Inhaltsübersicht für heute:

Anwendung: Das Heiratsproblem

Ganzzahligkeit von Polyedern

Anwendung: Netzwerkflüsse

Mehrgüterflussprobleme

Ganzzahlige Optimierung

Inhaltsübersicht für heute:

Anwendung: Das Heiratsproblem

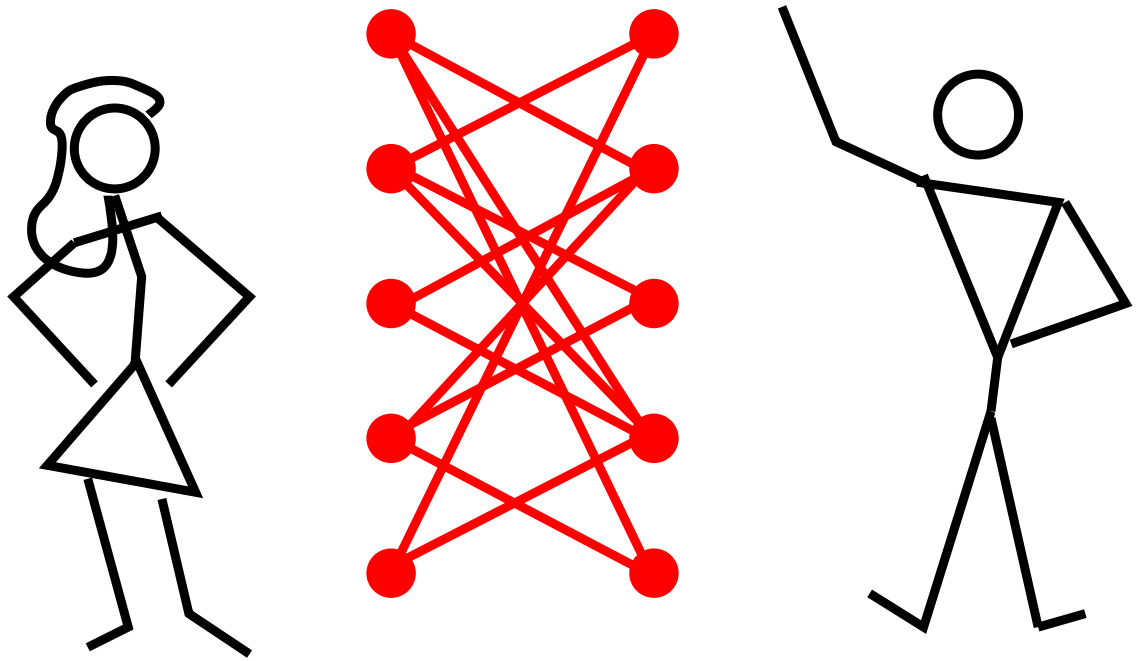
Ganzzahligkeit von Polyedern

Anwendung: Netzwerkflüsse

Mehrgüterflussprobleme

Ganzzahlige Optimierung

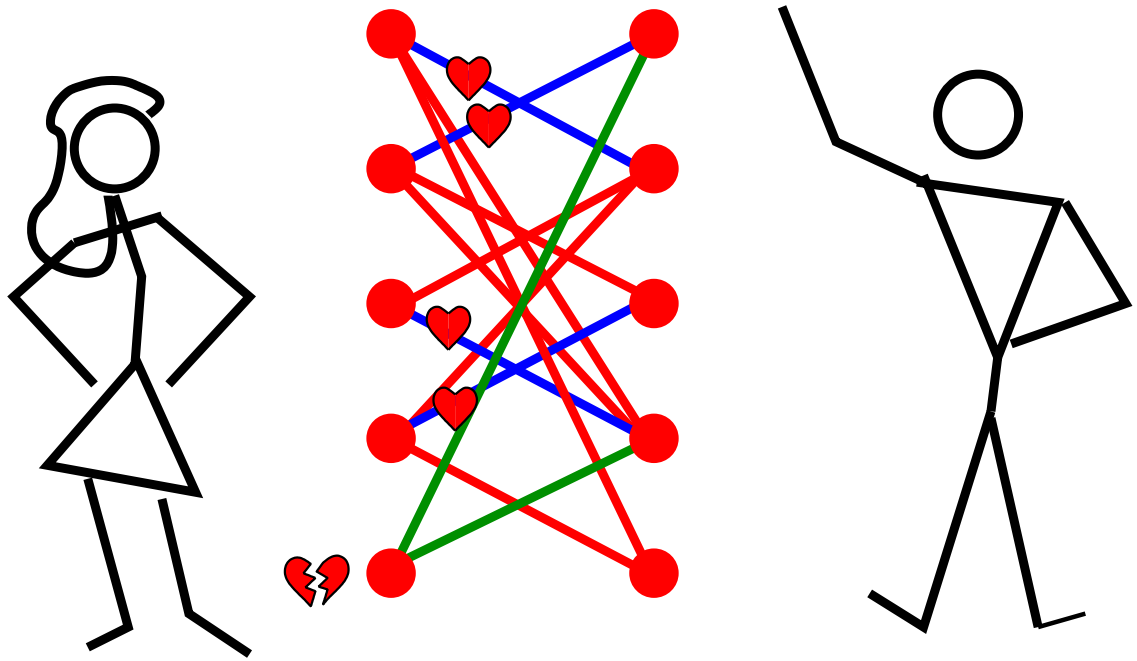
Anwendung: Das Heiratsproblem (bipartites Matching)



Bilde möglichst viele Paare!

Männer $\leftrightarrow$ Frauen, Arbeiter $\leftrightarrow$ Maschinen, Studierende $\leftrightarrow$ Studienplätze, ...
(geht auch gewichtet)

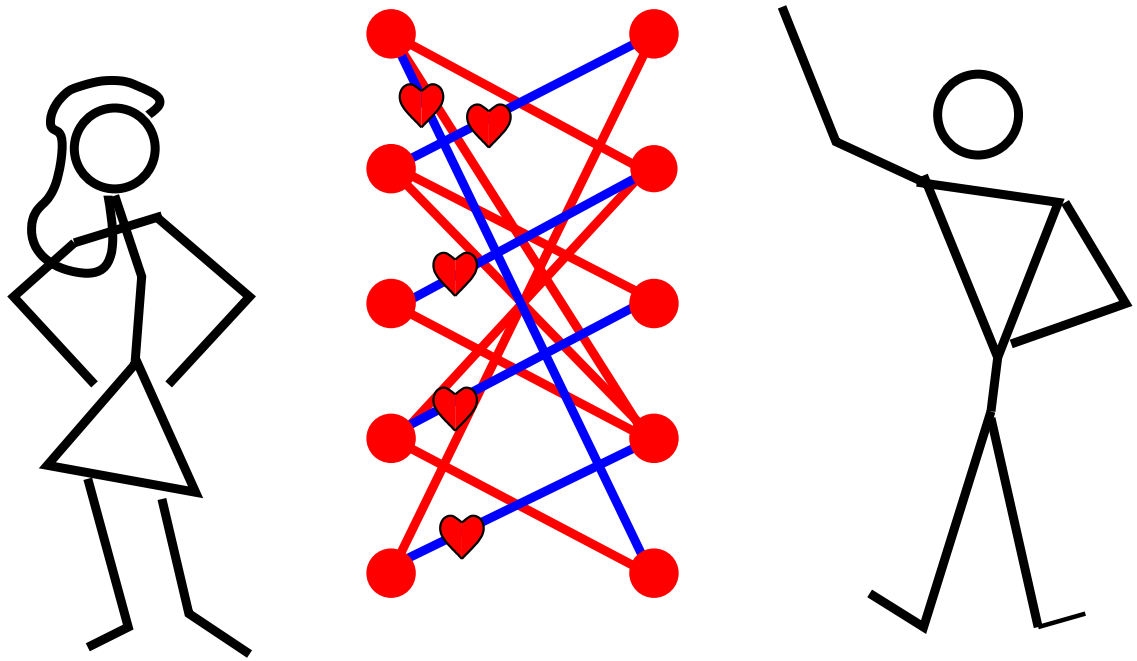
Anwendung: Das Heiratsproblem (bipartites Matching)



Bilde möglichst viele Paare!
Maximal (nicht vergrößerbar), aber kein Kardinalitätsmaximum

Männer $\leftrightarrow$ Frauen, Arbeiter $\leftrightarrow$ Maschinen, Studierende $\leftrightarrow$ Studienplätze, ...
(geht auch gewichtet)

Anwendung: Das Heiratsproblem (bipartites Matching)

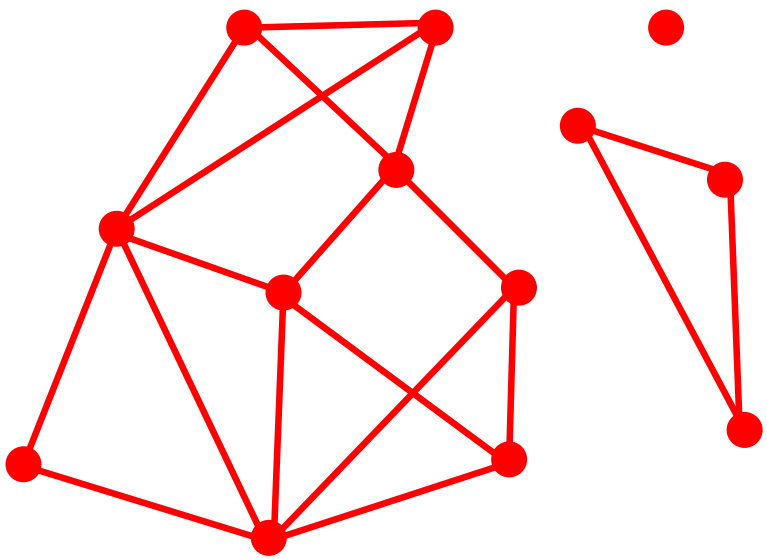


Bilde möglichst viele Paare!
Maximum Cardinality Matching (sogar perfekt)

Männer $\leftrightarrow$ Frauen, Arbeiter $\leftrightarrow$ Maschinen, Studierende $\leftrightarrow$ Studienplätze, ...
(geht auch gewichtet)

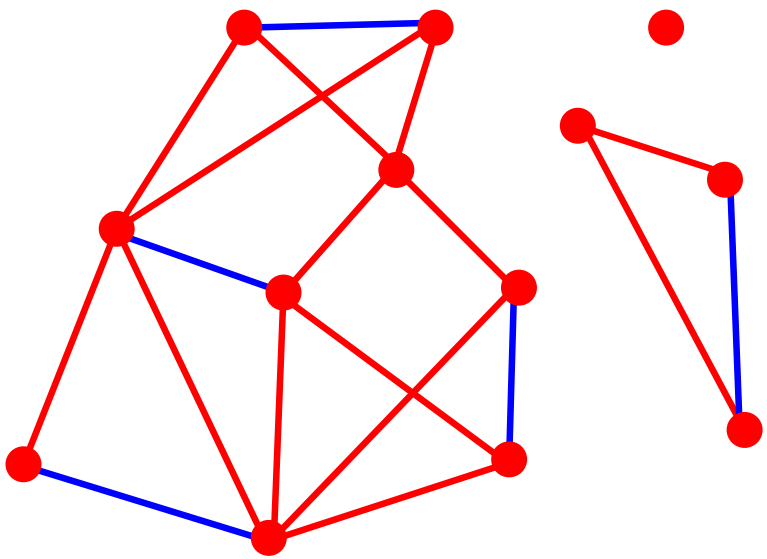
Bipartites Matching

- Ein (ungerichteter) Graph $G = (V, E)$ ist ein Paar bestehend aus Knotenmenge V und Kantenmenge $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$.
- Zwei Knoten $u, v \in V$ heißen adjazent/benachbart, falls $\{u, v\} \in E$.
- Ein Knoten $v \in V$ und eine Kante $e \in E$ heißen inzident, falls $v \in e$.
- Zwei Kanten $e, f \in V$ heißen inzident, falls $e \cap f \neq \emptyset$.



