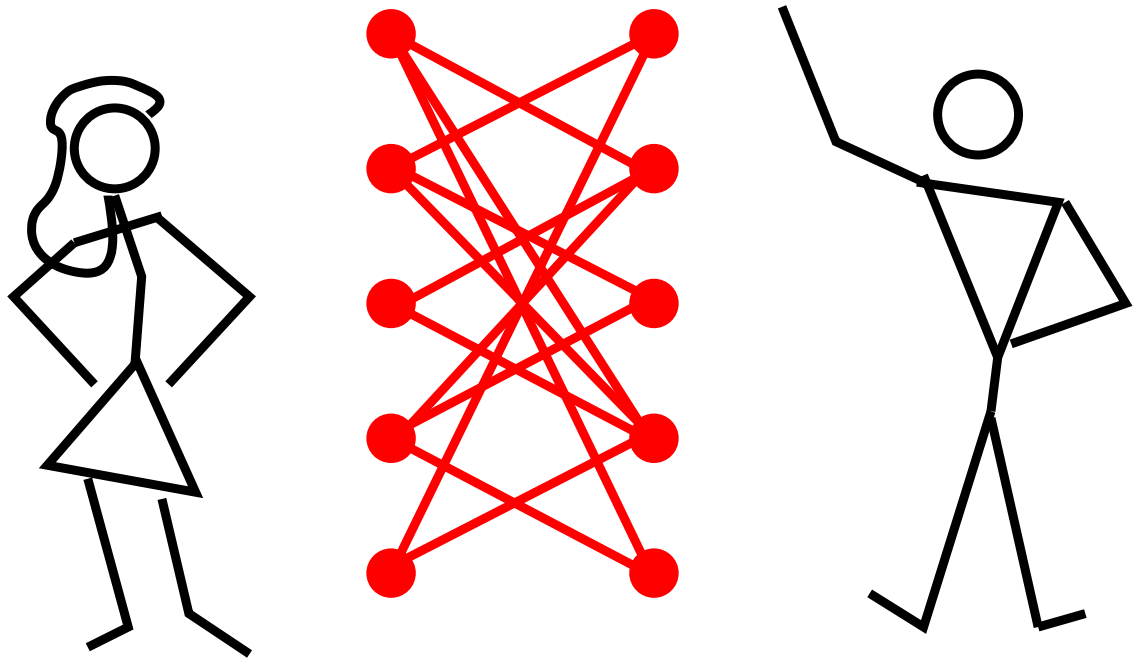
Diskrete Optimierung

(Einführung zur Vorlesung)

Christoph Helmberg

Inhaltsübersicht

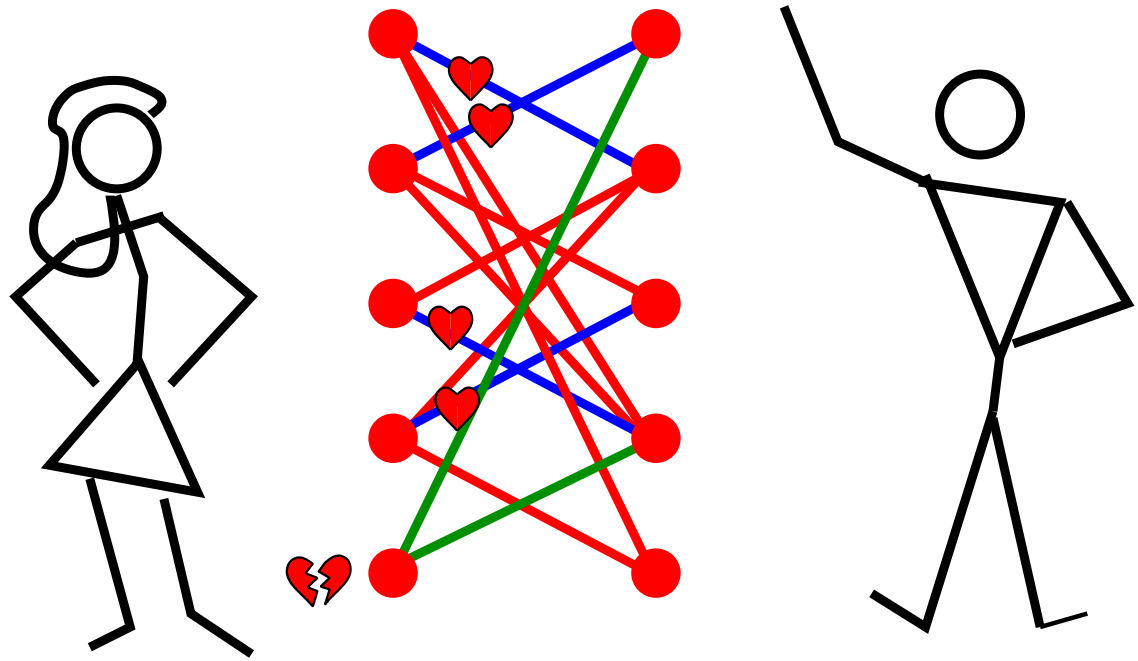
1.1 Anwendung: Das Heiratsproblem (bipartites Matching)



Bilde möglichst viele Paare!

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

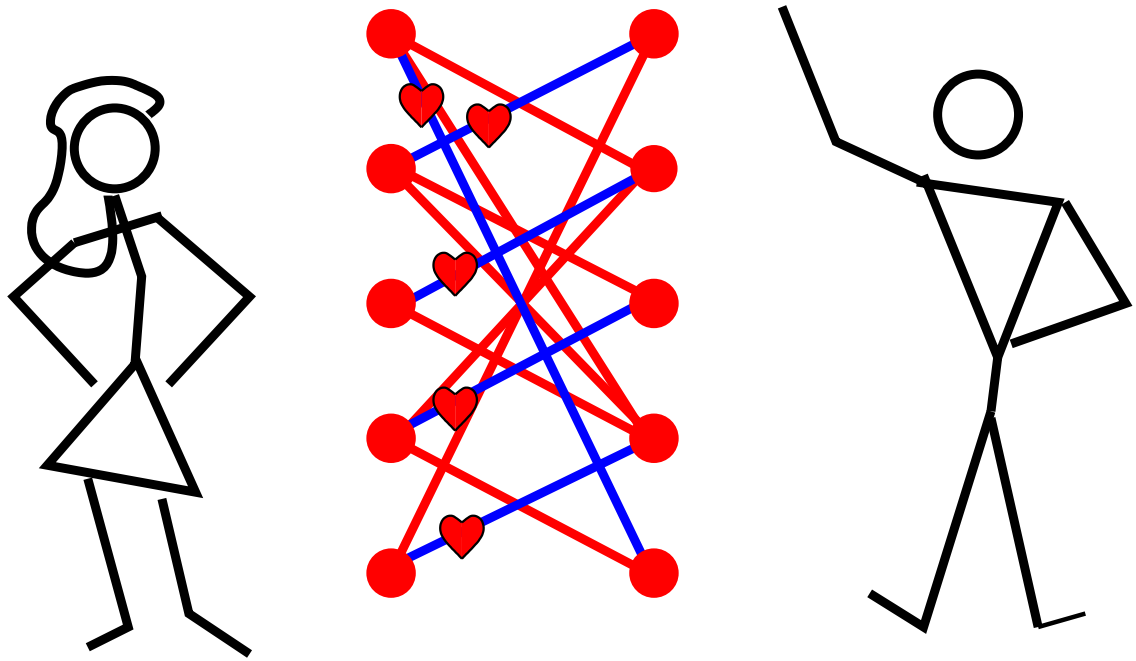
1.1 Anwendung: Das Heiratsproblem (bipartites Matching)



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

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

1.1 Anwendung: Das Heiratsproblem (bipartites Matching)

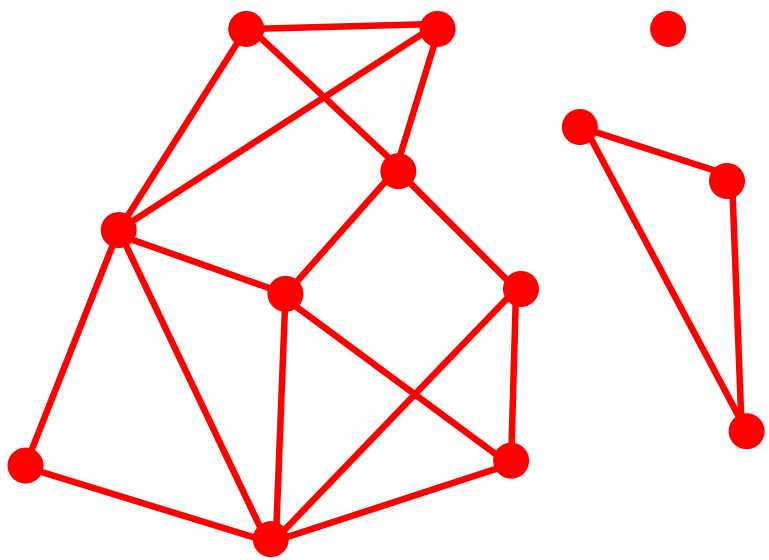


Bilde möglichst viele Paare!
Maximum Cardinality Matching (sogar perfekt)

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

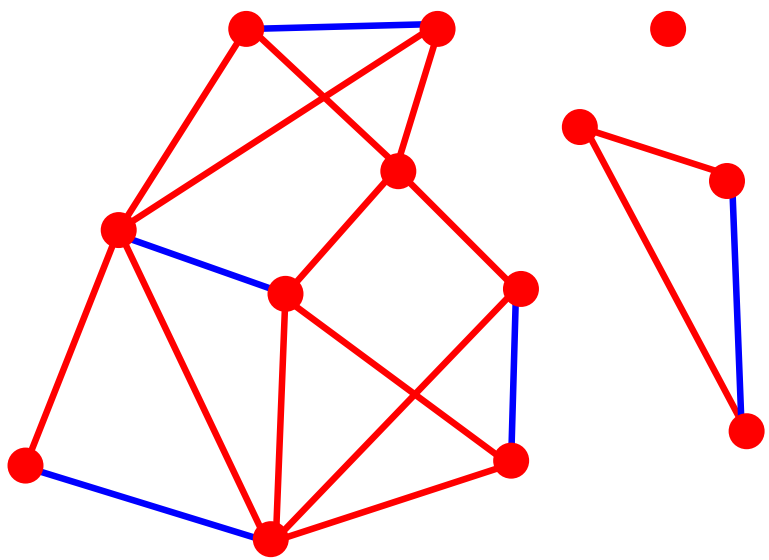
Bipartites Matching

- 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 $e \cap f \neq \emptyset$.



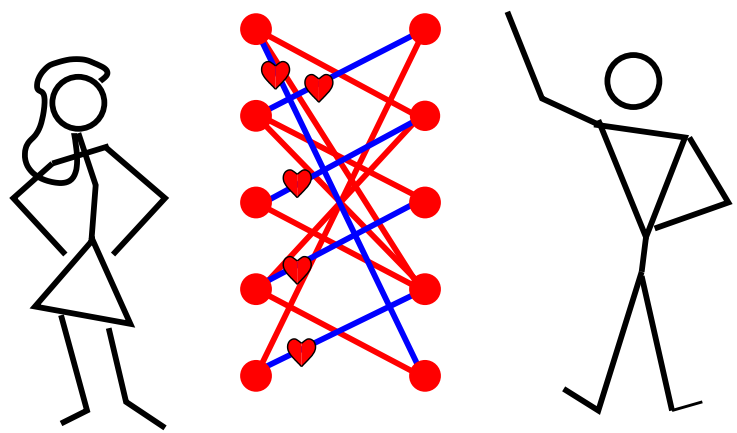
Bipartites Matching

- 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

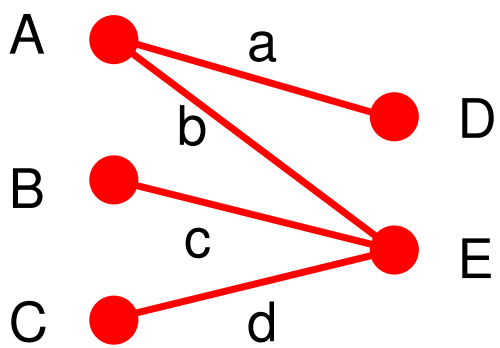
Variablen: $x \in \{0,1\}^E$ mit $x_e = \begin{cases} 1 & \text{falls } e \in M \\ 0 & \text{sonst.} \end{cases} \quad (e \in E)$

(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 $x_e = \begin{cases} 1 & \text{falls } e \in M \\ 0 & \text{sonst.} \end{cases} \quad (e \in E)$

(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

1.2 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?

1.2 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 $x_B \in \mathbb{Z}^m$.

1.2 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.

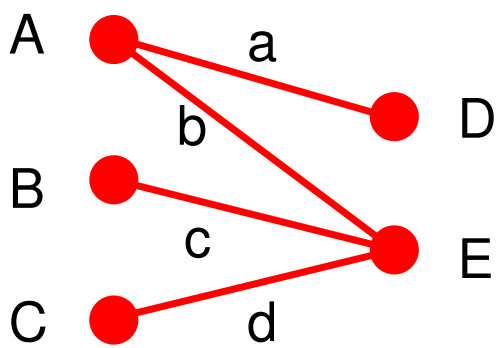
Beachte: A tot. unimod. $\Leftrightarrow$ A^T bzw. $[A, -A, I, -I]$ tot. unimod.
Konsequenz: duales LP, Gleichungsvarianten, etc. sind ganzzahlig

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:

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

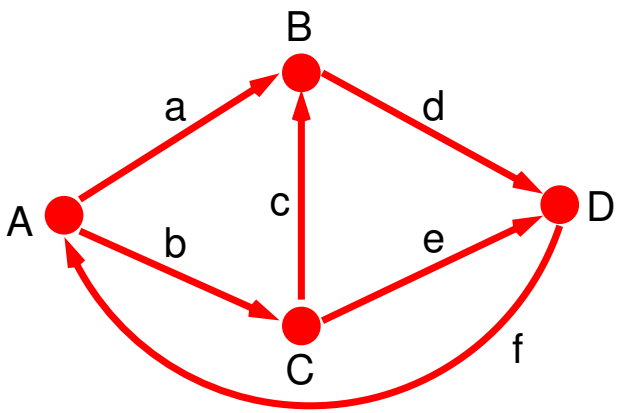
ist auch ganzzahlig, weil $[A; -A]$ tot. unimod.

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

1.3 Anwendung: Netzwerkflüsse

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

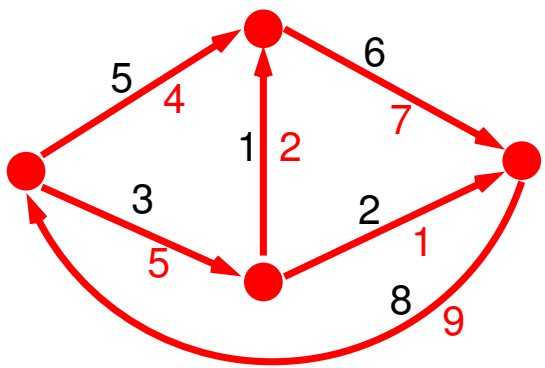
- Ein Netzwerk (D, w) besteht aus einem Digraphen $D = (V, E)$ und (Kanten-)Kapazitäten $w \in \mathbb{R}_+^E$.
- Ein Vektor $x \in \mathbb{R}^E$ heißt Fluss auf (D, w), falls er die Flusserhaltungsgleichungen

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

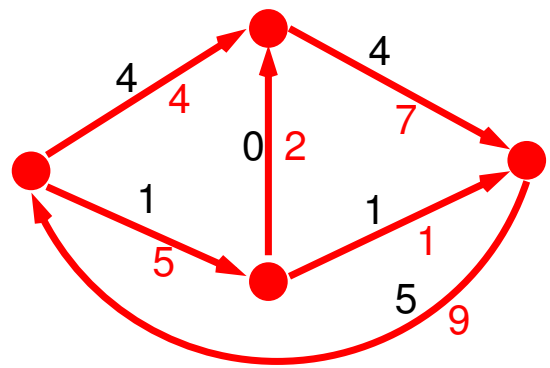
erfüllt. [$\Leftrightarrow Ax = 0$ für Knoten-Kanten-Inzidenzmatrix A]

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

[auch untere Schranken machbar]



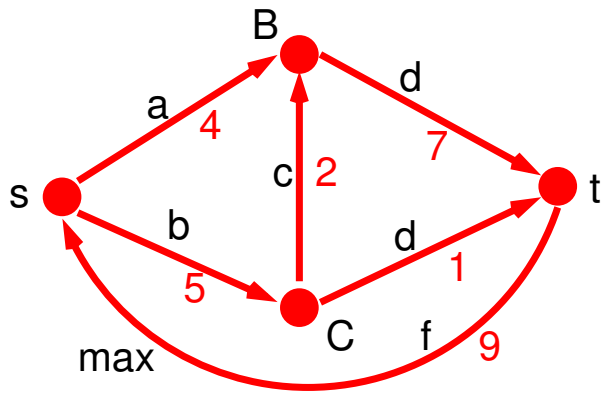
Fluss



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 $x_{(t,s)}$.



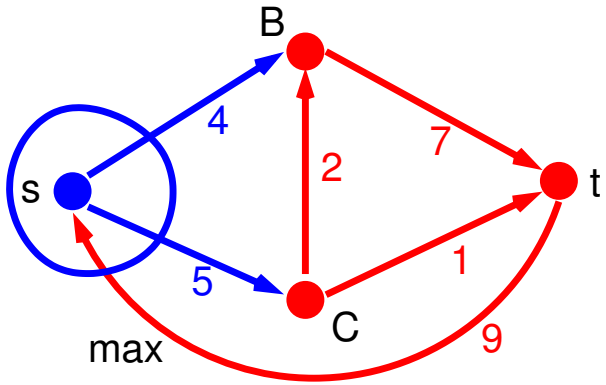
$$A = \begin{bmatrix} & (a) & (b) & (c) & (d) & (e) & (f) \\ (s) & -1 & -1 & 0 & 0 & 0 & 1 \\ (B) & 1 & 0 & 1 & -1 & 0 & 0 \\ (C) & 0 & 1 & -1 & 0 & -1 & 0 \\ (t) & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

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

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. $Ax = 0,\ 0 \leq x \leq w$,

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

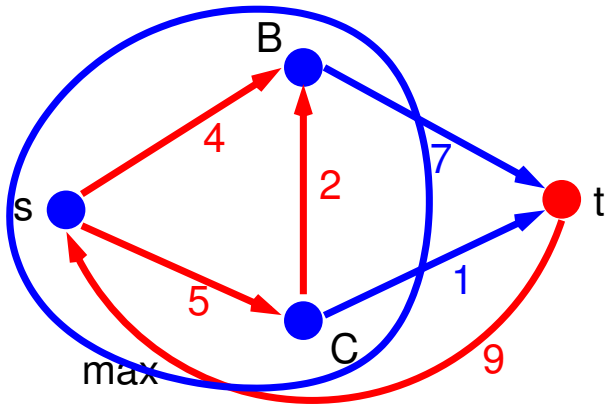
Jedes $S \subseteq V$ mit $s \in S$ und $t \notin S$ definiert einen s-t-Schnitt

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

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 $x_{(t,s)}$.



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

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.

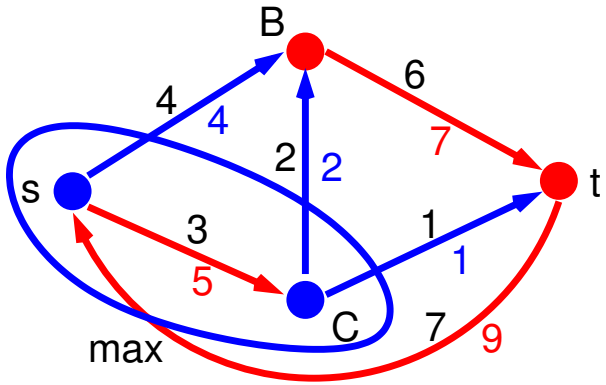
Jedes $S \subseteq V$ mit $s \in S$ und $t \notin S$ definiert einen s-t-Schnitt

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

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 $x_{(t,s)}$.



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

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

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

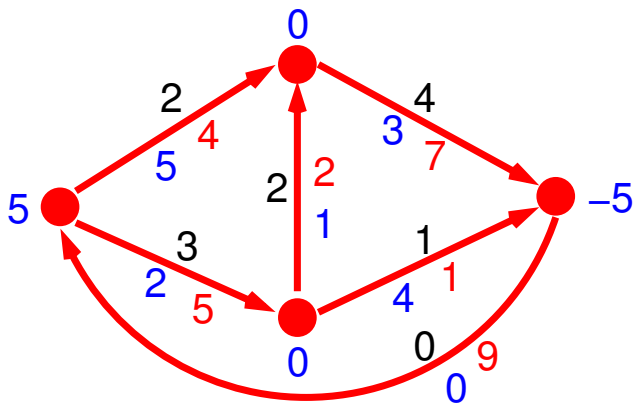
darüber fließt höchstens $w(\delta^+(S)) := \sum_{e \in \delta^+(S)} w_e$, der Wert des Schnitts.

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.



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

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

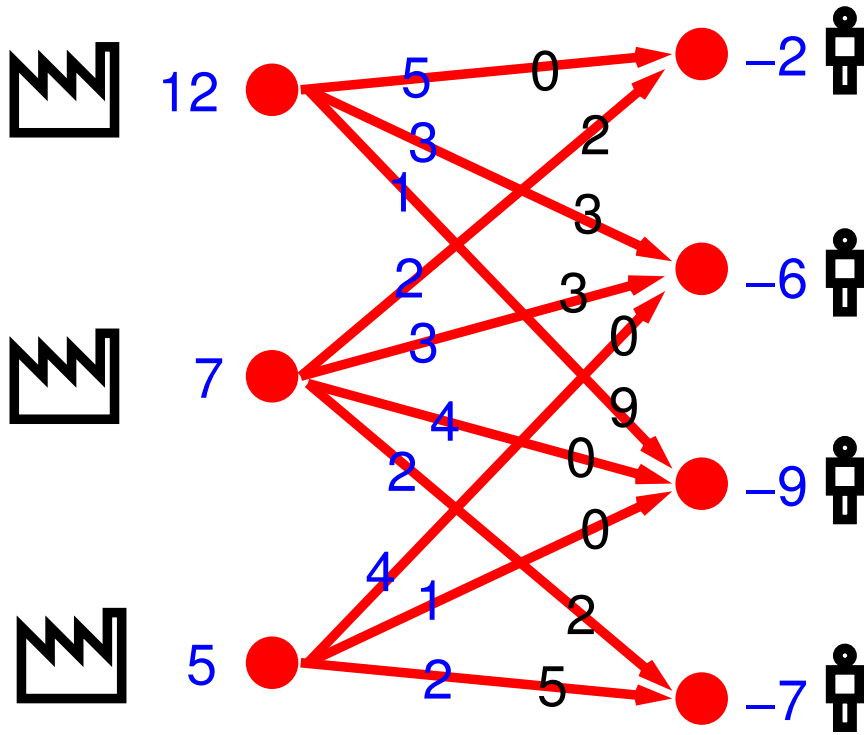
Für b, c und w ganzz. ist Simplex-OL $x^* \in \mathbb{Z}^E$, weil $[A; -A; I]$ tot. unimod.
[geht auch mit unteren Schranken auf den Kanten: $u \le x \le w$!]

