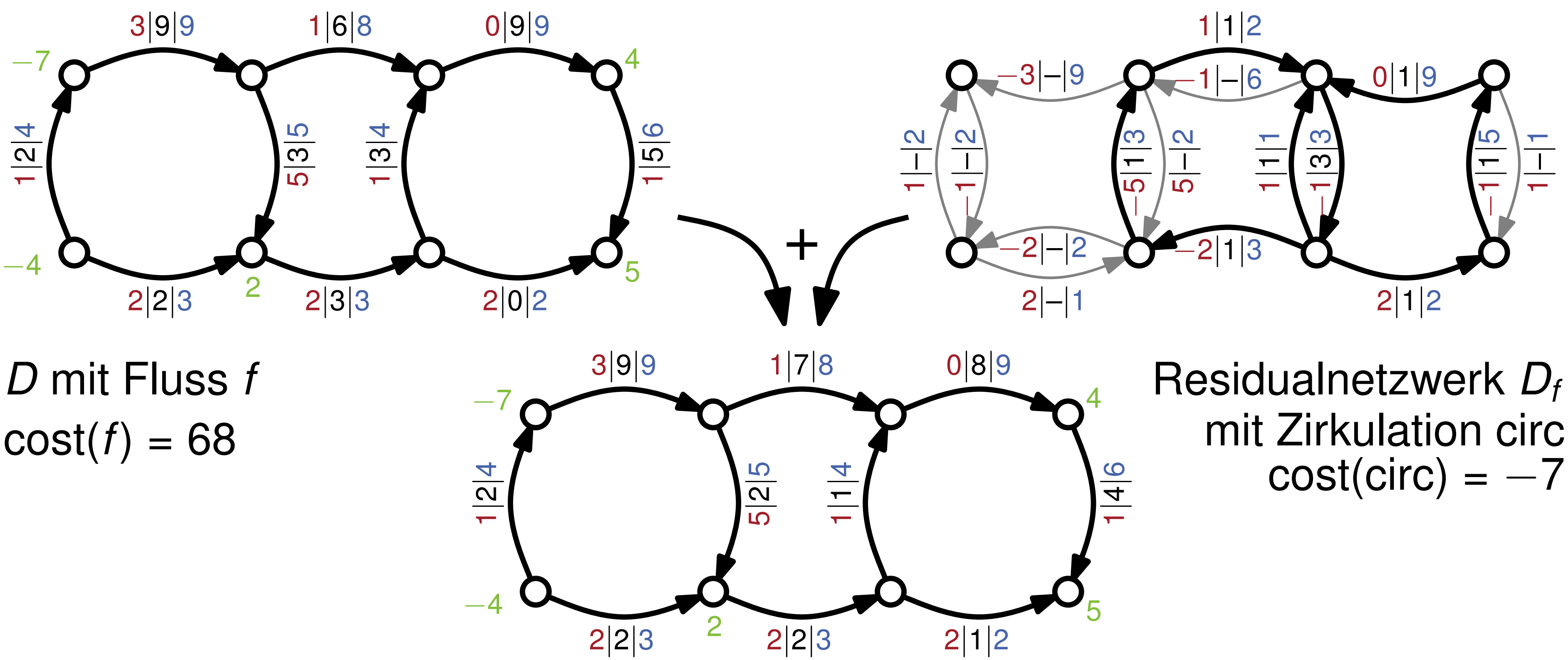
D mit Fluss f
$\text{cost}(f) = 68$

Residualnetzwerk D_f
mit Zirkulation circ
$\text{cost}(\text{circ}) = -7$

Erhöhende Zirkulation

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

„Beweis“ durch Beispiel:



D mit Fluss f
$\text{cost}(f) = 68$

Residualnetzwerk D_f
mit Zirkulation circ
$\text{cost}(\text{circ}) = -7$

D mit Fluss f + circ. Beachte: $\text{cost}(f + \text{circ}) = \text{cost}(f) + \text{cost}(\text{circ})$

Erhöhende Zirkulation & erhöhende Kreise

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \mathrm{circ}(e) - \mathrm{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\mathrm{cost}(f^\star) = \mathrm{cost}(f) + \mathrm{cost}(\mathrm{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \mathrm{circ}$.

Beweisidee:

Erhöhende Zirkulation & erhöhende Kreise

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \text{circ}$.

Beweisidee:

Betrachte alle Kanten $e \in E$ und definiere circ wie folgt:

- Falls $f^\star(e) - f(e) \geq 0$, setzte $\text{circ}(e) = f^\star(e) - f(e)$ und $\text{circ}(\bar{e}) = 0$
- Falls $f^\star(e) - f(e) < 0$, setzte $\text{circ}(e) = 0$ und $\text{circ}(\bar{e}) = -(f^\star(e) - f(e))$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Erhöhende Zirkulation & erhöhende Kreise

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \text{circ}$.

Beweisidee:

Betrachte alle Kanten $e \in E$ und definiere circ wie folgt:

- Falls $f^\star(e) - f(e) \geq 0$, setzte $\text{circ}(e) = f^\star(e) - f(e)$ und $\text{circ}(\bar{e}) = 0$
- Falls $f^\star(e) - f(e) < 0$, setzte $\text{circ}(e) = 0$ und $\text{circ}(\bar{e}) = -(f^\star(e) - f(e))$

Rechne nach, dass circ Zirkulation ist (Kapazitäts- & Flusserhaltungsbedingung).

Erhöhende Zirkulation & erhöhende Kreise

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \text{circ}$.

Man kann einen Fluss mit minimalen Kosten in D berechnen indem man einen gültigen Fluss f bestimmt und dann in D_f eine Zirkulation mit minimalen Kosten sucht.

Erhöhende Zirkulation & erhöhende Kreise

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \text{circ}$.

Definition: erhöhender Kreis
Ein *erhöhender Kreis* bezüglich eines Flusses f in D ist ein gerichteter Kreis C in D_f mit einer Zirkulation circ_C, sodass $\text{circ}_C(e) > 0$ falls $e \in C$, $\text{circ}_C(e) = 0$ sonst.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Erhöhende Zirkulation & erhöhende Kreise

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \text{circ}$.

Definition: erhöhender Kreis
Ein *erhöhender Kreis* bezüglich eines Flusses f in D ist ein gerichteter Kreis C in D_f mit einer Zirkulation circ_C, sodass $\text{circ}_C(e) > 0$ falls $e \in C$, $\text{circ}_C(e) = 0$ sonst.

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m $(= |E_f|)$ erhöhenden Kreisen.

Beweis: später

Erhöhende Zirkulation & erhöhende Kreise

Karlsruher Institut für Technologie

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \text{circ}$.

Definition: erhöhender Kreis
Ein *erhöhender Kreis* bezüglich eines Flusses f in D ist ein gerichteter Kreis C in D_f mit einer Zirkulation circ_C, sodass $\text{circ}_C(e) > 0$ falls $e \in C$, $\text{circ}_C(e) = 0$ sonst.

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m $(= |E_f|)$ erhöhenden Kreisen.

Folgerung: Optimalitätssatz vom erhöhenden Kreis
Ein Fluss f in D hat genau dann minimale Kosten, wenn D_f keinen erhöhenden Kreis mit negativen Kosten enthält.

Erhöhende Zirkulation & erhöhende Kreise

Lemma: erhöhende Zirkulation
Sei f ein Fluss in D und circ eine Zirkulation in D_f. Dann ist $f^\star$ mit $f^\star(e) = f(e) + \text{circ}(e) - \text{circ}(\bar{e})$ für alle $e \in E$ ein Fluss in D, mit $\text{cost}(f^\star) = \text{cost}(f) + \text{cost}(\text{circ})$.

Satz: erhöhende Zirkulation
Seien f und $f^\star$ Flüsse in D. Dann gibt es eine Zirkulation circ in D_f mit $f^\star = f + \text{circ}$.

Definition: erhöhender Kreis
Ein *erhöhender Kreis* bezüglich eines Flusses f in D ist ein gerichteter Kreis C in
D_f ... st.

Sa...
Je... n.

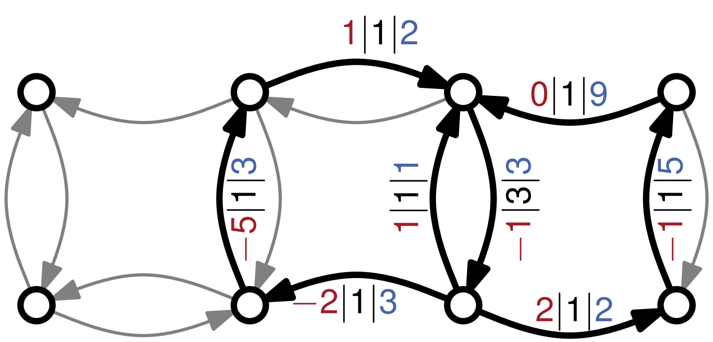
Cycle Canceling Algorithmus:
(1) Bestimme gültigen Fluss f.
(2) Solange D_f negative Kreise enthält, erhöhe f um negativen Kreis.

Folgerung: Optimalitätssatz vom erhöhenden Kreis
Ein Fluss f in D hat genau dann minimale Kosten, wenn D_f keinen erhöhenden Kreis mit negativen Kosten enthält.

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zerlegung in erhöhende Kreise – Beweis

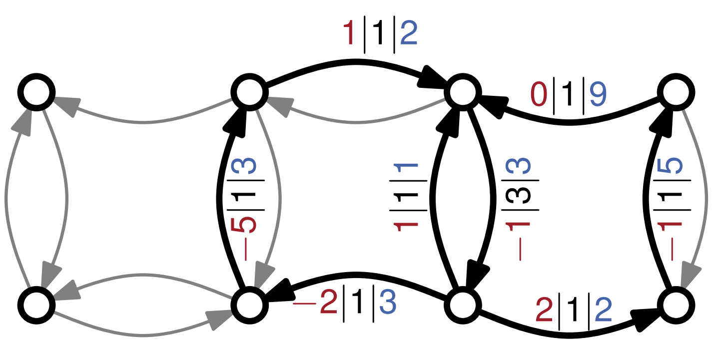
Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Alle Kanten, die wir mit erhöhenden Kreisen „abdecken“ müssen.

fett und schwarz



circ

Zerlegung in erhöhende Kreise – Beweis

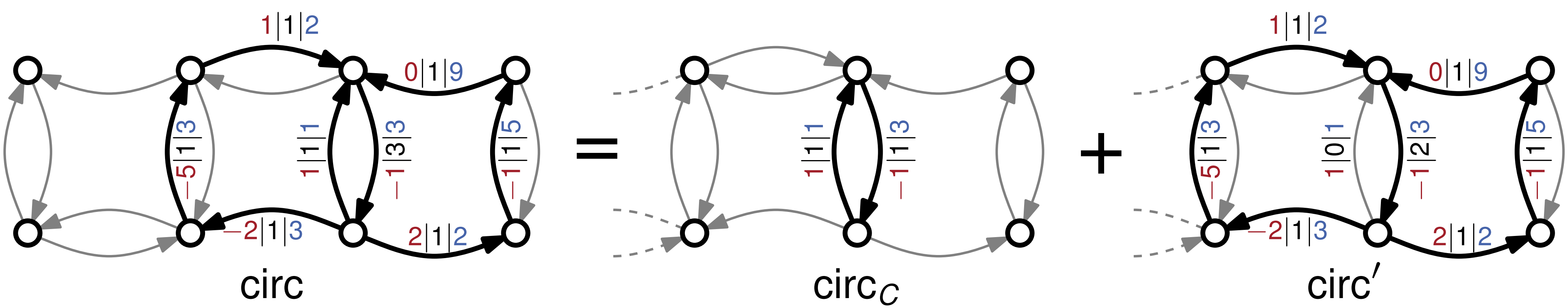
Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Zeige: Wenn $E(\text{circ}) \neq \emptyset$, dann gibt es erh. Kreis C mit Zirkulation circ_C sodass:

- $\text{circ} = \text{circ}_C + \text{circ}'$ (für eine andere Zirkulation circ' in D_f)
- $|E(\text{circ}')| \leq |E(\text{circ})| - 1$



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m ($= |E_f|$) erhöhenden Kreisen.