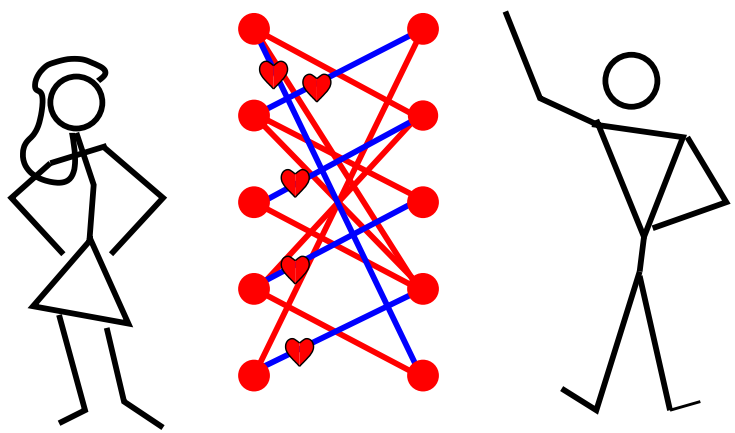
Bipartites Matching

- Ein (ungerichteter) Graph $G = (V, E)$ ist ein Paar bestehend aus Knotenmenge V und Kantenmenge $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$.
- Zwei Knoten $u, v \in V$ heißen adjazent/benachbart, falls $\{u, v\} \in E$.
- Ein Knoten $v \in V$ und eine Kante $e \in E$ heißen inzident, falls $v \in e$.
- Zwei Kanten $e, f \in V$ heißen inzident, falls $e \cap f \neq \emptyset$.
- Eine Kantenmenge $M \subseteq E$ heißt Matching/Paarung, falls für $e, f \in M$ mit $e \neq f$ stets $e \cap f = \emptyset$. Das Matching heißt perfekt, falls $|V| = 2|M|$.



Bipartites Matching

- Ein (ungerichteter) Graph $G = (V, E)$ ist ein Paar bestehend aus Knotenmenge V und Kantenmenge $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$.
- Zwei Knoten $u, v \in V$ heißen adjazent/benachbart, falls $\{u, v\} \in E$.
- Ein Knoten $v \in V$ und eine Kante $e \in E$ heißen inzident, falls $v \in e$.
- Zwei Kanten $e, f \in V$ heißen inzident, falls $e \cap f \neq \emptyset$.
- Eine Kantenmenge $M \subseteq E$ heißt Matching/Paarung, falls für $e, f \in M$ mit $e \neq f$ stets $e \cap f = \emptyset$. Das Matching heißt perfekt, falls $|V| = 2|M|$.
- $G = (V, E)$ heißt bipartit, falls $V = V_1 \cup V_2$ mit $V_1 \cap V_2 = \emptyset$ und $E \subseteq \{\{u, v\} : u \in V_1, v \in V_2\}$.



Modellierung: kardinalitätsmaximales bipartites Matching

gegeben: $G = (V_1 \dot\cup V_2, E)$ bipartit
gesucht: Matching $M \subseteq E$ mit $|M|$ maximal

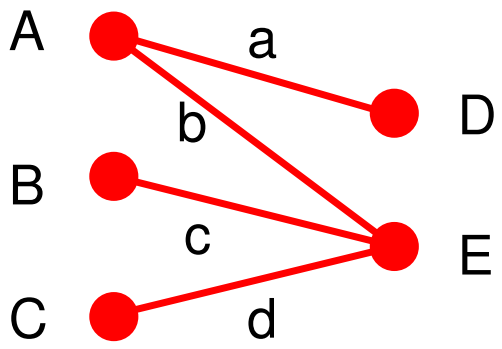
Variablen: $x \in \{0,1\}^E$ mit $x_e = \begin{cases} 1 & \text{falls } e \in M \\ 0 & \text{sonst.} \end{cases}$ $(e \in E)$

(steht für den Inzidenz-/charakteristischen Vektor von M bzgl. E)

Nebenbedingung: $Ax \leq \mathbf{1}$,

wobei $A \in \{0,1\}^{V \times E}$ Knoten-Kanten-Inzidenzmatrix zu G:

$$A_{v,e} = \begin{cases} 1 & \text{falls } v \in e \\ 0 & \text{sonst.} \end{cases} \quad (v \in V, e \in E)$$



$$A = \left[\begin{array}{c|cccc} & (a) & (b) & (c) & (d) \\ (A) & 1 & 1 & 0 & 0 \\ (B) & 0 & 0 & 1 & 0 \\ (C) & 0 & 0 & 0 & 1 \\ \hline (D) & 1 & 0 & 0 & 0 \\ (E) & 0 & 1 & 1 & 1 \end{array}\right]$$

Modellierung: kardinalitätsmaximales bipartites Matching

gegeben: $G = (V_1 \dot\cup V_2, E)$ bipartit
gesucht: Matching $M \subseteq E$ mit $|M|$ maximal

Variablen: $x \in \{0,1\}^E$ mit $x_e = \begin{cases} 1 & \text{falls } e \in M \\ 0 & \text{sonst.} \end{cases}$ $(e \in E)$

(steht für den Inzidenz-/charakteristischen Vektor von M bzgl. E)

Nebenbedingung: $Ax \leq \mathbf{1}$,

wobei $A \in \{0,1\}^{V \times E}$ Knoten-Kanten-Inzidenzmatrix zu G:

$$A_{v,e} = \begin{cases} 1 & \text{falls } v \in e \\ 0 & \text{sonst.} \end{cases} \quad (v \in V, e \in E)$$

Optimierungsproblem:

$$\begin{aligned} \max \quad & \mathbf{1}^T x \\ \text{s.t.} \quad & Ax \leq \mathbf{1} \\ & x \in \{0,1\}^E \end{aligned}$$

So kein LP! $x \in \{0,1\}^E$ zu $x \in [0,1]^E$ vergrößern $\rightarrow$ LP

Für G bipartit gilt: Simplex liefert immer eine Optimallösung $x^* \in \{0,1\}^E$!
(Für allgemeine Graphen G i.A. aber nicht!)

Inhaltsübersicht für heute:

Anwendung: Das Heiratsproblem

Ganzzahligkeit von Polyedern

Anwendung: Netzwerkflüsse

Mehrgüterflussprobleme

Ganzzahlige Optimierung

Ganzzahlige Polyeder

Simplex liefert automatisch eine ganzzahlige Lösung, wenn alle Ecken der zulässigen Menge ganzzahlig sind.

$$\min c^T x \quad \text{s.t.} \quad x \in \mathcal{X} := \{x \geq 0 : Ax = b\}$$

Hat $\mathcal{X}$ nur ganzzahlige Ecken?

Ganzzahlige Polyeder

Simplex liefert automatisch eine ganzzahlige Lösung, wenn alle Ecken der zulässigen Menge ganzzahlig sind.

$$\min c^T x \quad \text{s.t.} \quad x \in \mathcal{X} := \{x \geq 0 : Ax = b\}$$

Hat $\mathcal{X}$ nur ganzzahlige Ecken? Fast nie!
Aber es gibt wichtige Klassen von Matrizen $A \in \mathbb{Z}^{m \times n}$, für die $\mathcal{X}$ für jedes (!) $b \in \mathbb{Z}^m$ nur ganzzahlige Ecken hat:

Eine Ecke ist ganzzahlig $\Leftrightarrow$ Basislösung $x_B = A_B^{-1} b \in \mathbb{Z}^m$
Wenn $|\det(A_B)| = 1$, folgt mit der Cramer'schen Regel $x_B \in \mathbb{Z}^m$.

Ganzzahlige Polyeder

Simplex liefert automatisch eine ganzzahlige Lösung, wenn alle Ecken der zulässigen Menge ganzzahlig sind.

$$\min c^T x \quad \text{s.t.} \quad x \in \mathcal{X} := \{x \geq 0 : Ax = b\}$$

Hat $\mathcal{X}$ nur ganzzahlige Ecken? Fast nie!
Aber es gibt wichtige Klassen von Matrizen $A \in \mathbb{Z}^{m \times n}$, für die $\mathcal{X}$ für jedes (!) $b \in \mathbb{Z}^m$ nur ganzzahlige Ecken hat:

Eine Ecke ist ganzzahlig $\Leftrightarrow$ Basislösung $x_B = A_B^{-1} b \in \mathbb{Z}^m$
Wenn $|\det(A_B)| = 1$, folgt mit der Cramer'schen Regel $x_B \in \mathbb{Z}^m$.
Eine Matrix $A \in \mathbb{Z}^{m \times n}$ mit vollem Zeilenrang heißt unimodular, falls $|\det(A_B)| = 1$ für jede Basis B erfüllt ist.

Satz

$A \in \mathbb{Z}^{m \times n}$ ist genau dann unimodular, wenn für jedes $b \in \mathbb{Z}^m$ alle Ecken des Polyeders $\mathcal{X} := \{x \geq 0 : Ax = b\}$ ganzzahlig sind.

Gilt das auch für $\mathcal{X} := \{x \geq 0 : Ax \geq b\}$?

Total unimodulare Matrizen

$$\{x \geq 0 : Ax \geq b\} \quad \rightarrow \quad \left\{ \begin{bmatrix} x \\ s \end{bmatrix} \geq 0 : [A, I] \begin{bmatrix} x \\ s \end{bmatrix} = b \right\}$$

Sicher ganzzahlig, falls $\bar{A} = [A, I]$ unimodular ist.
Determinantenentwicklung nach Laplace für jede Basis $B \rightarrow$

Eine Matrix A heißt total unimodular, falls für jede quadratische Untermatrix von A die Determinante den Wert 0, 1 oder -1 hat.
(geht nur, wenn $A \in \{0, 1, -1\}^{m \times n}$)

Satz (Hoffmann und Kruskal)

$A \in \mathbb{Z}^{m \times n}$ ist genau dann total unimodular, wenn für jedes $b \in \mathbb{Z}^m$ alle Ecken des Polyeders $\mathcal{X} := \{x \geq 0 : Ax \geq b\}$ ganzzahlig sind.

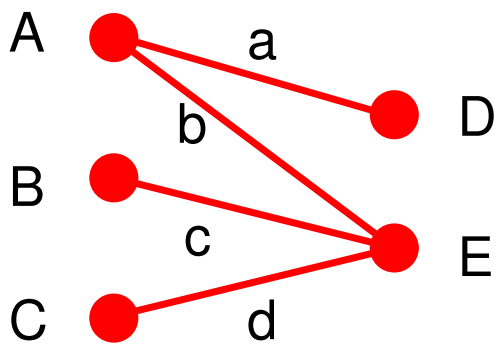
Beachte: A tot. unimod. $\Leftrightarrow A^T$ bzw. $[A, -A, I, -I]$ tot. unimod.
Konsequenz: duales LP, Gleichungsvarianten, etc. sind ganzzahlig

Total unimodulare Matrizen erkennen

Satz (Heller und Tompkins)

$A \in \{0, 1, -1\}^{m \times n}$ habe höchstens zwei Einträge pro Spalte.
A ist total unimodular $\Leftrightarrow$ Die Zeilen von A können in zwei Klassen eingeteilt werden, sodass
(i) Zeilen mit einem $+1$ und einem -1 Eintrag in derselben Spalte in dieselbe Klasse,
(ii) Zeilen mit zwei vorzeichengleichen Einträgen in der gleichen Spalte in unterschiedliche Klassen kommen.