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

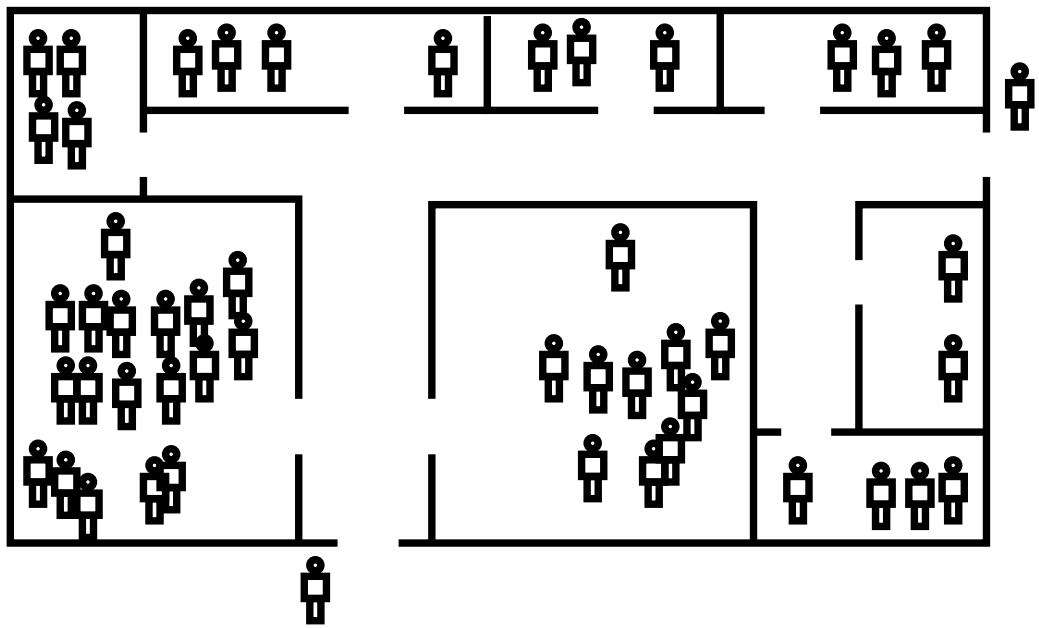
Eine Firma mit mehreren Produktionsstandorten hat mehrere Kunden zu beliefern. Wie geschieht dies am günstigsten unter Berücksichtigung der Transportkosten?



Beachte: Nur ein Produkttyp!

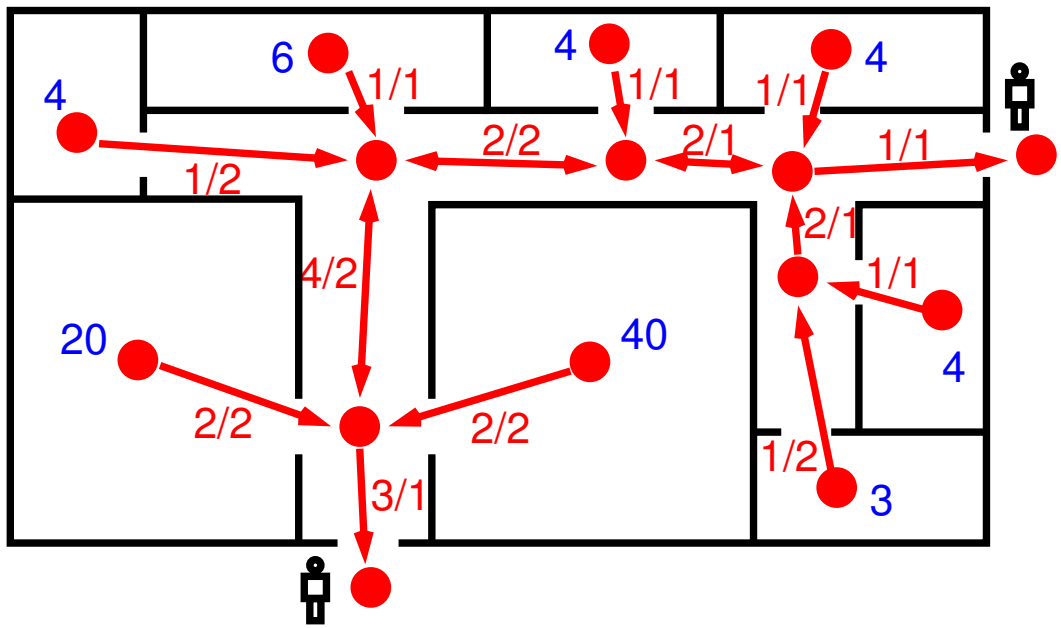
Beispiel: Evakuierungsplanung

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



Beispiel: Evakuierungsplanung

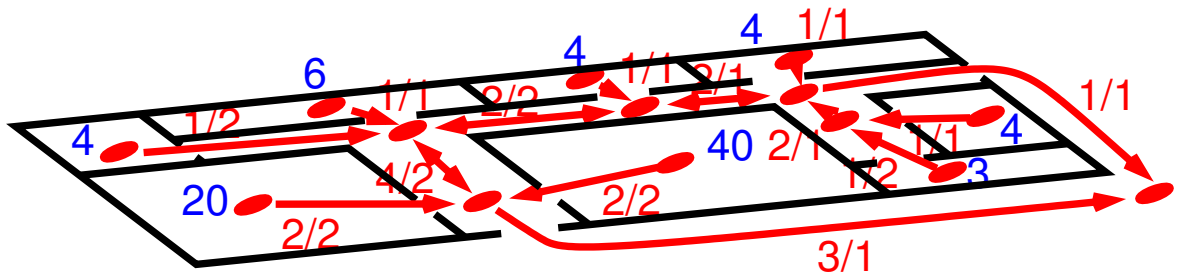
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Kanten mit Kapazität und Durchquerungszeit

Beispiel: Evakuierungsplanung

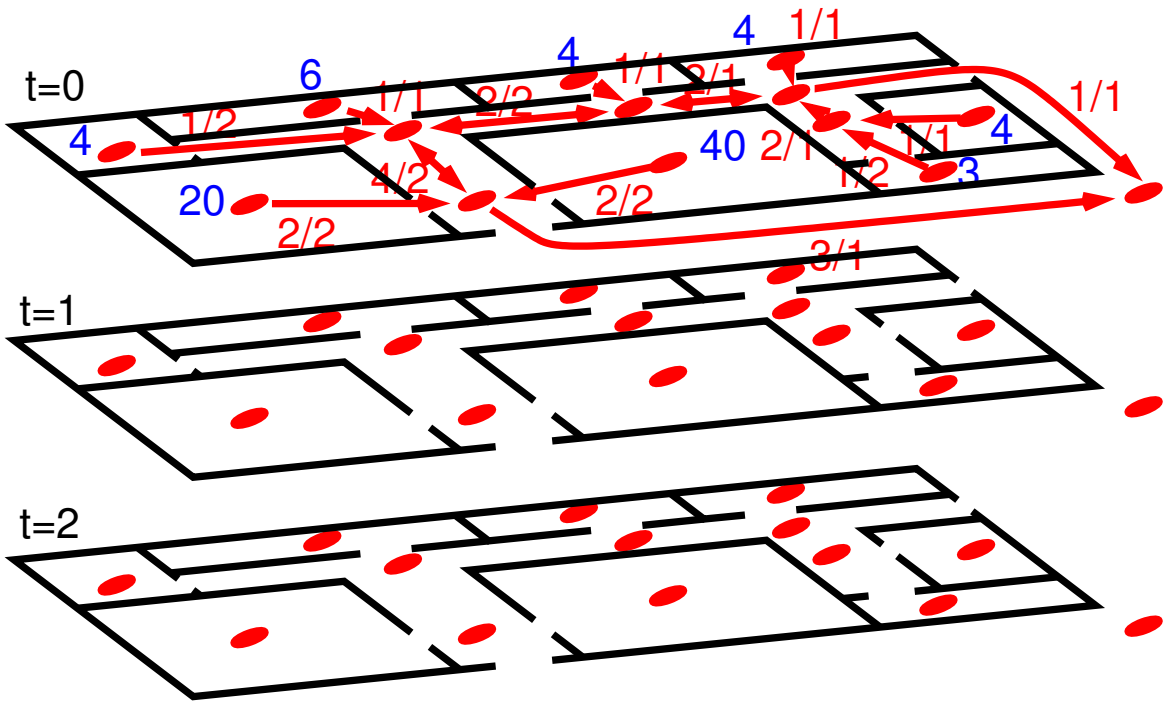
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Geometrie nicht wichtig, vereinfachbar

Beispiel: Evakuierungsplanung

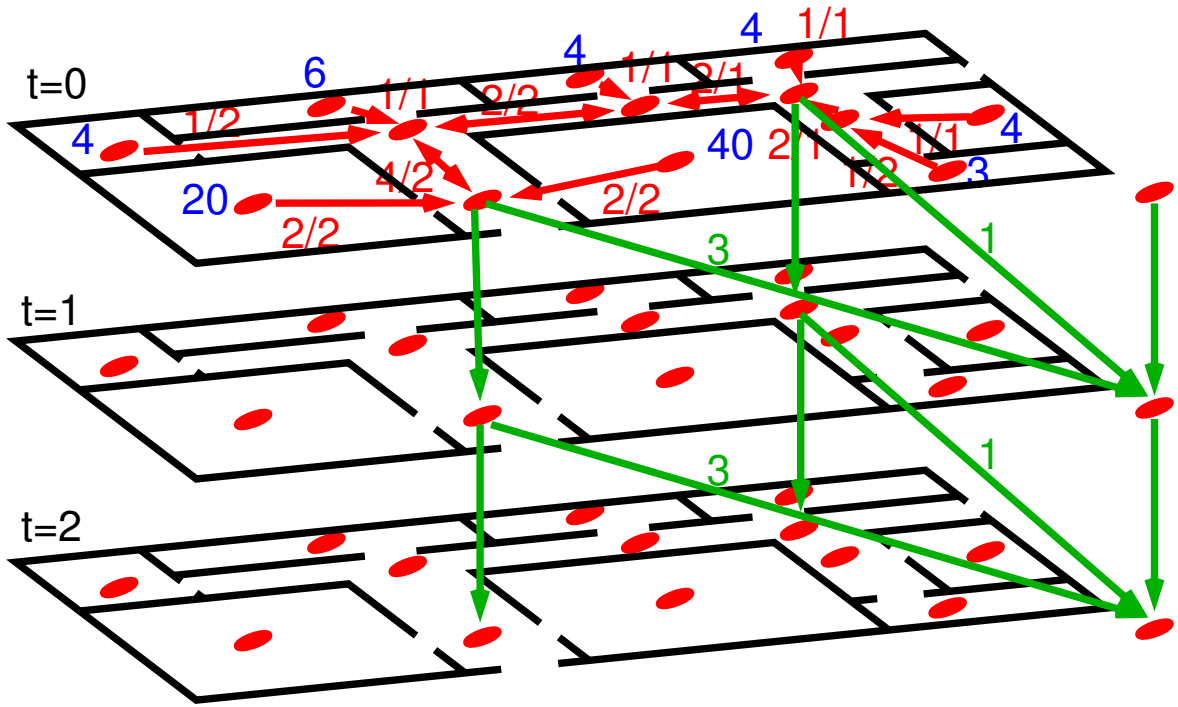
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Diskretisierung der Zeit, ein Niveau pro Zeiteinheit

Beispiel: Evakuierungsplanung

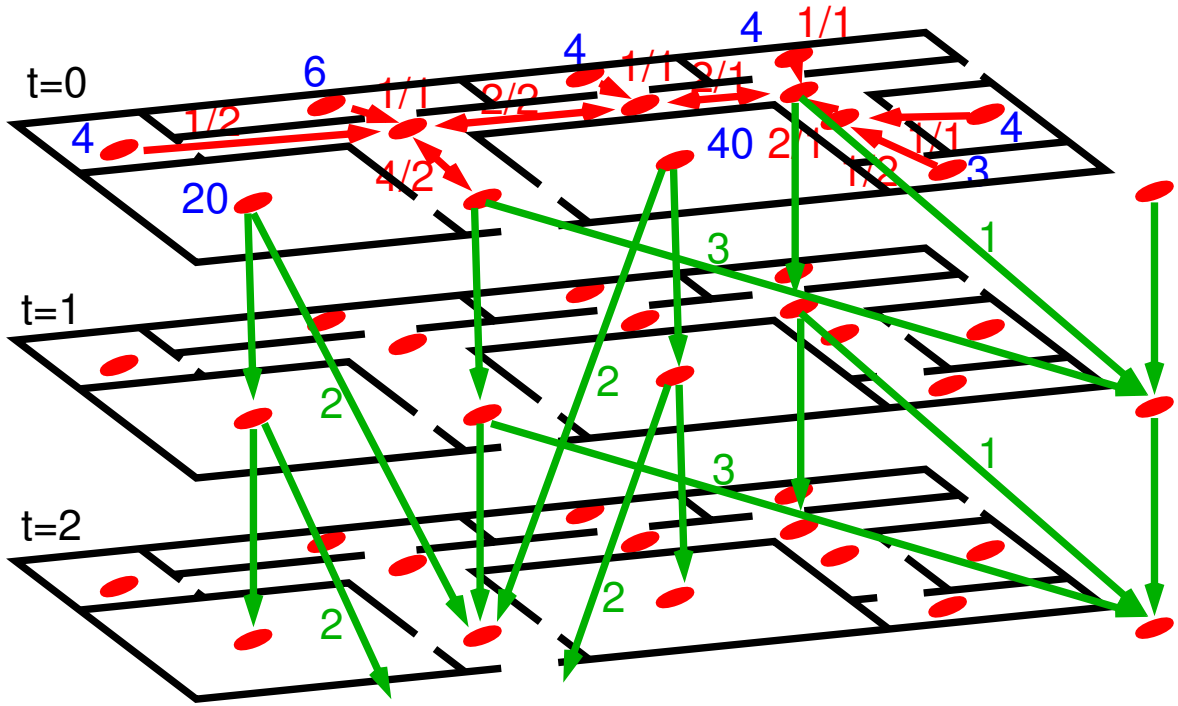
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Durchquerungszeit verbindet Niveaus, Kapazität bleibt

Beispiel: Evakuierungsplanung

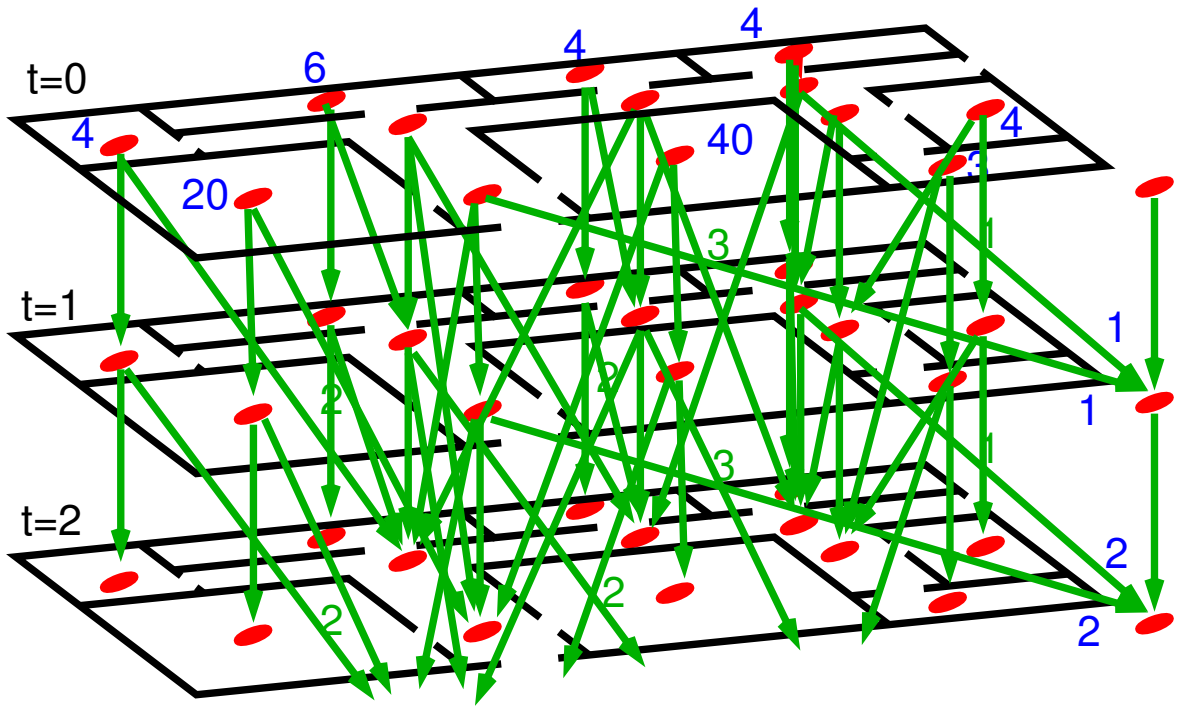
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



Durchquerungszeit verbindet Niveaus, Kapazität bleibt

Beispiel: Evakuierungsplanung

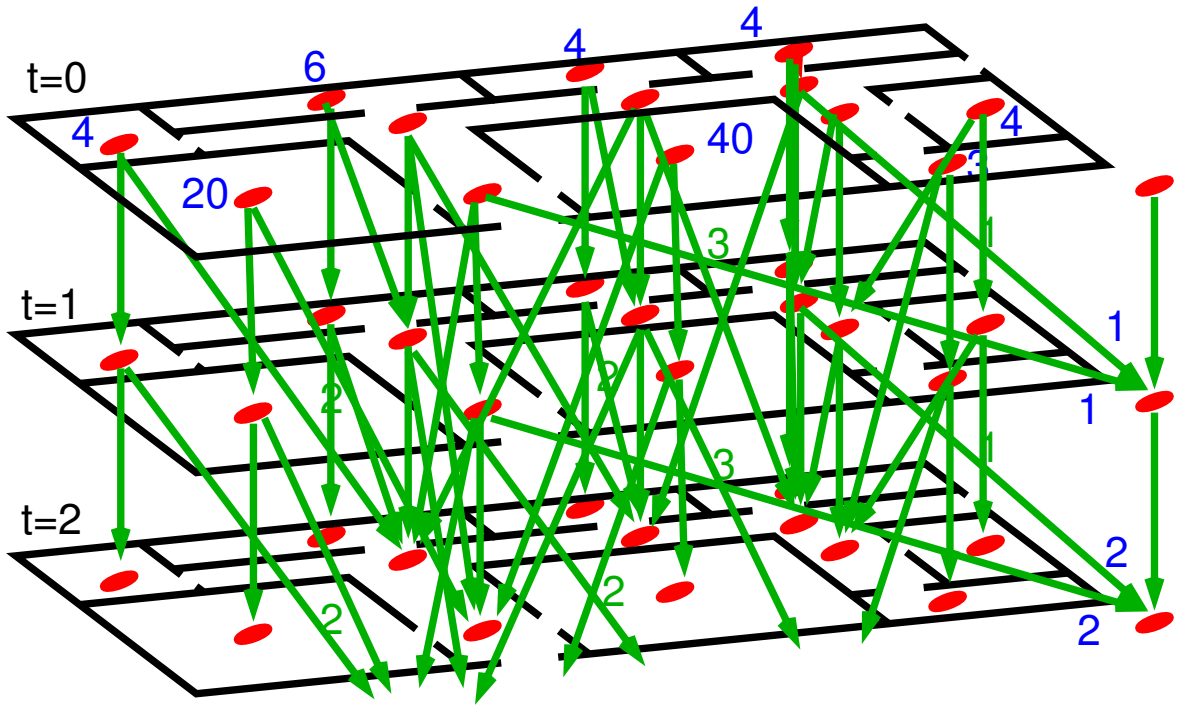
Bestimme für jeden Raum den Fluchtweg, sodass das Gebäude schnellstmöglich geräumt ist. Pro Abschnitt sind Kapazität pro Zeiteinheit und Durchquerungszeit bekannt.



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.



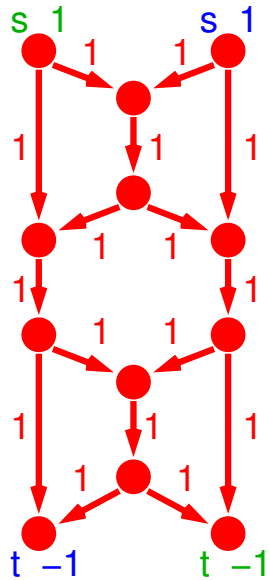
Steigende Kosten auf den Ausgangskanten für rasches Verlassen

Ansatz nur ok, wenn Personen nicht unterschieden werden müssen!

Inhaltsübersicht

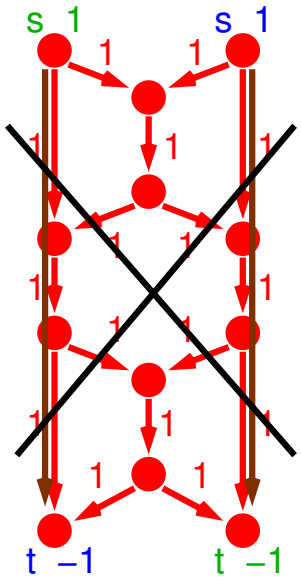
1.4 Mehrgüterflussprobleme (Multicommodity Flow)

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.



1.4 Mehrgüterflussprobleme (Multicommodity Flow)

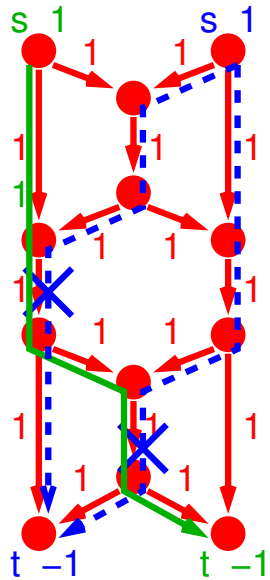
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



Mischen verboten!

1.4 Mehrgüterflussprobleme (Multicommodity Flow)

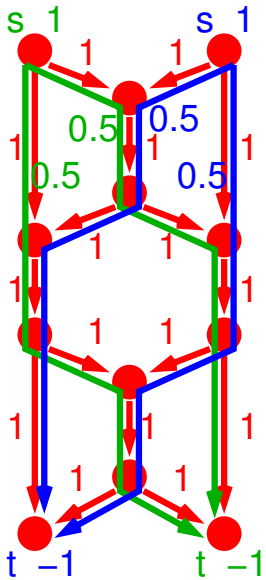
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



ganzz. geht nicht

1.4 Mehrgüterflussprobleme (Multicommodity Flow)

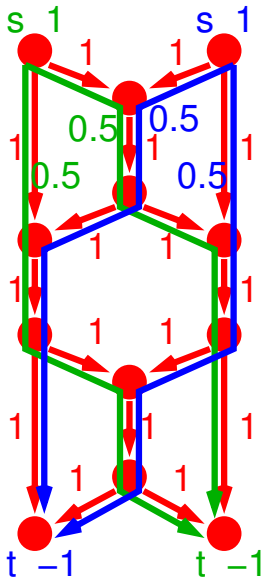
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



gebrochen geht

1.4 Mehrgüterflussprobleme (Multicommodity Flow)

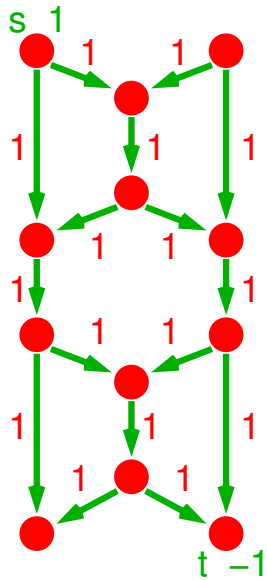
In einem Neztwerk (D, w) sind für unterschiedliche Güter $K = \{1, \ldots, k\}$ mit Quellen/Senken (s_i, t_i) und Flusswerten f_i, $i \in K$, zulässige Flüsse $x^{(i)} \in \mathbb{R}^E$ zu finden, die in Summe die Kapazitätsbedingungen einhalten.