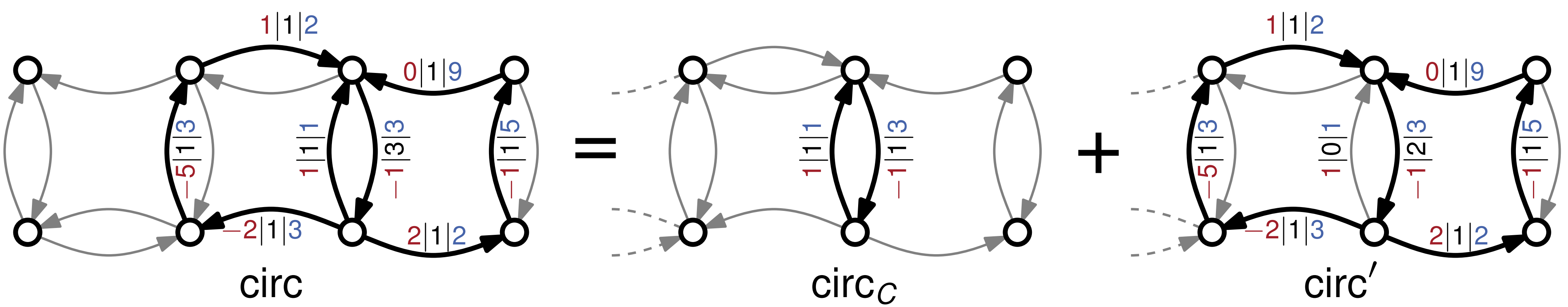
Beweis:

Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Zeige: Wenn $E(\text{circ}) \neq \emptyset$, dann gibt es erh. Kreis C mit Zirkulation circ_C sodass:

- $\text{circ} = \text{circ}_C + \text{circ}'$ (für eine andere Zirkulation circ' in D_f)
- $|E(\text{circ}')| \leq |E(\text{circ})| - 1$

Setzt man diese Zerlegung mit circ' fort, so ist nach spätestens m Schritten kein Fluss mehr übrig. Die Summe der zur Zerlegung genutzten erhöhenden Kreise (circ_C) ist dann gerade circ.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

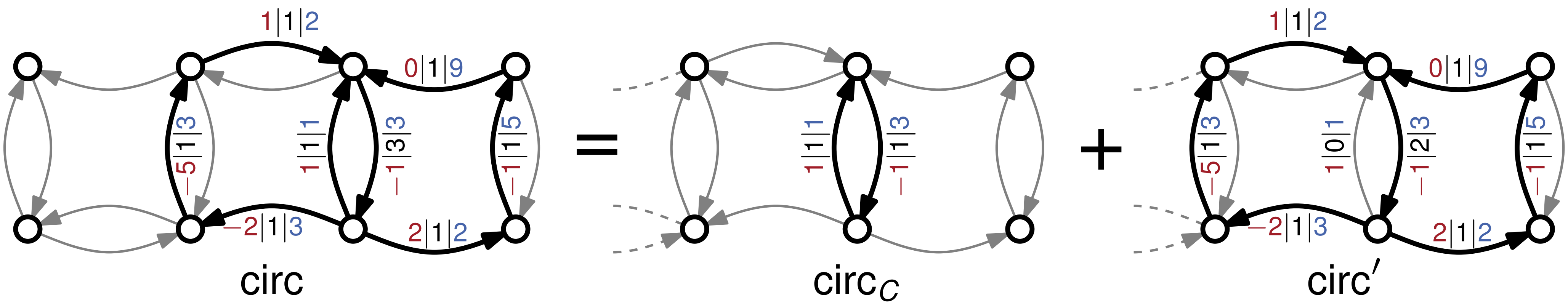
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Zeige: Wenn $E(\text{circ}) \neq \emptyset$, dann gibt es erh. Kreis C mit Zirkulation circ_C sodass:

- $\text{circ} = \text{circ}_C + \text{circ}'$ (für eine andere Zirkulation circ' in D_f)
- $|E(\text{circ}')| \leq |E(\text{circ})| - 1$

Existenz dieses erhöhenden Kreises:

1. Finde gerichteten Kreis C der nur aus Kanten in $E(\text{circ})$ besteht.
2. Sei $e_{\min}$ die Kante in C für die $\text{circ}(e_{\min})$ minimal ist. Setzte $\text{circ}_C(e) = \text{circ}(e_{\min})$ für alle Kanten e in C.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

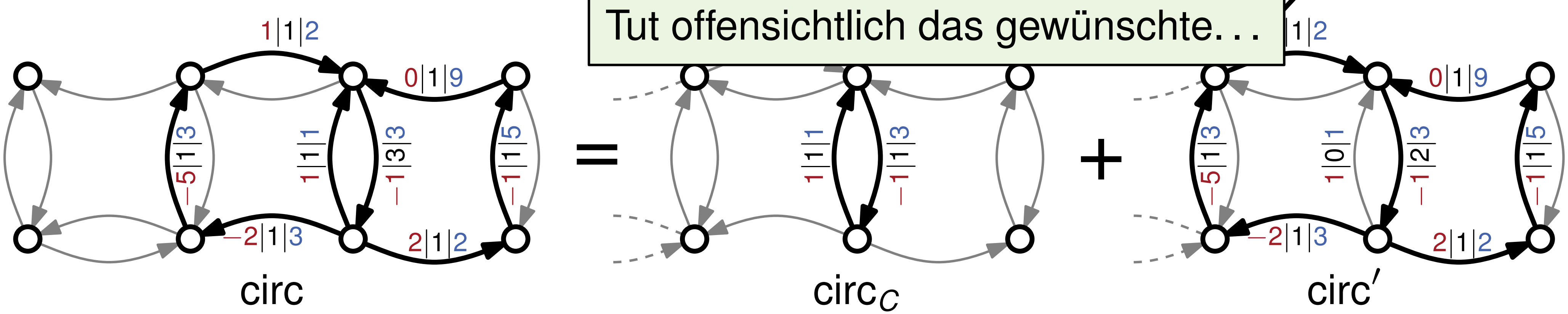
Zeige: Wenn $E(\text{circ}) \neq \emptyset$, dann gibt es erh. Kreis C mit Zirkulation circ_C sodass:

- $\text{circ} = \text{circ}_C + \text{circ}'$ (für eine andere Zirkulation circ' in D_f)
- $|E(\text{circ}')| \leq |E(\text{circ})| - 1$

Existenz dieses erhöhenden Kreises:

1. Finde gerichteten Kreis C der nur aus Kanten in $E(\text{circ})$ besteht.
2. Sei $e_{\min}$ die Kante in C für die $\text{circ}(e_{\min})$ minimal ist. Setzte $\text{circ}_C(e) = \text{circ}(e_{\min})$ für alle Kanten e in C.

Tut offensichtlich das gewünschte...



Zerlegung in erhöhende Kreise – Beweis

Karlsruher Institut für Technologie

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Zeige: Wenn $E(\text{circ}) \neq \emptyset$, dann gibt es erh. Kreis C mit Zirkulation circ_C sodass:

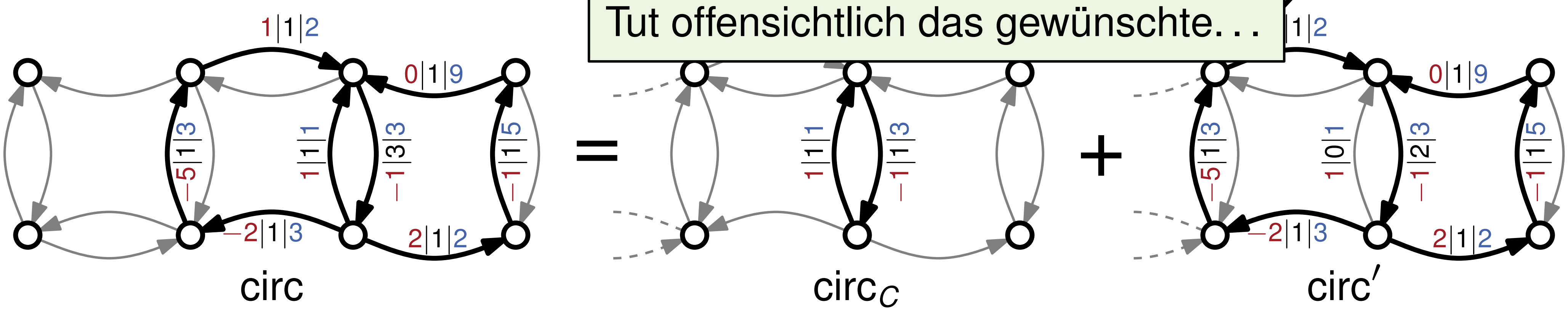
- $\text{circ} = \text{circ}_C + \text{circ}'$ (für eine andere Zirkulation circ' in D_f)
- $|E(\text{circ}')| \leq |E(\text{circ})| - 1$

. . . falls C existiert.

Existenz dieses erhöhenden Kreises:

1. Finde gerichteten Kreis C der nur aus Kanten in $E(\text{circ})$ besteht.
2. Sei $e_{\min}$ die Kante in C für die $\text{circ}(e_{\min})$ minimal ist. Setzte $\text{circ}_C(e) = \text{circ}(e_{\min})$ für alle Kanten e in C.

Tut offensichtlich das gewünschte. . .



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

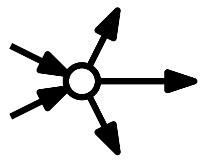
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m ($= |E_f|$) erhöhenden Kreisen.

