LINEARE PROGRAMMIERUNG

LINEARE PROGRAMMIERUNG

Definition (kanonische Form)

Beispiel

Beispiel

Minimiere $\quad 2x_1 - x_2 + 4x_3$

Beispiel

Minimiere $\quad 2x_1 - x_2 + 4x_3 \quad$ *(lineare Zielfunktion)*

Beispiel

Minimiere $\quad 2x_1 - x_2 + 4x_3 \quad$ *(lineare Zielfunktion)*

so dass

$$\begin{aligned} x_1 + x_2 + x_4 &\leq 2 \\ 3x_2 - x_3 &= 5 \\ x_3 + x_4 &\geq 3 \\ x_1 &\geq 0 \\ x_3 &\leq 0 \end{aligned}$$

Beispiel

Minimiere $\quad 2x_1 - x_2 + 4x_3 \quad$ *(lineare Zielfunktion)*

so dass

$$\begin{aligned} x_1 + x_2 + x_4 &\leq 2 \\ 3x_2 - x_3 &= 5 \\ x_3 + x_4 &\geq 3 \\ x_1 &\geq 0 \\ x_3 &\leq 0 \end{aligned}$$

(lineare Nebenbedingungen)

Beispiel

Minimiere $\quad 2x_1 - x_2 + 4x_3 \quad$ *(lineare Zielfunktion)*

so dass

$$\begin{aligned} x_1 + x_2 + x_4 &\leq 2 \\ 3x_2 - x_3 &= 5 \\ x_3 + x_4 &\geq 3 \\ x_1 &\geq 0 \\ x_3 &\leq 0 \end{aligned}$$

(lineare Nebenbedingungen)

Gesucht wird eine belegung der Variablen x_1, x_2, x_3, x_4 die die Nebenbedingungen erfüllt, und die Zielfunktion minimiert.

Beispiel

Minimiere $\quad 2x_1 - x_2 + 4x_3 \quad$ *(lineare Zielfunktion)*

so dass

$$\begin{aligned} x_1 + x_2 + x_4 &\leq 2 \\ 3x_2 - x_3 &= 5 \\ x_3 + x_4 &\geq 3 \\ x_1 &\geq 0 \\ x_3 &\leq 0 \end{aligned}$$

(lineare Nebenbedingungen)

Gesucht wird eine belegung der Variablen x_1, x_2, x_3, x_4 die die Nebenbedingungen erfüllt, und die Zielfunktion minimiert.

(Linear Program, LP)

Ein allgemeines lineares Programmierproblem

Ein allgemeines lineares Programmierproblem

Seien $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b \in \mathbb{Q}^m, \quad c, a_i \in \mathbb{Q}^n,$

Ein allgemeines lineares Programmierproblem

Seien $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b \in \mathbb{Q}^m, \quad c, a_i \in \mathbb{Q}^n,$

M_1, M_2, M_3 eine Partition des $\{1, 2, \ldots, m\}$ und
$N_1, N_2 \subseteq \{1, \ldots, n\}$.

Ein allgemeines lineares Programmierproblem

Seien $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b \in \mathbb{Q}^m, \quad c, a_i \in \mathbb{Q}^n,$

M_1, M_2, M_3 eine Partition des $\{1, 2, \ldots, m\}$ und $N_1, N_2 \subseteq \{1, \ldots, n\}$.

Wir betrachten Optimierungsprobleme der Form:

Minimiere/Maximiere $\quad c^T \cdot x$

Ein allgemeines lineares Programmierproblem

Seien $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b \in \mathbb{Q}^m, \quad c, a_i \in \mathbb{Q}^n,$

M_1, M_2, M_3 eine Partition des $\{1, 2, \ldots, m\}$ und
$N_1, N_2 \subseteq \{1, \ldots, n\}$.

Wir betrachten Optimierungsprobleme der Form:

Minimiere/Maximiere $\quad c^T \cdot x$ so dass

$$
\begin{array}{rcll}
a_i^T \cdot x & \geq & b_i & i \in M_1 \\
a_i^T \cdot x & = & b_i & i \in M_2 \\
a_i^T \cdot x & \leq & b_i & i \in M_3 \\
x_j & \geq & 0 & j \in N_1 \\
x_j & \leq & 0 & j \in N_2
\end{array}
$$

Ein allgemeines lineares Programmierproblem

Seien $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b \in \mathbb{Q}^m, \quad c, a_i \in \mathbb{Q}^n,$

M_1, M_2, M_3 eine Partition des $\{1, 2, \ldots, m\}$ und $N_1, N_2 \subseteq \{1, \ldots, n\}$.

Wir betrachten Optimierungsprobleme der Form:

Minimiere/Maximiere $\quad c^T \cdot x$ so dass

$$\begin{array}{rcll} a_i^T \cdot x & \geq & b_i & i \in M_1 \\ a_i^T \cdot x & = & b_i & i \in M_2 \\ a_i^T \cdot x & \leq & b_i & i \in M_3 \\ x_j & \geq & 0 & j \in N_1 \\ x_j & \leq & 0 & j \in N_2 \end{array}$$

Definition:

Definition:

Lineares Programm in *kanonischer Form:*

Definition:

Lineares Programm in *kanonischer Form:*

Minimiere $\quad c^T \cdot x \quad$ so dass $\quad A \cdot x \geq b$

Definition:

Lineares Programm in *kanonischer Form:*

Minimiere $\quad c^T \cdot x \quad$ so dass $\quad A \cdot x \geq b$

(hier $x^T = [x_1, x_2, \ldots, x_n]$, $\quad c \in \mathbb{Q}^n$, $\quad A \in \mathbb{Q}^{m \times n}$, $\quad b \in \mathbb{Q}^m$)

Definition:

Lineares Programm in *kanonischer Form:*

Minimiere $c^T \cdot x$ so dass $A \cdot x \geq b$

(hier $x^T = [x_1, x_2, \ldots, x_n]$, $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$)

Beobachtung: Jedes LP hat ein äquivalentes LP in kanonischer Form.

Definition:

Lineares Programm in *kanonischer Form:*

Minimiere $\quad c^T \cdot x \quad$ so dass $\quad A \cdot x \geq b$

(hier $x^T = [x_1, x_2, \ldots, x_n]$, $\quad c \in \mathbb{Q}^n$, $\quad A \in \mathbb{Q}^{m \times n}$, $\quad b \in \mathbb{Q}^m$)

Beobachtung: Jedes LP hat ein äquivalentes LP in kanonischer Form.

$\Rightarrow$ die kanonische Form ist allgemein genug!

Die kanonische Form für unser erstes Beispiel:

Minimiere $\quad [2, -1, 4, 0] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = c^T \cdot x \quad$ so dass

$$\begin{bmatrix} -1 & -1 & 0 & -1 \\ 0 & 3 & -1 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} -2 \\ 5 \\ -5 \\ 3 \\ 0 \\ 0 \end{bmatrix}$$

Die kanonische Form für unser erstes Beispiel:

Minimiere $\quad [2, -1, 4, 0] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = c^T \cdot x \quad$ so dass

$$\begin{bmatrix} -1 & -1 & 0 & -1 \\ 0 & 3 & -1 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} -2 \\ 5 \\ -5 \\ 3 \\ 0 \\ 0 \end{bmatrix}$$

$$A \cdot x \quad \geq \quad b$$

LINEARE PROGRAMMIERUNG

Anwendungsbeispiele

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Eine Firma herstellt n verschiedene Produkte aus m verschiedenen Rohstoffen.

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Eine Firma herstellt n verschiedene Produkte
aus m verschiedenen Rohstoffen.

– von Rohstoff i steht eine Menge von b_i zur Verfügung;

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Eine Firma herstellt n verschiedene Produkte
aus m verschiedenen Rohstoffen.

- von Rohstoff i steht eine Menge von b_i zur Verfügung;
- *eine Einheit* von Produkt j braucht a_{ij} von Rohstoff i;

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Eine Firma herstellt n verschiedene Produkte

aus m verschiedenen Rohstoffen.

- von Rohstoff i steht eine Menge von b_i zur Verfügung;
- *eine Einheit* von Produkt j braucht a_{ij} von Rohstoff i;
- und erzeugt den Erlös c_j.