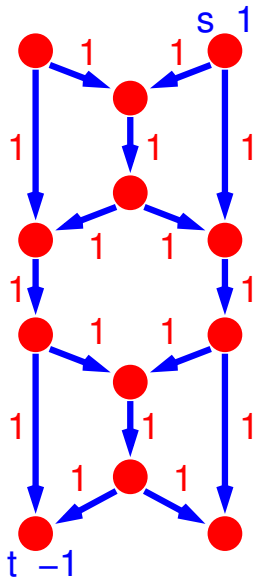
gebrochen geht

Umsetzung

$\longrightarrow$



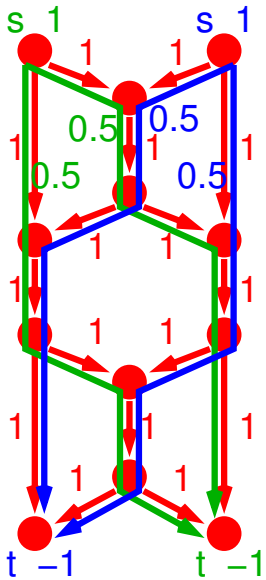
Kopie 1



Kopie 2

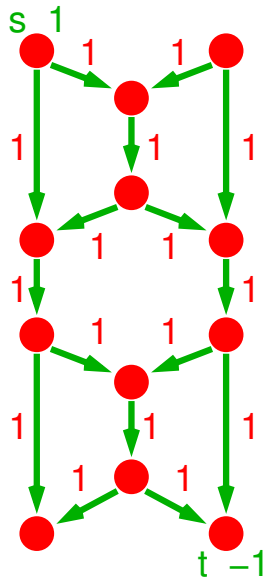
1.4 Mehrgüterflussprobleme (Multicommodity Flow)

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.

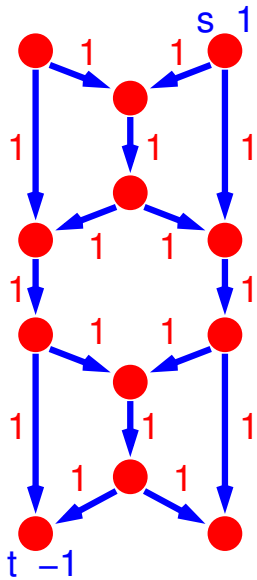


gebrochen geht

Umsetzung $\longrightarrow$



Kopie 1

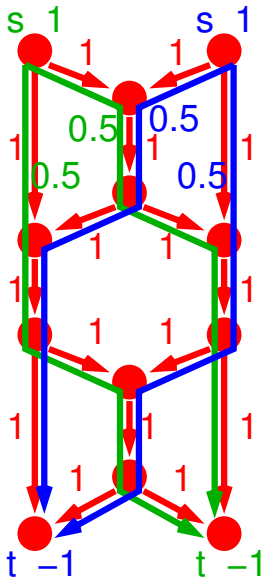


Kopie 2

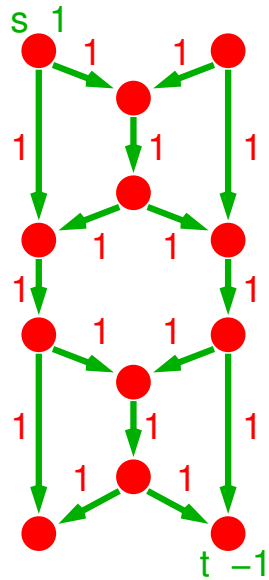
$$
\begin{array}{lccc}
\min & c^{(1)\,T} x^{(1)} + c^{(2)\,T} x^{(2)} & & \\
\text{s.t.} & Ax^{(1)} & = & b^{(1)} \\
 & Ax^{(2)} & = & b^{(2)} \\
 & Ix^{(1)} + Ix^{(2)} & \leq & w \\
 & x^{(1)} \geq 0, \quad x^{(2)} \geq 0. & &
\end{array}
$$

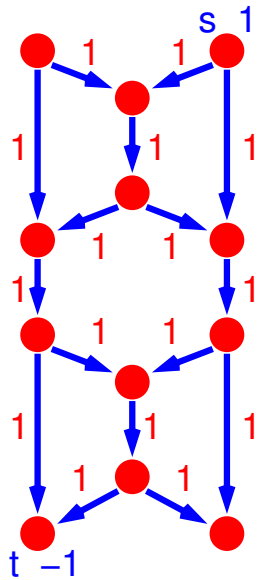
1.4 Mehrgüterflussprobleme (Multicommodity Flow)

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

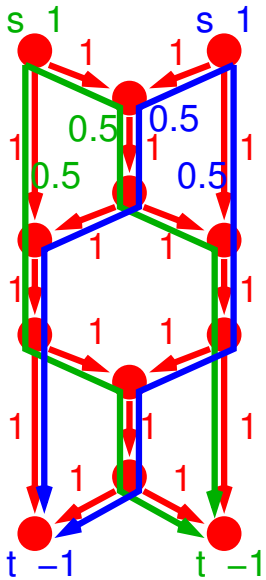
$$\begin{bmatrix} A & 0 \\ 0 & A \\ I & I \end{bmatrix}$$

i.A. nicht tot.unimod.!

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

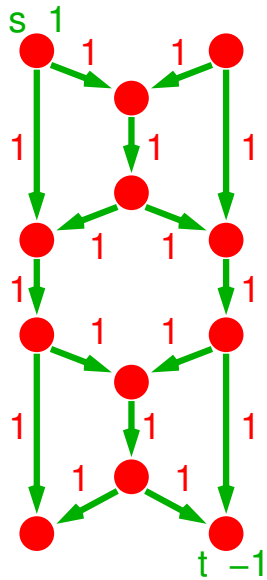
1.4 Mehrgüterflussprobleme (Multicommodity Flow)

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.

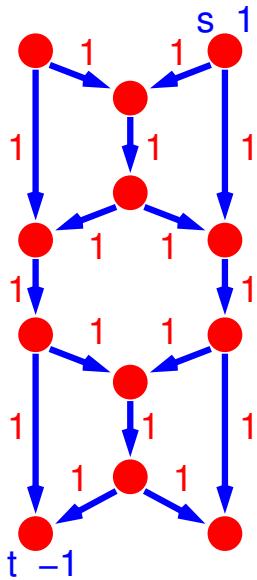


gebrochen geht

Umsetzung $\longrightarrow$



Kopie 1



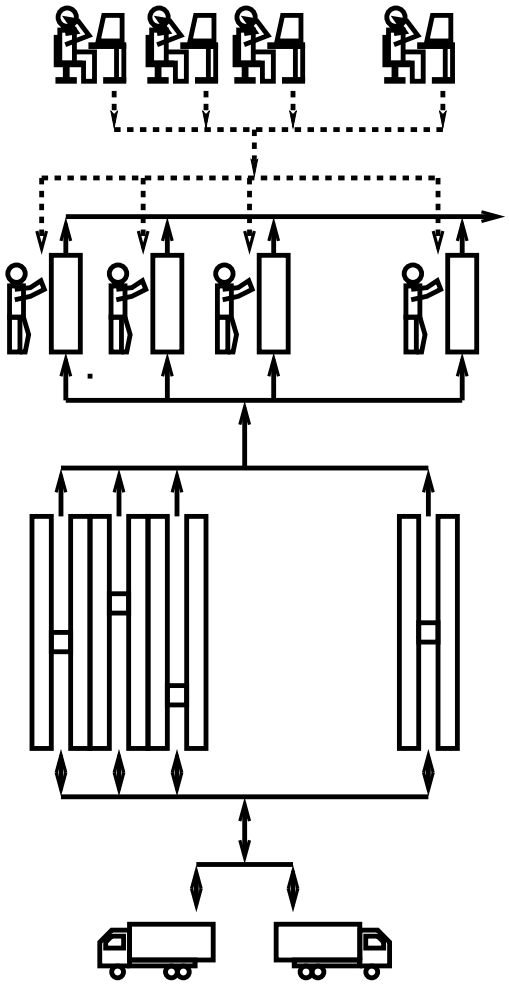
Kopie 2

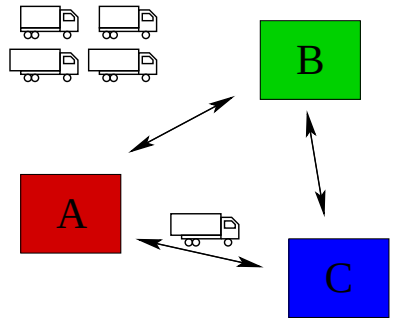
Gebrochen gut lösbar,
ganzzahlig SEHR schwer!

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

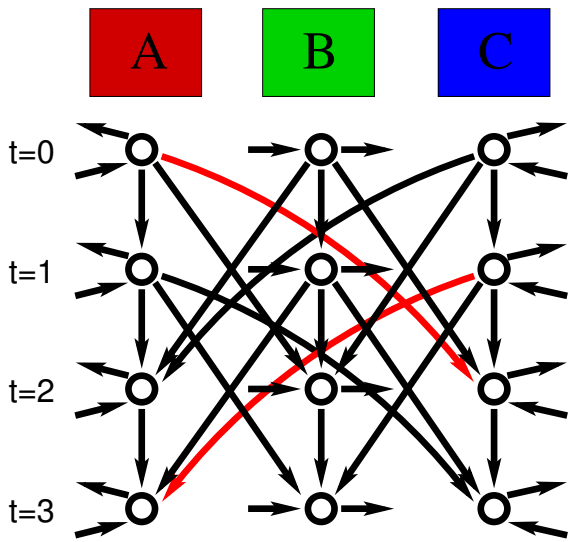
Beispiel: Logistik

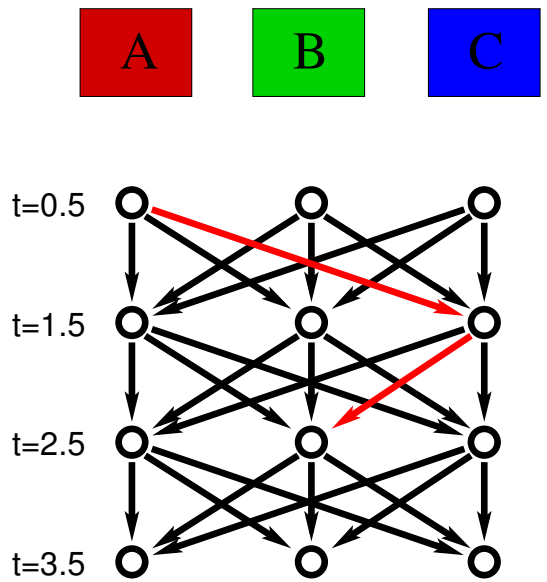
Paletten sind bedarfsgerecht mit LKWs zwischen Lagern zu verschieben





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

1.5 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 $|C| = |V|$.

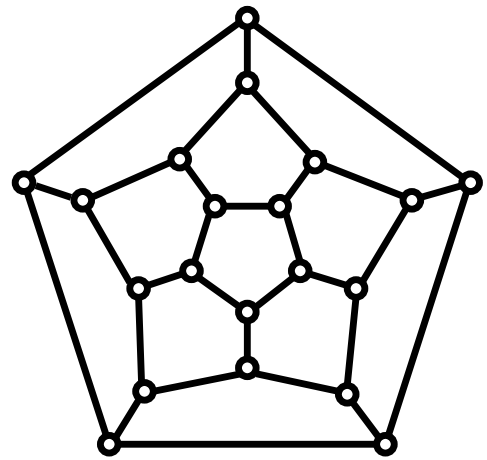
Entscheidungsprobleme und die Klasse *NP*

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*

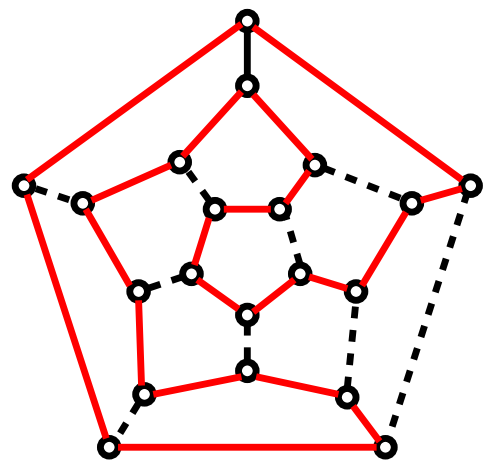
Entscheidungsprobleme und die Klasse *NP*

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*

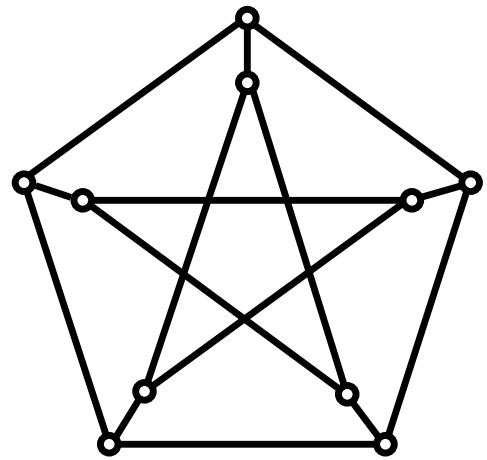
Entscheidungsprobleme und die Klasse *NP*

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.

Inhaltsübersicht

1.6 Branch-and-Bound

Beim systematischen Enumerieren aller Lösungen sollen möglichst viele frühzeitig durch obere und untere Schranken ausgeschlossen werden.

Bsp: $\{0,1\}$-Rucksack: Gewichte $a \in \mathbb{N}^n$, Kapazität $b \in \mathbb{N}$, Nutzen $c \in \mathbb{N}^n$,

$$\max\ c^T x \quad \text{s.t.} \quad a^T x \leq b,\ x \in \{0,1\}^n$$

Obere Schranke: $\max\ c^T x$ s.t. $a^T x \leq b,\ x \in [0,1]^n$ [LP-Relaxation]
Untere Schranke: Sortiere nach Nutzen/Gewicht und fülle danach auf

---

Ablaufskizze (für Maximierungsprobleme):
M ... Menge offener Probleme, anfangs $M = \{\text{Ursprungsproblem}\}$
$\underline{f}$ ... Wert der besten bekannten Lösung, anfangs $\underline{f} = -\infty$
1. Falls $M = \emptyset$ STOP, sonst wähle $P \in M$, $M \leftarrow M \setminus \{P\}$
2. Berechne obere Schranke $\bar{f}(P)$.
3. Falls $\bar{f}(P) < \underline{f}$ (P enthält keine OL), gehe zu 1.
4. Berechne zulässige Lösung $\hat{f}(P)$ für P (untere Schranke).
5. Ist $\hat{f}(P) > \underline{f}$ (neue beste Lösung), setze $\underline{f} \leftarrow \hat{f}(P)$
6. Ist $\bar{f}(P) = \hat{f}(P)$ (keine bessere Lösung in P), gehe zu 1.
7. Teile P in „kleinere“ Teilprobleme P_i, $M \leftarrow M \cup \{P_1, \ldots, P_k\}$
8. Gehe zu 1.

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

P_1: Originalproblem
OS: $C + \frac{8}{9}A = 34$
US: $C + D + F = 30$

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

P_1: Originalproblem
OS: $C + \frac{8}{9}A = 34$
US: $C + D + F = 30$

$$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$$

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

P_1: Originalproblem
OS: $C + \frac{8}{9}A = 34$
US: $C + D + F = 30$

$$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
OS: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
OS $< 30 \Rightarrow$ keine OL $\square$

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

P_1: Originalproblem
OS: $C + \frac{8}{9}A = 34$
US: $C + D + F = 30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
OS: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
OS $< 30 \Rightarrow$ keine OL $\square$

P_3: $x_A = 0$
OS: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
US: $C + D + F = 30$

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

P_1: Originalproblem
OS: $C + \frac{8}{9}A = 34$
US: $C + D + F = 30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
OS: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
OS $< 30 \Rightarrow$ keine OL $\square$

P_3: $x_A = 0$
OS: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
US: $C + D + F = 30$

$1 \quad \leftarrow \quad x_E \quad \rightarrow \quad 0$

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

P_1: Originalproblem
OS: $C + \frac{8}{9}A = 34$
US: $C + D + F = 30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
OS: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
OS $< 30 \Rightarrow$ keine OL □

P_3: $x_A = 0$
OS: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
US: $C + D + F = 30$

$1 \quad \leftarrow \quad x_E \quad \rightarrow \quad 0$

P_4: $x_A = 0,\ x_E = 1$
OS: $E + C + D = 31$
US: $E + C + D = 31$ □

Beispiel: $\{0,1\}$-Rucksackproblem

Gegenstand	A	B	C	D	E	F	Kapazität
Gewicht(a)	9	7	6	4	4	3	14
Nutzen (c)	18	6	18	7	6	5	

Sortierung Nutzen/Gewicht: $C > A > D > F > E > B$.

Obere Schranke: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-Relaxation]

Untere Schranke: Nach Sortierung möglichst lange auffüllen

P_1: Originalproblem
OS: $C + \frac{8}{9}A = 34$
US: $C + D + F = 30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
OS: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
OS $< 30 \Rightarrow$ keine OL □

P_3: $x_A = 0$
OS: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
US: $C + D + F = 30$

$1 \quad \leftarrow \quad x_E \quad \rightarrow \quad 0$

P_4: $x_A = 0,\ x_E = 1$
OS: $E + C + D = 31$
US: $E + C + D = 31$ □

P_5: $x_A = x_E = 0$
OS: $C + D + F + \frac{1}{7}B = 30\frac{6}{7}$
OS $< 31 \Rightarrow$ keine OL □

Immer dann wird der Branch&Bound Baum groß werden, wenn viele Lösungen sehr nahe an der Optimallösung sind.

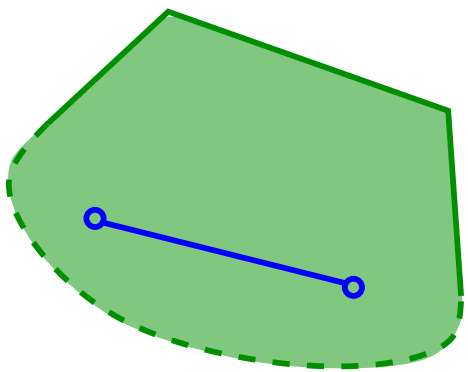
Entscheidend für den Erfolg von Branch&Bound:

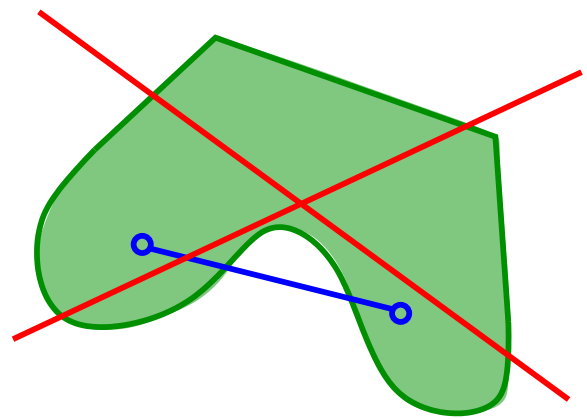
Wie kommt man zu guten oberen und unteren Schranken?

Inhaltsübersicht

1.7 Konvexe Mengen und konvexe Hülle

Eine Menge $C \subseteq \mathbb{R}^n$ heißt konvex, wenn für alle $x, y \in C$ auch die Verbindungsstrecke $[x, y] := \{\alpha x + (1 - \alpha)y : \alpha \in [0, 1]\}$ in C liegt.



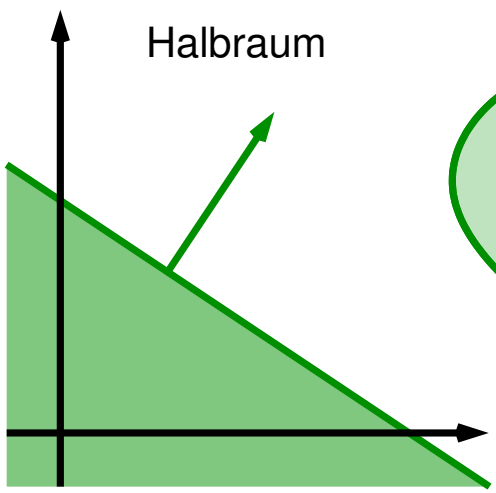


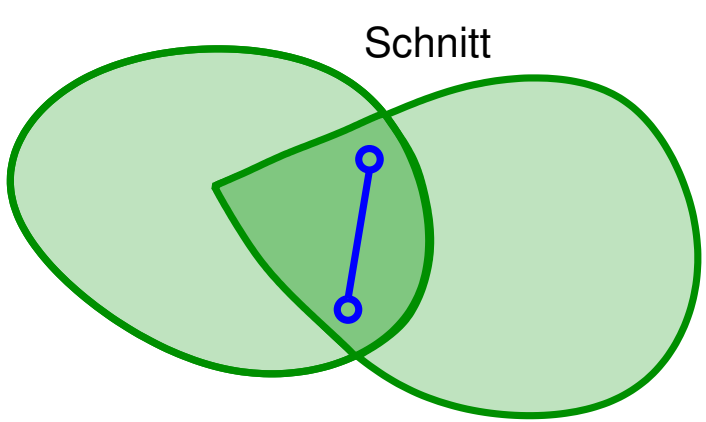
1.7 Konvexe Mengen und konvexe Hülle

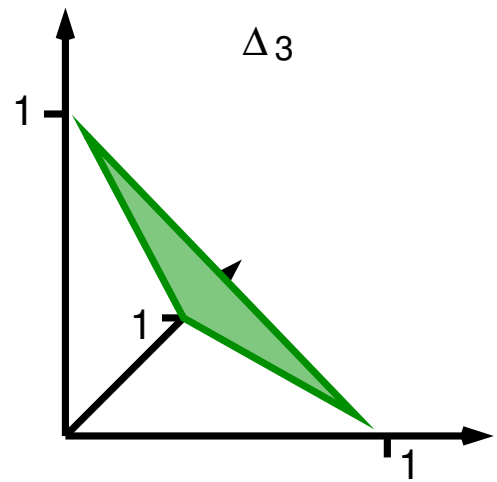
Eine Menge $C \subseteq \mathbb{R}^n$ heißt konvex, wenn für alle $x, y \in C$ auch die Verbindungsstrecke $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ in C liegt.

Bspe: $\emptyset$, $\mathbb{R}^n$, Halbräume, der Schnitt konvexer Mengen ist konvex,

Polyeder, der k-dim. Einheitssimplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$







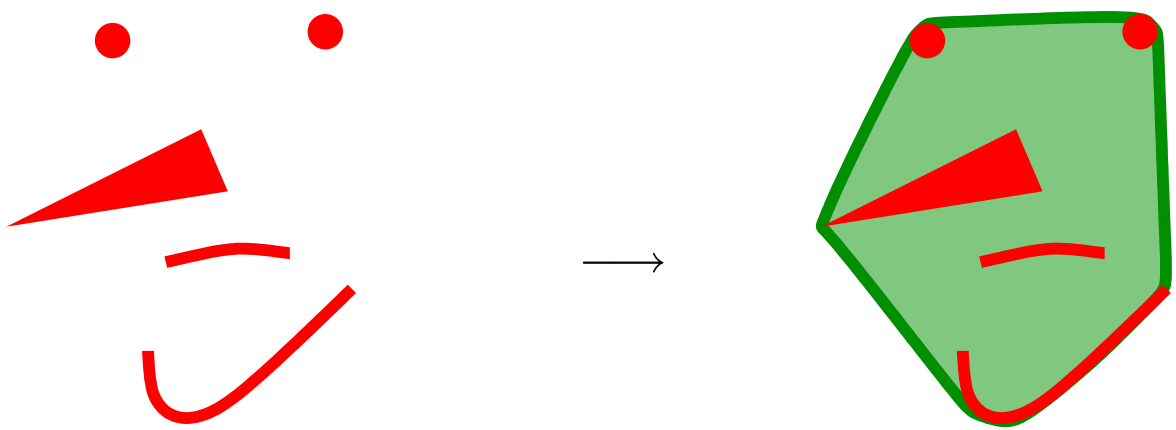
1.7 Konvexe Mengen und konvexe Hülle

Eine Menge $C \subseteq \mathbb{R}^n$ heißt konvex, wenn für alle $x, y \in C$ auch die Verbindungsstrecke $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ in C liegt.

Bspe: $\emptyset$, $\mathbb{R}^n$, Halbräume, der Schnitt konvexer Mengen ist konvex,

Polyeder, der k-dim. Einheitssimplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$

Für $S \subseteq \mathbb{R}^n$ ist die konvexe Hülle der Schnitt aller konvexen Mengen, die S enthalten, $\operatorname{conv} S := \bigcap\{C \text{ konvex} : S \subseteq C\}$.