Beweis:

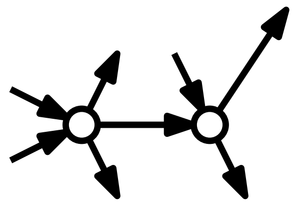
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m $(= |E_f|)$ erhöhenden Kreisen.

Beweis:

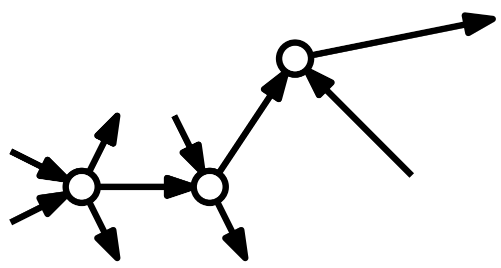
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m $(= |E_f|)$ erhöhenden Kreisen.

Beweis:

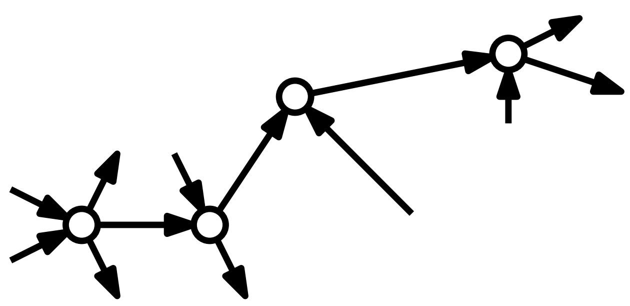
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



Zerlegung in erhöhende Kreise – Beweis

Karlsruher Institut für Technologie

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

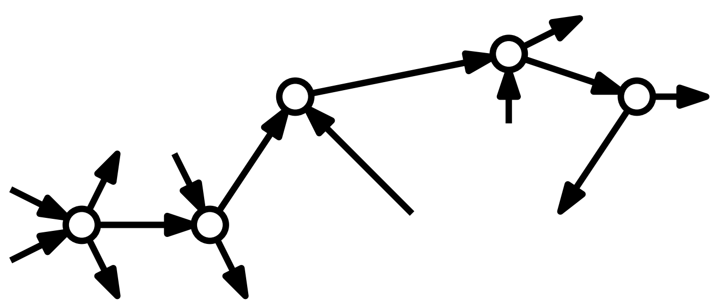
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

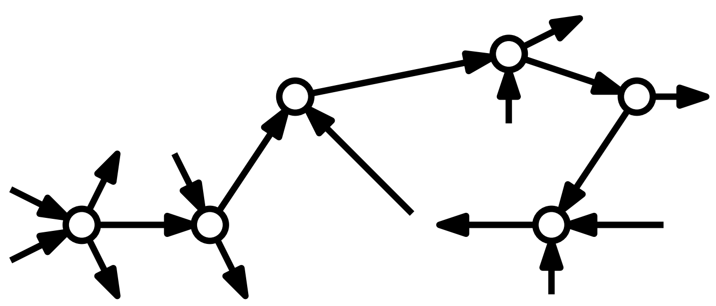
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m (= $|E_f|$) erhöhenden Kreisen.

Beweis:

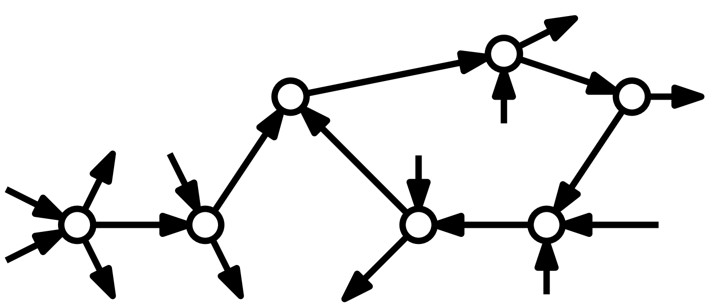
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



Zerlegung in erhöhende Kreise – Beweis

Satz: Zerlegung in erhöhende Kreise
Jede Zirkulation in D_f ist die Summe von maximal m $(= |E_f|)$ erhöhenden Kreisen.

Beweis:

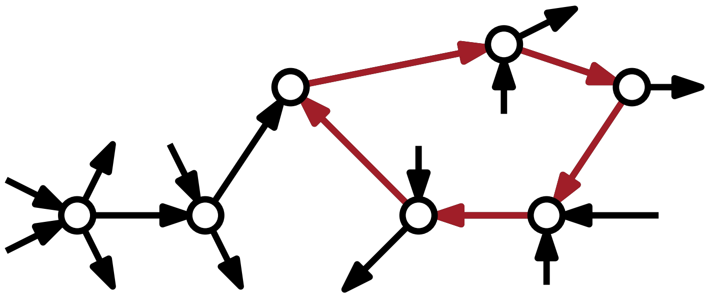
Gegeben eine Zirkulation circ in D_f. Sei $E(\text{circ}) = \{e \mid e \in E_f, \text{circ}(e) > 0\}$.

Noch zu zeigen: D_f enthält einen Kreis C der nur Kanten aus $E(\text{circ})$ benutzt.

Beobachtung: Wenn $v \in V$ eine eingehende Kante in $E(\text{circ})$ hat, dann auch eine ausgehende.

Strategie:

- Starte bei einem Knoten mit inzidenten Kanten in $E(\text{circ})$.
- Laufe auf ausgehenden Kanten weiter, bis schon besuchter Knoten erreicht wird. $\Rightarrow$ Kreis C gefunden.



$\square$

Cycle Canceling Algorithmus

Karlsruher Institut für Technologie

$\textsc{CycleCanceling}(D, c, \text{cost}, b)$

$f \leftarrow$ gültiger Fluss in D

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f

while D_f *enthält erhöhenden Kreis mit negativen Kosten* **do**

- $(C, \text{circ}_C) \leftarrow$ erhöhender Kreis mit negativen Kosten
- $f \leftarrow f + \text{circ}_C$
- $D_f \leftarrow$ Residualnetzwerk von D bezüglich f

Cycle Canceling Algorithmus

CycleCanceling(D, c, cost, b)

$f \leftarrow$ gültiger Fluss in D $\quad O(nm\log(n^2/m))$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f

while D_f *enthält erhöhenden Kreis mit negativen Kosten* **do**

  $(C, \text{circ}_C) \leftarrow$ erhöhender Kreis mit negativen Kosten

  $f \leftarrow f + \text{circ}_C$

  $D_f \leftarrow$ Residualnetzwerk von D bezüglich f

Berechnung eines maximalen Flusses
z.B. Goldberg & Tarjan

Cycle Canceling Algorithmus

CycleCanceling(D, c, cost, b)

$f \leftarrow$ gültiger Fluss in D $\quad O(nm\log(n^2/m))$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f $\quad O(m)$

while D_f *enthält erhöhenden Kreis mit negativen Kosten* **do**

$\quad (C, \text{circ}_C) \leftarrow$ erhöhender Kreis mit negativen Kosten

$\quad f \leftarrow f + \text{circ}_C$

$\quad D_f \leftarrow$ Residualnetzwerk von D bezüglich f

Berechnung eines maximalen Flusses
z.B. Goldberg & Tarjan

Cycle Canceling Algorithmus

CycleCanceling(D, c, cost, b)

$f \leftarrow$ gültiger Fluss in D — $O(nm \log(n^2/m))$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f — $O(m)$

while D_f *enthält erhöhenden Kreis mit negativen Kosten* **do**

- (C, circ$_C$) $\leftarrow$ erhöhender Kreis mit negativen Kosten — $O(nm)$
- $f \leftarrow f + \mathrm{circ}_C$
- $D_f \leftarrow$ Residualnetzwerk von D bezüglich f

Benutze kürzeste Wege Algorithmus
z.B. Bellman & Ford

Berechnung eines maximalen Flusses
z.B. Goldberg & Tarjan

Cycle Canceling Algorithmus

Karlsruher Institut für Technologie

$\textsc{CycleCanceling}(D, c, \text{cost}, b)$

$f \leftarrow$ gültiger Fluss in D — $O(nm\log(n^2/m))$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f — $O(m)$

while D_f *enthält erhöhenden Kreis mit negativen Kosten* **do**

$(C, \text{circ}_C) \leftarrow$ erhöhender Kreis mit negativen Kosten — $O(nm)$

$f \leftarrow f + \text{circ}_C$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f — $O(m)$

Benutze kürzeste Wege Algorithmus
z.B. Bellman & Ford

Berechnung eines maximalen Flusses
z.B. Goldberg & Tarjan

Cycle Canceling Algorithmus

CYCLECANCELING(D, c, cost, b)

$f \leftarrow$ gültiger Fluss in D — $O(nm \log(n^2/m))$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f — $O(m)$

while *D_f enthält erhöhenden Kreis mit negativen Kosten* **do**

- $(C, \mathrm{circ}_C) \leftarrow$ erhöhender Kreis mit negativen Kosten — $O(nm)$
- $f \leftarrow f + \mathrm{circ}_C$
- $D_f \leftarrow$ Residualnetzwerk von D bezüglich f — $O(m)$

$O(nm \cdot mc_{\max} \, \mathrm{cost}_{\max})$

Benutze kürzeste Wege Algorithmus
z.B. Bellman & Ford

Berechnung eines maximalen Flusses
z.B. Goldberg & Tarjan

$O(mc_{\max} \, \mathrm{cost}_{\max})$ Schleifendurchläufe
Grund: In jedem Schritt sinken die Kosten um ≥ 1 und es gilt:
$|\mathrm{cost}(f)| \leq mc_{\max} \, \mathrm{cost}_{\max}$
Kapazitäten und Bedarfe müssen ganzzahlig sein!

$c_{\max}$ / $\mathrm{cost}_{\max}$: maximale Kapazität / betragsmäßig maximale Kosten

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Cycle Canceling Algorithmus

$\textsc{CycleCanceling}(D, c, \text{cost}, b)$ $O(nm \cdot mc_{\max} \text{cost}_{\max})$

$f \leftarrow$ gültiger Fluss in D $O(nm \log(n^2/m))$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f $O(m)$

while D_f *enthält erhöhenden Kreis mit negativen Kosten* **do**

$(C, \text{circ}_C) \leftarrow$ erhöhender Kreis mit negativen Kosten $O(nm)$

$f \leftarrow f + \text{circ}_C$

$D_f \leftarrow$ Residualnetzwerk von D bezüglich f $O(m)$

$O(nm \cdot mc_{\max} \text{cost}_{\max})$

Benutze kürzeste Wege Algorithmus
z.B. Bellman & Ford

Berechnung eines maximalen Flusses
z.B. Goldberg & Tarjan

$O(mc_{\max} \text{cost}_{\max})$ Schleifendurchläufe
Grund: In jedem Schritt sinken die Kosten um ≥ 1 und es gilt:
$|\text{cost}(f)| \leq mc_{\max} \text{cost}_{\max}$
Kapazitäten und Bedarfe müssen ganzzahlig sein!