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Eine Firma herstellt n verschiedene Produkte
aus m verschiedenen Rohstoffen.

- von Rohstoff i steht eine Menge von b_i zur Verfügung;
- *eine Einheit* von Produkt j braucht a_{ij} von Rohstoff i;
- und erzeugt den Erlös c_j.

Formulierung als LP:

gesucht wird x_j die zu produzierende Menge des Produkt j;

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Eine Firma herstellt n verschiedene Produkte

aus m verschiedenen Rohstoffen.

- von Rohstoff i steht eine Menge von b_i zur Verfügung;
- *eine Einheit* von Produkt j braucht a_{ij} von Rohstoff i;
- und erzeugt den Erlös c_j.

Formulierung als LP:

gesucht wird x_j die zu produzierende Menge des Produkt j;

Maximiere $\sum_{j=1}^{n} c_j x_j$ so dass $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ für jeden Rohstoff i.

Beispiel 1: Produktionsplanung

(ein ökonomisches Problem)

Eine Firma herstellt n verschiedene Produkte

aus m verschiedenen Rohstoffen.

- von Rohstoff i steht eine Menge von b_i zur Verfügung;
- *eine Einheit* von Produkt j braucht a_{ij} von Rohstoff i;
- und erzeugt den Erlös c_j.

Formulierung als LP:

gesucht wird x_j die zu produzierende Menge des Produkt j;

Maximiere $\sum_{j=1}^{n} c_j x_j$ so dass $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ für jeden Rohstoff i.

Beispiel 2: Vertex Cover

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \dots n\}$ Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$ Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Formulierung als LP:

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \dots n\}$
Knotengewichten w_v, und $V' \subseteq V$ die gesuchte
Knotenüberdeckung.

Formulierung als LP:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \dots, x_n)$ ist

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$
Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Formulierung als LP:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \ldots, x_n)$ ist
$x_v = 1$ wenn $v \in V'$ und $x_v = 0$ sonst.

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$ Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Formulierung als LP:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \ldots, x_n)$ ist $x_v = 1$ wenn $v \in V'$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$ Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Formulierung als LP:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \ldots, x_n)$ ist

$x_v = 1$ wenn $v \in V'$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \quad \forall \{u, v\} \in E$

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \dots n\}$ Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Formulierung als LP:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \dots, x_n)$ ist $x_v = 1$ wenn $v \in V'$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \quad \forall \{u, v\} \in E$
- und $0 \leq x_v \leq 1 \;\; \forall v \in V$

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$ Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Formulierung als LP:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \ldots, x_n)$ ist $x_v = 1$ wenn $v \in V'$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \quad \forall \{u, v\} \in E$
- und $0 \leq x_v \leq 1 \ \ \forall v \in V$

Das LP erlaubt also fraktionale Lösungen...

Beispiel 2: Vertex Cover

Verschiedenste Optimierungsprobleme können als *ganzzahlige* Programme formuliert werden!

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$ Knotengewichten w_v, und $V' \subseteq V$ die gesuchte Knotenüberdeckung.

Formulierung als LP:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \ldots, x_n)$ ist $x_v = 1$ wenn $v \in V'$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \quad \forall \{u, v\} \in E$
- und $0 \leq x_v \leq 1 \;\; \forall v \in V$

Das LP erlaubt also fraktionale Lösungen... Aber: es ist effizient lösbar!

Formulierung als ganzzahliges Programm (*Integer Program, IP*):

Formulierung als ganzzahliges Programm (*Integer Program, IP*):

- Minimiere $\sum_{v=1}^{n} x_v w_v$

Formulierung als ganzzahliges Programm (*Integer Program, IP*):

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \;\; \forall \{u, v\} \in E$

Formulierung als ganzzahliges Programm (*Integer Program, IP*):

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \;\; \forall \{u, v\} \in E$
- $x_v \in \{0, 1\} \;\; \forall v \in V$

Formulierung als ganzzahliges Programm (*Integer Program, IP*):

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \;\; \forall \{u, v\} \in E$
- $x_v \in \{0, 1\} \;\; \forall v \in V$

Das IP ist 'richtig', aber nicht polynomiell lösbar

Formulierung als ganzzahliges Programm (*Integer Program, IP*):

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \;\; \forall \{u, v\} \in E$
- $x_v \in \{0, 1\} \;\; \forall v \in V$

Das IP ist 'richtig', aber nicht polynomiell lösbar

Die LP-Formulierung heißt eine *Relaxierung* der IP-Formulierung.

Formulierung als ganzzahliges Programm (*Integer Program, IP*):

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \;\; \forall \{u, v\} \in E$
- $x_v \in \{0, 1\} \;\; \forall v \in V$

Das IP ist 'richtig', aber nicht polynomiell lösbar

Die LP-Formulierung heißt eine *Relaxierung* der IP-Formulierung.

Durch das Lösen von LP werden wir eine 2-Approximation erhalten...

Beispiel 3: Gewichtetes Matching

Beispiel 3: Gewichtetes Matching

Für $G(V, E)$ mit Kantengewichten w_e,

finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)

mit maximalem Gewicht

Beispiel 3: Gewichtetes Matching

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)
mit maximalem Gewicht

Formulierung als LP (Relaxierung):

Beispiel 3: Gewichtetes Matching

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)
mit maximalem Gewicht

Formulierung als LP (Relaxierung):

Beabsichtigte Bedeutung der Variablen x_e ist

Beispiel 3: Gewichtetes Matching

Für $G(V,E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)
mit maximalem Gewicht

Formulierung als LP (Relaxierung):

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

Beispiel 3: Gewichtetes Matching

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)
mit maximalem Gewicht

Formulierung als LP (Relaxierung):

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

– Maximiere $\sum_{e \in E} x_e w_e$

Beispiel 3: Gewichtetes Matching

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)
mit maximalem Gewicht

Formulierung als LP (Relaxierung):

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

- Maximiere $\sum_{e \in E} x_e w_e$
- so dass
$$\sum_{u,\{u,v\} \in E} x_{\{u,v\}} \leq 1 \qquad \forall v \in E$$

Beispiel 3: Gewichtetes Matching

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)
mit maximalem Gewicht

Formulierung als LP (Relaxierung):

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

- Maximiere $\sum_{e \in E} x_e w_e$
- so dass
$$\sum_{u,\{u,v\} \in E} x_{\{u,v\}} \leq 1 \qquad \forall v \in E$$
- und $0 \leq x_e \leq 1 \qquad \forall e \in E$

Beispiel 3: Gewichtetes Matching

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (ohne gemeinsame Endknoten)
mit maximalem Gewicht

Formulierung als LP (Relaxierung):

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

- Maximiere $\sum_{e \in E} x_e w_e$
- so dass
$$\sum_{u,\{u,v\} \in E} x_{\{u,v\}} \leq 1 \qquad \forall v \in E$$
- und $0 \leq x_e \leq 1 \qquad \forall e \in E$

LINEARE PROGRAMMIERUNG

Geometrische Grundlagen

Geometrie in $\mathbb{R}^2$

Geometrie in $\mathbb{R}^2$

Sei $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, ein Vektor von Variablen und $a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \in \mathbb{R}^2$.

Geometrie in $\mathbb{R}^2$

Sei $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, ein Vektor von Variablen und $a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \in \mathbb{R}^2$.

- $\{x \in \mathbb{R}^2 \mid a^T \cdot x = 0\}$ ist eine zu a orthogonale Gerade die den $\mathbf{0}$ enthält

Geometrie in $\mathbb{R}^2$

Sei $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, ein Vektor von Variablen und $a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \in \mathbb{R}^2$.

- $\{x \in \mathbb{R}^2 \mid a^T \cdot x = 0\}$ ist eine zu a orthogonale Gerade die den $\mathbf{0}$ enthält
- $\{x \in \mathbb{R}^2 \mid a^T \cdot x > 0 \quad (\text{bzw.}\ a^T \cdot x < 0)\}$ ist eine offene Halbebene

Geometrie in $\mathbb{R}^2$

Sei $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, ein Vektor von Variablen und $a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \in \mathbb{R}^2$.