1.7 Konvexe Mengen und konvexe Hülle

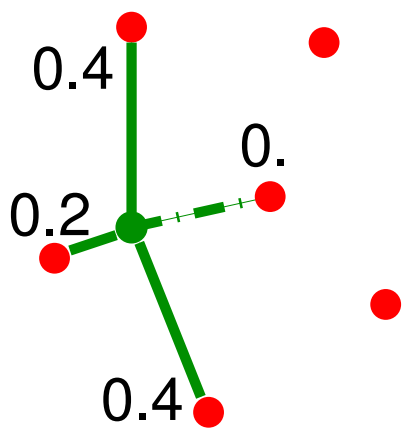
Eine Menge $C \subseteq \mathbb{R}^n$ heißt konvex, wenn für alle $x, y \in C$ auch die Verbindungsstrecke $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ in C liegt.

Bspe: $\emptyset$, $\mathbb{R}^n$, Halbräume, der Schnitt konvexer Mengen ist konvex,

Polyeder, der k-dim. Einheitssimplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$

Für $S \subseteq \mathbb{R}^n$ ist die konvexe Hülle der Schnitt aller konvexen Mengen, die S enthalten, $\operatorname{conv} S := \bigcap\{C \text{ konvex} : S \subseteq C\}$.

Für gegebene Punkte $x^{(i)} \in \mathbb{R}^n$, $i \in \{1, \ldots, k\}$ und $\alpha \in \Delta_k$ heißt $x = \sum \alpha_i x^{(i)}$ Konvexkombination der Punkte $x^{(i)}$.



1.7 Konvexe Mengen und konvexe Hülle

Eine Menge $C \subseteq \mathbb{R}^n$ heißt konvex, wenn für alle $x, y \in C$ auch die Verbindungsstrecke $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ in C liegt.

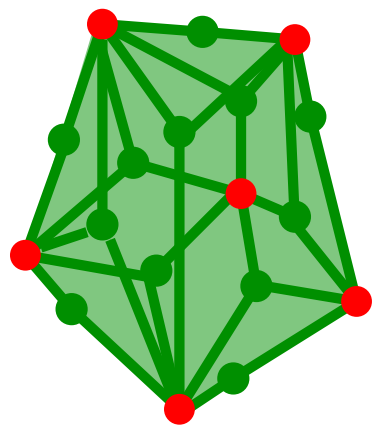
Bspe: $\emptyset$, $\mathbb{R}^n$, Halbräume, der Schnitt konvexer Mengen ist konvex,

Polyeder, der k-dim. Einheitssimplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$

Für $S \subseteq \mathbb{R}^n$ ist die konvexe Hülle der Schnitt aller konvexen Mengen, die S enthalten, $\operatorname{conv} S := \bigcap\{C \text{ konvex} : S \subseteq C\}$.

Für gegebene Punkte $x^{(i)} \in \mathbb{R}^n$, $i \in \{1, \ldots, k\}$ und $\alpha \in \Delta_k$ heißt $x = \sum \alpha_i x^{(i)}$ Konvexkombination der Punkte $x^{(i)}$.

$\operatorname{conv} S$ ist die Menge aller Konvexkombinationen endlich vieler Punkte aus S, $\operatorname{conv} S = \{\sum_{i=1}^k \alpha_i x^{(i)} : x^{(i)} \in S, i = 1, \ldots, k \in \mathbb{N}, \alpha \in \Delta_k\}$.



Konvexe Hülle und ganzz. Optimierung

Satz

Die konvexe Hülle endlich vieler Punkte ist ein (beschränktes) Polyeder.

Konvexe Hülle und ganzz. Optimierung

Satz

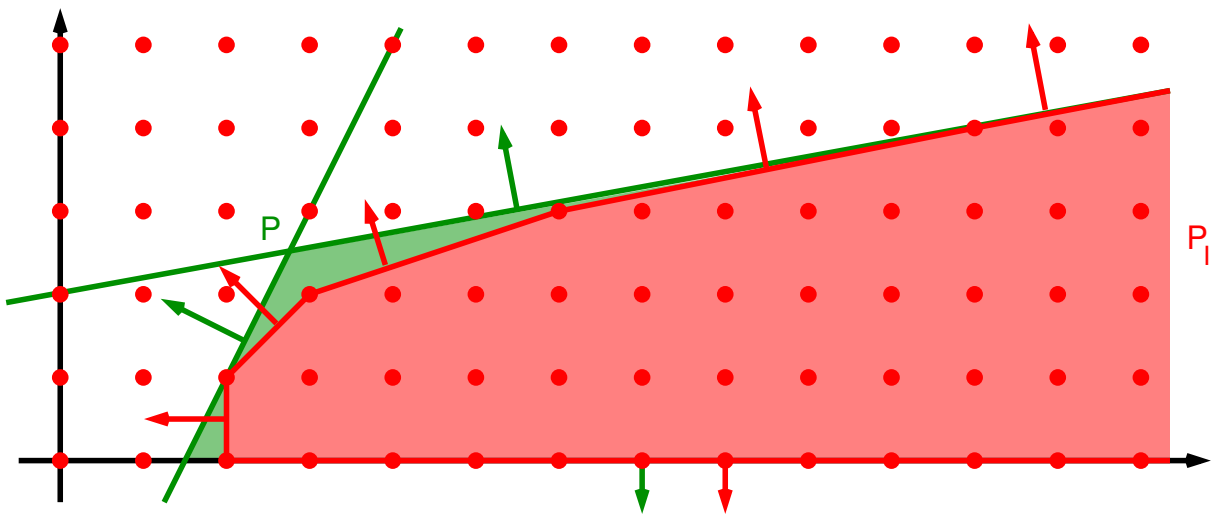
Die konvexe Hülle endlich vieler Punkte ist ein (beschränktes) Polyeder.

Die ganzzahlige Hülle eines Polyeders $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ ist die konvexe Hülle der in P enthaltenen ganzz. Punkte, $P_I := \text{conv}(P \cap \mathbb{Z}^n)$.

Satz

Ist $A \in \mathbb{Q}^{m \times n}$ und $b \in \mathbb{Q}^m$, dann ist die ganzz. Hülle P_I des Polyeders $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ selbst ein Polyeder.

Problem: Beschreibung von P_I meist unbekannt oder extrem groß!



[Ausnahme z.B. für A tot. unimod., $b \in \mathbb{Z}^n$ ist $P = P_I$]

Konvexe Hülle und ganzz. Optimierung

Satz

Die konvexe Hülle endlich vieler Punkte ist ein (beschränktes) Polyeder.

Die ganzzahlige Hülle eines Polyeders $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ ist die konvexe Hülle der in P enthaltenen ganzz. Punkte, $P_I := \operatorname{conv}(P \cap \mathbb{Z}^n)$.

Satz

Ist $A \in \mathbb{Q}^{m \times n}$ und $b \in \mathbb{Q}^m$, dann ist die ganzz. Hülle P_I des Polyeders $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ selbst ein Polyeder.

Problem: Beschreibung von P_I meist unbekannt oder extrem groß!

Falls die ganzzahlige Hülle gut linear beschreibbar ist, kann man das ganzzahlige Optimierungsproblem mit Simplex lösen:

Satz

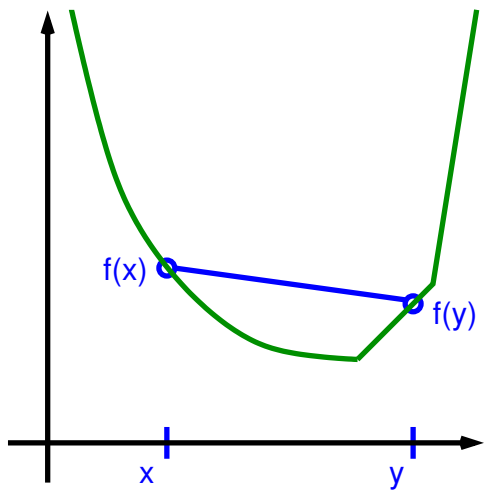
Ist für $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ die ganzzahlige Hülle durch $P_I = \{x \in \mathbb{R}^n : A_I x \leq b_I\}$ gegeben, so gilt:

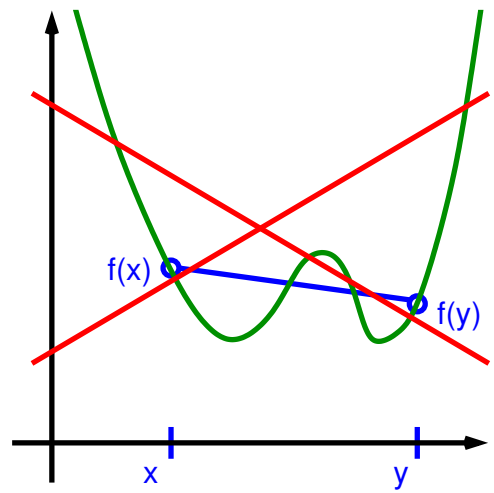
$$\begin{aligned} \sup\{c^T x : A_I x \leq b_I, x \in \mathbb{R}^n\} &= \sup\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\}, \\ \operatorname{Argmin}\{c^T x : A_I x \leq b_I, x \in \mathbb{R}^n\} &= \operatorname{conv}\operatorname{Argmin}\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\}. \end{aligned}$$

Konvexe Funktionen

Eine Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ heißt konvex, wenn $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$ und $\alpha \in [0,1]$.

Sie heißt streng konvex, wenn $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$, $x \neq y$ und $\alpha \in (0,1)$.





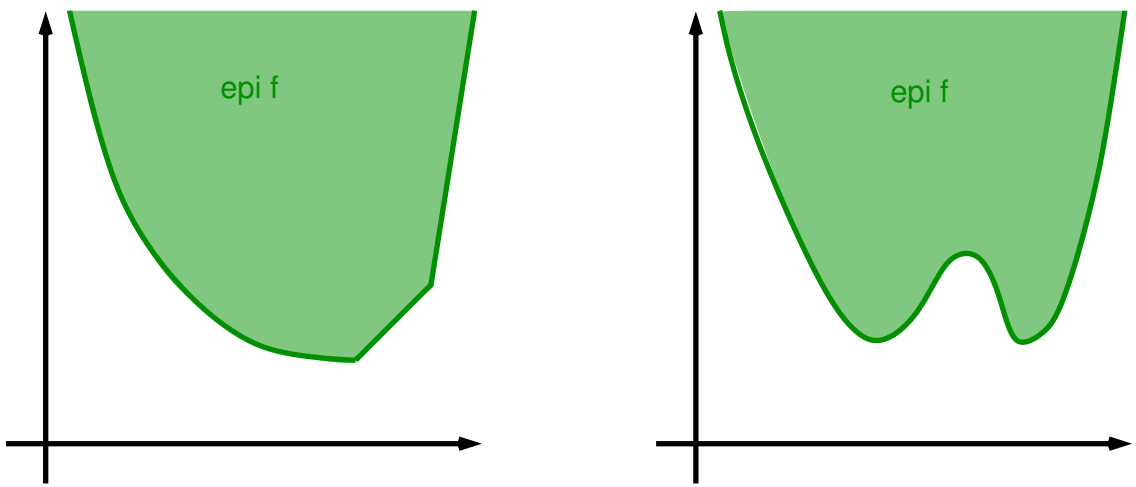
Konvexe Funktionen

Eine Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ heißt konvex, wenn $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$ und $\alpha \in [0,1]$.

Sie heißt streng konvex, wenn $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$, $x \neq y$ und $\alpha \in (0,1)$.

Der Epigraph einer Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ ist die Menge

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [die Punkte „oberhalb“von $f(x)$]



Konvexe Funktionen

Eine Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ heißt konvex, wenn $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$ und $\alpha \in [0,1]$.

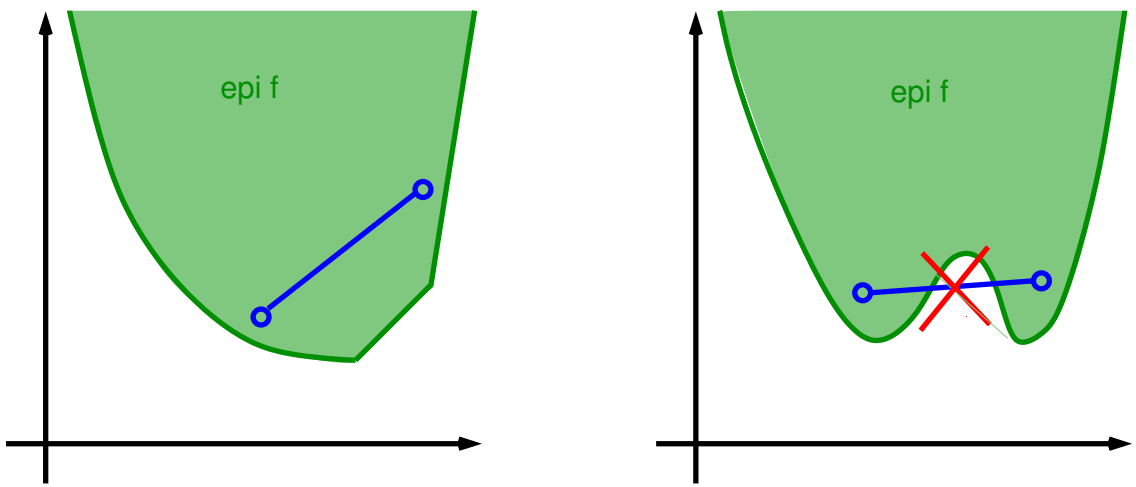
Sie heißt streng konvex, wenn $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$, $x \neq y$ und $\alpha \in (0,1)$.

Der Epigraph einer Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ ist die Menge

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [die Punkte „oberhalb“von $f(x)$]

Satz

Eine Funktion ist genau dann konvex, wenn ihr Epigraph konvex ist.



Konvexe Funktionen

Eine Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ heißt konvex, wenn $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$ und $\alpha \in [0,1]$.

Sie heißt streng konvex, wenn $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$, $x \neq y$ und $\alpha \in (0,1)$.

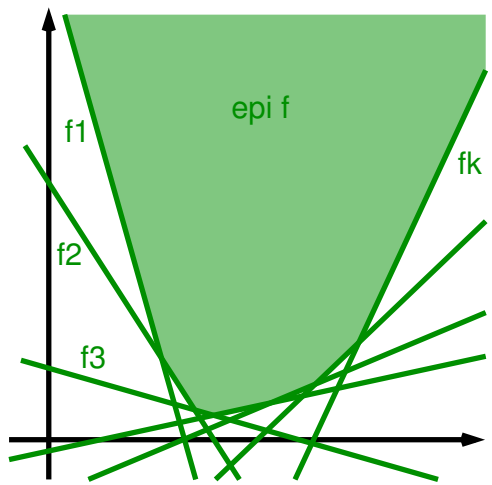
Der Epigraph einer Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ ist die Menge

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [die Punkte „oberhalb“von $f(x)$]

Satz

Eine Funktion ist genau dann konvex, wenn ihr Epigraph konvex ist.

Bsp: Sind $f_i : \mathbb{R}^n \to \bar{\mathbb{R}}$ konvex, so auch f mit $f(x) := \sup_i f_i(x)$, $x \in \mathbb{R}^n$.



Konvexe Funktionen

Eine Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ heißt konvex, wenn $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$ und $\alpha \in [0,1]$.

Sie heißt streng konvex, wenn $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ für $x, y \in \mathbb{R}^n$, $x \neq y$ und $\alpha \in (0,1)$.

Der Epigraph einer Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ ist die Menge

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [die Punkte „oberhalb“von $f(x)$]

Satz

Eine Funktion ist genau dann konvex, wenn ihr Epigraph konvex ist.

Bsp: Sind $f_i : \mathbb{R}^n \to \bar{\mathbb{R}}$ konvex, so auch f mit $f(x) := \sup_i f_i(x)$, $x \in \mathbb{R}^n$.

Satz

Jedes lokale Minimum einer konvexen Funktion ist auch globales Minimum, und für streng konvexe Funktionen ist es das einzige.

Für konvexe Funktionen gibt es gute Optimierungsverfahren.

Konvexe Funktionen

Eine Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ heißt konvex, wenn
$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ für $x, y \in \mathbb{R}^n$ und $\alpha \in [0,1]$.

Sie heißt streng konvex, wenn $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y)$
für $x, y \in \mathbb{R}^n$, $x \neq y$ und $\alpha \in (0,1)$.

Der Epigraph einer Funktion $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ ist die Menge

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [die Punkte „oberhalb“von $f(x)$]

Satz

Eine Funktion ist genau dann konvex, wenn ihr Epigraph konvex ist.

Bsp: Sind $f_i : \mathbb{R}^n \to \bar{\mathbb{R}}$ konvex, so auch f mit $f(x) := \sup_i f_i(x)$, $x \in \mathbb{R}^n$.

Satz

Jedes lokale Minimum einer konvexen Funktion ist auch globales Minimum, und für streng konvexe Funktionen ist es das einzige.

Für konvexe Funktionen gibt es gute Optimierungsverfahren.

Eine Funktion f heißt konkav, wenn $-f$ konvex ist.
(Jedes lokale Maximum einer konkaven Funktion ist auch globales.)

Inhaltsübersicht

1.8 Relaxation

Konzept auf beliebige Optimierungsprobleme anwendbar (hier Maximieren):

Definition

Gegeben zwei Optimierungsprobleme mit $\mathcal{X}, \mathcal{W} \subseteq \mathbb{R}^n$ *und* $f, f' : \mathbb{R}^n \to \mathbb{R}$

$$(OP) \quad \max \ f(x) \text{ s.t. } x \in \mathcal{X} \qquad \textit{bzw.} \qquad (RP) \quad \max \ f'(x) \text{ s.t. } x \in \mathcal{W},$$

(RP) *heißt Relaxation von* (OP), *falls*

1. $\mathcal{X} \subseteq \mathcal{W}$,
2. $f(x) \leq f'(x)$ *für alle* $x \in \mathcal{X}$.

1.8 Relaxation

Konzept auf beliebige Optimierungsprobleme anwendbar (hier Maximieren):

Definition

Gegeben zwei Optimierungsprobleme mit $\mathcal{X}, \mathcal{W} \subseteq \mathbb{R}^n$ *und* $f, f' : \mathbb{R}^n \to \mathbb{R}$

$$(OP) \quad \max \ f(x) \text{ s.t. } x \in \mathcal{X} \qquad \textit{bzw.} \qquad (RP) \quad \max \ f'(x) \text{ s.t. } x \in \mathcal{W},$$

(RP) *heißt Relaxation von* (OP), *falls*

1. $\mathcal{X} \subseteq \mathcal{W}$,
2. $f(x) \leq f'(x)$ *für alle* $x \in \mathcal{X}$.

Sei (RP) eine Relaxation von (OP).

Beobachtung

1. $v(RP) \geq v(OP)$. *[(RP) liefert obere Schranke]*
2. *Ist* (RP) *unzulässig, so auch* (OP),
3. *Ist* x^* *OL von (RP) und gilt* $x^* \in \mathcal{X}$ *sowie* $f'(x^*) = f(x^*)$, *dann ist* x^* *OL von (OP).*

1.8 Relaxation

Konzept auf beliebige Optimierungsprobleme anwendbar (hier Maximieren):

Definition

Gegeben zwei Optimierungsprobleme mit $\mathcal{X}, \mathcal{W} \subseteq \mathbb{R}^n$ *und* $f, f' : \mathbb{R}^n \to \mathbb{R}$

$$(OP) \quad \max \ f(x) \text{ s.t. } x \in \mathcal{X} \qquad \textit{bzw.} \qquad (RP) \quad \max \ f'(x) \text{ s.t. } x \in \mathcal{W},$$

(RP) *heißt Relaxation von* (OP)*, falls*

1. $\mathcal{X} \subseteq \mathcal{W}$,
2. $f(x) \leq f'(x)$ *für alle* $x \in \mathcal{X}$.

Sei (RP) eine Relaxation von (OP).

Beobachtung

1. $v(RP) \geq v(OP)$. *[(RP) liefert obere Schranke]*
2. *Ist* (RP) *unzulässig, so auch* (OP),
3. *Ist* x^* *OL von (RP) und gilt* $x^* \in \mathcal{X}$ *sowie* $f'(x^*) = f(x^*)$, *dann ist* x^* *OL von (OP).*

Man sucht nun möglichst „kleines“ $\mathcal{W} \supseteq \mathcal{X}$ und $f' \geq f$ so, dass (RP) noch gut lösbar ist.

Eine Relaxation (RP) von (OP) heißt exakt, falls $v(OP) = v(RP)$ gilt.

Allgemein: Konvexe Relaxation

Wird Konvexität für $\mathcal{W}$ und (bei max) Konkavität für f' gefordert, spricht man von konvexer Relaxation. Meist (aber nicht immer!) dient die Konvexität als Garant für vernünftige Lösbarkeit der Relaxation.

Allgemein: Konvexe Relaxation

Wird Konvexität für $\mathcal{W}$ und (bei max) Konkavität für f' gefordert, spricht man von konvexer Relaxation. Meist (aber nicht immer!) dient die Konvexität als Garant für vernünftige Lösbarkeit der Relaxation.

Bsp: Für ein kombinatorisches Max.-Problem mit endl. Grundmenge Ω, zulässigen Lösungen $\mathcal{F} \subseteq 2^\Omega$ und linearer Zielfunktion $c \in \mathbb{R}^\Omega$ ist

$$\max c^T x \text{ s.t. } x \in \operatorname{conv}\{\chi_\Omega(F) : F \in \mathcal{F}\}$$

eine exakte konvexe (sogar lineare) Relaxation, aber nur dann nützlich, wenn das Polyeder $\operatorname{conv}\{\chi(F) : F \in \mathcal{F}\}$ gut durch ein nicht zu großes Ungleichungssystem $Ax \leq b$ darstellbar ist.

Allgemein: Konvexe Relaxation

Wird Konvexität für $\mathcal{W}$ und (bei max) Konkavität für f' gefordert, spricht man von konvexer Relaxation. Meist (aber nicht immer!) dient die Konvexität als Garant für vernünftige Lösbarkeit der Relaxation.

Bsp: Für ein kombinatorisches Max.-Problem mit endl. Grundmenge Ω, zulässigen Lösungen $\mathcal{F} \subseteq 2^\Omega$ und linearer Zielfunktion $c \in \mathbb{R}^\Omega$ ist

$$\max c^T x \text{ s.t. } x \in \operatorname{conv}\{\chi_\Omega(F) : F \in \mathcal{F}\}$$

eine exakte konvexe (sogar lineare) Relaxation, aber nur dann nützlich, wenn das Polyeder $\operatorname{conv}\{\chi(F) : F \in \mathcal{F}\}$ gut durch ein nicht zu großes Ungleichungssystem $Ax \leq b$ darstellbar ist.

In der globalen Optimierung werden auf Teilintervallen nichtlineare Funktionen nach unten durch konvexe Funktionen abgeschätzt.

Bsp: Betrachte (OP) $\min f(x) := \frac{1}{2}x^T Qx + q^T x$ s.t. $x \in [0,1]^n$
mit f nicht konvex, d.h., $\lambda_{\min}(Q) < 0$. [$\lambda_{\min}$ . . . minimaler Eigenwert]
Es ist $Q - \lambda_{\min}(Q)I$ positiv semidefinit und wegen $x_i^2 \leq x_i$ auf $[0,1]^n$ gilt
$f'(x) := \frac{1}{2}x^T(Q - \lambda_{\min}(Q)I)x + (q + \lambda_{\min}(Q)\mathbf{1})^T x \leq f(x) \quad \forall x \in [0,1]^n$.
Damit ist $(RP)\min f'(x)$ s.t. $x \in [0,1]^n$ eine konvexe Relaxation von (OP).