$c_{\max}$ / $\text{cost}_{\max}$: maximale Kapazität / betragsmäßig maximale Kosten

Cycle Canceling Algorithmus

Karlsruher Institut für Technologie

$\textsc{CycleCanceling}(D, c, \text{cost}, b)$ — $O(nm \cdot mc_{\max} \,\text{cost}_{\max})$

- $f \leftarrow$ gültiger Fluss in D — $O(nm \log(n^2/m))$
- $D_f \leftarrow$ Residualnetzwerk von D bezüglich f — $O(m)$
- **while** D_f *enthält erhöhenden Kreis mit negativen Kosten* **do**
  - $(C, \text{circ}_C) \leftarrow$ erhöhender Kreis mit negativen Kosten — $O(nm)$
  - $f \leftarrow f + \text{circ}_C$
  - $D_f \leftarrow$ Residualnetzwerk von D bezüglich f — $O(m)$
  - $O(nm \cdot mc_{\max} \,\text{cost}_{\max})$

Satz: Cycle Canceling Algorithmus

Sei (D, c, cost, b) ein Flussnetzwerk mit ganzzahligen Kapazitäten und Bedarfen. Der Algorithmus $\textsc{CycleCanceling}$ berechnet Fluss mit minimalen Kosten in $O(nm^2 c_{\max} \,\text{cost}_{\max})$ Zeit.

Bemerkung: Es gibt diverse Algorithmen zur Lösung von $\textsc{MinCostFlow}$, sowohl pseudopolynomielle wie $\textsc{CycleCanceling}$ als auch streng-polynomielle.

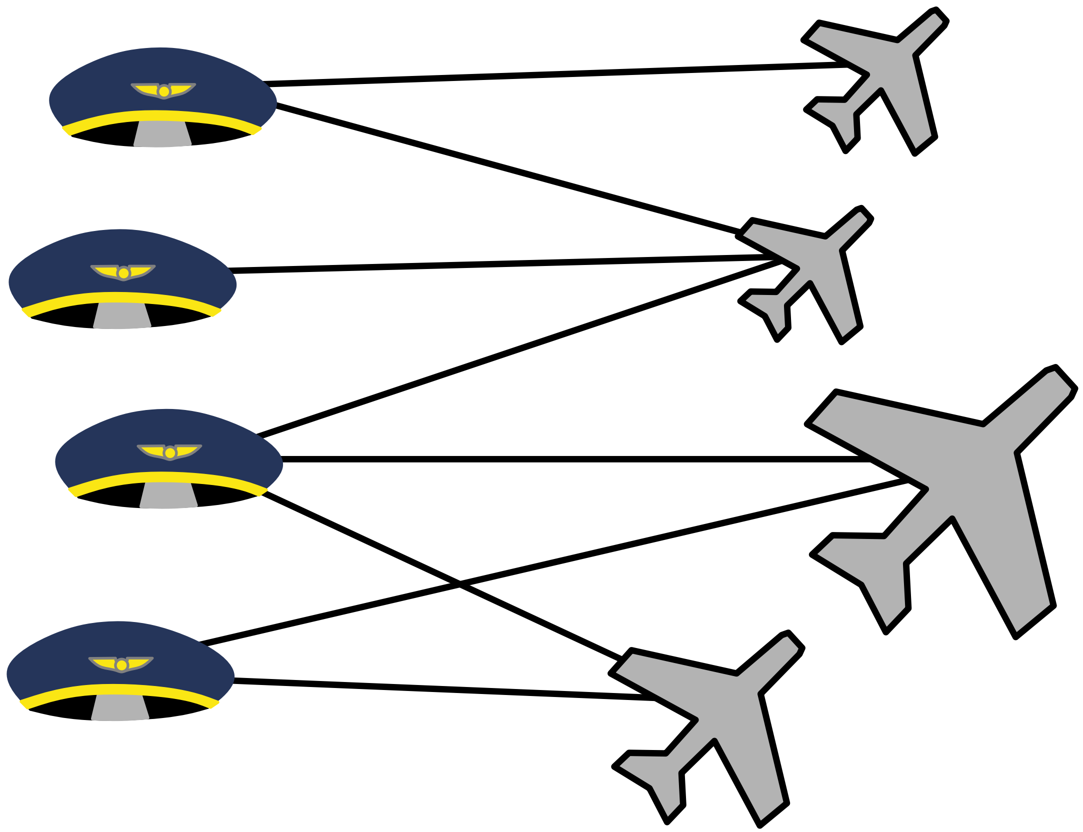
Beispiel: Der Algorithmus von Orlin hat eine Laufzeit von $O(m \log n(m + n \log n))$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

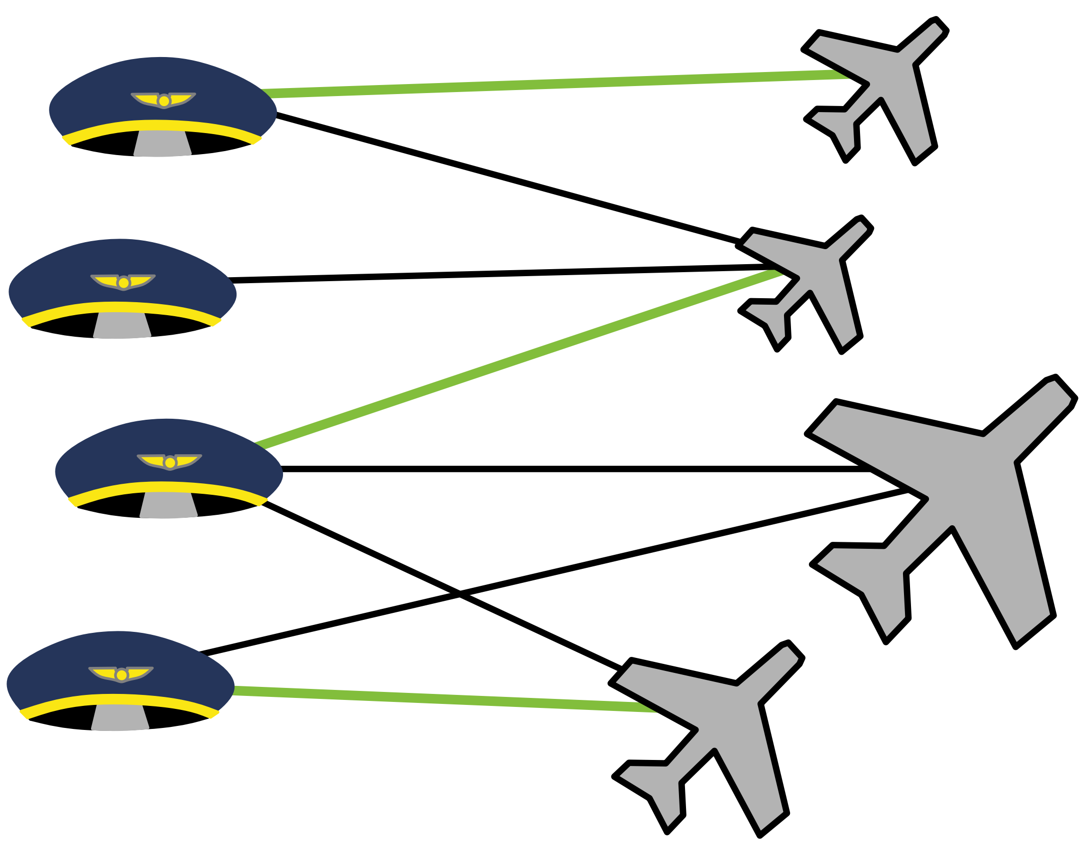
Matchings

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Motivation – Zuordnung von Piloten zu Flügen