- $\{x \in \mathbb{R}^2 \mid a^T \cdot x = 0\}$ ist eine zu a orthogonale Gerade die den $\mathbf{0}$ enthält

- $\{x \in \mathbb{R}^2 \mid a^T \cdot x > 0 \quad (\text{bzw. } a^T \cdot x < 0)\}$ ist eine offene Halbebene

- $\{x \in \mathbb{R}^2 \mid a^T \cdot x \geq 0 \quad (\text{bzw. } a^T \cdot x \leq 0)\}$ ist eine abgeschlossene Halbebene

Geometrie in $\mathbb{R}^2$

Sei $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, ein Vektor von Variablen und $a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \in \mathbb{R}^2$.

- $\{x \in \mathbb{R}^2 \mid a^T \cdot x = 0\}$ ist eine zu a orthogonale Gerade die den $\mathbf{0}$ enthält
- $\{x \in \mathbb{R}^2 \mid a^T \cdot x > 0 \quad (\text{bzw. } a^T \cdot x < 0)\}$ ist eine offene Halbebene
- $\{x \in \mathbb{R}^2 \mid a^T \cdot x \geq 0 \quad (\text{bzw. } a^T \cdot x \leq 0)\}$ ist eine abgeschlossene Halbebene
- $\{x \in \mathbb{R}^2 \mid a^T \cdot x = b\}$ ist eine *affine* Gerade, parallel zur Gerade $a^T \cdot x = 0$

Zusammenfassung, Verallgemeinerung

Zusammenfassung, Verallgemeinerung

(an $\mathbb{R}^3$ denken...)

Zusammenfassung, Verallgemeinerung

(an $\mathbb{R}^3$ denken...)

- $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$, usw. sind Vektoren, bzw. ihre Endpunkte in $\mathbb{R}^n$

Zusammenfassung, Verallgemeinerung

(an $\mathbb{R}^3$ denken...)

- $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$, usw. sind Vektoren, bzw.

  ihre Endpunkte in $\mathbb{R}^n$

- $a^T \cdot x = \sum_{i=1}^n a_i x_i = |a||x| \cos\gamma$, wobei γ der Winkel zwischen

  a und x ist

Zusammenfassung, Verallgemeinerung

(an $\mathbb{R}^3$ denken...)

- $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$, usw. sind Vektoren, bzw.

  ihre Endpunkte in $\mathbb{R}^n$

- $a^T \cdot x = \sum_{i=1}^n a_i x_i = |a||x| \cos \gamma$, wobei γ der Winkel zwischen

  a und x ist

- $a^T \cdot x = 0 \quad \Leftrightarrow \quad a$ und x sind *orthogonal*

– die zu a orthogonale Vektoren $\{x \in \mathbb{R}^n \,|\, a^T \cdot x = 0\}$

– die zu a orthogonale Vektoren $\{x \in \mathbb{R}^n \mid a^T \cdot x = 0\}$ formen eine Hyperebene (in $\mathbb{R}^3$ ist das eine Ebene)

- die zu a orthogonale Vektoren $\{x \in \mathbb{R}^n \mid a^T \cdot x = 0\}$ formen eine Hyperebene (in $\mathbb{R}^3$ ist das eine Ebene)

- Seien $a \in \mathbb{R}^n$ und $b \in \mathbb{R}$

- die zu a orthogonale Vektoren $\{x \in \mathbb{R}^n \mid a^T \cdot x = 0\}$ formen eine Hyperebene (in $\mathbb{R}^3$ ist das eine Ebene)

- Seien $a \in \mathbb{R}^n$ und $b \in \mathbb{R}$

  $\{x \in \mathbb{R}^n \mid a^T \cdot x = b\}$ ist eine (affine) Hyperebene

- die zu a orthogonale Vektoren $\{x \in \mathbb{R}^n \mid a^T \cdot x = 0\}$ formen eine Hyperebene (in $\mathbb{R}^3$ ist das eine Ebene)

- Seien $a \in \mathbb{R}^n$ und $b \in \mathbb{R}$

  $\{x \in \mathbb{R}^n \mid a^T \cdot x = b\}$ ist eine (affine) Hyperebene

  $\{x \in \mathbb{R}^n \mid a^T \cdot x \geq b\}$ und $\{x \in \mathbb{R}^n \mid a^T \cdot x \leq b\}$

- die zu a orthogonale Vektoren $\{x \in \mathbb{R}^n \,|\, a^T \cdot x = 0\}$ formen eine Hyperebene (in $\mathbb{R}^3$ ist das eine Ebene)

- Seien $a \in \mathbb{R}^n$ und $b \in \mathbb{R}$

  $\{x \in \mathbb{R}^n \,|\, a^T \cdot x = b\}$ ist eine (affine) Hyperebene

  $\{x \in \mathbb{R}^n \,|\, a^T \cdot x \geq b\}$ und $\{x \in \mathbb{R}^n \,|\, a^T \cdot x \leq b\}$

  sind (abgeschlossene) Halbräume

Die Lösungsmenge (Lösungsraum) eines linearen Programms

Die Lösungsmenge (Lösungsraum) eines linearen Programms

Definition 1:

Sei $A \cdot x \geq 0$ ein lineares Programm mit Zielfunktion $c^T \cdot x$.

Ein Vektor x mit $A \cdot x \geq b$ heißt eine *Lösung*.

Die Lösungsmenge (Lösungsraum) eines linearen Programms

Definition 1:

Sei $A \cdot x \geq 0$ ein lineares Programm mit Zielfunktion $c^T \cdot x$.

Ein Vektor x mit $A \cdot x \geq b$ heißt eine *Lösung*.

Das LP ist *lösbar* wenn lösungen existieren, und *unlösbar* sonst.

Die Lösungsmenge (Lösungsraum) eines linearen Programms

Definition 1:

Sei $A \cdot x \geq 0$ ein lineares Programm mit Zielfunktion $c^T \cdot x$.

Ein Vektor x mit $A \cdot x \geq b$ heißt eine *Lösung*.

Das LP ist *lösbar* wenn lösungen existieren, und *unlösbar* sonst.

Die *Lösungsmenge* ist $\{x \in \mathbb{R}^n \,|\, A \cdot x \geq b\}$.

Die Lösungsmenge (Lösungsraum) eines linearen Programms

Definition 1:

Sei $A \cdot x \geq 0$ ein lineares Programm mit Zielfunktion $c^T \cdot x$.

Ein Vektor x mit $A \cdot x \geq b$ heißt eine *Lösung*.

Das LP ist *lösbar* wenn lösungen existieren, und *unlösbar* sonst.

Die *Lösungsmenge* ist $\{x \in \mathbb{R}^n \mid A \cdot x \geq b\}$.

Definition 2: Einen Durchschnitt von Halbräumen nennt man *Polyeder*

Die Lösungsmenge (Lösungsraum) eines linearen Programms

Definition 1:

Sei $A \cdot x \geq 0$ ein lineares Programm mit Zielfunktion $c^T \cdot x$.

Ein Vektor x mit $A \cdot x \geq b$ heißt eine *Lösung*.

Das LP ist *lösbar* wenn lösungen existieren, und *unlösbar* sonst.

Die *Lösungsmenge* ist $\{x \in \mathbb{R}^n \,|\, A \cdot x \geq b\}$.

Definition 2: Einen Durchschnitt von Halbräumen nennt man *Polyeder*

(wenn er beschränkt ist, nennt man ihn auch *Polytop*).

Die Lösungsmenge (Lösungsraum) eines linearen Programms

Definition 1:

Sei $A \cdot x \geq 0$ ein lineares Programm mit Zielfunktion $c^T \cdot x$.

Ein Vektor x mit $A \cdot x \geq b$ heißt eine *Lösung*.

Das LP ist *lösbar* wenn lösungen existieren, und *unlösbar* sonst.

Die *Lösungsmenge* ist $\{x \in \mathbb{R}^n \mid A \cdot x \geq b\}$.

Definition 2: Einen Durchschnitt von Halbräumen nennt man *Polyeder*

(wenn er beschränkt ist, nennt man ihn auch *Polytop*).

Beobachtung: Die Lösungsmenge ist ein Polyeder, weil sie der Durchschnitt der Halbräume $\{x \mid a_i^T \cdot x \geq b_i\}$ für alle $i = 1, 2, \ldots, m$ ist.