Die LP-Relaxation für Ganzzahlige Programme

Für ein ganzzahliges Programm $\max\ c^T x$ s.t. $Ax \leq b,\ x \in \mathbb{Z}^n$
entsteht die LP-Relaxation durch Weglassen der Ganzz.-Bedingung,

$$\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{R}^n.$$

Ist Relaxation, denn

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \leq b\} \subseteq \{x \in \mathbb{R}^n : Ax \leq b\} =: \mathcal{W}.$$

[wird von allen Standardlösern für gemischt-ganzz. Programme verwendet]

Die LP-Relaxation für Ganzzahlige Programme

Für ein ganzzahliges Programm $\max\ c^T x$ s.t. $Ax \leq b,\ x \in \mathbb{Z}^n$
entsteht die LP-Relaxation durch Weglassen der Ganzz.-Bedingung,

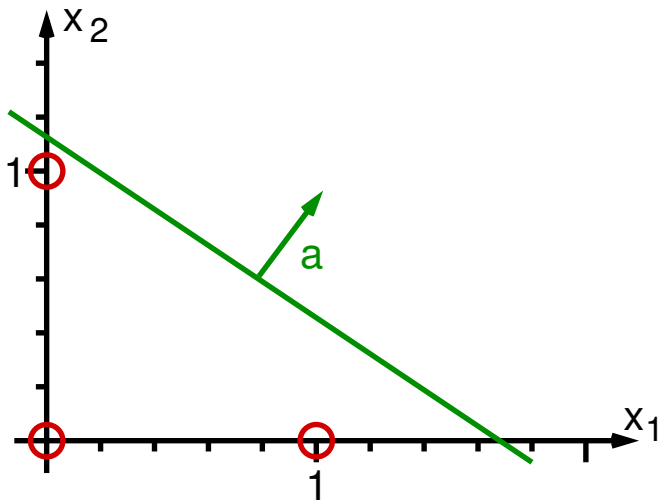
$$\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{R}^n.$$

Ist Relaxation, denn

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \leq b\} \subseteq \{x \in \mathbb{R}^n : Ax \leq b\} =: \mathcal{W}.$$

[wird von allen Standardlösern für gemischt-ganzz. Programme verwendet]

Bsp: Rucksackproblem: $n = 2$, Gewichte $a = (6, 8)^T$, Kapazität $b = 10$,
$\max\ c^T x$ s.t. $a^T x \leq b,\ x \in \mathbb{Z}^n_+ \quad \rightarrow \quad \max\ c^T x$ s.t. $a^T x \leq b,\ x \geq 0,$



zulässige ganzzahlige Punkte: ◯

Die LP-Relaxation für Ganzzahlige Programme

Für ein ganzzahliges Programm $\max\ c^T x$ s.t. $Ax \leq b,\ x \in \mathbb{Z}^n$
entsteht die LP-Relaxation durch Weglassen der Ganzz.-Bedingung,

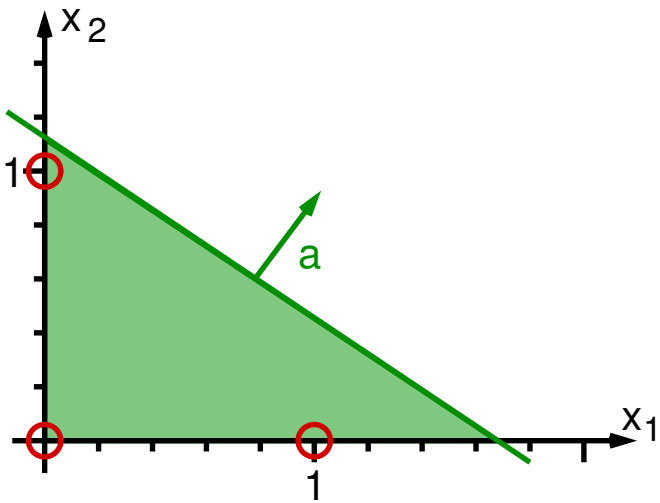
$$\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{R}^n.$$

Ist Relaxation, denn

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \leq b\} \subseteq \{x \in \mathbb{R}^n : Ax \leq b\} =: \mathcal{W}.$$

[wird von allen Standardlösern für gemischt-ganzz. Programme verwendet]

Bsp: Rucksackproblem: $n = 2$, Gewichte $a = (6, 8)^T$, Kapazität $b = 10$,
$\max\ c^T x$ s.t. $a^T x \leq b,\ x \in \mathbb{Z}^n_+ \quad \rightarrow \quad \max\ c^T x$ s.t. $a^T x \leq b,\ x \geq 0,$



zulässige ganzzahlige Punkte: ◯

LP-Relaxation: grün

Die LP-Relaxation für Ganzzahlige Programme

Für ein ganzzahliges Programm $\max\ c^T x$ s.t. $Ax \leq b,\ x \in \mathbb{Z}^n$
entsteht die LP-Relaxation durch Weglassen der Ganzz.-Bedingung,

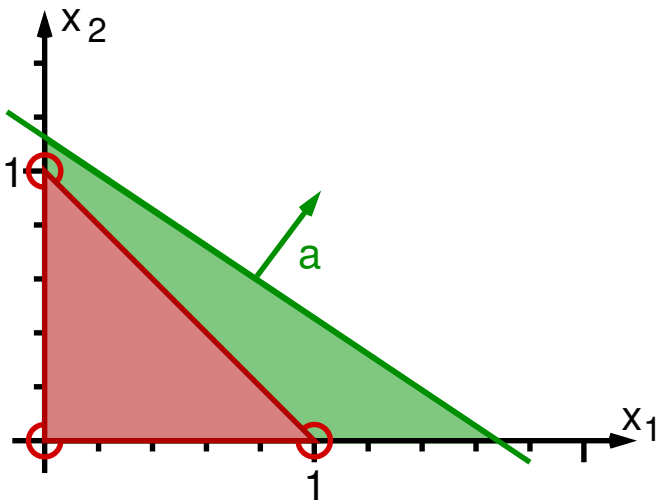
$$\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{R}^n.$$

Ist Relaxation, denn

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \leq b\} \subseteq \{x \in \mathbb{R}^n : Ax \leq b\} =: \mathcal{W}.$$

[wird von allen Standardlösern für gemischt-ganzz. Programme verwendet]

Bsp: Rucksackproblem: $n = 2$, Gewichte $a = (6, 8)^T$, Kapazität $b = 10$,
$\max\ c^T x$ s.t. $a^T x \leq b,\ x \in \mathbb{Z}^n_+ \quad \rightarrow \quad \max\ c^T x$ s.t. $a^T x \leq b,\ x \geq 0,$



zulässige ganzzahlige Punkte: ◯

LP-Relaxation: grün

beste Relaxation: die konvexe Hülle

Die LP-Relaxation für Ganzzahlige Programme

Für ein ganzzahliges Programm $\max\ c^T x$ s.t. $Ax \leq b,\ x \in \mathbb{Z}^n$
entsteht die LP-Relaxation durch Weglassen der Ganzz.-Bedingung,

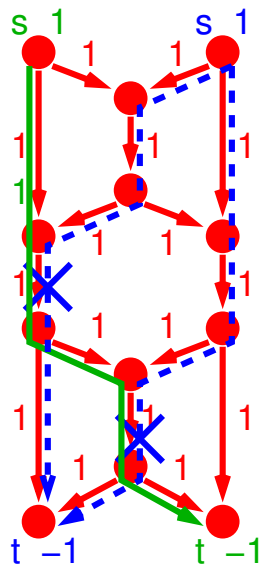
$$\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{R}^n.$$

Ist Relaxation, denn

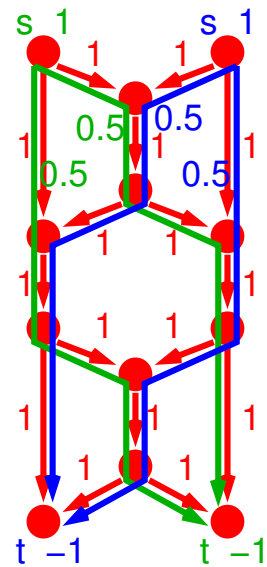
$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \leq b\} \subseteq \{x \in \mathbb{R}^n : Ax \leq b\} =: \mathcal{W}.$$

[wird von allen Standardlösern für gemischt-ganzz. Programme verwendet]

Bsp: ganzzahliger Mehrgüterfluss → gebrochener Mehrgüterfluss



ganzz. geht nicht, $\mathcal{X} = \emptyset$



gebrochen geht, $\mathcal{W} \neq \emptyset$

auch zu groß, bräuchten die konvexe Hülle

Lagrange-Relaxation

[Allg. für restr. Opt. verwendbar, hier vorerst nur Ungls.-Nebenbed.]

Unbequeme Nebenbedingungen werden mit einem Lagrangemultiplikator, der die Verletzung bestraft, in die Zielfunktion gehoben ($g : \mathbb{R}^n \to \mathbb{R}^k$):

$$(OP) \quad \begin{array}{ll} \max & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x \in \Omega \end{array} \quad \left| \cdot \lambda \geq 0 \right. \quad \to \quad (RP_\lambda) \quad \begin{array}{ll} \max & c^T x - \lambda^T g(x) \\ \text{s.t.} & x \in \Omega \end{array}$$

Ist Relaxation: $\mathcal{X} := \{x \in \Omega : g(x) \leq 0\} \subseteq \{x \in \Omega\} =: \mathcal{W}$ und für $x \in \mathcal{X}$, $\lambda \geq 0$ gilt $f(x) \leq f(x) - \lambda^T g(x) =: f'(x)$ wegen $g(x) \leq 0$.

Lagrange-Relaxation

[Allg. für restr. Opt. verwendbar, hier vorerst nur Unglgs.-Nebenbed.]

Unbequeme Nebenbedingungen werden mit einem Lagrangemultiplikator, der die Verletzung bestraft, in die Zielfunktion gehoben ($g : \mathbb{R}^n \to \mathbb{R}^k$):

$$(OP) \quad \begin{array}{ll} \max & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x \in \Omega \end{array} \quad \Big| \cdot \lambda \geq 0 \quad \to \quad (RP_\lambda) \quad \begin{array}{ll} \max & c^T x - \lambda^T g(x) \\ \text{s.t.} & x \in \Omega \end{array}$$

Ist Relaxation: $\mathcal{X} := \{x \in \Omega : g(x) \leq 0\} \subseteq \{x \in \Omega\} =: \mathcal{W}$ und für $x \in \mathcal{X}$, $\lambda \geq 0$ gilt $f(x) \leq f(x) - \lambda^T g(x) =: f'(x)$ wegen $g(x) \leq 0$.

Definiere die duale Funktion $\varphi(\lambda) := \sup_{x \in \Omega} \left[f(x) - g(x)^T \lambda \right] = v(RP_\lambda)$
[für jedes feste x linear in λ]

- Für jedes $\lambda \geq 0$ gilt $\varphi(\lambda) \geq v(OP)$ [obere Schranke]
- $\varphi(\lambda)$ gut berechenbar, falls (RP_λ) „leicht“ lösbar
- φ ist als sup von in λ linearen Funktionen konvex
- Beste Schranke ist $\inf\{\varphi(\lambda) : \lambda \geq 0\}$ [konvexes Problem!]

gut mit Verfahren der konvexen Optimierung bestimmbar!

Beispiel: Ganzzahliger Mehrgüterfluss

A sei die Knoten-Kanten-Inz.-Matrix zu $D = (V, E)$, 2 Güter,
Lagrange-Relaxation der koppelnden Kapazitätsbedingungen mit $\lambda \geq 0$:

$$\begin{array}{lll} \min & {c^{(1)}}^T x^{(1)} & + \; {c^{(2)}}^T x^{(2)} \\ \text{s.t.} & Ax^{(1)} & = b^{(1)} \\ & Ax^{(2)} & = b^{(2)} \\ \lambda \cdot | & x^{(1)} + x^{(2)} & \leq w \\ & x^{(1)} \leq w, & x^{(2)} \leq w \\ & x^{(1)} \in \mathbb{Z}_+^E, & x^{(2)} \in \mathbb{Z}_+^E. \end{array} \quad \rightarrow \quad \begin{array}{lll} \min & (c^{(1)}+\lambda)^T x^{(1)} + (c^{(2)}+\lambda)^T x^{(2)} - \lambda^T w & \\ \text{s.t.} & Ax^{(1)} & = b^{(1)} \\ & Ax^{(2)} & = b^{(2)} \\ & x^{(1)} \leq w, & x^{(2)} \leq w, \\ & x^{(1)} \in \mathbb{Z}_+^E, & x^{(2)} \in \mathbb{Z}_+^E. \end{array}$$

Relaxation zerfällt in zwei unabhängige Minimale-Kosten-Fluss-Probleme

$$(RP_\lambda^{(i)}) \quad \min (c^{(i)}+\lambda)^T x^{(i)} \quad \text{s.t.} \quad Ax^{(i)} = b^{(i)},\ w \geq x^{(i)} \in \mathbb{Z}_+^E \quad i \in \{1, 2\}$$

Diese sind effizient ganzzahlig lösbar!

Beispiel: Ganzzahliger Mehrgüterfluss

A sei die Knoten-Kanten-Inz.-Matrix zu $D = (V, E)$, 2 Güter,
Lagrange-Relaxation der koppelnden Kapazitätsbedingungen mit $\lambda \geq 0$:

$$\begin{array}{lll} \min & {c^{(1)}}^T x^{(1)} & + \; {c^{(2)}}^T x^{(2)} \\ \text{s.t.} & Ax^{(1)} & = b^{(1)} \\ & Ax^{(2)} & = b^{(2)} \\ \lambda \cdot | & x^{(1)} + x^{(2)} & \leq w \\ & x^{(1)} \leq w, & x^{(2)} \leq w \\ & x^{(1)} \in \mathbb{Z}^E_+, & x^{(2)} \in \mathbb{Z}^E_+. \end{array} \quad \rightarrow \quad \begin{array}{lll} \min & (c^{(1)}+\lambda)^T x^{(1)} + (c^{(2)}+\lambda)^T x^{(2)} - \lambda^T w & \\ \text{s.t.} & Ax^{(1)} & = b^{(1)} \\ & Ax^{(2)} & = b^{(2)} \\ & x^{(1)} \leq w, & x^{(2)} \leq w, \\ & x^{(1)} \in \mathbb{Z}^E_+, & x^{(2)} \in \mathbb{Z}^E_+. \end{array}$$

Relaxation zerfällt in zwei unabhängige Minimale-Kosten-Fluss-Probleme

$$(RP^{(i)}_\lambda) \quad \min (c^{(i)}+\lambda)^T x^{(i)} \quad \text{s.t.} \quad Ax^{(i)} = b^{(i)},\ w \geq x^{(i)} \in \mathbb{Z}^E_+ \quad i \in \{1, 2\}$$

Diese sind effizient ganzzahlig lösbar!

Zerfällt das Problem bei Lagrange-Relaxation in unabhängige Teilprobleme, nennt man das Lagrange-Dekomposition. Damit können oft deutlich größere Probleme sehr effizient gelöst werden.

Bekommt man dadurch auch eine bessere Schranke?

Vergleich Lagrange- und LP-Relaxation

Sei $\Omega \subset \mathbb{Z}^n$ endlich und $D \in \mathbb{Q}^{k\times n}$, $d \in \mathbb{Q}^k$,

$$(OP)\quad \begin{array}{ll}\max & c^T x \\ \text{s.t.} & Dx \le d \quad | \cdot \lambda \ge 0 \\ & x \in \Omega\end{array} \quad \rightarrow \quad (RP_\lambda)\quad \begin{array}{ll}\max & c^T x + \lambda^T(d - Dx) \\ \text{s.t.} & x \in \Omega\end{array}$$

Im Bsp: $\Omega = \Omega^{(1)} \times \Omega^{(2)}$ mit $\Omega^{(i)} = \{x \in \mathbb{Z}_+^E : Ax = b^{(i)}, x \le w\}$, $i \in \{1, 2\}$.

Satz

$$\inf_{\lambda \ge 0} v(RP_\lambda) = \sup\{c^T x : Dx \le d, x \in \operatorname{conv}\Omega\}.$$

Ist $\operatorname{conv}\Omega$ gleich der zulässigen Menge der LP-Relaxation von Ω, sind die Werte der besten Lagrange- und der LP-Relaxation gleich!

Vergleich Lagrange- und LP-Relaxation

Sei $\Omega \subset \mathbb{Z}^n$ endlich und $D \in \mathbb{Q}^{k\times n}$, $d \in \mathbb{Q}^k$,

$$(OP)\quad \begin{array}{ll} \max & c^T x \\ \text{s.t.} & Dx \le d \quad | \cdot \lambda \ge 0 \\ & x \in \Omega \end{array} \quad \rightarrow \quad (RP_\lambda)\quad \begin{array}{ll} \max & c^T x + \lambda^T (d - Dx) \\ \text{s.t.} & x \in \Omega \end{array}$$

Im Bsp: $\Omega = \Omega^{(1)} \times \Omega^{(2)}$ mit $\Omega^{(i)} = \{x \in \mathbb{Z}_+^E : Ax = b^{(i)}, x \le w\}$, $i \in \{1,2\}$.

Satz

$$\inf_{\lambda \ge 0} v(RP_\lambda) = \sup\{c^T x : Dx \le d, x \in \operatorname{conv}\Omega\}.$$

Ist $\operatorname{conv}\Omega$ gleich der zulässigen Menge der LP-Relaxation von Ω, sind die Werte der besten Lagrange- und der LP-Relaxation gleich!

Im Bsp ist A total unimodular, daher gilt für $i \in \{1,2\}$ und $w \in \mathbb{Z}^E$

$$\operatorname{conv}\{x \in \mathbb{Z}_+^E : Ax = b^{(i)}, x \le w\} = \{x \ge 0 : Ax = b^{(i)}, x \le w\}.$$

Das beste λ ergibt den Wert des gebrochenen Mehrgüterflussproblems!

Vergleich Lagrange- und LP-Relaxation

Sei $\Omega \subset \mathbb{Z}^n$ endlich und $D \in \mathbb{Q}^{k\times n}$, $d \in \mathbb{Q}^k$,

$$(OP)\quad \begin{array}{ll}\max & c^T x \\ \text{s.t.} & Dx \le d \quad | \cdot \lambda \ge 0 \\ & x \in \Omega\end{array} \quad \rightarrow \quad (RP_\lambda)\quad \begin{array}{ll}\max & c^T x + \lambda^T(d - Dx) \\ \text{s.t.} & x \in \Omega\end{array}$$

Im Bsp: $\Omega = \Omega^{(1)} \times \Omega^{(2)}$ mit $\Omega^{(i)} = \{x \in \mathbb{Z}_+^E : Ax = b^{(i)}, x \le w\}$, $i \in \{1,2\}$.

Satz

$$\inf_{\lambda \ge 0} v(RP_\lambda) = \sup\{c^T x : Dx \le d, x \in \operatorname{conv}\Omega\}.$$

Ist $\operatorname{conv}\Omega$ gleich der zulässigen Menge der LP-Relaxation von Ω, sind die Werte der besten Lagrange- und der LP-Relaxation gleich!

Im Bsp ist A total unimodular, daher gilt für $i \in \{1,2\}$ und $w \in \mathbb{Z}^E$

$$\operatorname{conv}\{x \in \mathbb{Z}_+^E : Ax = b^{(i)}, x \le w\} = \{x \ge 0 : Ax = b^{(i)}, x \le w\}.$$

Das beste λ ergibt den Wert des gebrochenen Mehrgüterflussproblems!

Allgemein: Sei $\{x \in \mathbb{Z}^n : Ax \le b\} = \Omega$ eine Formulierung von Ω. Nur falls $\{x \in \mathbb{R}^n : Ax \le b\} \ne \operatorname{conv}\Omega$, kann die Lagrange-Relaxation einen besseren Wert liefern als die LP-Relaxation.

Inhaltsübersicht

1.9 Anwendung: Rundreiseprobleme (TSP)

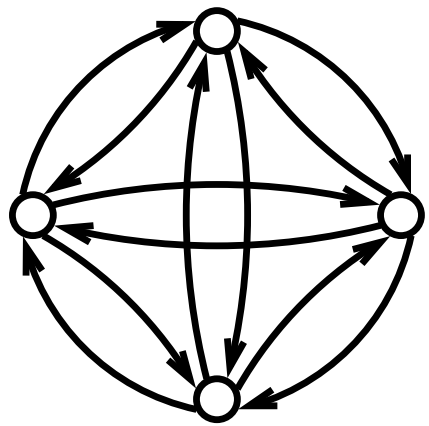
Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

1.9 Anwendung: Rundreiseprobleme (TSP)

Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

Komb. Opt.: $D = (V, E = \{(u, v) : u, v \in V, u \neq v\})$ vollst. Digraph, Kosten $c \in \mathbb{R}^E$, zul. Menge $\mathcal{F} = \{R \subset E : R \text{ (ger.) Kreis in } D, |R| = n\}$.
Finde $R \in \operatorname{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-vollständiges Problem
Anzahl Rundreisen?



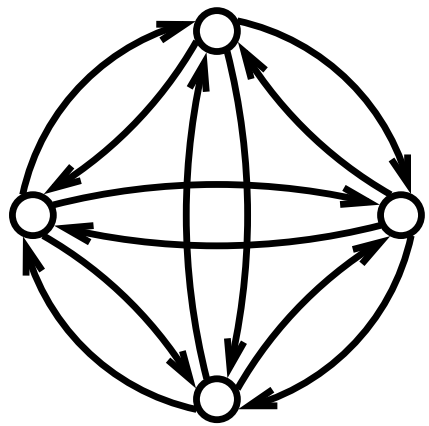
1.9 Anwendung: Rundreiseprobleme (TSP)

Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

Komb. Opt.: $D = (V, E = \{(u,v) : u,v \in V, u \neq v\})$ vollst. Digraph, Kosten $c \in \mathbb{R}^E$, zul. Menge $\mathcal{F} = \{R \subset E : R$ (ger.) Kreis in $D, |R| = n\}$.
Finde $R \in \operatorname{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-vollständiges Problem
Anzahl Rundreisen? $(n-1)!$

$\rightarrow$kombinatorische Explosion



$n = 3$:	2
$n = 4$:	$3 \cdot 2 = 6$
$n = 5$:	$4 \cdot 3 \cdot 2 = 24$
$n = 10$:	362880
$n = 11$:	3628800
$n = 12$:	39916800
$n = 13$:	479001600
$n = 14$:	6227020800
$n = 100$:	10^{158}

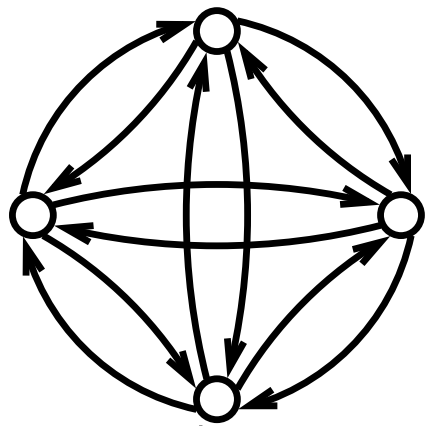
1.9 Anwendung: Rundreiseprobleme (TSP)

Problem des Handelsreisenden: Gegeben n Städte mit allen paarweisen Distanzen, finde eine kürzeste Rundreise, die jede genau einmal besucht.