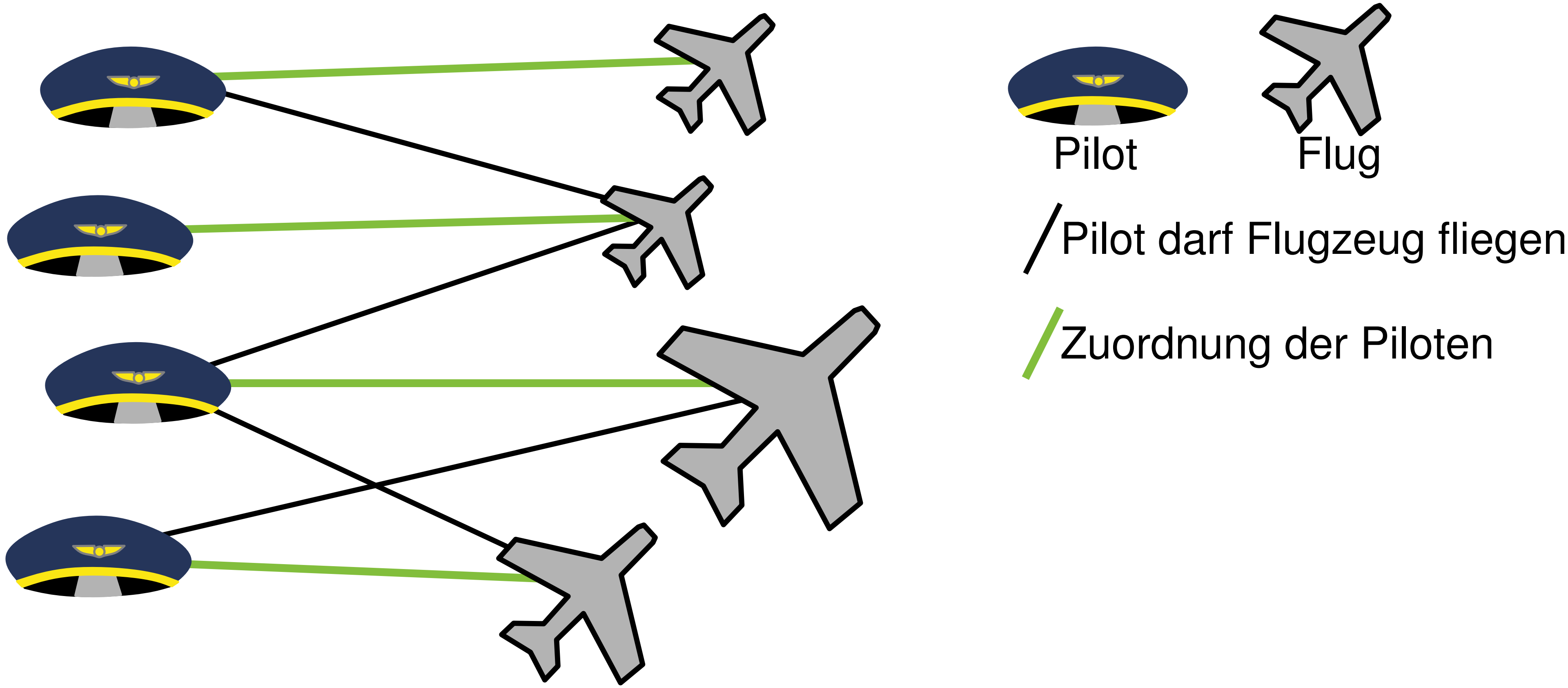


Motivation – Zuordnung von Piloten zu Flügen

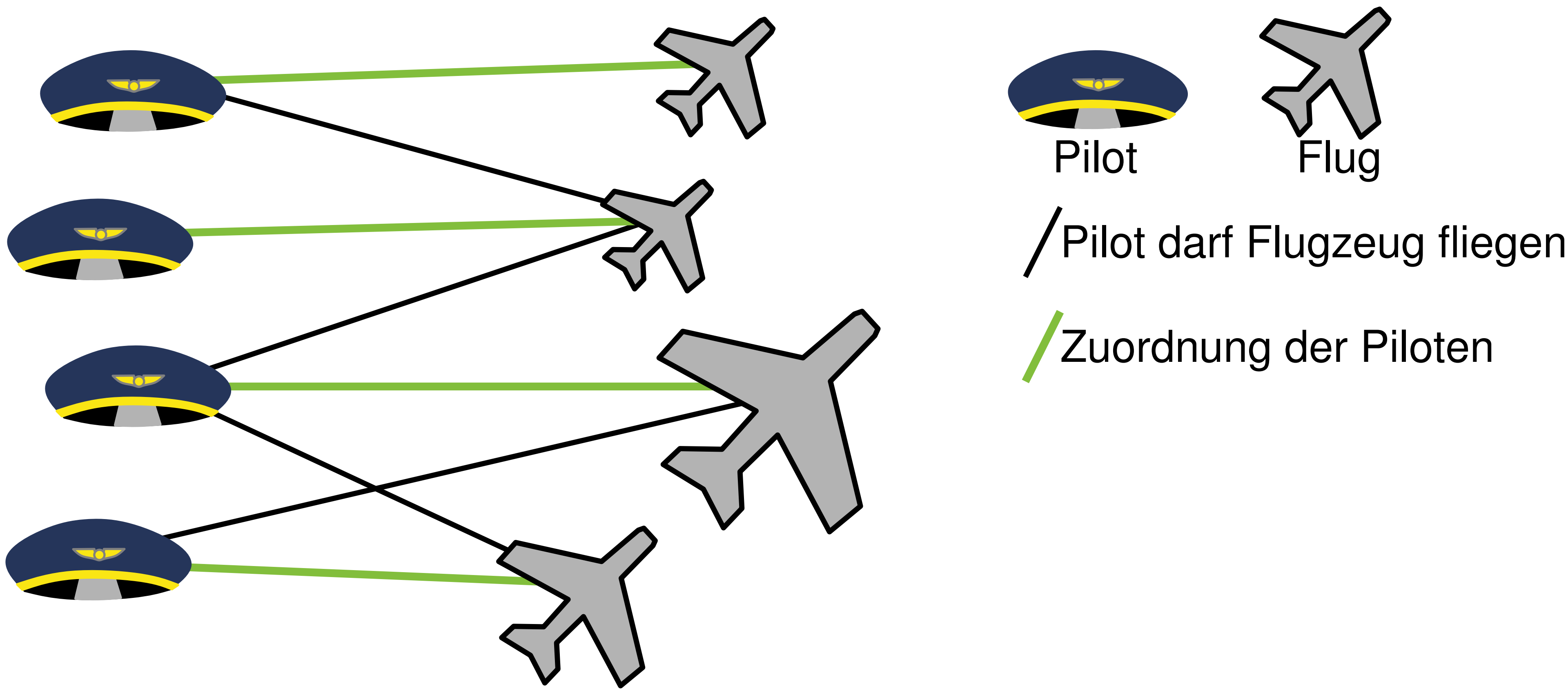


Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Motivation – Zuordnung von Piloten zu Flügen



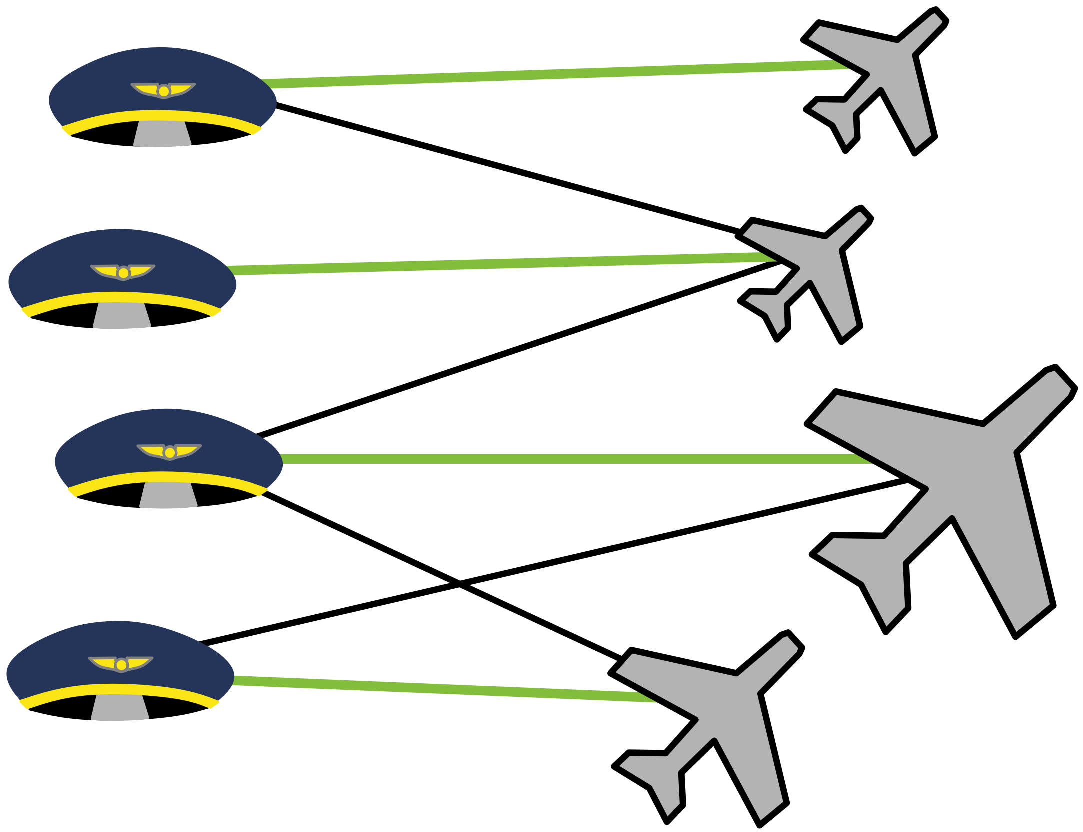
Motivation – Zuordnung von Piloten zu Flügen



Ziel: Finde eine Zuordnung, sodass:

- Jeder Pilot maximal ein Flugzeug fliegt, jedes Flugzeug von maximal einem Pilot geflogen wird.
- Möglichst viele Flugzeuge besetzt (bzw. Piloten beschäftigt) sind.

Motivation – Zuordnung von Piloten zu Flügen



maximales Matching

Ziel: Finde eine Zuordnung, sodass:

- Jeder Pilot maximal ein Flugzeug fliegt, jedes Flugzeug von maximal einem Pilot geflogen wird.
- Möglichst viele Flugzeuge besetzt (bzw. Piloten beschäftigt) sind.

Formale Definition

Definition: Matching
Sei $G = (V, E)$ ein Graph. Ein *Matching* in G ist eine Teilmenge $M \subseteq E$, sodass jeder Knoten zu maximal einer Kante in M inzident ist.

Problem: MAXIMALES MATCHING
Finde ein möglichst großes Matching M in G. (d. h. $|M| \geq |M'|$ für alle Matchings M')

Bemerkung: Man könnte auch jeder Kante ein Gewicht zuweisen und dann die Summe der Gewichte gewählter Kanten maximieren.

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Formale Definition

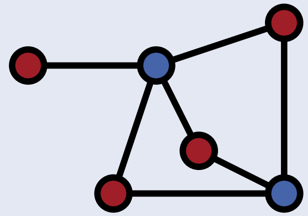
Definition: Matching
Sei $G = (V, E)$ ein Graph. Ein *Matching* in G ist eine Teilmenge $M \subseteq E$, sodass jeder Knoten zu maximal einer Kante in M inzident ist.

Problem: MAXIMALES MATCHING
Finde ein möglichst großes Matching M in G. (d. h. $|M| \geq |M'|$ für alle Matchings M')

Bemerkung: Man könnte auch jeder Kante ein Gewicht zuweisen und dann die Summe der Gewichte gewählter Kanten maximieren.

Einschränkung: Wir nehmen im Folgenden an, dass G *bipartit* ist.

Erinnerung: bipartite Graphen
Ein Graph $G = (R \cup B, E)$ ist *bipartit*, wenn jede Kante in E einen Knoten aus R mit einem Knoten aus B verbindet.



Formale Definition

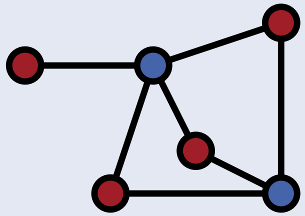
Definition: Matching
Sei $G = (V, E)$ ein Graph. Ein *Matching* in G ist eine Teilmenge $M \subseteq E$, sodass jeder Knoten zu maximal einer Kante in M inzident ist.

Problem: MAXIMALES MATCHING
Finde ein möglichst großes Matching M in G. (d. h. $|M| \geq |M'|$ für alle Matchings M')

Bemerkung: Man könnte auch jeder Kante ein Gewicht zuweisen und dann die Summe der Gewichte gewählter Kanten maximieren.

Einschränkung: Wir nehmen im Folgenden an, dass G *bipartit* ist.

Erinnerung: bipartite Graphen
Ein Graph $G = (R \cup B, E)$ ist *bipartit*, wenn jede Kante in E einen Knoten aus R mit einem Knoten aus B verbindet.



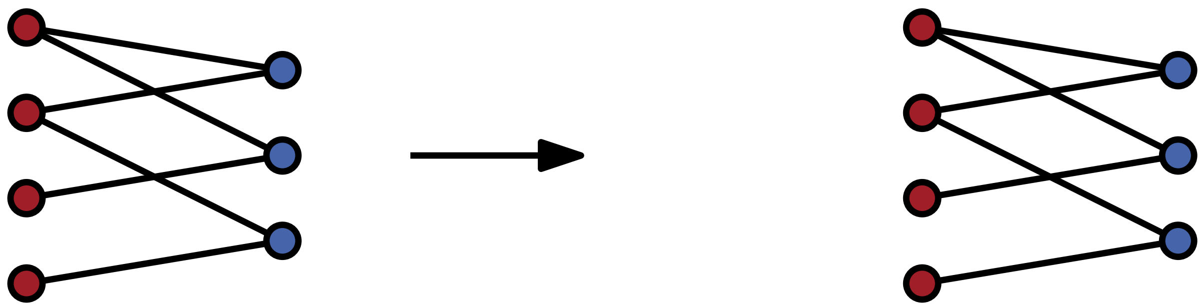
Problem: MAXIMALES BIPARTITES MATCHING
Finde ein möglichst großes Matching M in einem bipartiten Graphen G.