Beispiel 1: die Knoten-Kanten-Inzidenzmatrix eines bipartiten Graphen



$$A = \left[\begin{array}{c|cccc} & (a) & (b) & (c) & (d) \\ (A) & 1 & 1 & 0 & 0 \\ (B) & 0 & 0 & 1 & 0 \\ (C) & 0 & 0 & 0 & 1 \\ \hline (D) & 1 & 0 & 0 & 0 \\ (E) & 0 & 1 & 1 & 1 \end{array}\right]$$

Beispiel 1: bipartite Graphen

A ... Knoten-Kanten-Inzidenzmatrix von $G = (V_1 \dot\cup V_2, E)$ bipartit

---

Bipartites Matching maximaler Kardinalität:

$$\max \mathbf{1}^T x \quad \text{s.t.} \quad Ax \leq \mathbf{1},\ x \geq 0$$

Wegen A tot. unimod. hat die zul. Menge nur ganzzahlige Ecken
$\Rightarrow$ Simplex liefert Optimallösung $x^* \in \{0,1\}^E$.

Das Duale ist auch ganzzahlig, wegen A^T tot. unimod.:

$$\min \mathbf{1}^T y \quad \text{s.t.} \quad A^T y \geq \mathbf{1},\ y \geq 0$$

Interpretation: $y^* \in \{0,1\}^V$ ist Inzidenzvektor einer kleinsten Knotenmenge $V' \subseteq V$, sodass $\forall e \in E : e \cap V' \neq \emptyset$ (**Minimum Vertex Cover**)

Das **Zuweisungsproblem**: $|V_1| = |V_2| = n$, vollständig bipartit:
$E = \{\{u,v\} : u \in V_1, v \in V_2\}$; Kantengewichte $c \in \mathbb{R}^E$
Suche ein perfektes Matching minimalen Gesamtgewichts:

$$\min c^T x \quad \text{s.t.} \quad Ax = \mathbf{1},\ x \geq 0$$

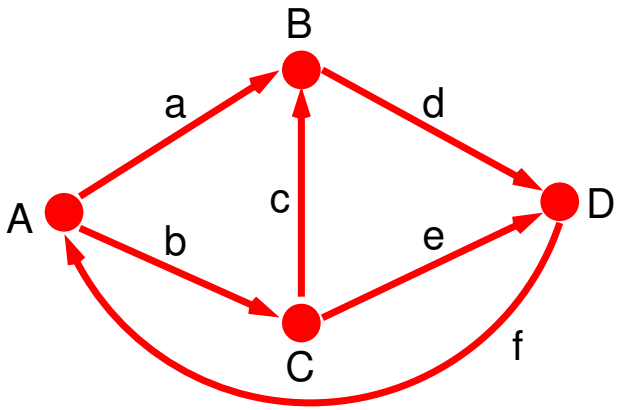
ist auch ganzzahlig, weil $[A; -A]$ tot. unimod.

Beispiel 2: Knoten-Kanten-Inzidenzmatrix von Digraphen

- Ein Digraph/gerichteter Graph $D = (V, E)$ ist ein Paar bestehend aus Knotenmenge V und einer (Multi-)Menge gerichteter Kanten/Pfeile $E \subseteq \{(u, v) : u, v \in V, u \neq v\}$. [Mehrfachkanten sind erlaubt!]
- Für $e = (u, v) \in E$ ist u der Schaft und v die Spitze von e.
- Die Knoten-Kanten-Inzidenzmatrix $A \in \{0, 1, -1\}^{V \times E}$ von D hat Einträge

$$A_{v,e} = \begin{cases} -1 & v \text{ ist Schaft von } e \\ 1 & v \text{ ist Spitze von } e \\ 0 & \text{sonst} \end{cases} \qquad (v \in V, e \in E).$$

Die Knoten-Kanten-Inzidenzmatrix eines Digraphen ist total unimodular.



$$A = \begin{bmatrix} & (a) & (b) & (c) & (d) & (e) & (f) \\ (A) & -1 & -1 & 0 & 0 & 0 & 1 \\ (B) & 1 & 0 & 1 & -1 & 0 & 0 \\ (C) & 0 & 1 & -1 & 0 & -1 & 0 \\ (D) & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

Inhaltsübersicht für heute:

Anwendung: Das Heiratsproblem

Ganzzahligkeit von Polyedern

Anwendung: Netzwerkflüsse

Mehrgüterflussprobleme

Ganzzahlige Optimierung

Anwendung: Netzwerkflüsse

Modellierungswerkzeug: Transportprobleme, Evakuierungspläne, Auftrags-, Verkehrsplanung, ...

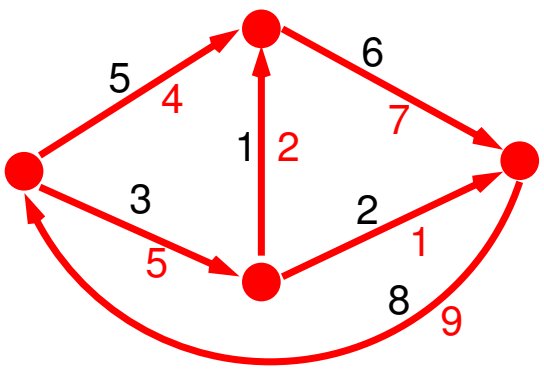
- Ein Netzwerk (D, w) besteht aus einem Digraphen $D = (V, E)$ und (Kanten-)Kapazitäten $w \in \mathbb{R}_+^E$.
- Ein Vektor $x \in \mathbb{R}^E$ heißt Fluss auf (D, w), falls er die Flusserhaltungsgleichungen

$$\sum_{e=(u,v)\in E} x_e = \sum_{e=(v,u)\in E} x_e \quad (v \in V)$$

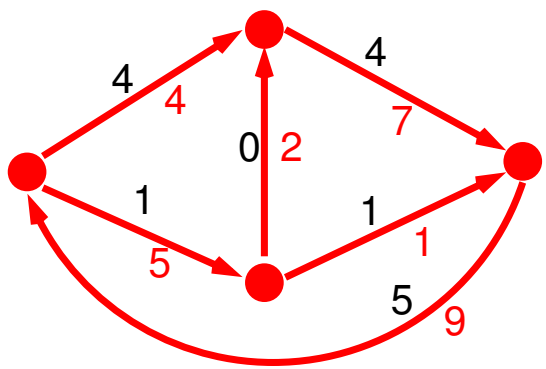
erfüllt. [$\Leftrightarrow Ax = 0$ für Knoten-Kanten-Inzidenzmatrix A]

- Ein Fluss $x \in \mathbb{R}^E$ auf (D,w) heißt zulässig, falls $0 \leq x \leq w$

[auch untere Schranken machbar]



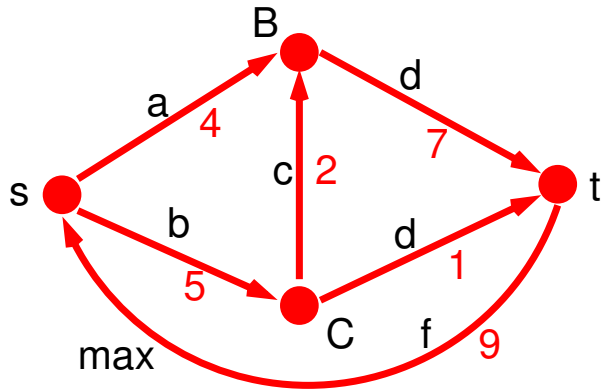
Fluss



zulässiger Fluss

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Gegeben Quelle $s \in V$ und Senke $t \in V$ mit $(t, s) \in E$, finde einen zulässigen Fluss $x \in \mathbb{R}^E$ auf (D, w) mit maximalem Flusswert $x_{(t,s)}$.



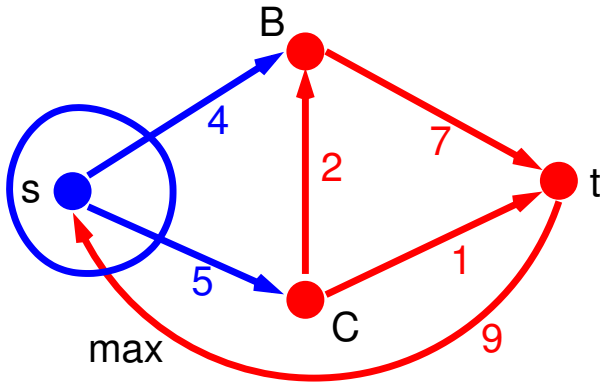
$$A = \begin{bmatrix} & (a) & (b) & (c) & (d) & (e) & (f) \\ (s) & -1 & -1 & 0 & 0 & 0 & 1 \\ (B) & 1 & 0 & 1 & -1 & 0 & 0 \\ (C) & 0 & 1 & -1 & 0 & -1 & 0 \\ (t) & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

LP: $\max\ x_{(t,s)}$ s.t. $Ax = 0,\ 0 \leq x \leq w,$

Falls $w \in \mathbb{Z}^E$, ist Simplex-OL $x^* \in \mathbb{Z}^E$, weil $[A; -A; I]$ tot. unimod.

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Gegeben Quelle $s \in V$ und Senke $t \in V$ mit $(t,s) \in E$, finde einen zulässigen Fluss $x \in \mathbb{R}^E$ auf (D,w) mit maximalem Flusswert $x_{(t,s)}$.



$S = \{s\}$,
$\delta^+(S) = \{(s,B),(s,C)\}$,
$w(\delta^+(S)) = 4+5 = 9$.

LP: max $x_{(t,s)}$ s.t. $Ax = 0,\ 0 \le x \le w$,

Falls $w \in \mathbb{Z}^E$, ist Simplex-OL $x^* \in \mathbb{Z}^E$, weil $[A;-A;I]$ tot. unimod.

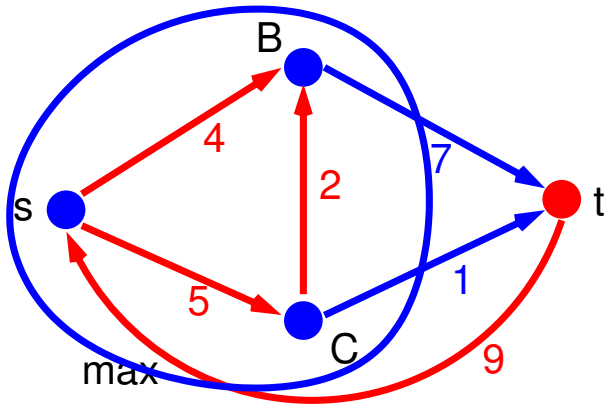
Jedes $S \subseteq V$ mit $s \in S$ und $t \notin S$ definiert einen s-t-Schnitt

$$\delta^+(S) := \{(u,v) \in E : u \in S, v \notin S\},$$

darüber fließt höchstens $w(\delta^+(S)) := \sum_{e \in \delta^+(S)} w_e$, der Wert des Schnitts.

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Gegeben Quelle $s \in V$ und Senke $t \in V$ mit $(t, s) \in E$, finde einen zulässigen Fluss $x \in \mathbb{R}^E$ auf (D, w) mit maximalem Flusswert $x_{(t,s)}$.



$S = \{s, B, C\}$,
$\delta^+(S) = \{(B, t), (C, t)\}$,
$w(\delta^+(S)) = 7 + 1 = 8$.

LP: max $x_{(t,s)}$ s.t. $Ax = 0,\ 0 \leq x \leq w$,

Falls $w \in \mathbb{Z}^E$, ist Simplex-OL $x^* \in \mathbb{Z}^E$, weil $[A; -A; I]$ tot. unimod.