Beispiel

Beispiel

Minimiere $\quad -x_1 - x_2$

Beispiel

Minimiere $\quad -x_1 - x_2$

so dass

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

LINEARE PROGRAMMIERUNG

Die Ecken der Lösungsmenge

Ecken: Definition 1

Ecken: Definition 1

Sei $P \subseteq \mathbb{R}^n$ ein Lösungspolyeder.

<u>Definition 1:</u> $x^* \in P$ ist eine *Ecke*, wenn in x^* n (linear unabhängige) Nebenbedingungen *exakt* erfüllt werden.

Ecken: Definition 1

Sei $P \subseteq \mathbb{R}^n$ ein Lösungspolyeder.

<u>Definition 1:</u> $x^* \in P$ ist eine *Ecke*, wenn in x^* n (linear unabhängige) Nebenbedingungen *exakt* erfüllt werden.

(d.h. die entsprechenden a_i^T Vektoren in den Bedingungen sind linear unabhängig in $\mathbb{R}^n$)

Ecken: Definition 1

Sei $P \subseteq \mathbb{R}^n$ ein Lösungspolyeder.

Definition 1: $x^* \in P$ ist eine *Ecke*, wenn in x^* n (linear unabhängige) Nebenbedingungen *exakt* erfüllt werden.

(d.h. die entsprechenden a_i^T Vektoren in den Bedingungen sind linear unabhängig in $\mathbb{R}^n$)

Beachte: Wir brauchen also n unabhängige Bedingungen damit P *überhaupt* Ecken hat!

Ecken: Definition 1

Sei $P \subseteq \mathbb{R}^n$ ein Lösungspolyeder.

Definition 1: $x^* \in P$ ist eine *Ecke*, wenn in x^* n (linear unabhängige) Nebenbedingungen *exakt* erfüllt werden.

(d.h. die entsprechenden a_i^T Vektoren in den Bedingungen sind linear unabhängig in $\mathbb{R}^n$)

Beachte: Wir brauchen also n unabhängige Bedingungen damit P *überhaupt* Ecken hat!

(Bemerkung:)

(Bemerkung:)

Es gibt $< n$ unabhängige Bedingungen

(Bemerkung:)

Es gibt $< n$ unabhängige Bedingungen

$$\Updownarrow$$

der Lösungspolyeder enthält eine Gerade

(Bemerkung:)

Es gibt $< n$ unabhängige Bedingungen

$$\Updownarrow$$

der Lösungspolyeder enthält eine Gerade

$$\Downarrow$$

es gibt keine Ecke, und das folgende Theorem *gilt nicht*

Ecken: Definition 2

Ecken: Definition 2

Definition 2: $x^* \in P$ ist eine *Ecke*, wenn es *keinen* Vektor $y \neq 0$ gibt mit $x^* + y \in P$ und $x^* - y \in P$.

Ecken: Definition 2

Definition 2: $x^* \in P$ ist eine *Ecke*, wenn es *keinen* Vektor $y \neq 0$ gibt mit $x^* + y \in P$ und $x^* - y \in P$.

Theorem: Wenn für ein LP mit n Variablen und n linear unabhängigen Nebenbedingungen ein endliches Optimum existiert, dann gibt es eine optimale Ecke des Lösungspolyeders.

Definition: Zwei Ecken $x^*, x^{**} \in P$ sind *benachbart*, oder *adjazent*, wenn es $n-1$ Bedingungen gibt, die x^* und x^{**} beide exakt erfüllen.

Definition: Zwei Ecken $x^*, x^{**} \in P$ sind *benachbart*, oder *adjazent*, wenn es $n-1$ Bedingungen gibt, die x^* und x^{**} beide exakt erfüllen.

Definition: Eine Ecke $x^* \in P$ ist *entartet*, wenn in x^* mehr als n Nebenbedingungen exakt erfüllt werden.

Definition: Zwei Ecken $x^*, x^{**} \in P$ sind *benachbart*, oder *adjazent*, wenn es $n-1$ Bedingungen gibt, die x^* und x^{**} beide exakt erfüllen.

Definition: Eine Ecke $x^* \in P$ ist *entartet*, wenn in x^* mehr als n Nebenbedingungen exakt erfüllt werden.

(natürlich nicht unabhängige)

LINEARE PROGRAMMIERUNG

Lineare Programme in Standardform

Definition:

Definition:

Lineares Programm in *Standardform:*

Definition:

Lineares Programm in *Standardform:*

Minimiere $\quad c^T \cdot x \quad$ so dass $\quad A \cdot x = b \quad$ und $\quad x \geq 0$

Definition:

Lineares Programm in *Standardform:*

Minimiere $\quad c^T \cdot x \quad$ so dass $\quad A \cdot x = b \quad$ und $\quad x \geq 0$

(hier $x^T = [x_1, x_2, \ldots, x_n], \quad c \in \mathbb{Q}^n, \quad A \in \mathbb{Q}^{m \times n}, \quad b \in \mathbb{Q}^m$)

Definition:

Lineares Programm in *Standardform:*

Minimiere $c^T \cdot x$ so dass $A \cdot x = b$ und $x \geq 0$

(hier $x^T = [x_1, x_2, \ldots, x_n]$, $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$)

Beobachtung: Jedes lineare Programm in kanonischer Form, hat ein äquivalentes lineares Programm in Standardform.

Definition:

Lineares Programm in *Standardform:*

Minimiere $c^T \cdot x$ so dass $A \cdot x = b$ und $x \geq 0$

(hier $x^T = [x_1, x_2, \ldots, x_n]$, $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$)

Beobachtung: Jedes lineare Programm in kanonischer Form, hat ein äquivalentes lineares Programm in Standardform.

$\Rightarrow$ die Standardform ist auch allgemein genug!

Beispiel

Beispiel

Minimiere $\qquad 2x_1 + 4x_2$

so dass

$$\begin{aligned} x_1 + x_2 &\geq 3 \\ 3x_1 + 2x_2 &= 14 \\ x_1 &\geq 0 \end{aligned}$$

Beispiel

Minimiere $\quad 2x_1 + 4x_2$

so dass

$$\begin{aligned} x_1 + x_2 &\geq 3 \\ 3x_1 + 2x_2 &= 14 \\ x_1 &\geq 0 \end{aligned}$$

Die äquivalente Standardform:

Beispiel

Minimiere $\quad 2x_1 + 4x_2$

so dass

$$\begin{aligned} x_1 + x_2 &\geq 3 \\ 3x_1 + 2x_2 &= 14 \\ x_1 &\geq 0 \end{aligned}$$

Die äquivalente Standardform:

Minimiere $\quad 2x_1 + 4x_2^+ - 4x_2^-$

Beispiel

Minimiere $\quad 2x_1 + 4x_2$

so dass

$$\begin{aligned} x_1 + x_2 &\geq 3 \\ 3x_1 + 2x_2 &= 14 \\ x_1 &\geq 0 \end{aligned}$$

Die äquivalente Standardform:

Minimiere $\quad 2x_1 + 4x_2^+ - 4x_2^-$

so dass

$$\begin{aligned} x_1 + x_2^+ - x_2^- - s_1 = 3 &= 3 \\ 3x_1 + 2x_2^+ - 2x_2^- &= 14 \\ x_1 &\geq 0 \\ s_1 &\geq 0 \\ x_2^+ &\geq 0 \\ x_2^- &\geq 0 \end{aligned}$$

Geometrische Darstellung der Standardform

Geometrische Darstellung der Standardform

Sei $\quad A \cdot x = b, \quad x \geq 0 \quad$ ein LP in Standardform, mit n Variablen und m unabhängigen Gleichungen $A \in \mathbb{Q}^{m \times n}$.

Der Lösungspolyeder (falls nichtleer) ist $n - m$ dimensional.

Geometrische Darstellung der Standardform

Sei $A \cdot x = b, \quad x \geq 0$ ein LP in Standardform, mit
n Variablen und m unabhängigen Gleichungen $A \in \mathbb{Q}^{m \times n}$.
Der Lösungspolyeder (falls nichtleer) ist $n - m$ dimensional.

[Jede Gleichung in $A \cdot x = b$ nimmt einen Freiheitsgrad (eine Dimension) weg.]

Geometrische Darstellung der Standardform

Sei $A \cdot x = b, \quad x \geq 0$ ein LP in Standardform, mit n Variablen und m unabhängigen Gleichungen $A \in \mathbb{Q}^{m \times n}$.
Der Lösungspolyeder (falls nichtleer) ist $n - m$ dimensional.