Lösungsansatz mittels Flussberechnung

Definition: zugehöriges Flussnetzwerk

Sei $G = (V = R \cup B, E)$ ein bipartiter Graph. Das *zugehörige Flussnetzwerk* $G = (V', E', c, s, t)$ ist wie folgt definiert.

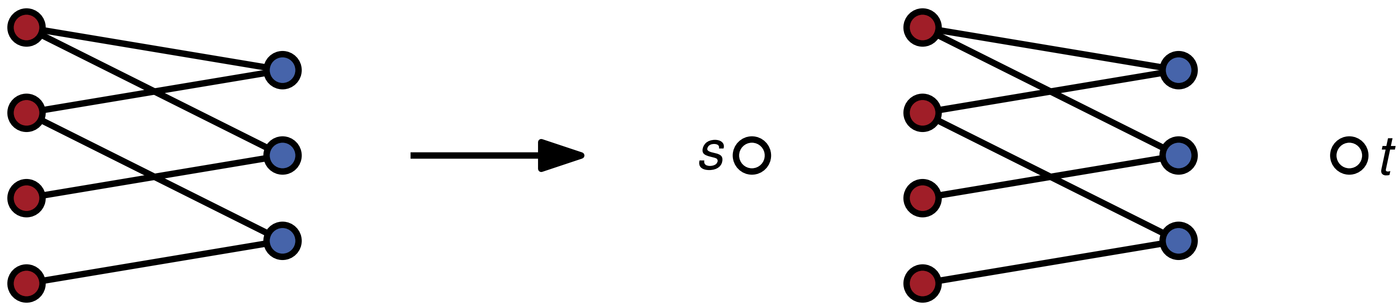
- $V' = V \cup \{s, t\}$
- $(r, b) \in E'$ für jede Kante $\{r, b\} \in E$ mit $r \in R$, $b \in B$.
- $(s, r) \in E'$ für alle Knoten $r \in R$ und $(b, t) \in E'$ für alle Knoten $b \in B$.
- $c(e) = 1$ für alle Kanten $e \in E'$.



Lösungsansatz mittels Flussberechnung

Definition: zugehöriges Flussnetzwerk
Sei $G = (V = R \cup B, E)$ ein bipartiter Graph. Das *zugehörige Flussnetzwerk* $G = (V', E', c, s, t)$ ist wie folgt definiert.

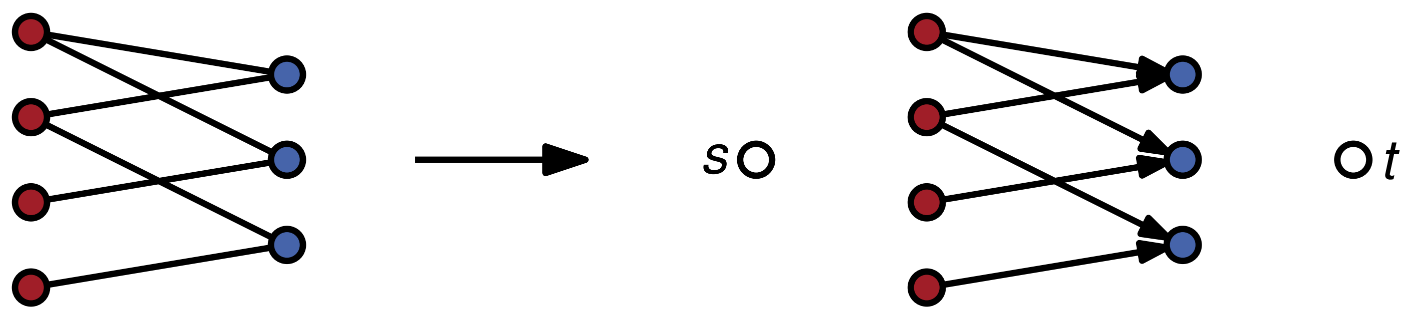
- $V' = V \cup \{s, t\}$
- $(r, b) \in E'$ für jede Kante $\{r, b\} \in E$ mit $r \in R$, $b \in B$.
- $(s, r) \in E'$ für alle Knoten $r \in R$ und $(b, t) \in E'$ für alle Knoten $b \in B$.
- $c(e) = 1$ für alle Kanten $e \in E'$.



Lösungsansatz mittels Flussberechnung

Definition: zugehöriges Flussnetzwerk
Sei $G = (V = R \cup B, E)$ ein bipartiter Graph. Das *zugehörige Flussnetzwerk* $G = (V', E', c, s, t)$ ist wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $(r, b) \in E'$ für jede Kante $\{r, b\} \in E$ mit $r \in R$, $b \in B$.
- $(s, r) \in E'$ für alle Knoten $r \in R$ und $(b, t) \in E'$ für alle Knoten $b \in B$.
- $c(e) = 1$ für alle Kanten $e \in E'$.

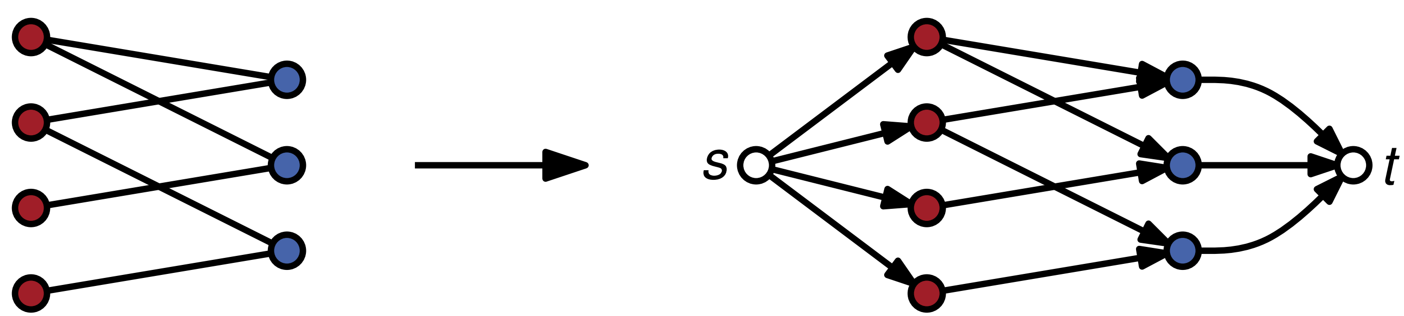


Lösungsansatz mittels Flussberechnung

Definition: zugehöriges Flussnetzwerk

Sei $G = (V = R \cup B, E)$ ein bipartiter Graph. Das *zugehörige Flussnetzwerk* $G = (V', E', c, s, t)$ ist wie folgt definiert.

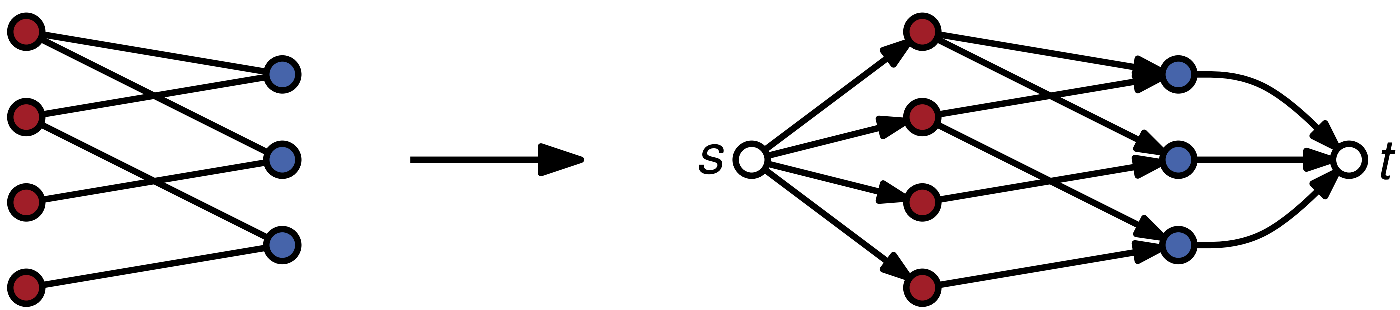
- $V' = V \cup \{s, t\}$
- $(r, b) \in E'$ für jede Kante $\{r, b\} \in E$ mit $r \in R$, $b \in B$.
- $(s, r) \in E'$ für alle Knoten $r \in R$ und $(b, t) \in E'$ für alle Knoten $b \in B$.
- $c(e) = 1$ für alle Kanten $e \in E'$.



Lösungsansatz mittels Flussberechnung

Definition: zugehöriges Flussnetzwerk
Sei $G = (V = R \cup B, E)$ ein bipartiter Graph. Das *zugehörige Flussnetzwerk* $G = (V', E', c, s, t)$ ist wie folgt definiert.

- $V' = V \cup \{s, t\}$
- $(r, b) \in E'$ für jede Kante $\{r, b\} \in E$ mit $r \in R$, $b \in B$.
- $(s, r) \in E'$ für alle Knoten $r \in R$ und $(b, t) \in E'$ für alle Knoten $b \in B$.
- $c(e) = 1$ für alle Kanten $e \in E'$.



Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

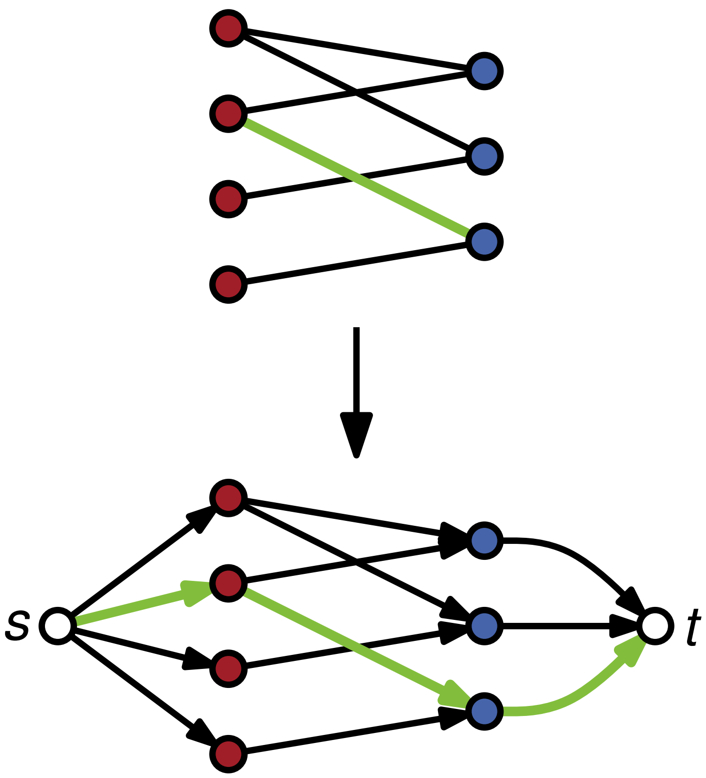
Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

Definiere f wie folgt:

- Für $\{r, b\} \in M$ setze $f(s, r) = f(r, b) = f(b, t) = 1$.
- Für alle anderen Kanten $e \in E'$ setzte $f(e) = 0$.