Komb. Opt.: $D = (V, E = \{(u,v) : u,v \in V, u \neq v\})$ vollst. Digraph, Kosten $c \in \mathbb{R}^E$, zul. Menge $\mathcal{F} = \{R \subset E : R$ (ger.) Kreis in $D, |R| = n\}$.
Finde $R \in \operatorname{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-vollständiges Problem
Anzahl Rundreisen? $(n-1)!$

$\rightarrow$kombinatorische Explosion



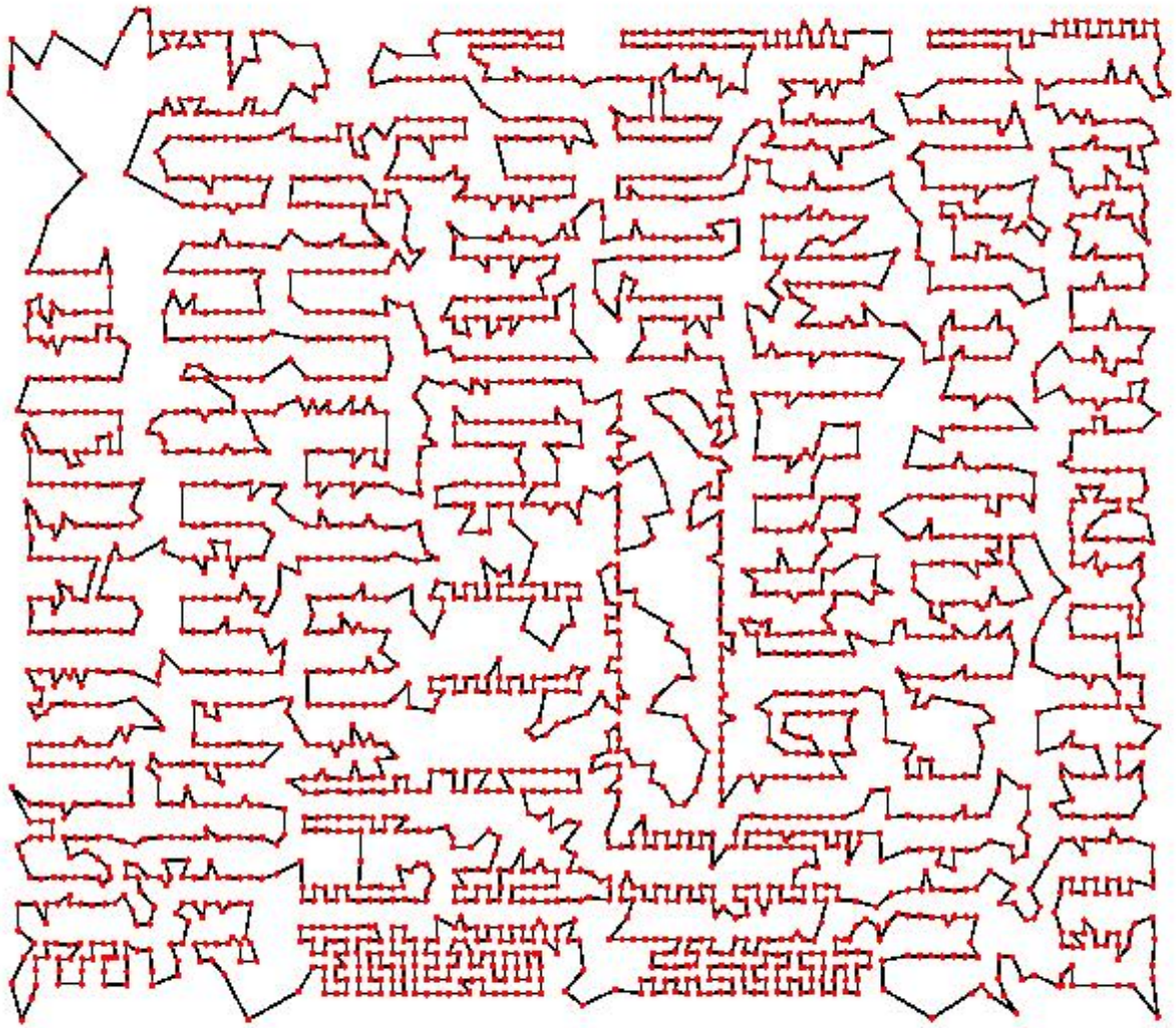
$n = 3$:	2
$n = 4$:	$3 \cdot 2 = 6$
$n = 5$:	$4 \cdot 3 \cdot 2 = 24$
$n = 10$:	362880
$n = 11$:	3628800
$n = 12$:	39916800
$n = 13$:	479001600
$n = 14$:	6227020800
$n = 100$:	10^{158}

Typisches Grundproblem in:

- Zustelldiensten (Paket-, Arznei-Transporte)
- Taxiservice, Havarie-Dienste
- Auftragsplanung mit Rüstkosten [z.B. Autos färben]
- Robotersteuerung (meist nichtlinear)

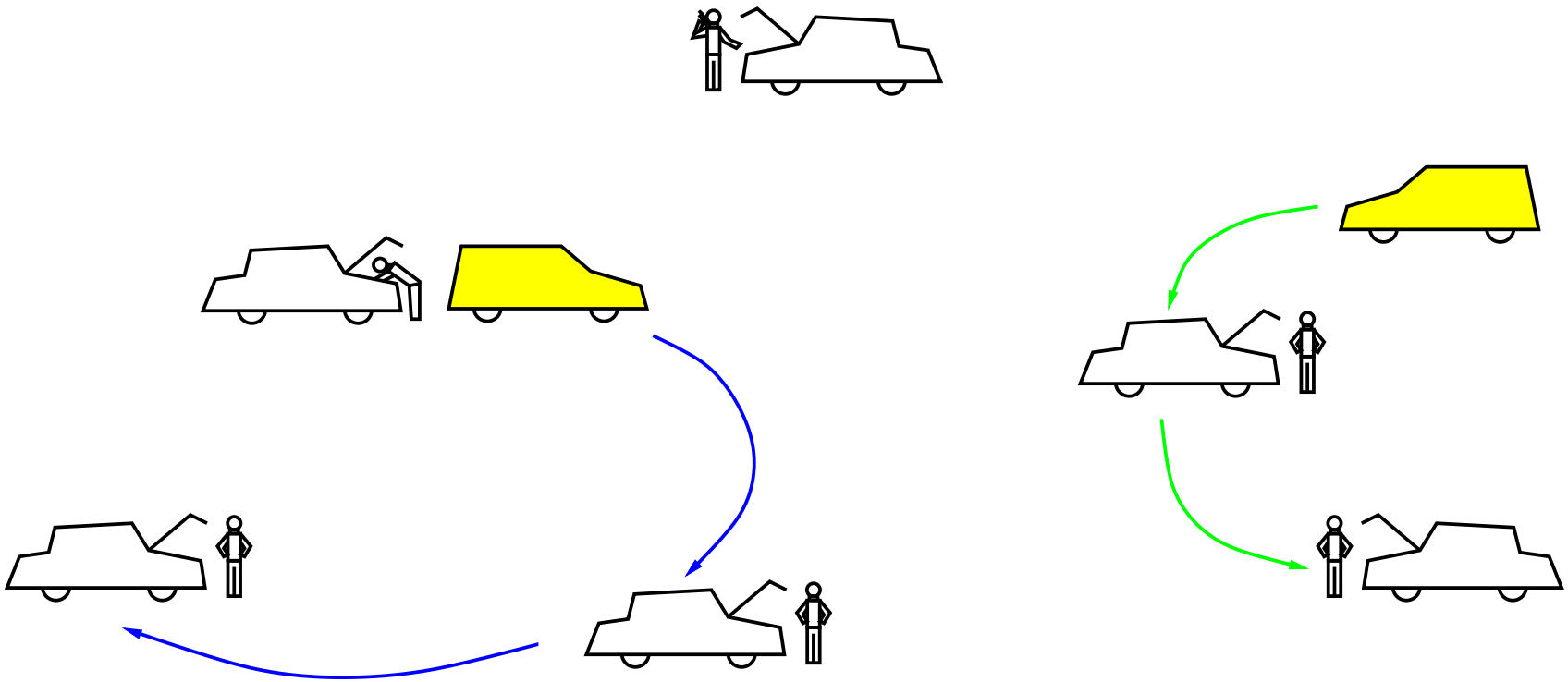
[oft mit mehreren „Fahrzeugen“ und Zeitfenstern]

Löcher in Platinen bohren



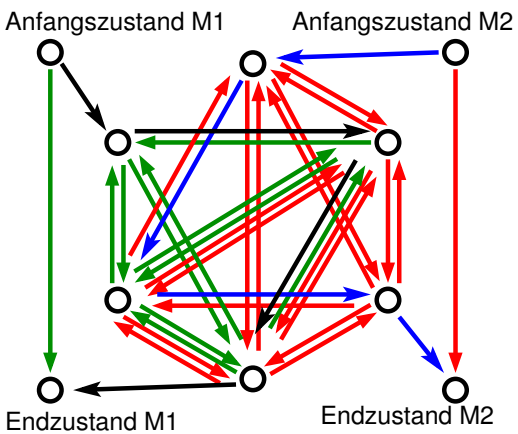
[http://www.math.princeton.edu/tsp]

Mit Online-Aspekten: Pannenhilfe

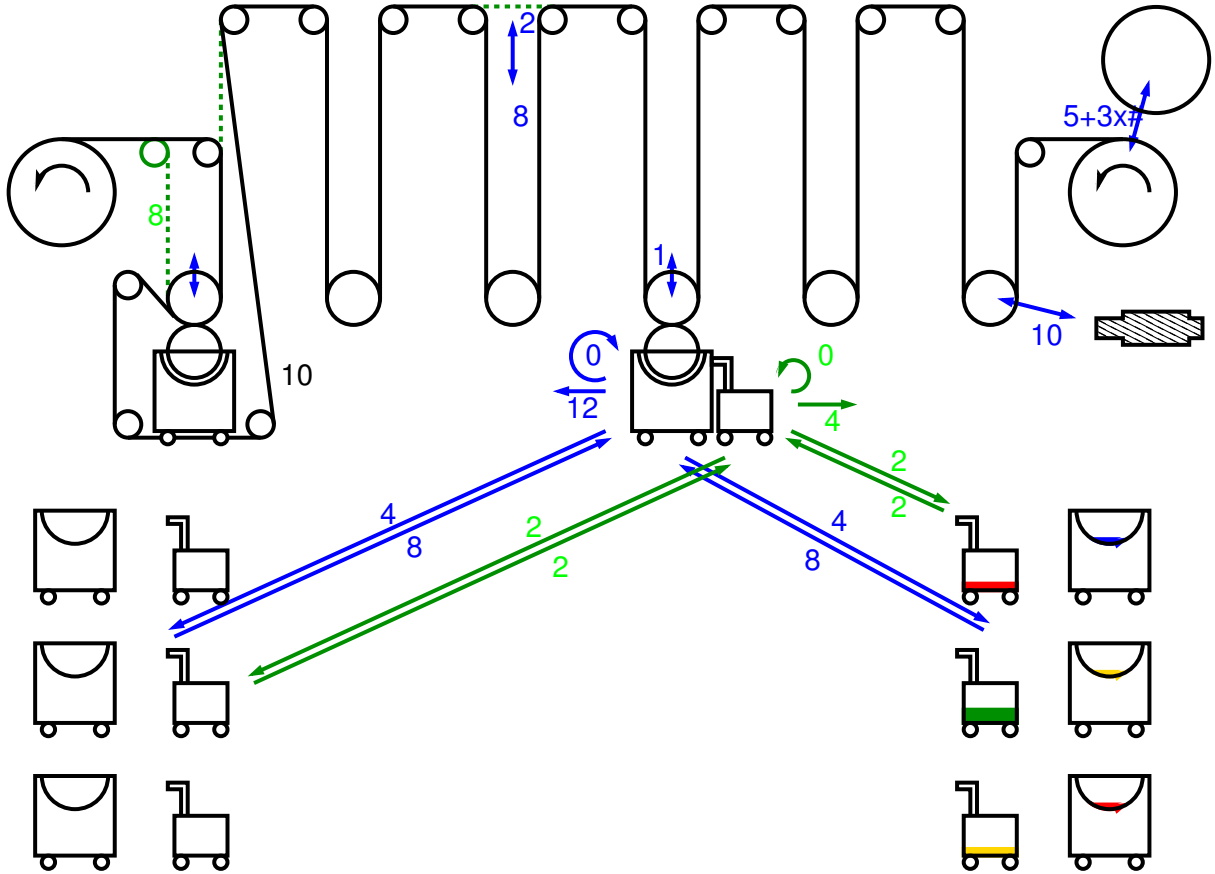


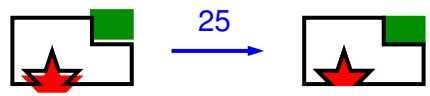
Bestimme Autozuweisung und Reihenfolge für jedes Pannenfahrzeug so, dass bereits versprochene Wartezeiten nicht überschritten werden.

Reihenfolgeoptimierung bei langen Rüstzeiten



Zwei Rotationstiefdruckmaschinen
für den Druck von Geschenkpapier



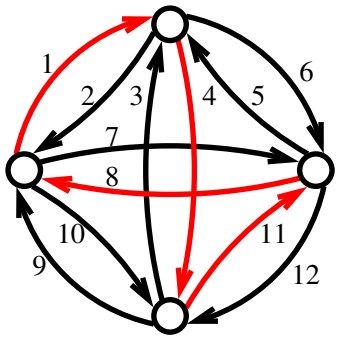


Passer

Farbanpassung neue Farben: 20

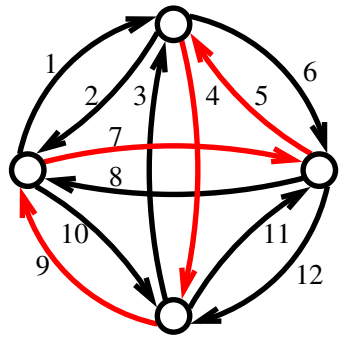
Modellierung als ganzzahliges Programm

Abstrakte Formulierung über konvexe Hülle der Inzidenzvektoren:



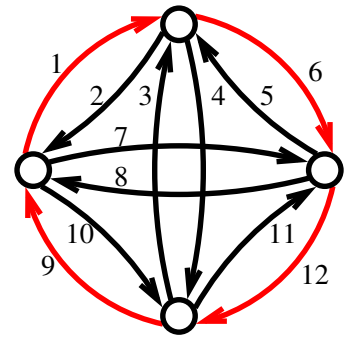
$R_1 = \{1, 4, 8, 11\}$

$$\chi(R_1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



$R_2 = \{4, 5, 7, 9\}$

$$\chi(R_2) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



$R_3 = \{1, 6, 9, 12\}$

$$\chi(R_3) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Für Kantenlängen $c \in \mathbb{R}^E$ ist die Länge von Rundreise R: $\sum_{e \in R} c_e = c^T \chi(R)$.

(TSP) min $c^T x$ s.t. $x \in \text{conv}\{\chi(R) : R \text{ Rundreise in } D = (V, E)\} =: P_{TSP}$

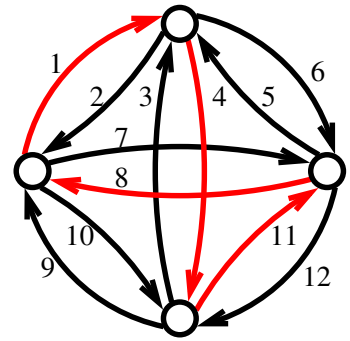
Wäre exakt, aber keine lineare Beschreibung $A_I x \leq b_I$ von P_{TSP} bekannt!

Ganzzahlige Formulierung für (TSP)

Ziel: P_{TSP} durch ein größeres Polytop $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_{TSP}$ möglichst eng einfassen, sodass wenigstens $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$.

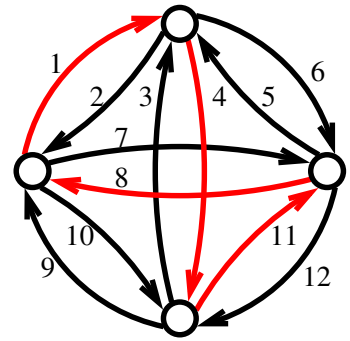
Eine Gleichung/Ungleichung heißt gültig für P_{TSP}, wenn sie für alle $x \in \{\chi(R) : R \text{ Rundreise}\}$ erfüllt ist.

Vorschläge?



Ganzzahlige Formulierung für (TSP)

Ziel: P_{TSP} durch ein größeres Polytop $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_{TSP}$ möglichst eng einfassen, sodass wenigstens $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$.



Eine Gleichung/Ungleichung heißt gültig für P_{TSP}, wenn sie für alle $x \in \{\chi(R) : R \text{ Rundreise}\}$ erfüllt ist.

0-1 Würfel: $0 \leq x \leq 1$ ist gültig

Gradgleichungen:
aus jedem und in jeden Knoten geht genau ein Pfeil,

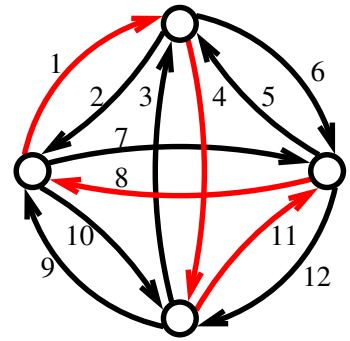
$$\text{für } v \in V: \quad \sum_{(v,u)\in E} x_{(v,u)} = 1, \quad \sum_{(u,v)\in E} x_{(u,v)} = 1$$

(pro Zeile/Spalte genau eine 1 $\leftrightarrow$ Zuweisungsproblem)

Schon Formulierung? $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$?

Ganzzahlige Formulierung für (TSP)

Ziel: P_{TSP} durch ein größeres Polytop $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_{TSP}$ möglichst eng einfassen, sodass wenigstens $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$.



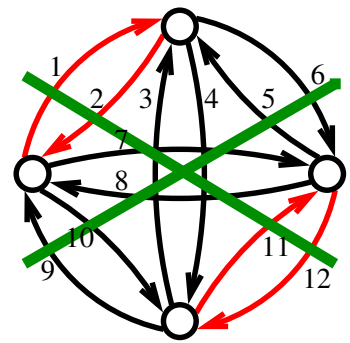
Eine Gleichung/Ungleichung heißt gültig für P_{TSP}, wenn sie für alle $x \in \{\chi(R) : R \text{ Rundreise}\}$ erfüllt ist.

0-1 Würfel: $0 \leq x \leq 1$ ist gültig

Gradgleichungen:
aus jedem und in jeden Knoten geht genau ein Pfeil,

$$\text{für } v \in V: \quad \sum_{(v,u)\in E} x_{(v,u)} = 1, \quad \sum_{(u,v)\in E} x_{(u,v)} = 1$$

(pro Zeile/Spalte genau eine 1 $\leftrightarrow$ Zuweisungsproblem)



Schon Formulierung? $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ Rundreise}\}$?

Kurzzyklusungleichungen:
Aus jeder Knoten-Teilmenge muss mindestens eine Kante hinausführen,

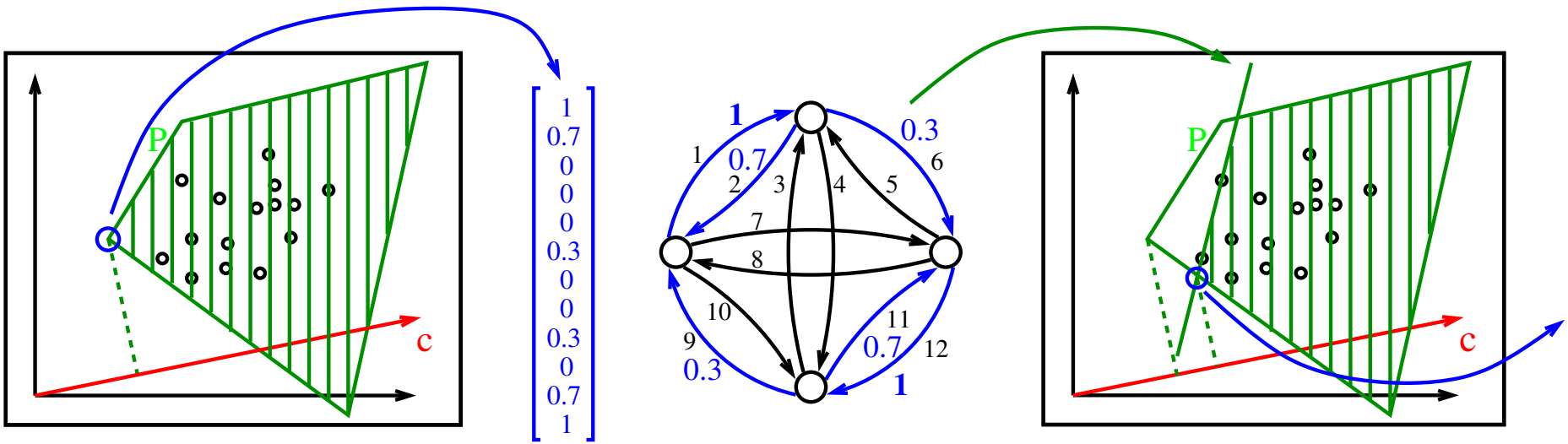
$$\text{für } S \subset V, 2 \leq |S| \leq n-2: \quad \sum_{e\in\delta^+(S)} x_e \geq 1$$

Ist nun eine Formulierung, aber mit etwa 2^n Ungleichungen!

Lösen der TSP LP-Relaxation

Geht nur mit Schnittebenenverfahren:
Die 1. Relaxation bildet das Zuweisungsproblem (Box+Gradgleichungen)
Dessen Lösung ist ganzzahlig und besteht meist aus getrennten Kreisen.

Ab jetzt die Schranke iterativ durch Kurzzyklus-Schnittebenen verbessern

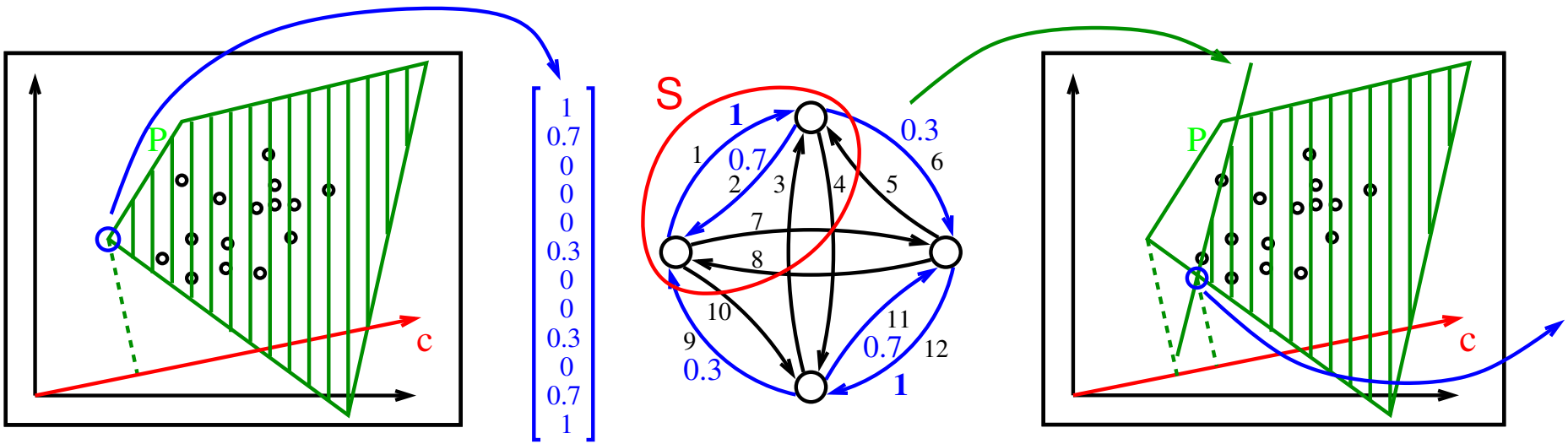


Separationsproblem: Finde ein $S \subset V, 2 \leq |S| \leq n-2$: $\sum_{e \in \delta^+(S)} x_e \not\geq 1$.

Lösen der TSP LP-Relaxation

Geht nur mit Schnittebenenverfahren:
Die 1. Relaxation bildet das Zuweisungsproblem (Box+Gradgleichungen)
Dessen Lösung ist ganzzahlig und besteht meist aus getrennten Kreisen.

Ab jetzt die Schranke iterativ durch Kurzzyklus-Schnittebenen verbessern



Separationsproblem: Finde ein $S \subset V, 2 \leq |S| \leq n-2$: $\sum_{e \in \delta^+(S)} x_e \not\geq 1$.