[Jede Gleichung in $A \cdot x = b$ nimmt einen Freiheitsgrad (eine Dimension) weg.]

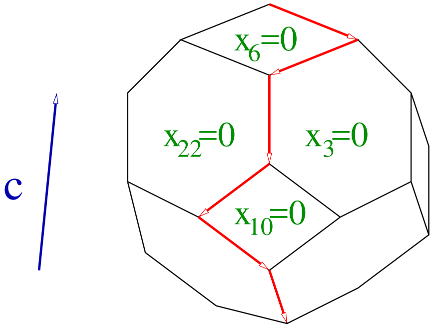
$\rightarrow$ Wir werden Beispiele mit $n - m = 2$ und $n - m = 3$ geometrisch darstellen.

LINEARE PROGRAMMIERUNG

Der Simplex-Algorithmus

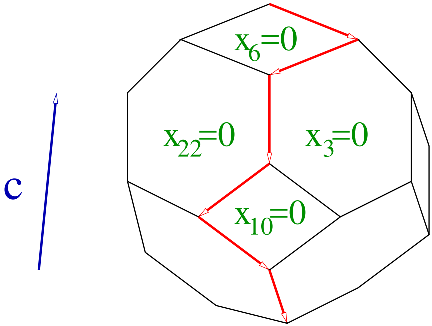
Die Grobstruktur des Simplex-Algorithmus

Die Grobstruktur des Simplex-Algorithmus



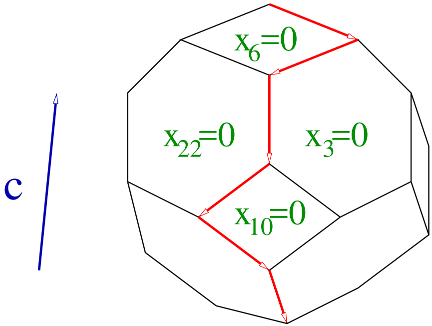
1. Wähle eine beliebige Ecke x^* des Lösungspolyeders

Die Grobstruktur des Simplex-Algorithmus



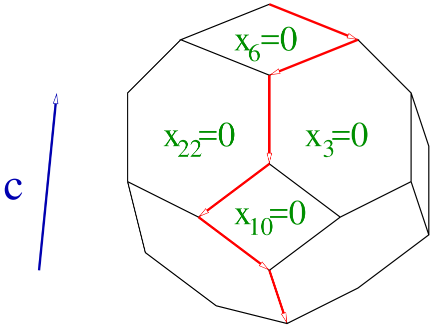
1. Wähle eine beliebige Ecke x^* des Lösungspolyeders
2. Bestimme eine benachbarte Ecke x^{**} mit niedrigerem Zielwert

Die Grobstruktur des Simplex-Algorithmus



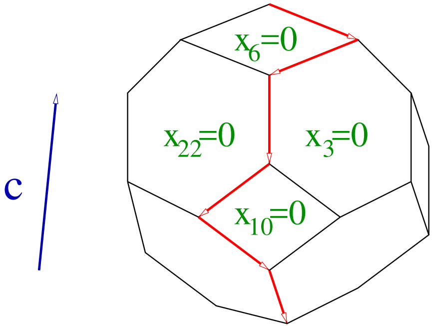
1. Wähle eine beliebige Ecke x^* des Lösungspolyeders
2. Bestimme eine benachbarte Ecke x^{**} mit niedrigerem Zielwert
3. Falls keine existiert, gib x^* als optimale Lösung aus;

Die Grobstruktur des Simplex-Algorithmus



1. Wähle eine beliebige Ecke x^* des Lösungspolyeders
2. Bestimme eine benachbarte Ecke x^{**} mit niedrigerem Zielwert
3. Falls keine existiert, gib x^* als optimale Lösung aus; sonst setze $x^* := x^{**}$ und GOTO 2.

Die Grobstruktur des Simplex-Algorithmus



1. Wähle eine beliebige Ecke x^* des Lösungspolyeders
2. Bestimme eine benachbarte Ecke x^{**} mit niedrigerem Zielwert
3. Falls keine existiert, gib x^* als optimale Lösung aus; sonst setze $x^* := x^{**}$ und GOTO 2.

Simplex wendet lokale Suche an.

Die gefundene lokale optimale Lösung ist sogar global optimal!

[Einige algebraische Details]

Eingabe: ein LP in Standardform

1. von x^* laufen wir entlang einer 'Kante' über Lösungen $x^* + \lambda d$

   (hier $d_j = 1$ für eine $j \notin B$, und $d_B = -A_B^{-1} \cdot a^j$)

2. Welche $j \notin B$ (Richtung) nehmen wir?

   der Zielwert wächst um $\lambda c^T \cdot d = \lambda(c_B^T \cdot d_B + c_j) =: \lambda \tilde{c}_j$

   Theorem: Wenn $\tilde{c}_j \geq 0$ für jede $j \notin B$, dann wächst der Zielwert

   in jeder Richtung $\Rightarrow$ in jeder Lösung $x \in P$, $\Rightarrow$ x^* war minimal.

[Weiter...]

Sonst wähle die *kleinste* $j \notin B$ mit $\tilde{c}_j < 0$

(Diese *smallest subscript rule* schließt *cycling* aus!)

Sei diese j fixiert:

- falls $d_B \geq 0$ dann $x^* + \lambda d \geq 0$
  wir können unendlich weiterlaufen und $\min c^T \cdot x = -\infty$

- sonst existiert $\lambda_{\max}$ maximal so dass $x^* + \lambda d \geq 0$

  setze $x^{**} = x^* + \lambda_{\max} d$

  bei $\lambda_{\max}$ gibt es eine neue k mit $x_k^{**} = 0$
  (k wird gegen j ausgetauscht in der Basis B)

Laufzeit

Laufzeit

– In jeder Iteration (neue Ecke) werden Matrizen invertiert und multipliziert in Laufzeit Poly(n, m)

Laufzeit

- In jeder Iteration (neue Ecke) werden Matrizen invertiert und multipliziert in Laufzeit Poly(n, m)
- Simplex kann aber im Worst-Case $\mathcal{O}(2^n)$ Ecken durchlaufen! (zB. die Ecken eines verzerrten Würfels)

Laufzeit

- In jeder Iteration (neue Ecke) werden Matrizen invertiert und multipliziert in Laufzeit Poly(n, m)
- Simplex kann aber im Worst-Case $\mathcal{O}(2^n)$ Ecken durchlaufen! (zB. die Ecken eines verzerrten Würfels)
- Exponentielle Laufzeit *im Worst-Case*: $\mathcal{O}(\mathrm{Poly}(n) \cdot 2^n)$

Laufzeit

- In jeder Iteration (neue Ecke) werden Matrizen invertiert und multipliziert in Laufzeit $\mathrm{Poly}(n, m)$
- Simplex kann aber im Worst-Case $\mathcal{O}(2^n)$ Ecken durchlaufen! (zB. die Ecken eines verzerrten Würfels)
- Exponentielle Laufzeit *im Worst-Case*: $\mathcal{O}(\mathrm{Poly}(n) \cdot 2^n)$
- Jedoch wird Simplex in der Praxis erfolgreich angewandt.

Geschichte

Geschichte

– 1947 (Dantzig): Simplex Algorithmus

Geschichte

– 1947 (Dantzig): Simplex Algorithmus

– 1979 (Khachiyan): Ellipsoid-Methode

(erster polynomieller Algorithmus; nicht praktisch)

Geschichte

– 1947 (Dantzig): Simplex Algorithmus

– 1979 (Khachiyan): Ellipsoid-Methode

(erster polynomieller Algorithmus; nicht praktisch)

– 1984 (Karmarkar): Interior-Point Methode

Geschichte

- 1947 (Dantzig): Simplex Algorithmus
- 1979 (Khachiyan): Ellipsoid-Methode

  (erster polynomieller Algorithmus; nicht praktisch)
- 1984 (Karmarkar): Interior-Point Methode

  (für große LP-s; approximative Lösungen)

LINEARE PROGRAMMIERUNG

Lineare Programmierung und Approximation

"In general it is not known what properties of integer programs allow approximation with an arbitrary ε, a fixed ε, or no approximation at all. This is an active research area."