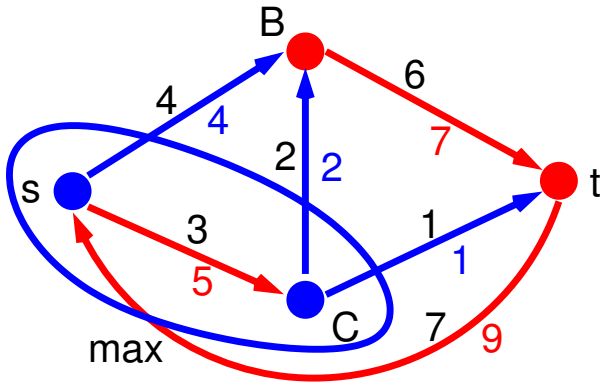
Jedes $S \subseteq V$ mit $s \in S$ und $t \notin S$ definiert einen s-t-Schnitt

$$\delta^+(S) := \{(u, v) \in E : u \in S, v \notin S\},$$

darüber fließt höchstens $w(\delta^+(S)) := \sum_{e \in \delta^+(S)} w_e$, der Wert des Schnitts.

Maximale s-t-Flüsse, Minimale s-t-Schnitte

Gegeben Quelle $s \in V$ und Senke $t \in V$ mit $(t,s) \in E$, finde einen zulässigen Fluss $x \in \mathbb{R}^E$ auf (D,w) mit maximalem Flusswert $x_{(t,s)}$.



$S = \{s, C\}$,
$\delta^+(S) = \{(s,B),(C,B),(C,t)\}$,
$w(\delta^+(S)) = 4+2+1 = 7 = x^*_{(t,s)}$

LP: max $x_{(t,s)}$ s.t. $Ax = 0,\ 0 \le x \le w$,

Falls $w \in \mathbb{Z}^E$, ist Simplex-OL $x^* \in \mathbb{Z}^E$, weil $[A; -A; I]$ tot. unimod.

Jedes $S \subseteq V$ mit $s \in S$ und $t \notin S$ definiert einen s-t-Schnitt

$$\delta^+(S) := \{(u,v) \in E : u \in S, v \notin S\},$$

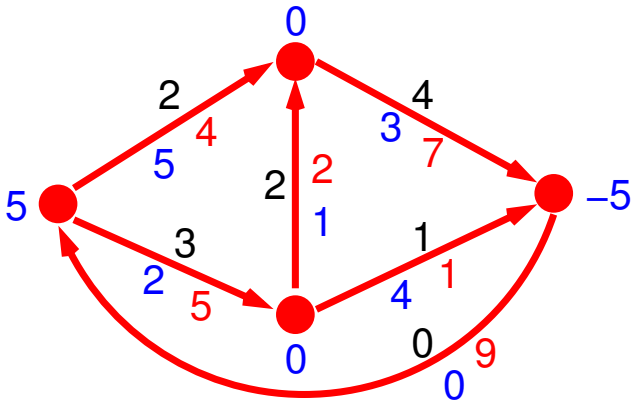
darüber fließt höchstens $w(\delta^+(S)) := \sum_{e \in \delta^+(S)} w_e$, der Wert des Schnitts.

Satz (Max-Flow Min-Cut Theorem von Ford und Fulkerson)

Der maximale Wert eines s-t-Flusses ist gleich dem minimalen Werte eines s-t-Schnitts. [beide durch Simplex bestimmbar]

Minimale-Kosten-Flüsse (Min-Cost-Flow)

Die Flussmenge wird durch Balancen $b \in \mathbb{R}^V$ $(\mathbf{1}^T b = 0)$ auf den Knoten vorgegeben, pro Flusseinheit fallen Kantenkosten $c \in \mathbb{R}^E$ an.
Finde den günstigsten Fluss.



$$\begin{array}{lll} \min & \begin{bmatrix} 5 & 2 & 1 & 3 & 4 & 0 \end{bmatrix} x & \\ \text{s.t.} & \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} x = \begin{bmatrix} 5 \\ 0 \\ 0 \\ -5 \end{bmatrix} & \\ & 0 \le x \le w & \end{array}$$

LP: $\min\ c^T x \quad \text{s.t.} \quad Ax = b,\ 0 \le x \le w,$

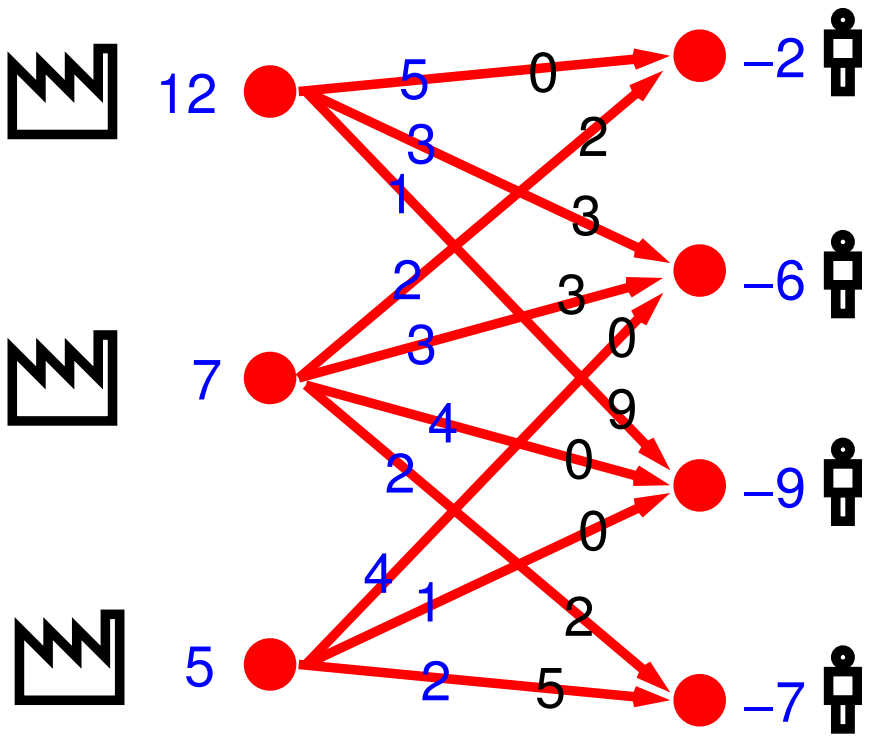
Für b, c und w ganzz. ist Simplex-OL $x^* \in \mathbb{Z}^E$, weil $[A; -A; I]$ tot. unimod.
[geht auch mit unteren Schranken auf den Kanten: $u \le x \le w$!]

Für LPs $\min\ c^T x \quad \text{s.t.} \quad Ax = b,\ u \le x \le w,$ A Knoten-Kanten-Inz. gibt es eine besonders effiziente Simplex-Variante, den Netzwerksimplex, dieser braucht nur Addition, Subtraktion und Vergleiche!

Extrem breite Anwendungsmöglichkeiten, beliebtes Modellierungswerkzeug

Beispiel: Transportproblem

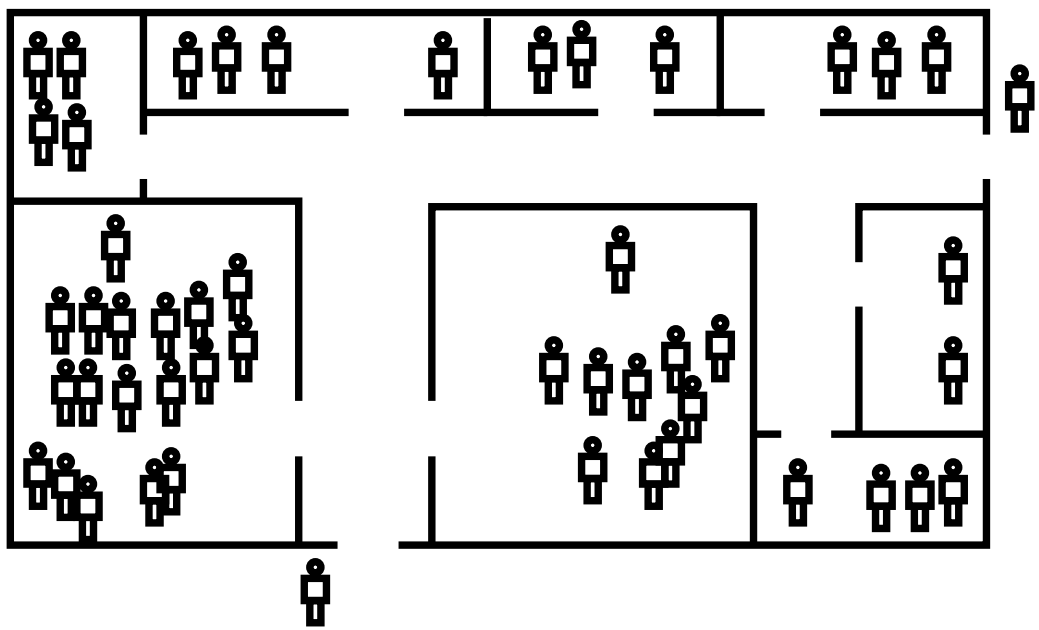
Eine Firma mit mehreren Produktionsstandorten hat mehrere Kunden zu beliefern. Wie geschieht dies am günstigsten unter Berücksichtigung der Transportkosten?



Beachte: Nur ein Produkttyp!

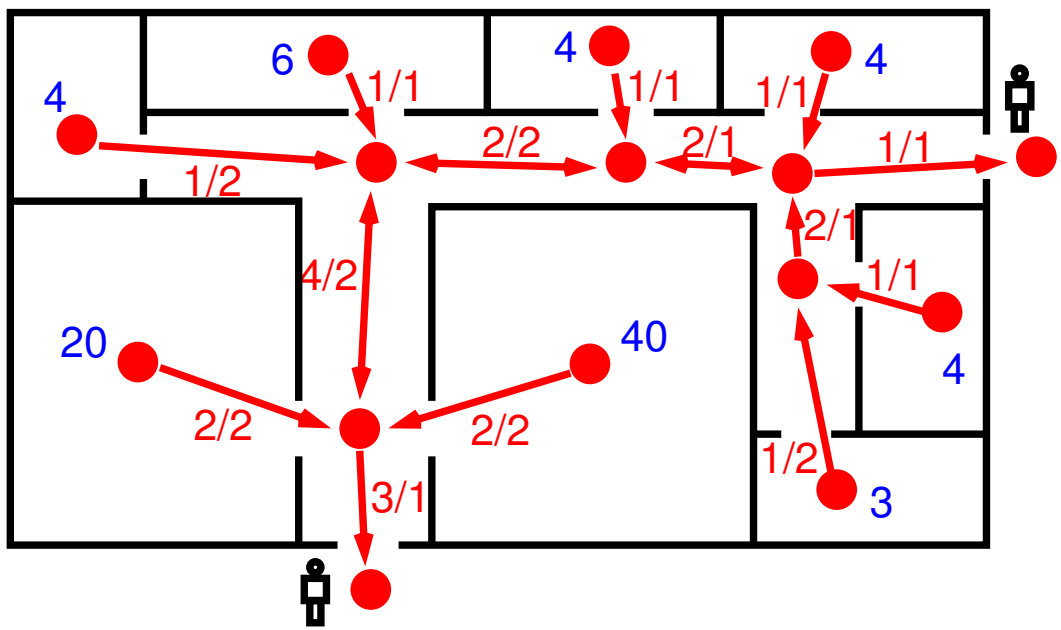
Beispiel: Evakuierungsplanung

Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Beispiel: Evakuierungsplanung