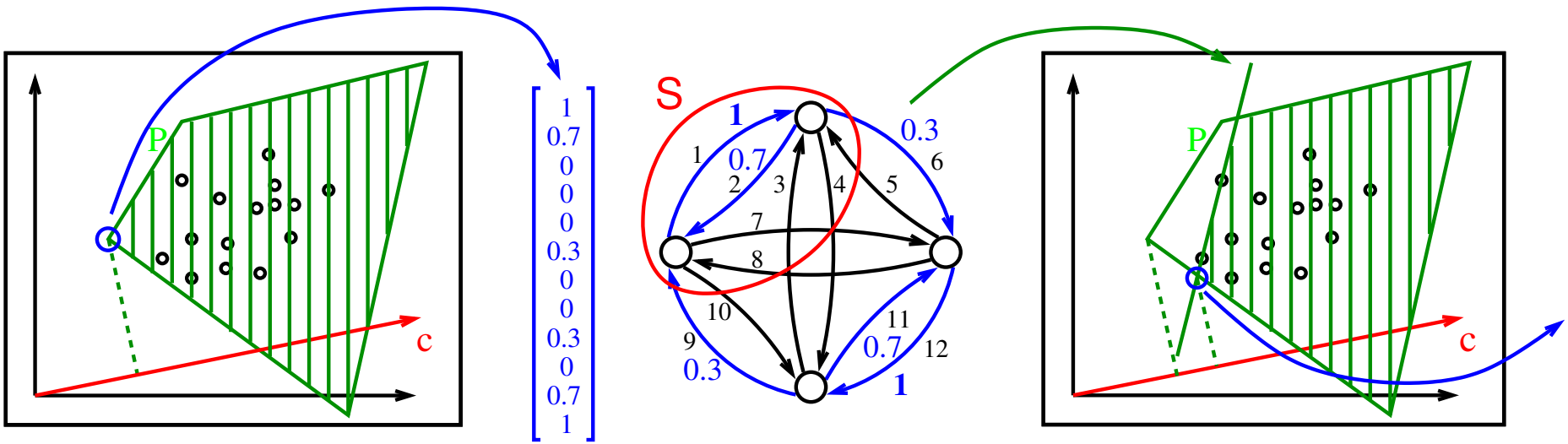
$\Leftrightarrow$ Finde im Netzwerk $D = (V, E)$ mit Kantenkap. x einen Schnitt $x(\delta^+(S)) < 1$.

$\rightarrow$ Maximale s-t-Flüsse/minimale s-t-Schnitte für $s, t \in V$, exakt lösbar!

Lösen der TSP LP-Relaxation

Geht nur mit Schnittebenenverfahren:
Die 1. Relaxation bildet das Zuweisungsproblem (Box+Gradgleichungen)
Dessen Lösung ist ganzzahlig und besteht meist aus getrennten Kreisen.

Ab jetzt die Schranke iterativ durch Kurzzyklus-Schnittebenen verbessern



Separationsproblem: Finde ein $S \subset V, 2 \leq |S| \leq n-2$: $\sum_{e \in \delta^+(S)} x_e \not\geq 1$.

$\Leftrightarrow$ Finde im Netzwerk $D = (V, E)$ mit Kantenkap. x einen Schnitt $x(\delta^+(S)) < 1$.
$\rightarrow$ Maximale s-t-Flüsse/minimale s-t-Schnitte für $s, t \in V$, exakt lösbar!

Grad+Kurzz. liefern sehr gute Schranken, aber noch lange nicht P_{TSP}!
Schranke mit weiteren Unglg. verbesserbar (Kamm-, etc.),
die Lösung der Relaxation wird aber fast nie ganzzahlig!

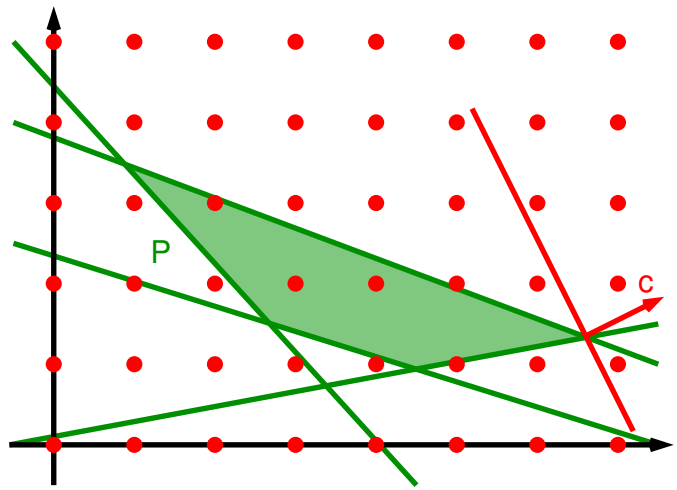
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



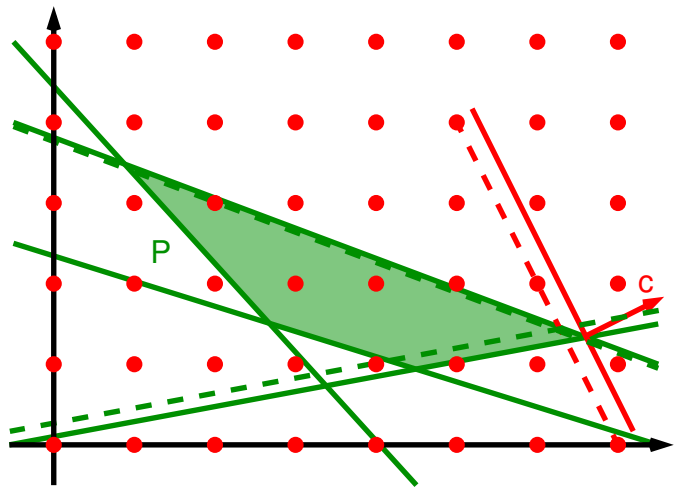
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



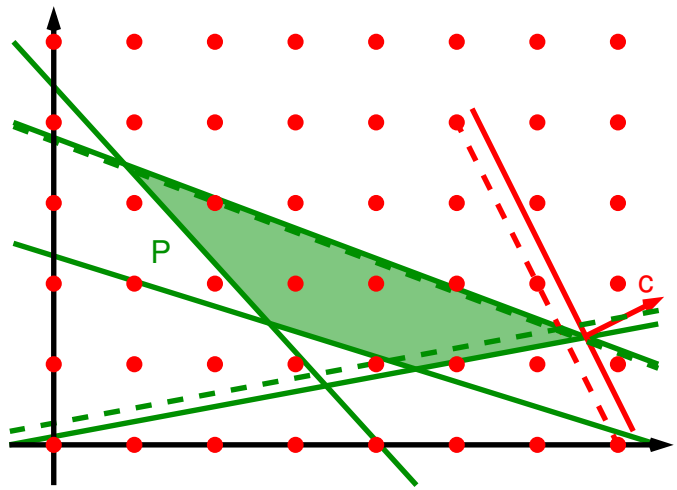
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



- Lift-and-Project Cuts,
- Clique-Ungleichungen,

etc.

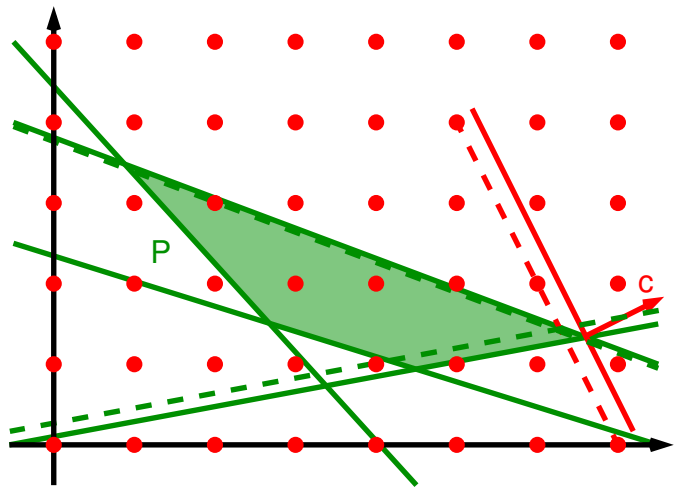
Allgemeine Schnittebenen

Es gibt auch problemübergreifende allgemeine Schnittebenen, die in state-of-the-art Lösern für ganzz. Opt. eingesetzt werden:

- Gomory-Schnitte [Abrunden der Koeffizienten durch $\lfloor \cdot \rfloor$]

  Ist $a^T x \leq \beta$ gültig für $x \in P \cap \mathbb{Z}^n_+$ dann ist wegen $\lfloor a \rfloor^T x \leq a^T x$ auch $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ gültig.

  So eine verletzte Unglg. ist automatisch aus nicht ganzz. OLs erzeugbar.



- Lift-and-Project Cuts,
- Clique-Ungleichungen,

etc.

LP-Relaxation mit Schnittebenen erzeugt gute Schranken (obere für Maximierungs-, untere für Minimierungsprobleme), die Lösungen der Relaxation sind (fast) nie ganzzahlig, führen aber oft in die Nähe guter ganzzahliger Lösungen.

Inhaltsübersicht

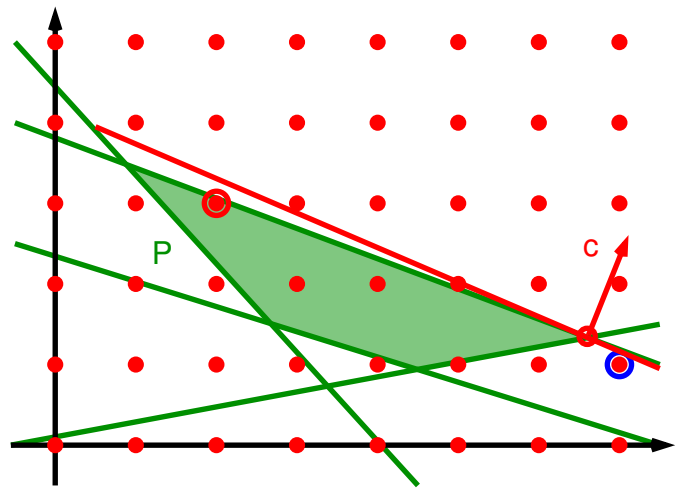
1.10 Finden „guter“ Lösungen, Heuristiken

[Heuristik hat den griechischen Wortstamm finden/erfinden]

Für „kleine“ $x \in \mathbb{Z}^n$ liefert normales Runden der LP-Lösung meist unzulässige oder schlechte Lösungen (selbst bei guter Schranke).

Es kann vorkommen, dass kein zulässiger Punkt in der Nähe der LP-Lösung ist!

In state-of-the-art Lösern gibt es aufwendige Standard-Rundeheuristiken (feasibility pump).



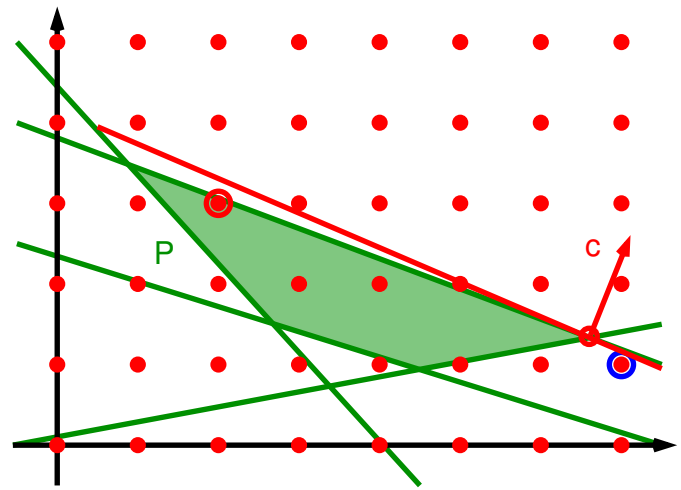
1.10 Finden „guter“ Lösungen, Heuristiken

[Heuristik hat den griechischen Wortstamm finden/erfinden]

Für „kleine“ $x \in \mathbb{Z}^n$ liefert normales Runden der LP-Lösung meist unzulässige oder schlechte Lösungen (selbst bei guter Schranke).

Es kann vorkommen, dass kein zulässiger Punkt in der Nähe der LP-Lösung ist!

In state-of-the-art Lösern gibt es aufwendige Standard-Rundeheuristiken (feasibility pump).



Generell: Ganzzahlige Probleme sind meist *NP*-schwer, haben viele „lokale Optima“ (keine benachbarte bessere Lösung) und es ist unwahrscheinlich, dass man die Optimallösung ohne Enumeration findet. Es kann sogar schwer sein, überhaupt eine zulässige Lösung zu finden!

$\rightarrow$ In Anwendungen nützt man möglichst viel problemspezifisches Wissen!

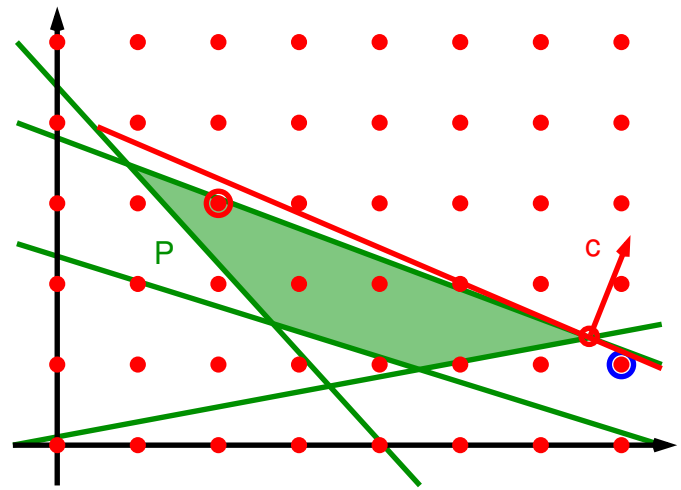
1.10 Finden „guter“ Lösungen, Heuristiken

[Heuristik hat den griechischen Wortstamm finden/erfinden]

Für „kleine“ $x \in \mathbb{Z}^n$ liefert normales Runden der LP-Lösung meist unzulässige oder schlechte Lösungen (selbst bei guter Schranke).

Es kann vorkommen, dass kein zulässiger Punkt in der Nähe der LP-Lösung ist!

In state-of-the-art Lösern gibt es aufwendige Standard-Rundeheuristiken (feasibility pump).



Generell: Ganzzahlige Probleme sind meist *NP*-schwer, haben viele „lokale Optima“ (keine benachbarte bessere Lösung) und es ist unwahrscheinlich, dass man die Optimallösung ohne Enumeration findet. Es kann sogar schwer sein, überhaupt eine zulässige Lösung zu finden!

$\longrightarrow$ In Anwendungen nützt man möglichst viel problemspezifisches Wissen!

Grober Ablauf:

- (zulässige?) Startlösung erzeugen (meist aus der LP-Lösung)
- Iterative Verbesserung der Lösung durch lokale Suche (lokal exakt, Simulated Annealing, Tabu Search, Genetische Algorithmen, etc.)

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:

  LP-Relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:

  LP-Relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.
- Abweichungen von Nebenbedingungen mit großen Dualvariablen erzeugen große Verluste im Zielfunktionswert.

  $\Rightarrow$ Versuche, diese beim Runden zu erfüllen.

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:
  $$\text{LP-Relaxation:} \qquad \sum_{i \in N} x_i = 1,\ x_i \in [0,1]$$
  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.
- Abweichungen von Nebenbedingungen mit großen Dualvariablen erzeugen große Verluste im Zielfunktionswert.
  $\Rightarrow$ Versuche, diese beim Runden zu erfüllen.
- Sukzessives Fixieren: Setze eine oder mehrere Variablen, deren Wert „fast“ ganzzahlig ist, auf den gerundeten Wert und löse das LP für die restlichen Variablen erneut (u.U. rückgängig machen, falls nun unzulässig).

Startlösung aus der LP-Relaxation

Typische Ansätze:

- Oft darf aus mehreren $\{0,1\}$-Variablen nur eine gewählt werden:
  LP-Relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$
  Interpretiere x_i als Wahrscheinlichkeit, dass x_i auf 1 gesetzt werden soll und erzeuge damit mehrere Lösungen zufällig, nimm die beste.
- Abweichungen von Nebenbedingungen mit großen Dualvariablen erzeugen große Verluste im Zielfunktionswert.
  $\Rightarrow$ Versuche, diese beim Runden zu erfüllen.
- Sukzessives Fixieren: Setze eine oder mehrere Variablen, deren Wert „fast“ ganzzahlig ist, auf den gerundeten Wert und löse das LP für die restlichen Variablen erneut (u.U. rückgängig machen, falls nun unzulässig).

Für einige grundlegende Probleme gibt es Rundungsverfahren, die aus LP-Lösung gerundete Lösungen mit Gütegarantie erzeugen (Approximationsalgorithmen), diese sind gute Ideenlieferanten für eigene Rundungsverfahren.

Verbesserungsverfahren allgemein

Gemeinsame Grundelemente:

- **Suchumgebung/Nachbarschaft erklären:** beschreibt, welche Lösungen ausgehend von der derzeitigen untersucht werden können oder sollen (etwa alle durch gewisse Austauschschritte erreichbaren Lösungen, oder Freigeben gewisser Variablen für lokales Nachoptimieren, etc.)
  Mathematisch: Jeder Lösung $\hat{x}$ wird eine (Nachbarschafts-) Menge $\mathcal{N}(\hat{x})$ von benachbarten Lösungen zugeordnet.

Verbesserungsverfahren allgemein

Gemeinsame Grundelemente:

- **Suchumgebung/Nachbarschaft erklären:** beschreibt, welche Lösungen ausgehend von der derzeitigen untersucht werden können oder sollen (etwa alle durch gewisse Austauschschritte erreichbaren Lösungen, oder Freigeben gewisser Variablen für lokales Nachoptimieren, etc.)
  Mathematisch: Jeder Lösung $\hat{x}$ wird eine (Nachbarschafts-) Menge $\mathcal{N}(\hat{x})$ von benachbarten Lösungen zugeordnet.
- **Fortschrittsmaß definieren:** Bewertungsfunktion $f(x)$ für neu erzeugte Lösungen, die Zielfunktion und Bestrafung von Unzulässigkeiten kombiniert
  [s. Merit- und Filter-Ansatz der nichtlin. Opt.]

Verbesserungsverfahren allgemein

Gemeinsame Grundelemente:

- **Suchumgebung/Nachbarschaft erklären:** beschreibt, welche Lösungen ausgehend von der derzeitigen untersucht werden können oder sollen (etwa alle durch gewisse Austauschschritte erreichbaren Lösungen, oder Freigeben gewisser Variablen für lokales Nachoptimieren, etc.)
  Mathematisch: Jeder Lösung $\hat{x}$ wird eine (Nachbarschafts-) Menge $\mathcal{N}(\hat{x})$ von benachbarten Lösungen zugeordnet.
- **Fortschrittsmaß definieren:** Bewertungsfunktion $f(x)$ für neu erzeugte Lösungen, die Zielfunktion und Bestrafung von Unzulässigkeiten kombiniert
  [s. Merit- und Filter-Ansatz der nichtlin. Opt.]
- **Akzeptanz-Schema festlegen:** Gibt an, welche neu erzeugten Lösungen zur Fortsetzung der Suche verwendet werden sollen. Um lokale Optima verlassen zu können, werden ab und zu auch Verschlechterungen akzeptiert.

Lokal exakte Verfahren/lokales Enumerieren

Bestimme $\mathcal{N}(\cdot)$ so, dass $(P_{\hat{x}})$ $\min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$ für jedes $\hat{x}$ durch polynomiale Verfahren oder vollständige Enumeration exakt lösbar ist.

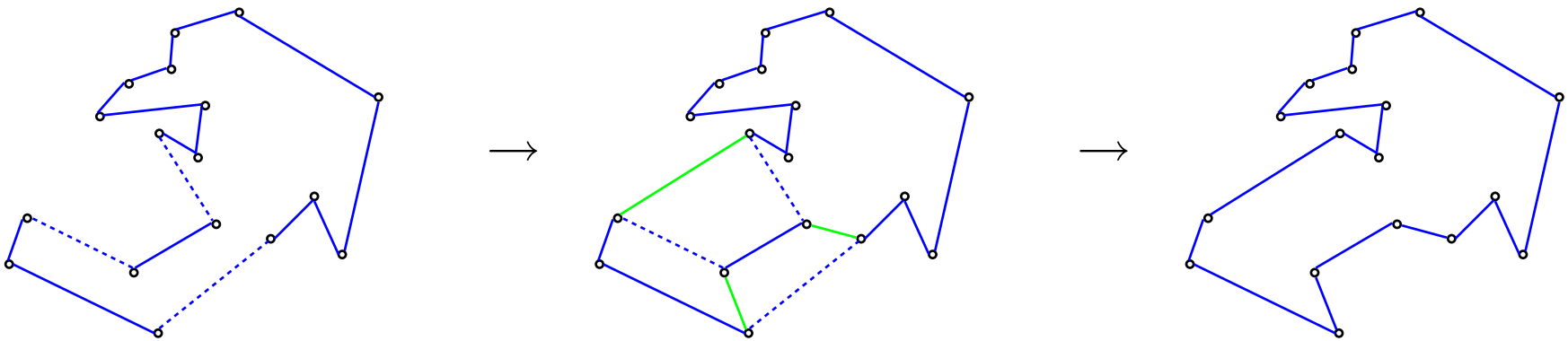
0. Bestimme eine Startlösung $\hat{x}$
1. Löse $(P_{\hat{x}}) \rightarrow \bar{x}$
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1., sonst STOP.

Lokal exakte Verfahren/lokales Enumerieren

Bestimme $\mathcal{N}(\cdot)$ so, dass $(P_{\hat{x}})$ $\min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$ für jedes $\hat{x}$ durch polynomiale Verfahren oder vollständige Enumeration exakt lösbar ist.

0. Bestimme eine Startlösung $\hat{x}$
1. Löse $(P_{\hat{x}}) \rightarrow \bar{x}$
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1., sonst STOP.

Bsp: 3-opt für TSP: 3 Kanten entfernen und neu zusammensetzen

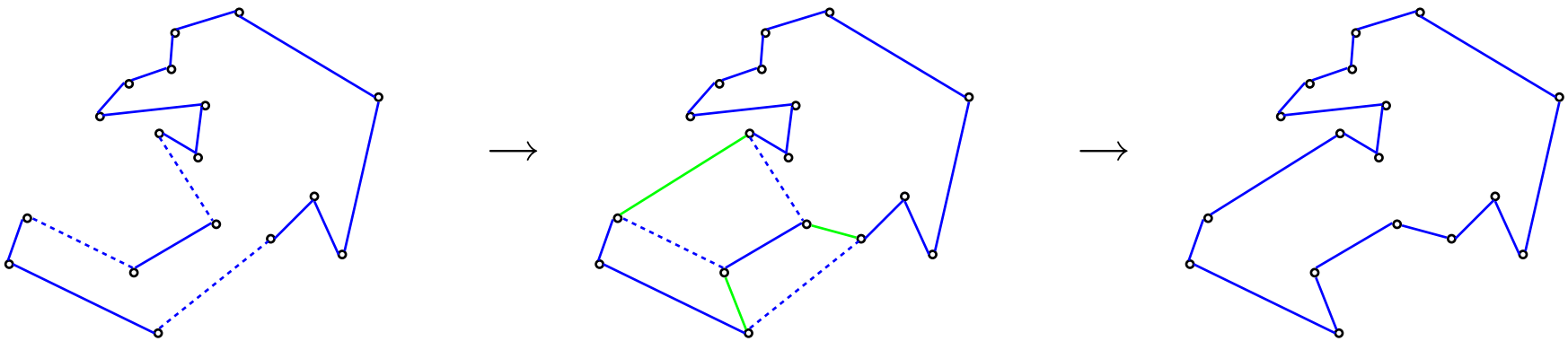


Lokal exakte Verfahren/lokales Enumerieren

Bestimme $\mathcal{N}(\cdot)$ so, dass $(P_{\hat{x}})$ $\min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$ für jedes $\hat{x}$ durch polynomiale Verfahren oder vollständige Enumeration exakt lösbar ist.

0. Bestimme eine Startlösung $\hat{x}$
1. Löse $(P_{\hat{x}}) \rightarrow \bar{x}$
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1., sonst STOP.

Bsp: 3-opt für TSP: 3 Kanten entfernen und neu zusammensetzen



---

die Kunst: Finde eine möglichst mächtige Nachbarschaft, für die $(P_{\hat{x}})$ noch polynomial lösbar ist.

Die Anzahl der Iterationen kann dennoch exponentiell sein!