Deterministisches Runden

Beispiel 1: min-VERTEX COVER

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \dots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte
Knotenüberdeckung.

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte
Knotenüberdeckung.

LP-Relaxierung:

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \dots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte
Knotenüberdeckung.

LP-Relaxierung:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \dots, x_n)$ ist

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \dots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte Knotenüberdeckung.

LP-Relaxierung:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \dots, x_n)$ ist

$x_v = 1$ wenn $v \in C$ und $x_v = 0$ sonst.

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \dots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte Knotenüberdeckung.

LP-Relaxierung:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \dots, x_n)$ ist

$x_v = 1$ wenn $v \in C$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte
Knotenüberdeckung.

LP-Relaxierung:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \ldots, x_n)$ ist

$x_v = 1$ wenn $v \in C$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \quad \forall \{u, v\} \in E$

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \dots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte
Knotenüberdeckung.

LP-Relaxierung:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \dots, x_n)$ ist

$x_v = 1$ wenn $v \in C$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \quad \forall \{u, v\} \in E$
- und $0 \leq x_v \leq 1 \ \ \forall v \in V$

Beispiel 1: min-VERTEX COVER

Sei G ein Graph mit $V = \{1, 2, \ldots n\}$
Knotengewichten w_v, und $C \subseteq V$ die gesuchte Knotenüberdeckung.

LP-Relaxierung:

Beabsichtigte Interpretation der Variablen $x = (x_1, x_2, \ldots, x_n)$ ist

$x_v = 1$ wenn $v \in C$ und $x_v = 0$ sonst.

- Minimiere $\sum_{v=1}^{n} x_v w_v$
- so dass $x_v + x_u \geq 1 \quad \forall \{u, v\} \in E$
- und $0 \leq x_v \leq 1 \;\; \forall v \in V$

Ein Approximationsalgorithmus

Ein Approximationsalgorithmus

deterministisches Runden

– löse die LP Relaxierung;
  sei $x^* = (x_1^*, \ldots, x_n^*)$ die optimale fraktionale Lösung

Ein Approximationsalgorithmus

deterministisches Runden

- löse die LP Relaxierung;
  sei $x^* = (x_1^*, \ldots, x_n^*)$ die optimale fraktionale Lösung

- setze $v \in C \quad \Leftrightarrow \quad x_v^* \geq 1/2$

Ein Approximationsalgorithmus

deterministisches Runden

- löse die LP Relaxierung;
  sei $x^* = (x_1^*, \ldots, x_n^*)$ die optimale fraktionale Lösung
- setze $v \in C \quad \Leftrightarrow \quad x_v^* \geq 1/2$

Dann ist C eine 2-approximative Knotenüberdeckung!

Ein Approximationsalgorithmus

deterministisches Runden

- löse die LP Relaxierung;
  sei $x^* = (x_1^*, \ldots, x_n^*)$ die optimale fraktionale Lösung

- setze $v \in C \quad \Leftrightarrow \quad x_v^* \geq 1/2$

Dann ist C eine 2-approximative Knotenüberdeckung!

(Übrigens: jede Ecke hat Koordinaten in $\{0, 1/2, 1\}$)

Beispiel 2: max-INDEPENDENT SET

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Ausgabe: unabhängige Knotenmenge $I \subset V$ maximaler Grösse

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Ausgabe: unabhängige Knotenmenge $I \subset V$ maximaler Grösse
$(u, v \in I \Rightarrow \{u, v\} \notin E)$

LP-Relaxierung:

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Ausgabe: unabhängige Knotenmenge $I \subset V$ maximaler Grösse
$(u, v \in I \Rightarrow \{u, v\} \notin E)$

LP-Relaxierung:

– Maximiere $\sum_{v=1}^{n} x_v$

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Ausgabe: unabhängige Knotenmenge $I \subset V$ maximaler Grösse
$(u, v \in I \Rightarrow \{u, v\} \notin E)$

LP-Relaxierung:

- Maximiere $\sum_{v=1}^{n} x_v$

- so dass $x_v + x_u \leq 1 \quad \forall \{u, v\} \in E$

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Ausgabe: unabhängige Knotenmenge $I \subset V$ maximaler Grösse
$(u, v \in I \Rightarrow \{u, v\} \notin E)$

LP-Relaxierung:

- Maximiere $\sum_{v=1}^{n} x_v$
- so dass $x_v + x_u \leq 1 \quad \forall \{u, v\} \in E$
- und $0 \leq x_v \leq 1 \;\; \forall v \in V$

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Ausgabe: unabhängige Knotenmenge $I \subset V$ maximaler Grösse
$(u, v \in I \Rightarrow \{u, v\} \notin E)$

LP-Relaxierung:

- Maximiere $\sum_{v=1}^{n} x_v$
- so dass $x_v + x_u \leq 1 \quad \forall \{u, v\} \in E$
- und $0 \leq x_v \leq 1 \;\; \forall v \in V$

Beobachtung: Es gibt fraktionale Lösungen, die um den Faktor $\Omega(n)$ besser sind als $\max |I|$

Beispiel 2: max-INDEPENDENT SET

Eingabe: $G(V, E)$

Ausgabe: unabhängige Knotenmenge $I \subset V$ maximaler Grösse
$(u, v \in I \Rightarrow \{u, v\} \notin E)$

LP-Relaxierung:

– Maximiere $\sum_{v=1}^{n} x_v$

– so dass $x_v + x_u \leq 1 \quad \forall \{u, v\} \in E$

– und $0 \leq x_v \leq 1 \;\; \forall v \in V$

Beobachtung: Es gibt fraktionale Lösungen, die um den Faktor $\Omega(n)$ besser sind als $\max |I|$

(zB. für den vollständigen Graphen die Lösung $(1/2, 1/2, \ldots, 1/2)$)

Randomisiertes Runden

Beispiel 3: Ein ROUTING Problem

Beispiel 3: Ein ROUTING Problem

Eingabe: ein gerichteter graph $\vec{G}(V, E)$ mit ausgezeichneten Knoten $s_1, s_2, \ldots, s_r$ $\quad$ (*Quellen*)

Beispiel 3: Ein ROUTING Problem

Eingabe: ein gerichteter graph $\vec{G}(V, E)$ mit ausgezeichneten Knoten

$s_1, s_2, \ldots, s_r$ (*Quellen*)

$t_1, t_2, \ldots, t_r$ (*Senken*)

Beispiel 3: Ein ROUTING Problem

Eingabe: ein gerichteter graph $\vec{G}(V, E)$ mit ausgezeichneten Knoten

$s_1, s_2, \ldots, s_r$ $\quad$ (*Quellen*)

$t_1, t_2, \ldots, t_r$ $\quad$ (*Senken*)

Ausgabe: bestimme Wege $P_1, P_2, \ldots P_r$, so dass

Beispiel 3: Ein ROUTING Problem

Eingabe: ein gerichteter graph $\vec{G}(V,E)$ mit ausgezeichneten Knoten
$s_1, s_2, \ldots, s_r$ $\quad$ (*Quellen*)
$t_1, t_2, \ldots, t_r$ $\quad$ (*Senken*)

Ausgabe: bestimme Wege $P_1, P_2, \ldots P_r$, so dass
P_i in s_i beginnt und in t_i endet,
und die maximale Belastung einer Kante minimiert wird.

Beispiel 3: Ein ROUTING Problem

Eingabe: ein gerichteter graph $\vec{G}(V, E)$ mit ausgezeichneten Knoten
$s_1, s_2, \ldots, s_r$ (*Quellen*)
$t_1, t_2, \ldots, t_r$ (*Senken*)

Ausgabe: bestimme Wege $P_1, P_2, \ldots P_r$, so dass
P_i in s_i beginnt und in t_i endet,
und die maximale Belastung einer Kante minimiert wird.

(Belastung(e)= Anzahl der P_i die über e laufen.)