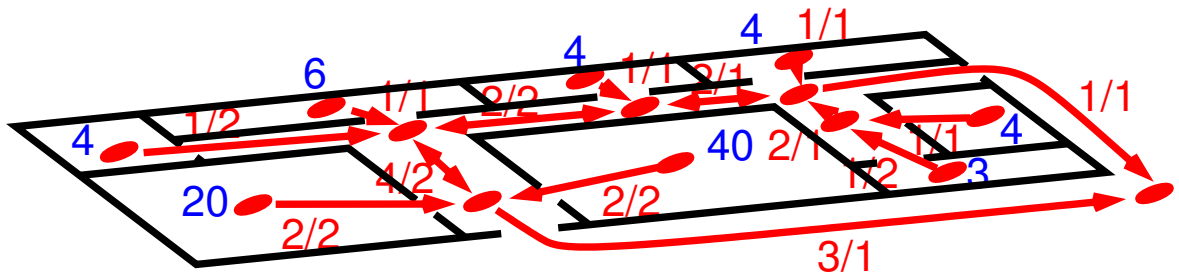
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Kanten mit Kapazität und Durchquerungszeit

Beispiel: Evakuierungsplanung

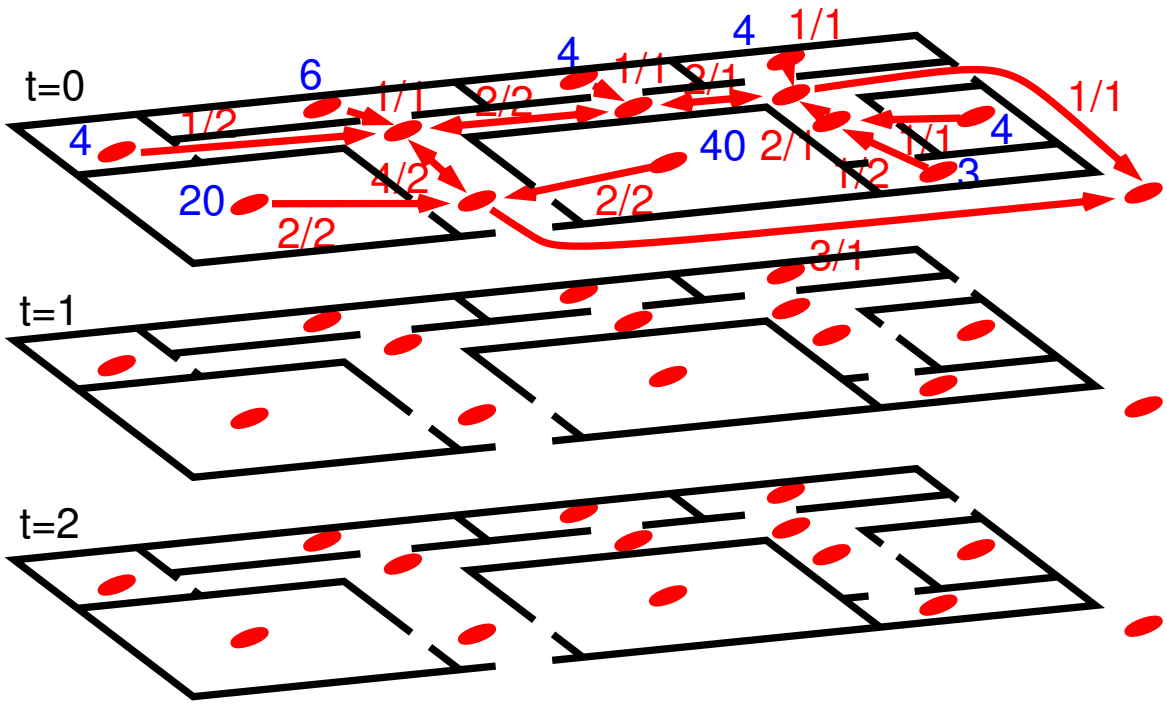
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Geometrie nicht wichtig, vereinfachbar

Beispiel: Evakuierungsplanung

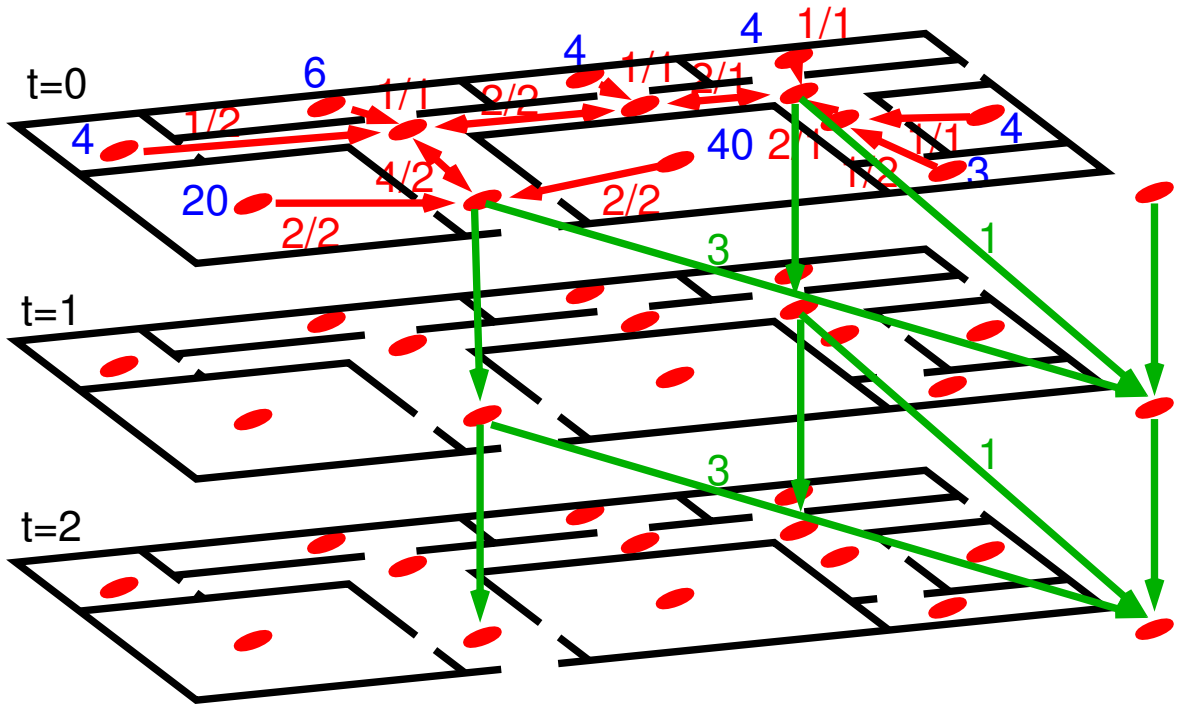
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Diskretisierung der Zeit, ein Niveau pro Zeiteinheit

Beispiel: Evakuierungsplanung

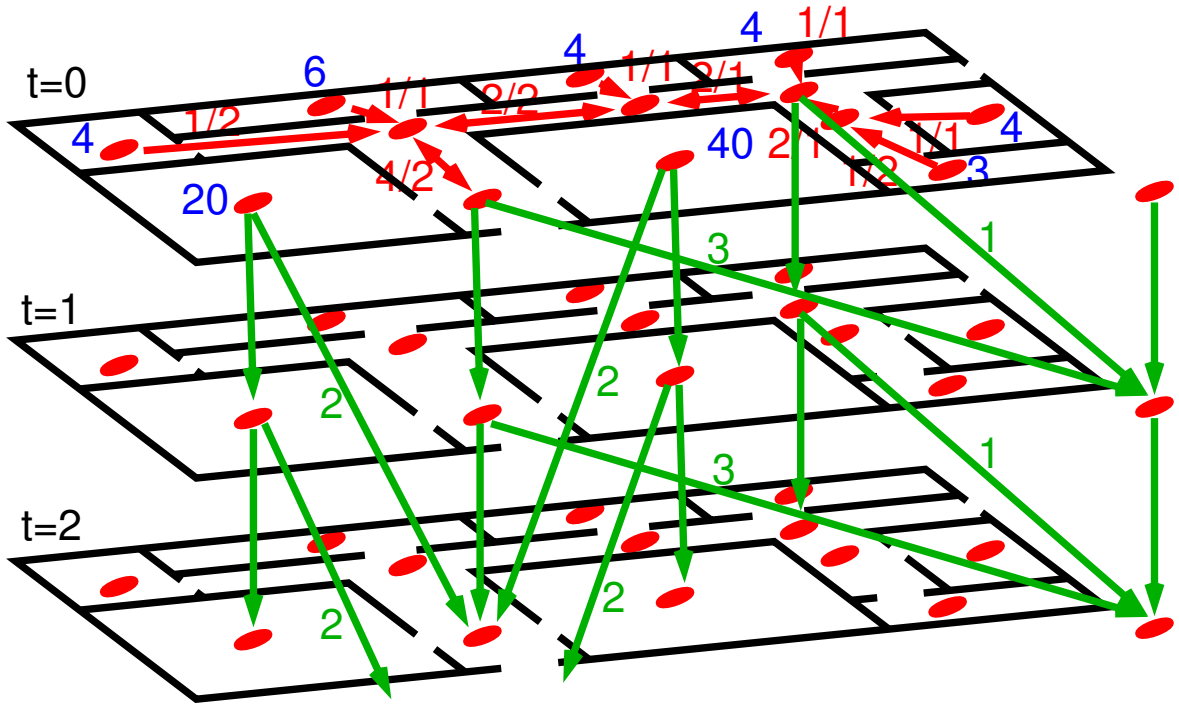
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Durchquerungszeit verbindet Niveaus, Kapazität bleibt

Beispiel: Evakuierungsplanung

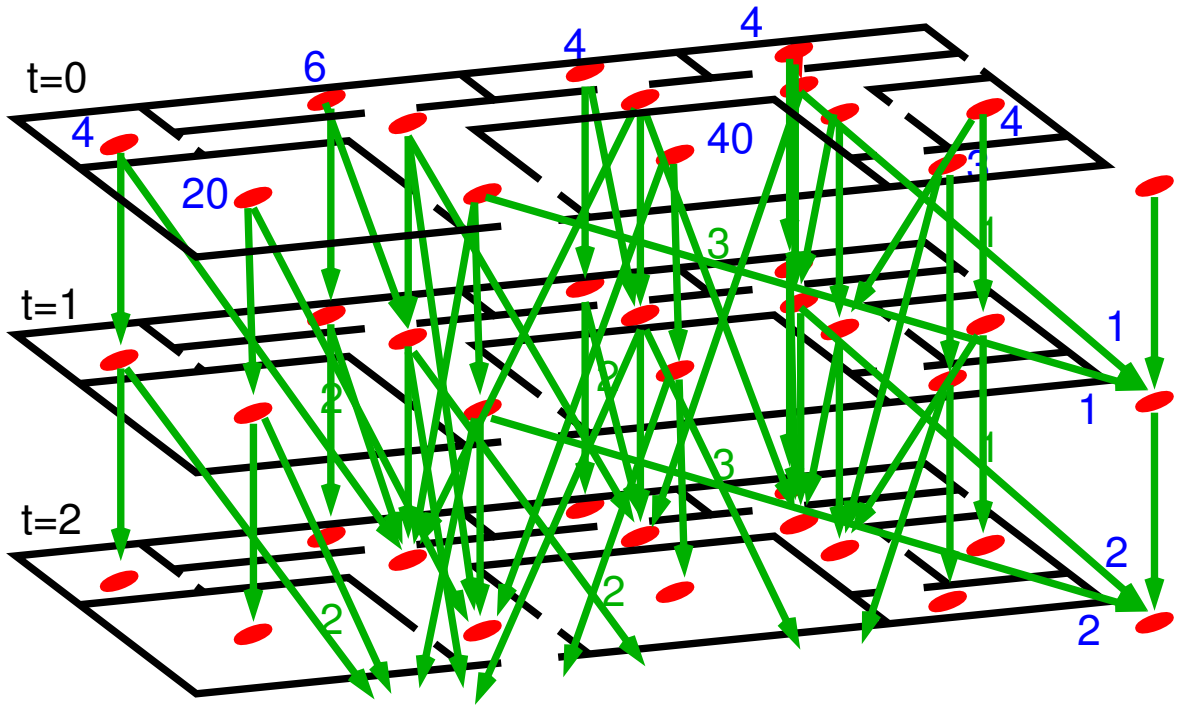
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Durchquerungszeit verbindet Niveaus, Kapazität bleibt