Simulated Annealing (simuliertes langsames Abkühlen)

Erzeuge, in Schritt k, zufällig ein $\bar{x}$ aus $\mathcal{N}(\hat{x})$. Akzeptiere es, falls $f(\bar{x})$ besser als $f(\hat{x})$, sonst akzeptiere es nur mit Wahrscheinlichkeit

$$\exp\left(\frac{-|f(\hat{x}) - f(\bar{x})|}{T_k}\right) \qquad (0 < T_k \to 0 \text{ für } k \to \infty).$$

0. Bestimme Startlösung $\hat{x}$, Nullfolge $\{T_k > 0\}_{k\in\mathbb{N}}$, setze $k = 0$.
1. Wähle zufällig (gleichverteilt) $\bar{x} \in \mathcal{N}(\hat{x})$ setze $k \leftarrow k + 1$.
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
3. Ziehe gleichverteilt eine Zufallszahl $\zeta \in [0, 1]$.
   Ist $\zeta < \exp(\frac{-|f(\hat{x})-f(\bar{x})|}{c_k})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
4. Gehe zu 1. (ohne $\hat{x}$ zu ändern).

Wähle $\mathcal{N}(\cdot)$ so, dass jedes x über Zwischenstationen erreichbar ist.

Simulated Annealing (simuliertes langsames Abkühlen)

Erzeuge, in Schritt k, zufällig ein $\bar{x}$ aus $\mathcal{N}(\hat{x})$. Akzeptiere es, falls $f(\bar{x})$ besser als $f(\hat{x})$, sonst akzeptiere es nur mit Wahrscheinlichkeit

$$\exp\left(\frac{-|f(\hat{x}) - f(\bar{x})|}{T_k}\right) \qquad (0 < T_k \to 0 \text{ für } k \to \infty).$$

0. Bestimme Startlösung $\hat{x}$, Nullfolge $\{T_k > 0\}_{k\in\mathbb{N}}$, setze $k = 0$.
1. Wähle zufällig (gleichverteilt) $\bar{x} \in \mathcal{N}(\hat{x})$ setze $k \leftarrow k + 1$.
2. Ist $f(\bar{x})$ besser als $f(\hat{x})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
3. Ziehe gleichverteilt eine Zufallszahl $\zeta \in [0, 1]$.
   Ist $\zeta < \exp(\frac{-|f(\hat{x})-f(\bar{x})|}{c_k})$, setze $\hat{x} \leftarrow \bar{x}$ und gehe zu 1.
4. Gehe zu 1. (ohne $\hat{x}$ zu ändern).

Wähle $\mathcal{N}(\cdot)$ so, dass jedes x über Zwischenstationen erreichbar ist.

Geht die (Temperatur-/Abkühlungs-)Folge T_k sehr langsam gegen Null, wird jedes x mit positiver Wahrscheinlichkeit irgendwann besucht (vollständige Enumeration), also auch das Optimum. (Aber wann?)

Tabu-Search

Idee: Versuche möglichst unterschiedliche Lösungen zu erzeugen.

Beschreibe $\mathcal{N}(\cdot)$ durch Änderungsregeln $r \in \mathcal{R}$ und speichere verwendete Regeln in einer Tabuliste $\mathcal{L}$. Für ein neues $\bar{x}$ sollte wenigstens eine Regel $r \in \mathcal{R} \setminus \mathcal{L}$ verwendet werden oder es muss besseren Wert haben.

0. Bestimme Startlösung $\hat{x}$, setze $\mathcal{L} = \emptyset$.
1. Erzeuge einige $x \in \mathcal{N}(\hat{x})$ durch mehrmaliges zufälliges Anwenden von Regeln aus $\mathcal{R}$, sammle diese in einer Menge S.
2. Wähle ein $\bar{x}$ aus S nach Tabuliste $\mathcal{L}$ und $f(\cdot)$.
3. Aktualisiere die Tabuliste $\mathcal{L}$, setze $\hat{x} \leftarrow \bar{x}$, gehe zu 1.

Tabu-Search

Idee: Versuche möglichst unterschiedliche Lösungen zu erzeugen.

Beschreibe $\mathcal{N}(\cdot)$ durch Änderungsregeln $r \in \mathcal{R}$ und speichere verwendete Regeln in einer Tabuliste $\mathcal{L}$. Für ein neues $\bar{x}$ sollte wenigstens eine Regel $r \in \mathcal{R} \setminus \mathcal{L}$ verwendet werden oder es muss besseren Wert haben.

0. Bestimme Startlösung $\hat{x}$, setze $\mathcal{L} = \emptyset$.
1. Erzeuge einige $x \in \mathcal{N}(\hat{x})$ durch mehrmaliges zufälliges Anwenden von Regeln aus $\mathcal{R}$, sammle diese in einer Menge S.
2. Wähle ein $\bar{x}$ aus S nach Tabuliste $\mathcal{L}$ und $f(\cdot)$.
3. Aktualisiere die Tabuliste $\mathcal{L}$, setze $\hat{x} \leftarrow \bar{x}$, gehe zu 1.

Bsp. TSP: $\mathcal{R} = \{r_{ij} := \text{vertausche Reihenfolge von Stadt } i \text{ und } j\}$.
$\mathcal{L} = \{r_{ij} : r_{ij} \text{ wurde in den letzten } n/10 \text{ Schritten verwendet}\}$

Tabu-Search

Idee: Versuche möglichst unterschiedliche Lösungen zu erzeugen.

Beschreibe $\mathcal{N}(\cdot)$ durch Änderungsregeln $r \in \mathcal{R}$ und speichere verwendete Regeln in einer Tabuliste $\mathcal{L}$. Für ein neues $\bar{x}$ sollte wenigstens eine Regel $r \in \mathcal{R} \setminus \mathcal{L}$ verwendet werden oder es muss besseren Wert haben.

0. Bestimme Startlösung $\hat{x}$, setze $\mathcal{L} = \emptyset$.
1. Erzeuge einige $x \in \mathcal{N}(\hat{x})$ durch mehrmaliges zufälliges Anwenden von Regeln aus $\mathcal{R}$, sammle diese in einer Menge S.
2. Wähle ein $\bar{x}$ aus S nach Tabuliste $\mathcal{L}$ und $f(\cdot)$.
3. Aktualisiere die Tabuliste $\mathcal{L}$, setze $\hat{x} \leftarrow \bar{x}$, gehe zu 1.

Bsp. TSP: $\mathcal{R} = \{r_{ij} := \text{vertausche Reihenfolge von Stadt } i \text{ und } j\}$.
$\mathcal{L} = \{r_{ij} : r_{ij} \text{ wurde in den letzten } n/10 \text{ Schritten verwendet}\}$

Über die Regeln $\mathcal{R}$ sollte jede Lösung erreichbar sein.
Allgemeine theoretische Resultate oder Qualitätsgarantien gibt es nicht.

Genetische Algorithmen

Idee: Lasse die Evolution für Dich arbeiten (und warte derweilen).

Erzeuge aus einer Population von Lösungen durch Selektion (Auswahl der nächsten Eltern), Rekombination/Vermehrung (Austausch von Teillösungen), Mutation (zufälliges Verändern) die nächste Population.

0. Wähle $k \in \mathbb{N}$ und bestimme eine Startpopulation $\mathcal{P}$, $|\mathcal{P}| \geq 2k$.
1. Bestimme die durchschnittliche Fitness $\bar{f} = \sum_{x \in \mathcal{P}} f(x)/|\mathcal{P}|$
2. Lösche x aus $\mathcal{P}$ mit Wahrscheinlichkeit prop. zu $\frac{f(x)}{\bar{f}}$, bis $|P| = 2k$.
3. Bilde zufällig k Paare aus $\mathcal{P}$, erzeuge für jedes Paar einige Nachkommen durch Rekombination und Mutation $\longrightarrow \bar{\mathcal{P}}$
4. Setze $\mathcal{P} \leftarrow \bar{\mathcal{P}}$, gehe zu 1.

Genetische Algorithmen

Idee: Lasse die Evolution für Dich arbeiten (und warte derweilen).

Erzeuge aus einer Population von Lösungen durch Selektion (Auswahl der nächsten Eltern), Rekombination/Vermehrung (Austausch von Teillösungen), Mutation (zufälliges Verändern) die nächste Population.

0. Wähle $k \in \mathbb{N}$ und bestimme eine Startpopulation $\mathcal{P}$, $|\mathcal{P}| \geq 2k$.
1. Bestimme die durchschnittliche Fitness $\bar{f} = \sum_{x \in \mathcal{P}} f(x)/|\mathcal{P}|$
2. Lösche x aus $\mathcal{P}$ mit Wahrscheinlichkeit prop. zu $\frac{f(x)}{\bar{f}}$, bis $|P| = 2k$.
3. Bilde zufällig k Paare aus $\mathcal{P}$, erzeuge für jedes Paar einige Nachkommen durch Rekombination und Mutation $\longrightarrow \bar{\mathcal{P}}$
4. Setze $\mathcal{P} \leftarrow \bar{\mathcal{P}}$, gehe zu 1.

- Viele Experimente mit Population der Größe 1 (!!!)
- Theorie besagt, dass Simulated Annealing eher Optima findet.

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.
- Man kann viele Parameter einstellen.

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.
- Man kann viele Parameter einstellen.

Nachteile:

- Man muss viele Parameter einstellen, ohne gute Richtschnur!

Bemerkungen

- SA, TS, GA sind Meta-Heuristiken (Schema ohne Problembezug).
- Meta-Heuristik-Industrie betrachtet beliebige Kombinationen, jede Menge „neue“ Verfahren (Ant-Colony-, Particle-swarm-, etc.)
- Fast alle hoffen, mit (leicht gesteuertem) Zufall eine Optimallösung zu finden (passt zu *NP*!).

Vorteile:

- Auch bei wenig Problemverständnis ist rasch etwas implementiert, erste Lösungen sind schnell erzeugt, Regeln schnell geändert.
- Man kann viele Parameter einstellen.

Nachteile:

- Man muss viele Parameter einstellen, ohne gute Richtschnur!
- Konvergenz der Verfahren sagt nichts über die Qualität der Lösung. Ohne dazupassende Relaxation hat man keine Ahnung, wie weit man vom Optimum entfernt ist (manchmal sehr weit).

Inhaltsübersicht

1.11 Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

1.11 Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten

Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$. Welche Standorte sollen geöffnet werden?

Variablen:

1.11 Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten
Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$.
Welche Standorte sollen geöffnet werden?

Variablen:
$x_m \in \{0, 1\}$, $m \in M$ ... Versandlager wird in m errichtet.
$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ... Menge, die Kunde k von Standort m erhält.

Nebenbedingungen:

1.11 Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten

Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$. Welche Standorte sollen geöffnet werden?

Variablen:

$x_m \in \{0, 1\}$, $m \in M$ ... Versandlager wird in m errichtet.

$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ... Menge, die Kunde k von Standort m erhält.

Nebenbedingungen:

$\sum_{m \in M} x_{km} = b_k$, $k \in K$ ... Kunde k erhält seine Bestellung

$\sum_{k \in K} x_{km} \leq b_m x_m$, $m \in M$ ... Standort m verteilt maximal b_m.

Zielfunktion:

1.11 Gemischt-ganzzahlige Optimierung (MIP)

Für eine Teilmenge der Indices $G \subseteq \{1, \ldots, n\}$ soll x ganzzahlig sein.

$$\max\ c^T x \text{ s.t. } Ax \geq b,\ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Enthält ganzz. Optimierung als Spezialfall, ist aber deutlich breiter.

Anwendungsbeispiel: Standortoptimierung mit Fixkosten

Gegeben eine Menge K an Kunden mit Bedarfen b_k und eine Menge M möglicher Standorte für Versandlager, jeweils mit Mietkosten c_m, Kapazität b_m und Transportkosten c_{km} pro Einheit für $k \in K, m \in M$. Welche Standorte sollen geöffnet werden?

Variablen:
$x_m \in \{0, 1\}$, $m \in M$ ... Versandlager wird in m errichtet.
$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ... Menge, die Kunde k von Standort m erhält.

Nebenbedingungen:
$\sum_{m \in M} x_{km} = b_k$, $k \in K$ ... Kunde k erhält seine Bestellung
$\sum_{k \in K} x_{km} \leq b_m x_m$, $m \in M$ ... Standort m verteilt maximal b_m.

Zielfunktion:
$\min \sum_{k \in K, m \in M} c_{km} x_{km} + \sum_{m \in M} c_m x_m = c^T x$

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

Bsp: Fährt Zug A mit Abfahrtszeit t_A vor Zug B mit Abfahrtszeit t_B, ist eine Mindestzugfolgezeit $t_{AB} > 0$ einzuhalten, also $t_B \geq t_A + t_{AB}$. Für Zug B vor Zug A muss wiederum $t_A \geq t_B + t_{BA}$ erfüllt sein. Die nachfolgenden Züge dürfen natürlich auch viel später fahren.

Variablen

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

Bsp: Fährt Zug A mit Abfahrtszeit t_A vor Zug B mit Abfahrtszeit t_B, ist eine Mindestzugfolgezeit $t_{AB} > 0$ einzuhalten, also $t_B \geq t_A + t_{AB}$. Für Zug B vor Zug A muss wiederum $t_A \geq t_B + t_{BA}$ erfüllt sein. Die nachfolgenden Züge dürfen natürlich auch viel später fahren.

Variablen ($M \gg 0$, größer als späteste Abfahrtszeit von t_A und t_B):
$x_{AB} \in \{0, 1\}$ . . . 1 falls A vor B fährt, sonst 0
$t_A, t_B \in [0, M]$. . . Abfahrtszeit

Nebenbedingungen:
$t_A + t_{AB} \leq t_B + M(1 - x_{AB})$ . . . nur wichtig, wenn $x_{AB} = 1$
$t_B + t_{BA} \leq t_A + M x_{AB}$ . . . nur wichtig, wenn $x_{AB} = 0$.

MIP-Modellierungstechniken:

Bedingte Ungleichungen: Ungleichungen, die nur bei bestimmten Entscheidungen beachtet werden müssen, können bei entgegengesetzter Entscheidung durch einen big M-Term erfüllt werden.

Bsp: Fährt Zug A mit Abfahrtszeit t_A vor Zug B mit Abfahrtszeit t_B, ist eine Mindestzugfolgezeit $t_{AB} > 0$ einzuhalten, also $t_B \geq t_A + t_{AB}$. Für Zug B vor Zug A muss wiederum $t_A \geq t_B + t_{BA}$ erfüllt sein. Die nachfolgenden Züge dürfen natürlich auch viel später fahren.

Variablen ($M \gg 0$, größer als späteste Abfahrtszeit von t_A und t_B):
$x_{AB} \in \{0, 1\}$ ... 1 falls A vor B fährt, sonst 0
$t_A, t_B \in [0, M]$... Abfahrtszeit

Nebenbedingungen:
$t_A + t_{AB} \leq t_B + M(1 - x_{AB})$ ... nur wichtig, wenn $x_{AB} = 1$
$t_B + t_{BA} \leq t_A + M x_{AB}$ ... nur wichtig, wenn $x_{AB} = 0$.

- M zu groß $\rightarrow$ Unglg. in LP-Relaxation wirkungslos $\rightarrow$ schlechte Schranke
- gut, falls Verletzungsspielraum der Unglg. gut abschätzbar (siehe das Standortplanungsbeispiel)
- in Branch&Bound hilfreich, wenn auf der Variable gebrancht wird

Modellierung logischer Bedingungen

Für ein $x_i \in \{0, 1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0, 1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0, 1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	$x_3 \geq x_1,\ x_3 \geq x_2,\ x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ und } x_2)$	$x_3 \leq x_1,\ x_3 \leq x_2,\ x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0,1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	$x_3 \geq x_1,\ x_3 \geq x_2,\ x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ und } x_2)$	$x_3 \leq x_1,\ x_3 \leq x_2,\ x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	$x_1 \leq x_2$
$x_1 \Leftrightarrow x_2$	

Modellierung logischer Bedingungen

Für ein $x_i \in \{0,1\}$ steht $x_i = 1$ oft für „Aussage i ist wahr“.
Logische Verknüpfungen sind dann so erzeugbar:

Log. Bedingung	Formulierung
$x_2 = (\text{nicht } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ oder } x_2)$	$x_3 \geq x_1,\ x_3 \geq x_2,\ x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ und } x_2)$	$x_3 \leq x_1,\ x_3 \leq x_2,\ x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	$x_1 \leq x_2$
$x_1 \Leftrightarrow x_2$	$x_1 = x_2$

Bemerkung: Zusammen mit $0 \leq x_i \leq 1$ beschreiben die Nebenbedinungen

$$\operatorname{conv}\left\{ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \{0,1\}^3 : \text{die } x_i \text{ erfüllen die logische Bedingung} \right\}.$$

Mit dieser Technik sind Formulierungen weiterer Beziehungen ableitbar.
Übung: $x_3 = (x_1 \text{ xor } x_2)$

Schnittebenen für MIP

Wie in der ganzz. Optimierung ist „conv“ die beste lineare Relaxation,

$$P_G := \mathrm{conv}\{x \in \mathbb{R}^n : Ax \leq b, x_G \in \mathbb{Z}^G\}.$$

$Ax \leq b$ ist die LP-Relaxation, P_G wird durch Schnittebenen angenähert.

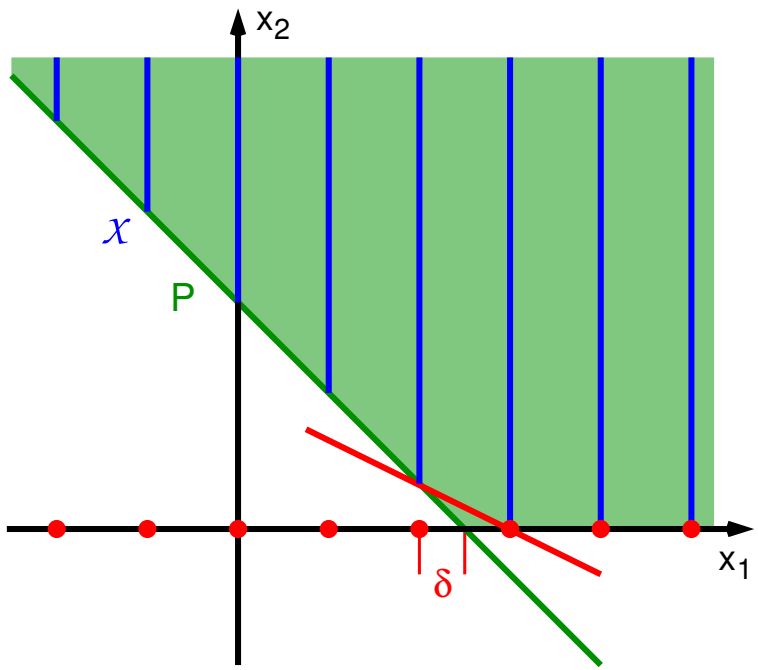
Schnittebenen für MIP

Wie in der ganzz. Optimierung ist „conv“ die beste lineare Relaxation,

$$P_G := \mathrm{conv}\{x \in \mathbb{R}^n : Ax \le b, x_G \in \mathbb{Z}^G\}.$$

$Ax \le b$ ist die LP-Relaxation, P_G wird durch Schnittebenen angenähert.

Bsp: Mixed Integer Rounding Ungleichung (MIR)
Einfachste Form: $\mathcal{X} = \{(x_1, x_2) \in \mathbb{Z} \times \mathbb{R}_+ : x_1 + x_2 \ge \beta\}$, $\beta \in \mathbb{R}$



Setze $\delta := \beta - \lfloor \beta \rfloor$,
dann ist die Unglg.

$$x_1 + \frac{1}{\delta} x_2 \ge \lceil \beta \rceil$$

gültig für $\mathcal{X}$.

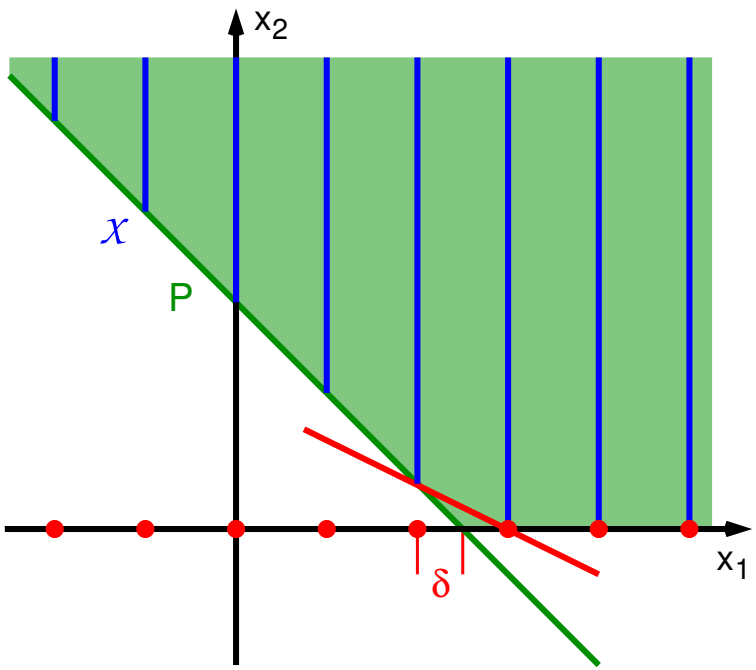
Schnittebenen für MIP

Wie in der ganzz. Optimierung ist „conv“ die beste lineare Relaxation,

$$P_G := \mathrm{conv}\{x \in \mathbb{R}^n : Ax \leq b, x_G \in \mathbb{Z}^G\}.$$

$Ax \leq b$ ist die LP-Relaxation, P_G wird durch Schnittebenen angenähert.

Bsp: Mixed Integer Rounding Ungleichung (MIR)
Einfachste Form: $\mathcal{X} = \{(x_1, x_2) \in \mathbb{Z} \times \mathbb{R}_+ : x_1 + x_2 \geq \beta\}$, $\beta \in \mathbb{R}$



Setze $\delta := \beta - \lfloor \beta \rfloor$,
dann ist die Unglg.

$$x_1 + \frac{1}{\delta} x_2 \geq \lceil \beta \rceil$$

gültig für $\mathcal{X}$.

In state-of-the-art Paketen sind viele weitere enthalten
(flow cover, cliques, etc.)

Branch-and-Cut Frameworks

Wird in einem Branch&Bound-Verfahren für jedes Teilproblem die LP-Relaxation durch Schnittebenen verschärft, spricht man von Branch&Cut-Verfahren. Dies sind derzeit die besten Verfahren für allgemeine gemischt-ganzzahlige Optimierungsprobleme.

Branch-and-Cut Frameworks

Wird in einem Branch&Bound-Verfahren für jedes Teilproblem die LP-Relaxation durch Schnittebenen verschärft, spricht man von Branch&Cut-Verfahren. Dies sind derzeit die besten Verfahren für allgemeine gemischt-ganzzahlige Optimierungsprobleme.

Eine effiziente Branch&Cut Implementierung ist sehr schwer:

- Auswahl des nächsten Teilproblems
- Auswahl der Verzweigungsvariable, Fixieren von Variablen
- Teilprobleme effizient inkrementell speichern
- Schnittebenen über mehrere Teilprobleme verwalten
- effiziente Heuristiken für zulässige Lösungen

etc.

Branch-and-Cut Frameworks

Wird in einem Branch&Bound-Verfahren für jedes Teilproblem die LP-Relaxation durch Schnittebenen verschärft, spricht man von Branch&Cut-Verfahren. Dies sind derzeit die besten Verfahren für allgemeine gemischt-ganzzahlige Optimierungsprobleme.

Eine effiziente Branch&Cut Implementierung ist sehr schwer:

- Auswahl des nächsten Teilproblems
- Auswahl der Verzweigungsvariable, Fixieren von Variablen
- Teilprobleme effizient inkrementell speichern
- Schnittebenen über mehrere Teilprobleme verwalten
- effiziente Heuristiken für zulässige Lösungen

etc.

Es gibt Software-Pakete, die alle Verwaltungsaspekte übernehmen und einem den Einbau weiterer Schnittebenen und Heuristiken ermöglichen.

z.B.: SCIP, Cplex, Gurobi, Abacus ...