IP-Formulierung

IP-Formulierung

Variablen: $x_i(e)$ für $1 \leq i \leq r$ und $e \in E$

IP-Formulierung

Variablen: $x_i(e)$ für $1 \leq i \leq r$ und $e \in E$

Interpretation: $x_i(e) = 1$ wenn P_i über e läuft, und 0 sonst

IP-Formulierung

Variablen: $x_i(e)$ für $1 \leq i \leq r$ und $e \in E$

Interpretation: $x_i(e) = 1$ wenn P_i über e läuft, und 0 sonst

<u>Bedingungen B_i für Pfad i :</u>

IP-Formulierung

Variablen: $x_i(e)$ für $1 \leq i \leq r$ und $e \in E$

Interpretation: $x_i(e) = 1$ wenn P_i über e läuft, und 0 sonst

Bedingungen B_i für Pfad i :

$$\sum_{e \text{ endet in } w} x_i(e) - \sum_{e \text{ startet in } w} x_i(e) = 0 \quad \forall w \in V$$

IP-Formulierung

Variablen: $x_i(e)$ für $1 \leq i \leq r$ und $e \in E$

Interpretation: $x_i(e) = 1$ wenn P_i über e läuft, und 0 sonst

Bedingungen B_i für Pfad i :

$$\begin{aligned}
\sum_{e \text{ endet in } w} x_i(e) - \sum_{e \text{ startet in } w} x_i(e) &= 0 \quad \forall w \in V \\
\sum_{e \text{ endet in } s_i} x_i(e) - \sum_{e \text{ startet in } s_i} x_i(e) &= -1
\end{aligned}$$

IP-Formulierung

Variablen: $x_i(e)$ für $1 \le i \le r$ und $e \in E$

Interpretation: $x_i(e) = 1$ wenn P_i über e läuft, und 0 sonst

Bedingungen B_i für Pfad i :

$$\begin{aligned}
\sum_{e \,\text{endet in}\, w} x_i(e) - \sum_{e \,\text{startet in}\, w} x_i(e) &= 0 \quad \forall w \in V \\
\sum_{e \,\text{endet in}\, s_i} x_i(e) - \sum_{e \,\text{startet in}\, s_i} x_i(e) &= -1 \\
\sum_{e \,\text{endet in}\, t_i} x_i(e) - \sum_{e \,\text{startet in}\, t_i} x_i(e) &= 1
\end{aligned}$$

IP-Formulierung

Variablen: $x_i(e)$ für $1 \leq i \leq r$ und $e \in E$

Interpretation: $x_i(e) = 1$ wenn P_i über e läuft, und 0 sonst

Bedingungen B_i für Pfad i :

$$
\begin{aligned}
\sum_{e \text{ endet in} w} x_i(e) - \sum_{e \text{ startet in} w} x_i(e) &= 0 \quad \forall w \in V \\
\sum_{e \text{ endet in} s_i} x_i(e) - \sum_{e \text{ startet in} s_i} x_i(e) &= -1 \\
\sum_{e \text{ endet in} t_i} x_i(e) - \sum_{e \text{ startet in} t_i} x_i(e) &= 1
\end{aligned}
$$

Minimiere W so dass $\sum_{i=1}^{r} x_i(e) \leq W \quad \forall e \in E$

und die Bedingungen B_i gelten für jedes i

LP-Relaxierung:

LP-Relaxierung:

Fordere $0 \leq x_i(e) \leq 1$ (statt $x_i \in \{0, 1\}$)

LP-Relaxierung:

Fordere $0 \leq x_i(e) \leq 1$ (statt $x_i \in \{0,1\}$)

Definition: Ein Fluss von s nach t ist eine Funktion $f : E \to \mathbb{R}$ so dass die *Flusserhaltung* (Bedingungen B_i) gilt;

LP-Relaxierung:

Fordere $0 \leq x_i(e) \leq 1$ (statt $x_i \in \{0,1\}$)

<u>Definition:</u> Ein Fluss von s nach t ist eine Funktion $f : E \to \mathbb{R}$ so dass die *Flusserhaltung* (Bedingungen B_i) gilt;

(und generell, auch vorgegebene Kapazitätsschranken $f(e) \leq c(e)$ eingehalten werden).

LP-Relaxierung:

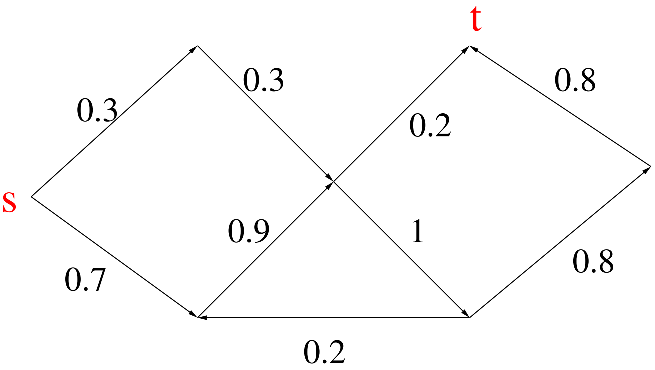
Fordere $0 \leq x_i(e) \leq 1$ (statt $x_i \in \{0,1\}$)

<u>Definition:</u> Ein Fluss von s nach t ist eine Funktion $f : E \to \mathbb{R}$ so dass die *Flusserhaltung* (Bedingungen B_i) gilt;

(und generell, auch vorgegebene Kapazitätsschranken $f(e) \leq c(e)$ eingehalten werden).

Eine optimale *fraktionale* Lösung bestimmt jeweils einen Fluss x_i mit Wert 1 von s_i nach t_i. Mit (Gesamt-)Kapazität W für jede Kante.

Pfad-Zerlegung (*path-decomposition*)