Beispiel: Evakuierungsplanung

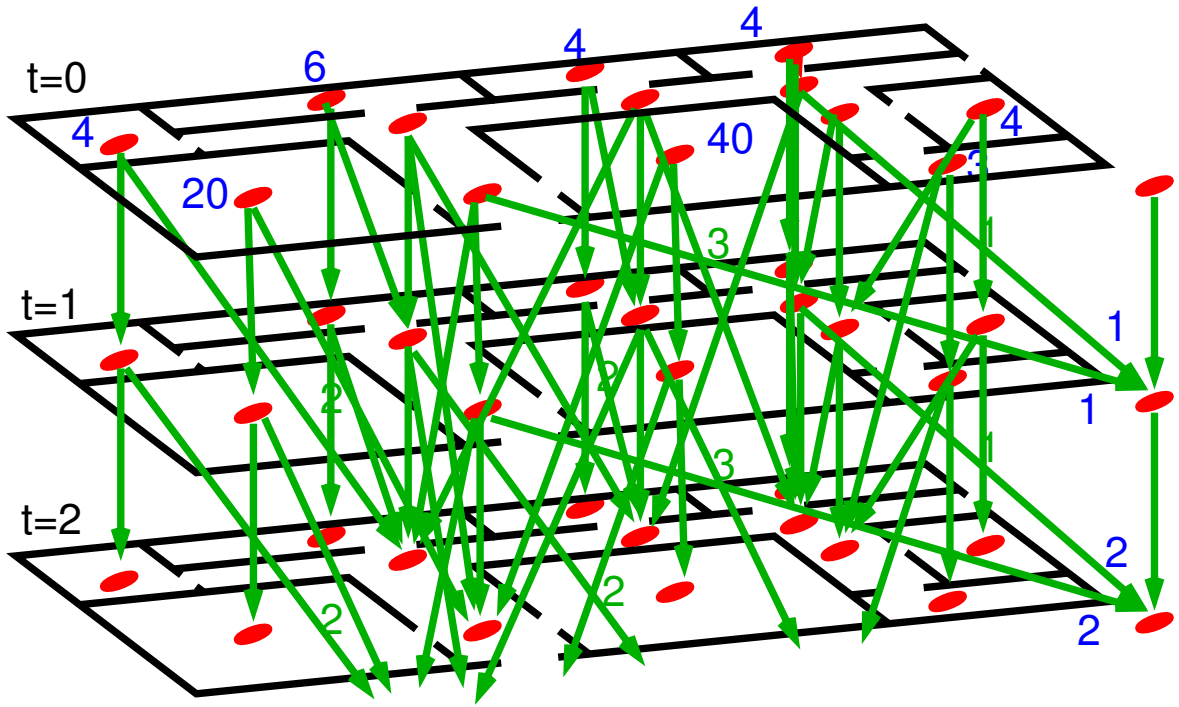
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Steigende Kosten auf den Ausgangskanten für rasches Verlassen

Beispiel: Evakuierungsplanung

Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Steigende Kosten auf den Ausgangskanten für rasches Verlassen

Ansatz nur ok, wenn Personen nicht unterschieden werden müssen!

Inhaltsübersicht für heute:

Anwendung: Das Heiratsproblem

Ganzzahligkeit von Polyedern

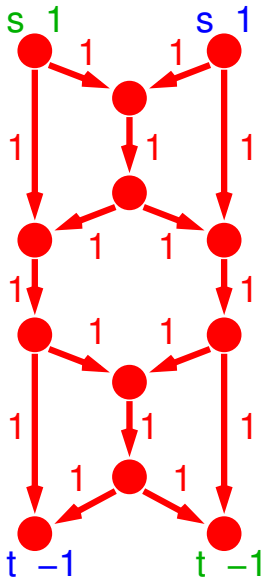
Anwendung: Netzwerkflüsse

Mehrgüterflussprobleme

Ganzzahlige Optimierung

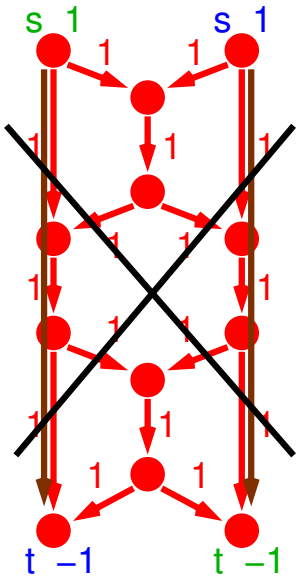
Mehrgüterflussprobleme (Multicommodity Flow)

In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



Mehrgüterflussprobleme (Multicommodity Flow)

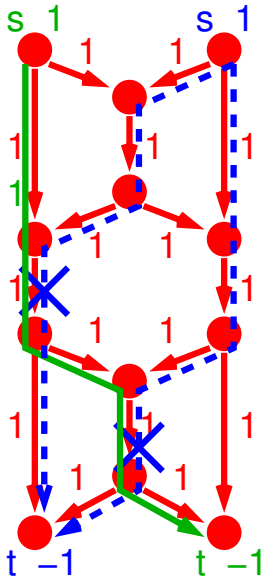
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



Mischen verboten!

Mehrgüterflussprobleme (Multicommodity Flow)

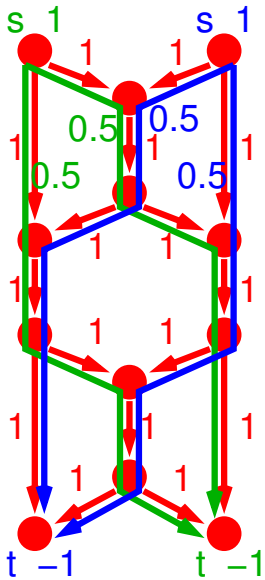
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



ganzz. geht nicht

Mehrgüterflussprobleme (Multicommodity Flow)

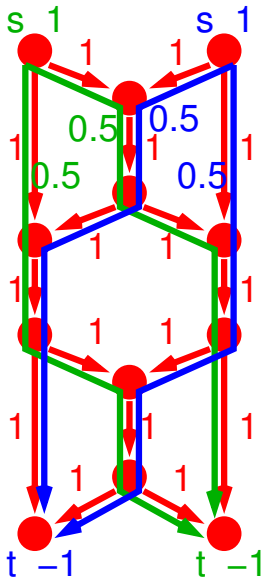
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



gebrochen geht

Mehrgüterflussprobleme (Multicommodity Flow)

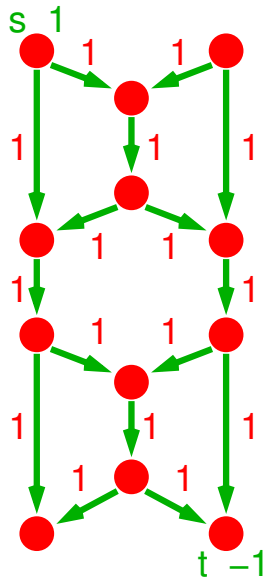
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



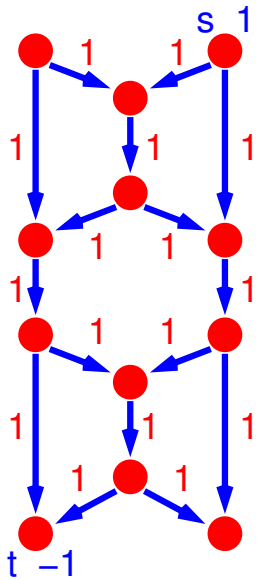
gebrochen geht

Umsetzung

$\longrightarrow$



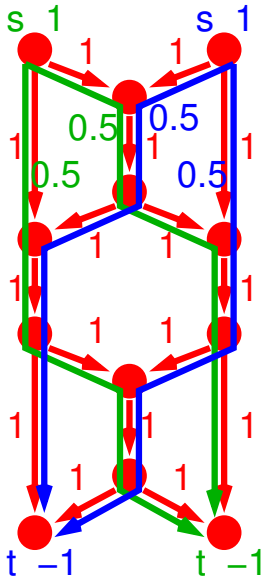
Kopie 1



Kopie 2

Mehrgüterflussprobleme (Multicommodity Flow)

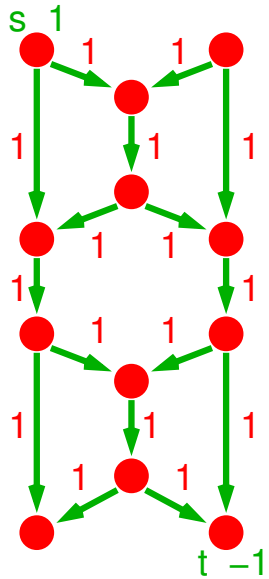
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



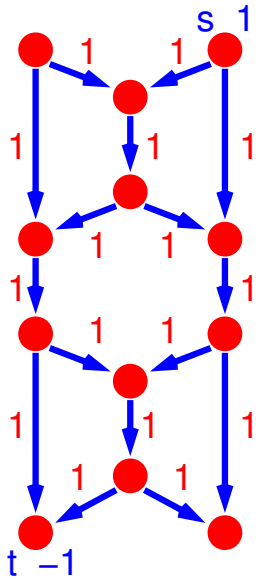
gebrochen geht

Umsetzung

$\longrightarrow$



Kopie 1

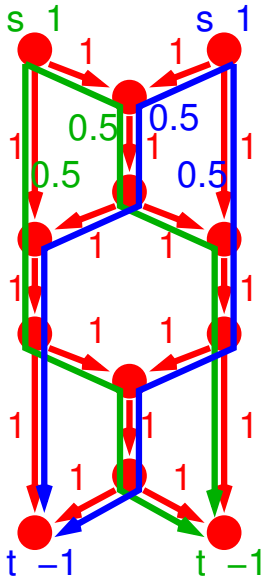


Kopie 2

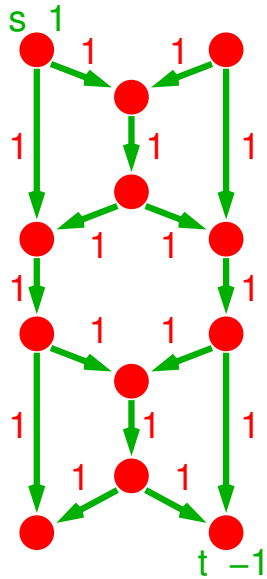
$$
\begin{array}{lccccc}
\min & c^{(1)\,T} x^{(1)} & + & c^{(2)\,T} x^{(2)} & & \\
\text{s.t.} & A x^{(1)} & & & = & b^{(1)} \\
 & & & A x^{(2)} & = & b^{(2)} \\
 & I x^{(1)} & + & I x^{(2)} & \leq & w \\
 & x^{(1)} \geq 0, & & x^{(2)} \geq 0. & &
\end{array}
$$

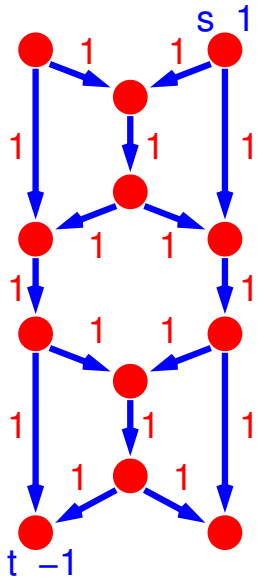
Mehrgüterflussprobleme (Multicommodity Flow)