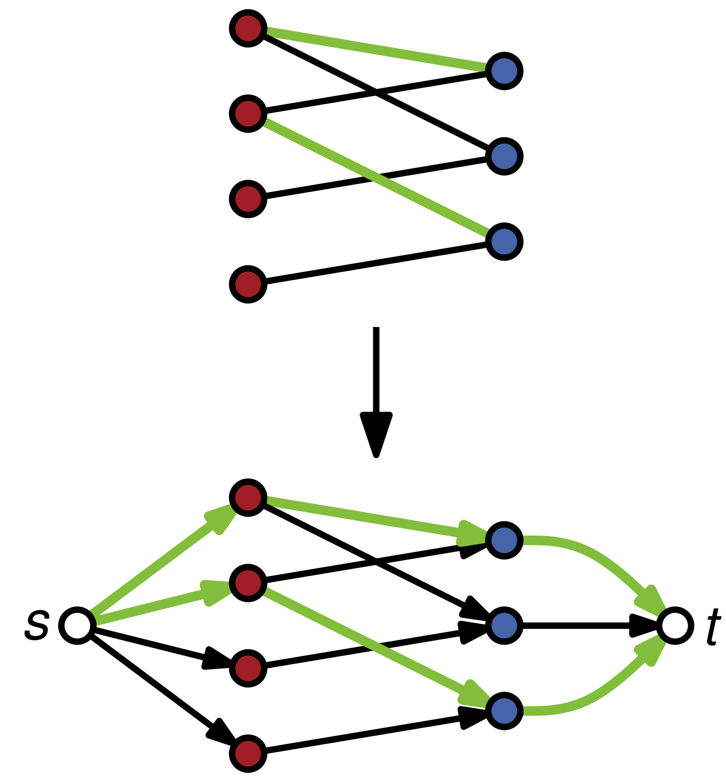
Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

Definiere f wie folgt:

- Für $\{r, b\} \in M$ setze $f(s, r) = f(r, b) = f(b, t) = 1$.
- Für alle anderen Kanten $e \in E'$ setzte $f(e) = 0$.



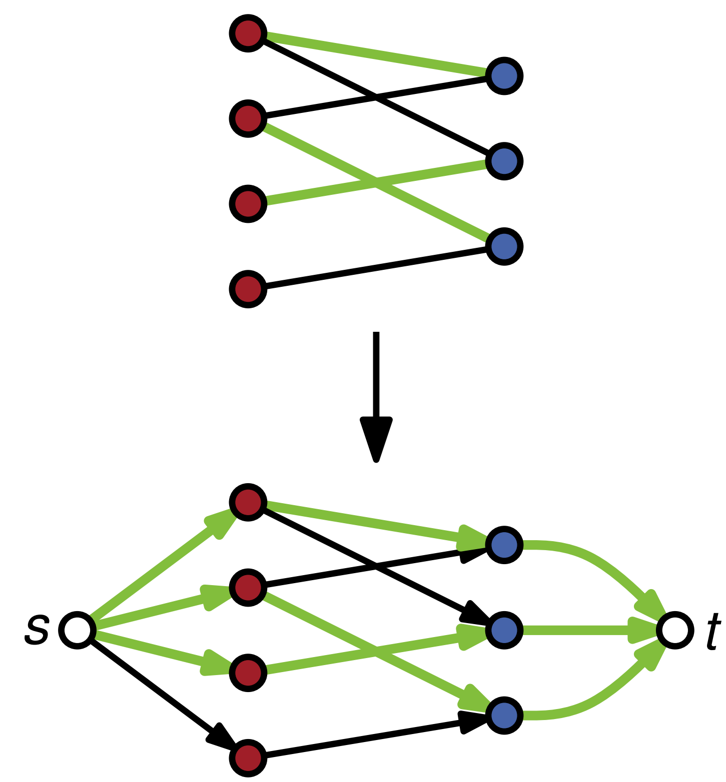
Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

Definiere f wie folgt:

- Für $\{r, b\} \in M$ setze $f(s, r) = f(r, b) = f(b, t) = 1$.
- Für alle anderen Kanten $e \in E'$ setzte $f(e) = 0$.



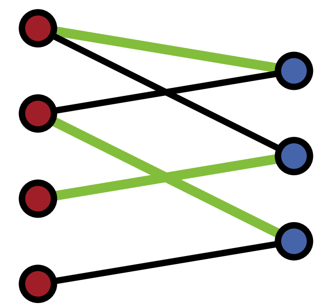
Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

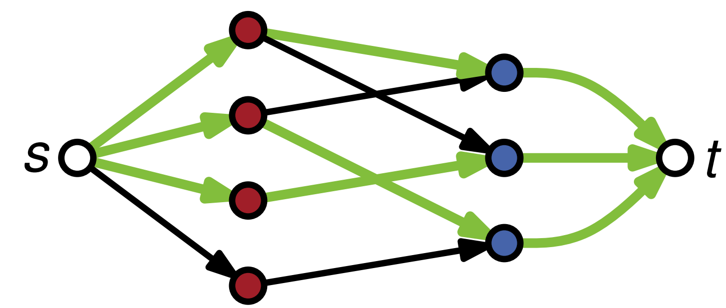
Definiere f wie folgt:

- Für $\{r, b\} \in M$ setze $f(s, r) = f(r, b) = f(b, t) = 1$.
- Für alle anderen Kanten $e \in E'$ setzte $f(e) = 0$.



Die Abbildung f tut das richtige, denn:

- Kapazitäts- und Flusserhaltungsbedingung sind erfüllt $\Rightarrow f$ ist ein Fluss
- f ist ganzzahlig
- $w(f) = |M|$



Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$.
(Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

Für alle Kanten $e \in E'$ gilt $c(e) = 1$. Daher folgt aus der ganzzahligkeit von f dass $f(e) = 0$ oder $f(e) = 1$ gilt.

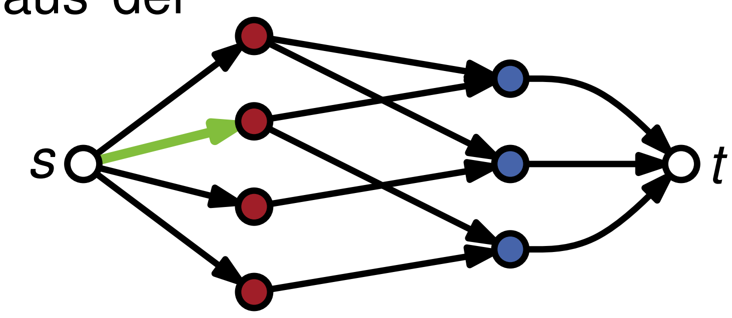
Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$.
(Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

Für alle Kanten $e \in E'$ gilt $c(e) = 1$. Daher folgt aus der ganzzahligkeit von f dass $f(e) = 0$ oder $f(e) = 1$ gilt.

Sei (s, r) eine Kante mit $f(s, r) = 1$.



Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$.
(Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

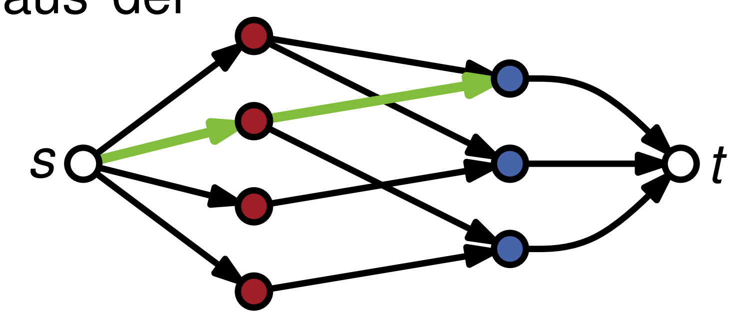
Beweis:

Für alle Kanten $e \in E'$ gilt $c(e) = 1$. Daher folgt aus der ganzzahligkeit von f dass $f(e) = 0$ oder $f(e) = 1$ gilt.

Sei (s, r) eine Kante mit $f(s, r) = 1$.

$\Rightarrow$ r hat einen Nachbar b mit $f(r, b) = 1$,
für alle anderen Nachbarn b' gilt $f(r, b') = 0$

Flusserhaltung



Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

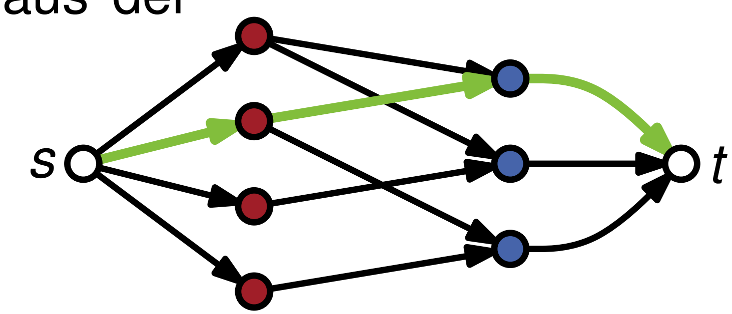
Äquivalenz des Flussnetzwerks – Beweis

Karlsruher Institut für Technologie

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

Für alle Kanten $e \in E'$ gilt $c(e) = 1$. Daher folgt aus der ganzzahligkeit von f dass $f(e) = 0$ oder $f(e) = 1$ gilt.



Sei (s, r) eine Kante mit $f(s, r) = 1$.