In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



Umsetzung $\longrightarrow$





gebrochen geht

Kopie 1 Kopie 2

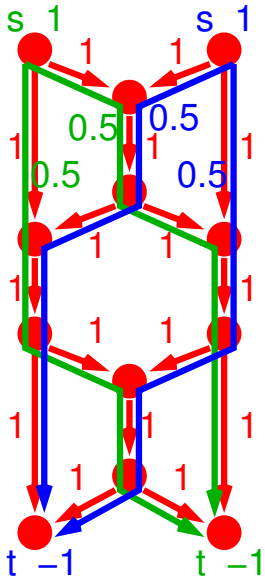
$$\begin{bmatrix} A & 0 \\ 0 & A \\ I & I \end{bmatrix}$$

i.A. nicht tot.unimod.!

$$\begin{array}{lrcl} \min & {c^{(1)}}^T x^{(1)} + {c^{(2)}}^T x^{(2)} & & \\ \text{s.t.} & Ax^{(1)} & = & b^{(1)} \\ & Ax^{(2)} & = & b^{(2)} \\ & Ix^{(1)} + Ix^{(2)} & \leq & w \\ & x^{(1)} \geq 0, \quad x^{(2)} \geq 0. & & \end{array}$$

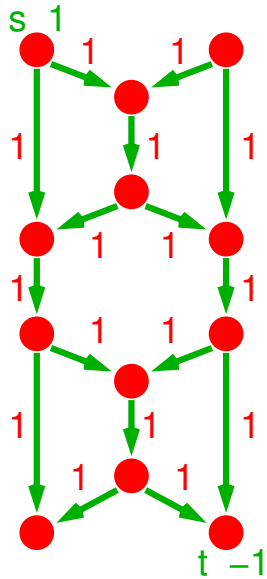
Mehrgüterflussprobleme (Multicommodity Flow)

In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.

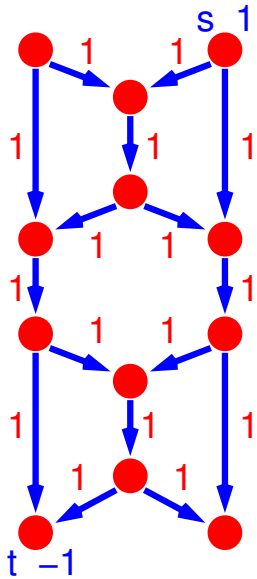


gebrochen geht

Umsetzung $\longrightarrow$



Kopie 1



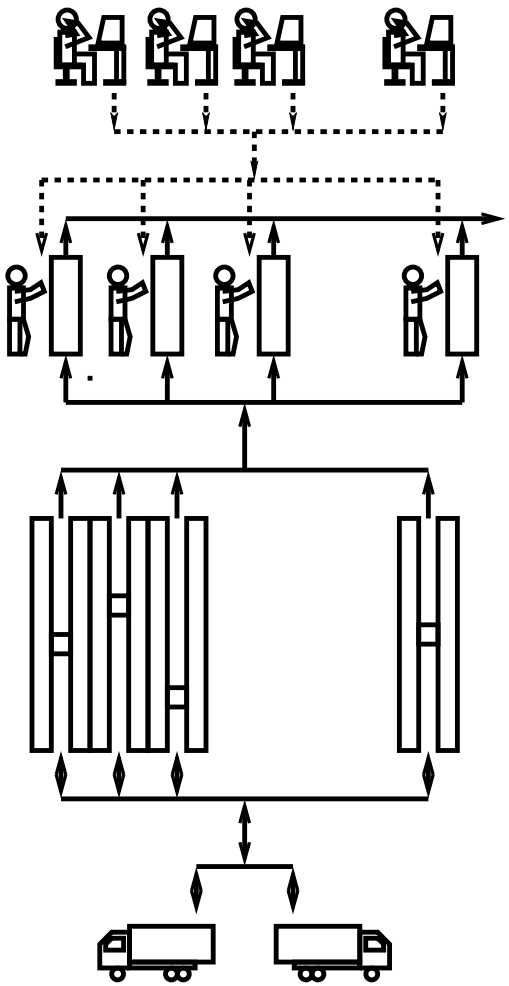
Kopie 2

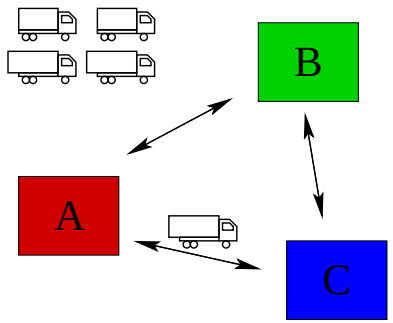
Gebrochen gut lösbar,
ganzzahlig SEHR schwer!

$$
\begin{array}{lrcl}
\min & {c^{(1)}}^T x^{(1)} & + & {c^{(2)}}^T x^{(2)} \\
\text{s.t.} & Ax^{(1)} & & = b^{(1)} \\
& & Ax^{(2)} & = b^{(2)} \\
& Ix^{(1)} & + & Ix^{(2)} \leq w \\
& x^{(1)} \geq 0, & & x^{(2)} \geq 0.
\end{array}
$$

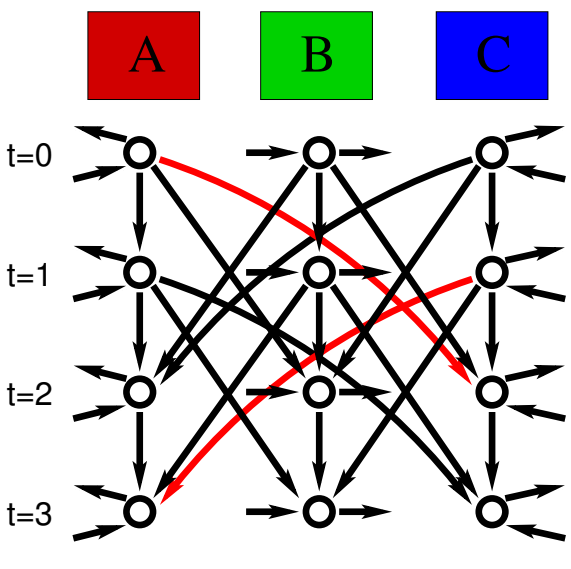
Beispiel: Logistik

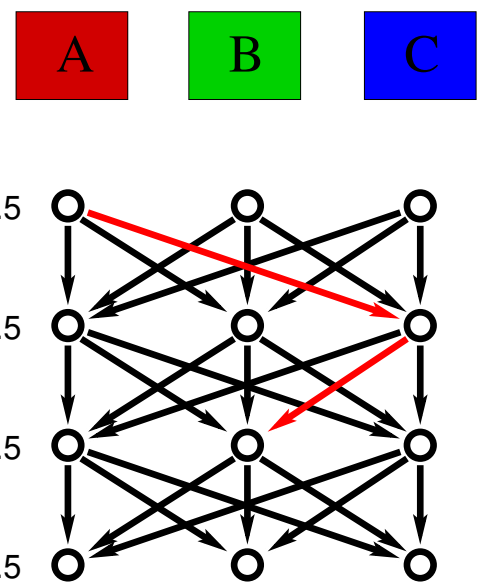
Paletten sind bedarfsgerecht mit LKWs zwischen Lagern zu verschieben





pro Artikel ein Palettengraph





Weitere Anwendungsbereiche

- gebrochen: Kapazitätsplanung
- ganzzahlig: Zeitdiskretisierte Routen- und Ablaufplanung

} für

  - Straßenverkehr
  - Schienenverkehr
  - Internet
  - Logistik (Engpassanalyse/Steuerung)
  - Produktion (Maschinenauslastung/-belegung)

Network-Design:
Auslegung soll Erfüllung möglichst aller Bedarfe auch bei Störung erlauben. [„Robuste“ Varianten sind extrem schwer!]

Mehrgüterflussprobleme werden oft als zugrundeliegendes Modell eingesetzt, das mit weiteren Bedingungen kombiniert wird.

Inhaltsübersicht für heute:

Anwendung: Das Heiratsproblem

Ganzzahligkeit von Polyedern

Anwendung: Netzwerkflüsse

Mehrgüterflussprobleme

Ganzzahlige Optimierung

Ganzzahlige Optimierung (Integer Programming)

vorwiegend: Lineare Programme mit ausschließlich ganzzahligen Variablen
(sonst gemischt-ganzzahlige Optimierung/Mixed Integer Programming)

$$\begin{array}{ll} \max & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \in \mathbb{Z}^n \end{array}$$

Enthält meist viele Binär-Variablen ($\{0,1\}$) für ja/nein Entscheidungen

---

Schwierigkeit: i.A. nicht „effizient“ lösbar, Komplexitätsklasse *NP*
$\Rightarrow$ exakte Lösung oft sehr Enumerations-lastig (system. Durchprobieren)

Exakte Lösung durch Kombination folgender Techniken:

- (obere) Schranken durch lineare/konvexe Relaxation
  verbessert durch Schnittebenenansätze
- zulässige Lösungen (untere Schranken) durch Rundungs- und Lokale-Suche-Heuristiken
- Enumeration durch Branch&Bound, Branch&Cut

Kombinatorische Optimierung

Mathematisch: Gegeben eine endliche Grundmenge Ω, eine Menge zulässiger Teilmengen $\mathcal{F} \subseteq 2^\Omega$ [Potenzmenge, Menge aller Teilmengen] und eine Zielfunktion $c : \mathcal{F} \to \mathbb{Q}$, bestimme

$$\max\{c(F) : F \in \mathcal{F}\} \quad \text{oder} \quad F \in \operatorname{Argmax}\{c(F) : F \in \mathcal{F}\}$$

Hier nur lineares c: $c(F) := \sum_{e \in F} c_e$ mit $c \in \mathbb{Q}^\Omega$.

Bsp1: Matching maximaler Kardinalität in $G = (V, E)$:
$\Omega = E$, $\mathcal{F} = \{M \subseteq E : M \text{ Matching in } G\}$, $c = \mathbf{1}$

Bsp2: Minimum Vertex Cover für $G = (V, E)$:
$\Omega = V$, $\mathcal{F} = \{V' \subseteq V : e \cap V' \neq \emptyset \text{ für } e \in E\}$, $c = -\mathbf{1}$

Formulierung über Binäre Programme:

Notation: Inzidenz-/Charakteristischer Vektor $\chi_\Omega(F) \in \{0,1\}^\Omega$ für $F \subseteq \Omega$
[kurz $\chi(F)$, erfüllt $[\chi(F)]_e = 1 \Leftrightarrow e \in F$]

Ein lineares Programm $\max\{c^T x : Ax \leq b, x \in [0,1]^\Omega\}$ heißt Formulierung des kombinatorischen Optimierungsproblems, falls

$$\{x \in \{0,1\}^\Omega : Ax \leq b\} = \{\chi(F) : F \in \mathcal{F}\}.$$

(Algorithmische) Komplexität von Problemen

Eine Instanz I eines Problems ist eine konkrete Wertebelegung der Problemdaten; ihre Größe $|I|$ ist die Kodierungslänge, also die Anzahl der Zeichen im Beschreibungsstring nach einem sinnvollen Kodierungsschema.

Die Laufzeit eines Algorithmus für eine Instanz ist die Anzahl der ausgeführten elementaren Operationen (ein Symbol lesen/schreiben, Bytes addieren/multiplizieren/vergleichen, etc.)

Ein Algorithmus löst ein Problem polynomial oder effizient, wenn er für jede Instanz die richtige Antwort innerhalb einer Laufzeit liefert, die durch ein Polynom in der Kodierungslänge beschränkt ist.

Ein Problem heißt polynomial/effizient lösbar, wenn es einen Algorithmus gibt, der es effizient löst. Die Klasse P umfasst alle Probleme, die effizient lösbar sind.

Bsp1: Lineare Optimierung ist in P, ABER Simplex löst es nicht effizient.
Bsp2: Kardinalitätsmaximales Matching in allgemeinen Graphen ist in P.
Bsp3: Minimum Vertex Cover in bipartiten Graphen ist in P.

Entscheidungsprobleme und die Klasse *NP*

In einem Entscheidungsproblem ist jede Instanz eine Frage, die entweder mit „Ja“ oder mit „Nein“ zu beantworten ist.

Ein Entscheidungsproblem ist nichtdeterministisch polynomial lösbar, wenn für jede „Ja“-Instanz I ein Lösungsstring (das Zertifikat) existiert, mit dem die Richtigkeit der „Ja“-Antwort in Laufzeit polynomial beschränkt in $|I|$ nachgewiesen werden kann. [Es geht nur um „Ja“!]

Die Klasse NP umfasst alle Entscheidungsprobleme, die nichtdeterministisch polynomial lösbar sind. Es gilt: $P \subseteq NP$

Entscheidungsprobleme und die Klasse *NP*

In einem Entscheidungsproblem ist jede Instanz eine Frage, die entweder mit „Ja“ oder mit „Nein“ zu beantworten ist.

Ein Entscheidungsproblem ist nichtdeterministisch polynomial lösbar, wenn für jede „Ja“-Instanz I ein Lösungsstring (das Zertifikat) existiert, mit dem die Richtigkeit der „Ja“-Antwort in Laufzeit polynomial beschränkt in $|I|$ nachgewiesen werden kann. [Es geht nur um „Ja“!]

Die Klasse NP umfasst alle Entscheidungsprobleme, die nichtdeterministisch polynomial lösbar sind. Es gilt: $P \subseteq NP$

Bsp: **Hamilton'scher Kreis:** In einem Graphen $G = (V, E)$ heißt eine Kantenmenge $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ ein Kreis (der Länge k), falls $v_i \neq v_j$ für $i \neq j$.
Ein Kreis C heißt hamiltonsch, falls $|C| = |V|$.

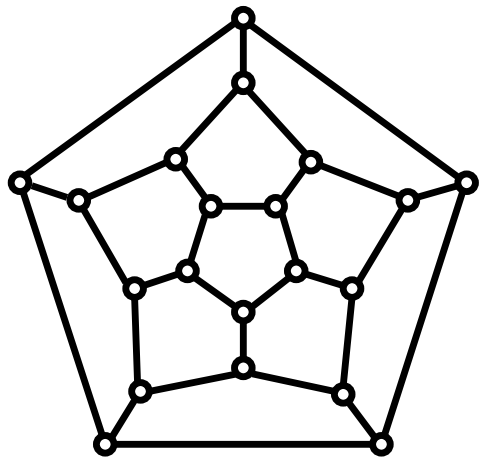
Entscheidungsprobleme und die Klasse *NP*

In einem Entscheidungsproblem ist jede Instanz eine Frage, die entweder mit „Ja“ oder mit „Nein“ zu beantworten ist.

Ein Entscheidungsproblem ist nichtdeterministisch polynomial lösbar, wenn für jede „Ja“-Instanz I ein Lösungsstring (das Zertifikat) existiert, mit dem die Richtigkeit der „Ja“-Antwort in Laufzeit polynomial beschränkt in $|I|$ nachgewiesen werden kann. [Es geht nur um „Ja“!]

Die Klasse NP umfasst alle Entscheidungsprobleme, die nichtdeterministisch polynomial lösbar sind. Es gilt: $P \subseteq NP$

Bsp: **Hamilton'scher Kreis:** In einem Graphen $G = (V, E)$ heißt eine Kantenmenge $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ ein Kreis (der Länge k), falls $v_i \neq v_j$ für $i \neq j$.
Ein Kreis C heißt hamiltonsch, falls $|C| = |V|$.



Entscheidungsproblem:
Enthält G einen Hamilton'schen Kreis?

Ja-Antwort ist effizient überprüfbar,
das Problem ist in *NP*

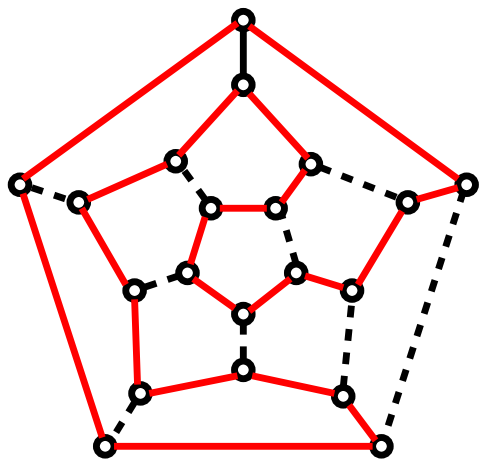
Entscheidungsprobleme und die Klasse *NP*

In einem Entscheidungsproblem ist jede Instanz eine Frage, die entweder mit „Ja“ oder mit „Nein“ zu beantworten ist.

Ein Entscheidungsproblem ist nichtdeterministisch polynomial lösbar, wenn für jede „Ja“-Instanz I ein Lösungsstring (das Zertifikat) existiert, mit dem die Richtigkeit der „Ja“-Antwort in Laufzeit polynomial beschränkt in $|I|$ nachgewiesen werden kann. [Es geht nur um „Ja“!]

Die Klasse NP umfasst alle Entscheidungsprobleme, die nichtdeterministisch polynomial lösbar sind. Es gilt: $P \subseteq NP$

Bsp: **Hamilton'scher Kreis:** In einem Graphen $G = (V, E)$ heißt eine Kantenmenge $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ ein Kreis (der Länge k), falls $v_i \neq v_j$ für $i \neq j$.
Ein Kreis C heißt hamiltonsch, falls $|C| = |V|$.



Entscheidungsproblem:
Enthält G einen Hamilton'schen Kreis?

Ja-Antwort ist effizient überprüfbar,
das Problem ist in *NP*

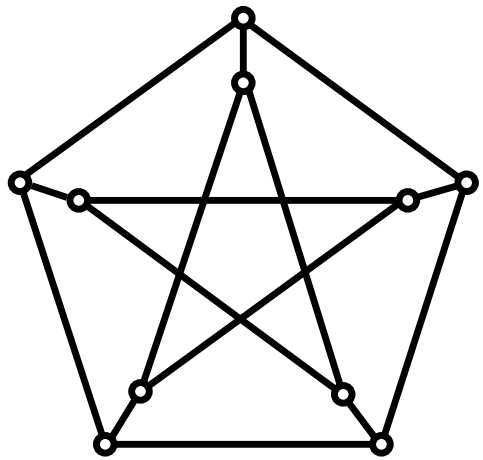
Entscheidungsprobleme und die Klasse *NP*

In einem Entscheidungsproblem ist jede Instanz eine Frage, die entweder mit „Ja“ oder mit „Nein“ zu beantworten ist.

Ein Entscheidungsproblem ist nichtdeterministisch polynomial lösbar, wenn für jede „Ja“-Instanz I ein Lösungsstring (das Zertifikat) existiert, mit dem die Richtigkeit der „Ja“-Antwort in Laufzeit polynomial beschränkt in $|I|$ nachgewiesen werden kann. [Es geht nur um „Ja“!]

Die Klasse NP umfasst alle Entscheidungsprobleme, die nichtdeterministisch polynomial lösbar sind. Es gilt: $P \subseteq NP$

Bsp: **Hamilton'scher Kreis:** In einem Graphen $G = (V, E)$ heißt eine Kantenmenge $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ ein Kreis (der Länge k), falls $v_i \neq v_j$ für $i \neq j$.
Ein Kreis C heißt hamiltonsch, falls $|C| = |V|$.



Entscheidungsproblem:
Enthält G einen Hamilton'schen Kreis?

Ja-Antwort ist effizient überprüfbar,
das Problem ist in *NP*

NP-vollständige Probleme

Ein Entscheidungsproblem P_1 ist polynomial transformierbar auf ein Entscheidungsproblem P_2, wenn es einen Algorithmus gibt, der jede Instanz I_1 von P_1 in Laufzeit polynomial in $|I_1|$ in eine Instanz I_2 von P_2 transformiert, sodass I_2 genau dann Ja-Instanz von P_2 ist, wenn I_1 Ja-Instanz von P_1 ist.

Ist P_1 polynomial auf P_2 transformierbar, dann löst ein effizienter Algorithmus für P_2 auch P_1 effizient; P_2 ist mindestens so schwer wie P_1.

Ist $\bar{P} \in NP$ und ist jedes $\hat{P} \in NP$ polynomial auf $\bar{P}$ transformierbar, so heißt $\bar{P}$ *NP*-vollständig.

Kann ein *NP*-vollständiges Problem $\bar{P}$ polynomial auf ein Problem $\hat{P} \in NP$ transformiert werden, ist auch $\hat{P}$ *NP*-vollständig; sie sind gleich schwer.

NP-vollständige Probleme

Ein Entscheidungsproblem P_1 ist polynomial transformierbar auf ein Entscheidungsproblem P_2, wenn es einen Algorithmus gibt, der jede Instanz I_1 von P_1 in Laufzeit polynomial in $|I_1|$ in eine Instanz I_2 von P_2 transformiert, sodass I_2 genau dann Ja-Instanz von P_2 ist, wenn I_1 Ja-Instanz von P_1 ist.

Ist P_1 polynomial auf P_2 transformierbar, dann löst ein effizienter Algorithmus für P_2 auch P_1 effizient; P_2 ist mindestens so schwer wie P_1.

Ist $\bar{P} \in NP$ und ist jedes $\hat{P} \in NP$ polynomial auf $\bar{P}$ transformierbar, so heißt $\bar{P}$ *NP*-vollständig.

Kann ein *NP*-vollständiges Problem $\bar{P}$ polynomial auf ein Problem $\hat{P} \in NP$ transformiert werden, ist auch $\hat{P}$ *NP*-vollständig; sie sind gleich schwer.

Es gibt dicke Sammlungen *NP*-vollständiger Probleme, dazu gehören:

- ganzzahlige Optimierung (in Entscheidungsversion)
- ganzzahliger Mehrgüterfluss
- Hamilton'scher Kreis
- Minimum Vertex Cover auf allgemeinen Graphen
- Das Rucksack-Problem (für große Zahlen)

NP-vollständige Probleme

Ein Entscheidungsproblem P_1 ist polynomial transformierbar auf ein Entscheidungsproblem P_2, wenn es einen Algorithmus gibt, der jede Instanz I_1 von P_1 in Laufzeit polynomial in $|I_1|$ in eine Instanz I_2 von P_2 transformiert, sodass I_2 genau dann Ja-Instanz von P_2 ist, wenn I_1 Ja-Instanz von P_1 ist.

Ist P_1 polynomial auf P_2 transformierbar, dann löst ein effizienter Algorithmus für P_2 auch P_1 effizient; P_2 ist mindestens so schwer wie P_1.

Ist $\bar{P} \in NP$ und ist jedes $\hat{P} \in NP$ polynomial auf $\bar{P}$ transformierbar, so heißt $\bar{P}$ *NP*-vollständig.

Kann ein *NP*-vollständiges Problem $\bar{P}$ polynomial auf ein Problem $\hat{P} \in NP$ transformiert werden, ist auch $\hat{P}$ *NP*-vollständig; sie sind gleich schwer.

Gibt es einen effizienten Algorithmus für eines der *NP*-vollständigen Probleme, sind alle effizient lösbar. Man vermutet seit Jahren: $P \neq NP$.

Will man alle Instanzen lösen können, ist wahrscheinlich eine teilweise Enumeration unvermeidbar.

Ein Problem ist *NP*-schwer, wenn damit ein *NP*-vollständiges lösbar wäre.