$\Rightarrow$ r hat einen Nachbar b mit $f(r, b) = 1$,
für alle anderen Nachbarn b' gilt $f(r, b') = 0$
$\Rightarrow$ $f(b, t) = 1$

Flusserhaltung

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Äquivalenz des Flussnetzwerks – Beweis

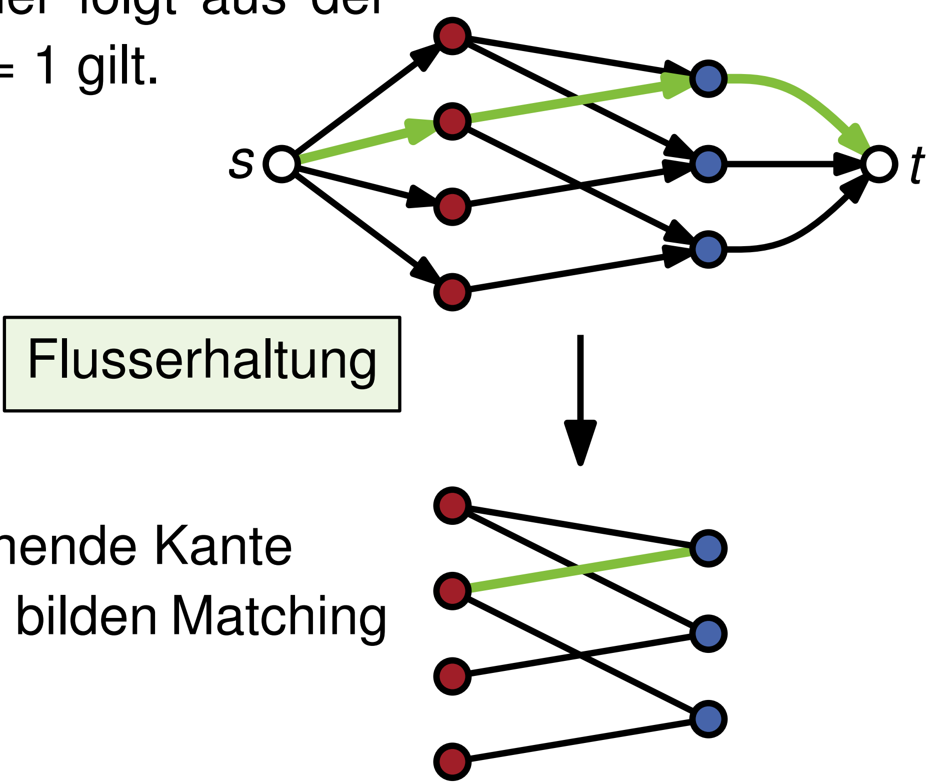
Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

Für alle Kanten $e \in E'$ gilt $c(e) = 1$. Daher folgt aus der ganzzahligkeit von f dass $f(e) = 0$ oder $f(e) = 1$ gilt.

Sei (s, r) eine Kante mit $f(s, r) = 1$.

$\Rightarrow$ r hat einen Nachbar b mit $f(r, b) = 1$,
für alle anderen Nachbarn b' gilt $f(r, b') = 0$
$\Rightarrow$ $f(b, t) = 1$



Jeder Knoten in R/B hat nur eine ein-/ausgehende Kante
$\Rightarrow$ Kanten zwischen R und B mit Fluss 1 in G' bilden Matching M in G mit $|M| = w(f)$.

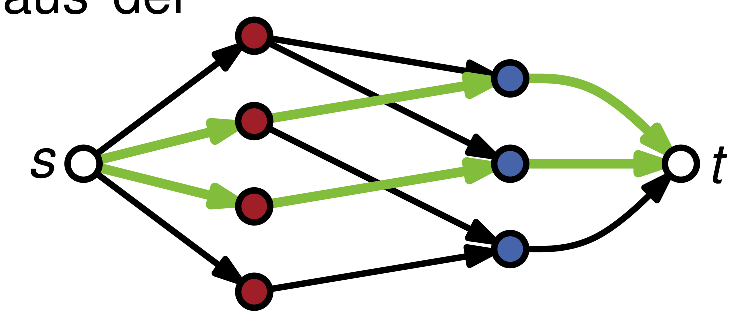
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$.
(Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

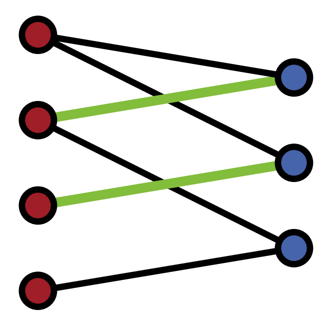
Für alle Kanten $e \in E'$ gilt $c(e) = 1$. Daher folgt aus der ganzzahligkeit von f dass $f(e) = 0$ oder $f(e) = 1$ gilt.



Sei (s, r) eine Kante mit $f(s, r) = 1$.

$\Rightarrow$ r hat einen Nachbar b mit $f(r, b) = 1$,
für alle anderen Nachbarn b' gilt $f(r, b') = 0$
$\Rightarrow f(b, t) = 1$

Flusserhaltung



Jeder Knoten in R/B hat nur eine ein-/ausgehende Kante
$\Rightarrow$ Kanten zwischen R und B mit Fluss 1 in G' bilden Matching M in G mit $|M| = w(f)$.

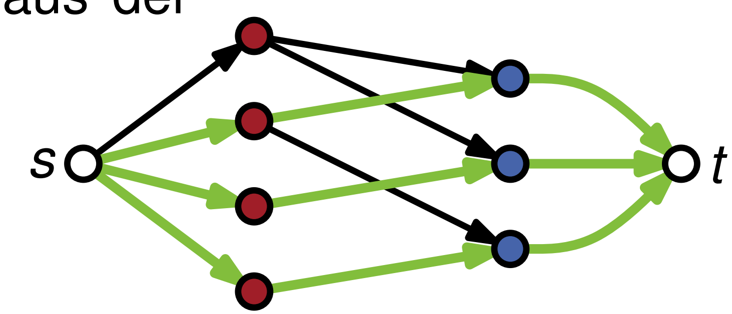
Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Äquivalenz des Flussnetzwerks – Beweis

Lemma: Äquivalenz des zugehörigen Flussnetzwerks
Sei M ein Matching in G. Dann gibt es einen ganzzahligen Fluss f in G' mit Wert $w(f) = |M|$. Sei f ein ganzzahliger Fluss in G', dann gibt es in G ein Matching M mit $|M| = w(f)$. (Erinnerung: $w(f)$ ist der bei s ausgehende Fluss)

Beweis:

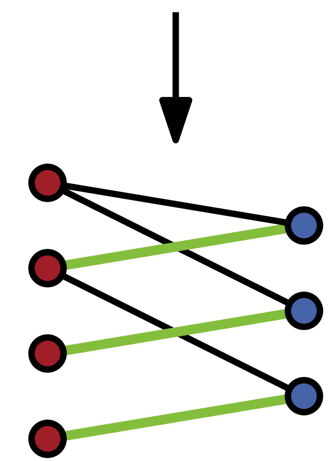
Für alle Kanten $e \in E'$ gilt $c(e) = 1$. Daher folgt aus der ganzzahligkeit von f dass $f(e) = 0$ oder $f(e) = 1$ gilt.



Sei (s, r) eine Kante mit $f(s, r) = 1$.

$\Rightarrow$ r hat einen Nachbar b mit $f(r, b) = 1$,
für alle anderen Nachbarn b' gilt $f(r, b') = 0$
$\Rightarrow$ $f(b, t) = 1$

Flusserhaltung



Jeder Knoten in R/B hat nur eine ein-/ausgehende Kante
$\Rightarrow$ Kanten zwischen R und B mit Fluss 1 in G' bilden Matching M in G mit $|M| = w(f)$.

$\square$

Institut für Theoretische Informatik
Prof. Dr. Dorothea Wagner

Der Algorithmus

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

Der Algorithmus

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING$(G = (R \cup B, E))$

$(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G

$f \leftarrow$ 0-Fluss

while *Residualnetzwerk von f enthält st-Weg* **do**

  $f \leftarrow f$ erhöht um st-Weg

$M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$

Der Algorithmus

Karlsruher Institut für Technologie

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING$(G = (R \cup B, E))$

$(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G $\quad O(m)$

$f \leftarrow$ 0-Fluss

while *Residualnetzwerk von f enthält st-Weg* **do**

  $f \leftarrow f$ erhöht um st-Weg

$M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$

Der Algorithmus

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING($G = (R \cup B, E)$)

$(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G	$O(m)$
$f \leftarrow$ 0-Fluss	$O(m)$

while *Residualnetzwerk von f enthält st-Weg* **do**

  $f \leftarrow f$ erhöht um st-Weg

$M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$

Der Algorithmus

Karlsruher Institut für Technologie

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING($G = (R \cup B, E)$)

$(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G $\quad O(m)$

$f \leftarrow$ 0-Fluss $\quad O(m)$

while *Residualnetzwerk von f enthält st-Weg* **do**

  $f \leftarrow f$ erhöht um st-Weg $\quad O(m)$

$M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$

Der Algorithmus

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING$(G = (R \cup B, E))$

$(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G	$O(m)$
$f \leftarrow$ 0-Fluss	$O(m)$
while *Residualnetzwerk von f enthält st-Weg* **do**	$O(nm)$
$f \leftarrow f$ erhöht um st-Weg	$O(m)$

$M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$

Es gilt $w(f) = |M| \leq n/2$, da jede Kante in M zwei Knoten in V besetzt.

Der Algorithmus

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING$(G = (R \cup B, E))$

$(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G	$O(m)$
$f \leftarrow$ 0-Fluss	$O(m)$
while *Residualnetzwerk von f enthält st-Weg* **do**	$O(nm)$
$f \leftarrow f$ erhöht um st-Weg	$O(m)$
$M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$	$O(m)$

Es gilt $w(f) = |M| \leq n/2$, da jede Kante in M zwei Knoten in V besetzt.

Der Algorithmus

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING($G = (R \cup B, E)$) $O(nm)$

- $(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G — $O(m)$
- $f \leftarrow$ 0-Fluss — $O(m)$
- **while** *Residualnetzwerk von f enthält st-Weg* **do** — $O(nm)$
  - $f \leftarrow f$ erhöht um st-Weg — $O(m)$
- $M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$ — $O(m)$

Es gilt $w(f) = |M| \leq n/2$, da jede Kante in M zwei Knoten in V besetzt.

Der Algorithmus

Der Ford-Fulkerson-Algorithmus liefert bei ganzzahligen Kapazitäten einen ganzzahligen maximalen Fluss. (siehe Vorlesung 2)

$\Rightarrow$ Er kann benutzt werden um MAXIMALES BIPARTITES MATCHING zu lösen.

MAXIMALES BIPARTITES MATCHING($G = (R \cup B, E)$) $O(nm)$

- $(G', c, s, t) \leftarrow$ zugehöriges Flussnetzwerk von G — $O(m)$
- $f \leftarrow$ 0-Fluss — $O(m)$
- **while** *Residualnetzwerk von f enthält st-Weg* **do** — $O(nm)$
  - $f \leftarrow f$ erhöht um st-Weg — $O(m)$
- $M \leftarrow$ alle Kanten $\{r, b\}$ mit $r \in R$, $b \in B$ und $f(r, b) = 1$ — $O(m)$

Satz: Maximales Bipartites Matching
Sei G ein bipartiter Graph. Der Algorithmus MAXIMALES BIPARTITES MATCHING berechnet ein maximales Matching in G in $O(nm)$ Zeit.

Bemerkung: Es gibt diverse effiziente Algorithmen zur Bestimmung von maximalen Matchings auch auf allgemeinen Graphen.
Beispiel: Gewichtsmaximale Matchings können in $O(m\sqrt{n})$ Zeit berechnet werden.