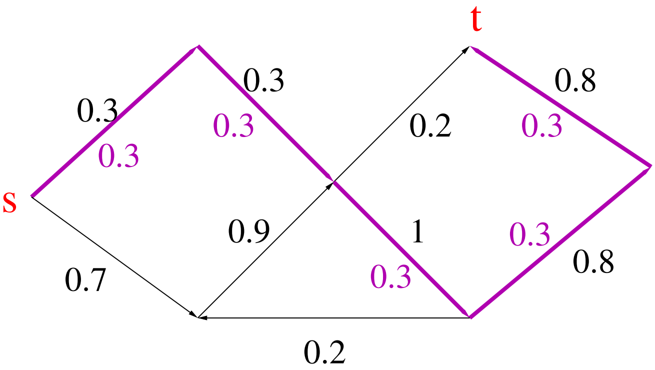


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

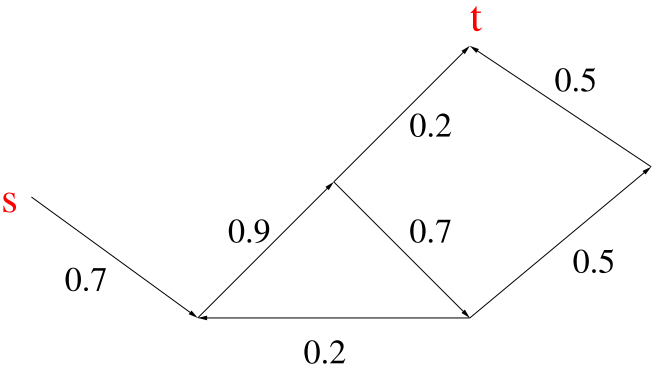


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

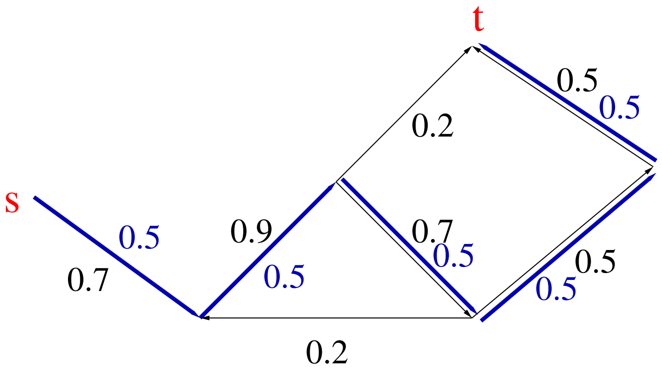


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

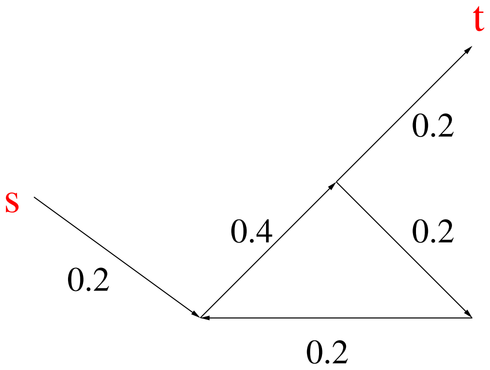


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

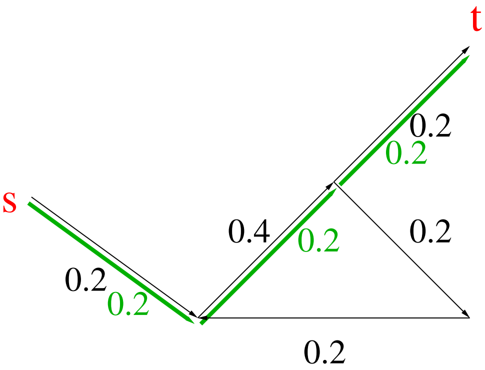


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

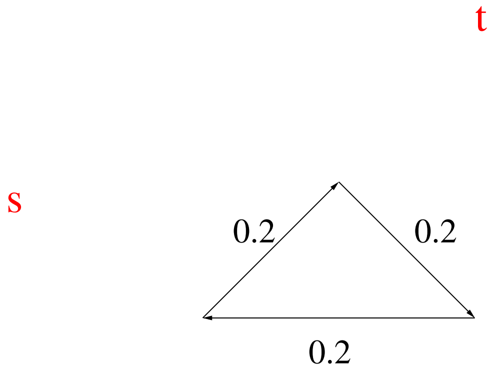


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

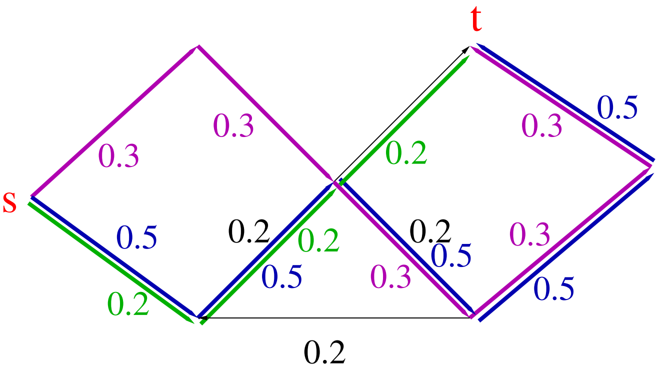


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

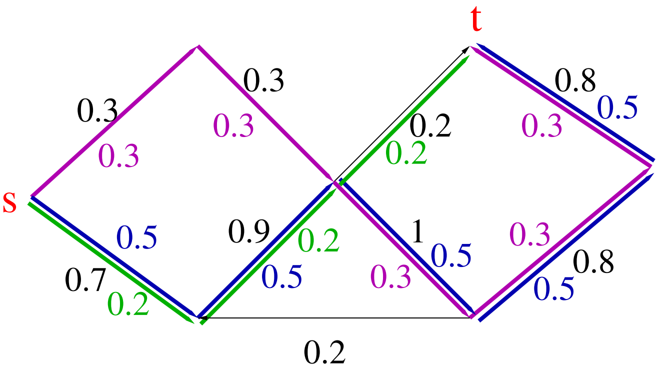


sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)



sei x^* eine optimale fraktionale Lösung

Sei i fixiert, und

G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Es gibt maximal $|E|$ Iterationen

Pfad-Zerlegung (*path-decomposition*)

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Sei x_i^* fixiert.

$k := 1$
REPEAT

- sei G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Sei x_i^* fixiert.

$k := 1$
REPEAT

- sei G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;
- nimm einen Pfad $P_{i,k}$ von s_i nach t_i

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Sei x_i^* fixiert.

$k := 1$
REPEAT

- sei G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;
- nimm einen Pfad $P_{i,k}$ von s_i nach t_i
- sei $\alpha_{i,k} = \min_{e \in P_{i,k}} x_i^*(e)$

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Sei x_i^* fixiert.

$k := 1$
REPEAT

- sei G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;
- nimm einen Pfad $P_{i,k}$ von s_i nach t_i
- sei $\alpha_{i,k} = \min_{e \in P_{i,k}} x_i^*(e)$
- reduziere $x_i^*(e)$ um $\alpha_{i,k}$ entlang $P_{i,k}$

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Sei x_i^* fixiert.

$k := 1$
REPEAT

- sei G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;
- nimm einen Pfad $P_{i,k}$ von s_i nach t_i
- sei $\alpha_{i,k} = \min_{e \in P_{i,k}} x_i^*(e)$
- reduziere $x_i^*(e)$ um $\alpha_{i,k}$ entlang $P_{i,k}$
- $k := k + 1$;

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Sei x_i^* fixiert.

$k := 1$
REPEAT

- sei G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;
- nimm einen Pfad $P_{i,k}$ von s_i nach t_i
- sei $\alpha_{i,k} = \min_{e \in P_{i,k}} x_i^*(e)$
- reduziere $x_i^*(e)$ um $\alpha_{i,k}$ entlang $P_{i,k}$
- $k := k + 1$;

UNTIL es keinen Weg von s_i nach t_i gibt.

Pfad-Zerlegung (*path-decomposition*)

sei $x^* = \{x_1^*, x_2^*, \ldots, x_r^*\}$ eine optimale fraktionale Lösung

Sei x_i^* fixiert.

$k := 1$
REPEAT

- sei G_i der Teilgraph aller Kanten mit $x_i^*(e) > 0$;
- nimm einen Pfad $P_{i,k}$ von s_i nach t_i
- sei $\alpha_{i,k} = \min_{e \in P_{i,k}} x_i^*(e)$
- reduziere $x_i^*(e)$ um $\alpha_{i,k}$ entlang $P_{i,k}$
- $k := k + 1$;

UNTIL es keinen Weg von s_i nach t_i gibt.

Routing durch randomisiertes Runden

Routing durch randomisiertes Runden

– Für jedes i zerlege x_i^* in die Pfade $P_{i,1}, P_{i,2}, \ldots, P_{i,r}$ mit Werten

$\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,r}$;

Routing durch randomisiertes Runden

- Für jedes i zerlege x_i^* in die Pfade $P_{i,1}, P_{i,2}, \ldots, P_{i,r}$ mit Werten

  $\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,r}$;

- es gilt $\sum_j \alpha_{i,j} = 1$; setze $P_i := P_{i,j}$ mit Wahrscheinlichkeit $\alpha_{i,j}$

Routing durch randomisiertes Runden

– Für jedes i zerlege x_i^* in die Pfade $P_{i,1}, P_{i,2}, \ldots, P_{i,r}$ mit Werten

$\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,r}$;

– es gilt $\sum_j \alpha_{i,j} = 1$; setze $P_i := P_{i,j}$ mit Wahrscheinlichkeit $\alpha_{i,j}$

Beobachtung: Für fixierte $e \in E$ ist

$$\mathbb{E}(\text{Belastung}(e)) \leq W_{\mathrm{OPT}}^{\mathrm{frac}} \leq W_{\mathrm{OPT}}.$$

Routing durch randomisiertes Runden

- Für jedes i zerlege x_i^* in die Pfade $P_{i,1}, P_{i,2}, \ldots, P_{i,r}$ mit Werten

  $\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,r}$;

- es gilt $\sum_j \alpha_{i,j} = 1$; setze $P_i := P_{i,j}$ mit Wahrscheinlichkeit $\alpha_{i,j}$

Beobachtung: Für fixierte $e \in E$ ist

$$\mathbb{E}(\text{Belastung}(e)) \leq W_{\mathrm{OPT}}^{\mathrm{frac}} \leq W_{\mathrm{OPT}}.$$

Beweis: es gilt

$$\sum_{j: e \in P_{i,j}} \alpha_{i,j} \leq x_i^*(e)$$

(Die erwartete *maximale* Belastung ist auch nicht groß.)

Beispiel: 4. Das MAX-SAT Problem

Beispiel: 4. Das MAX-SAT Problem

Eingabe: eine Menge K von *Klauseln* mit *Literalen* aus $\{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$
wobei Klausel k das Gewicht w_k besitzt

Beispiel: 4. Das MAX-SAT Problem

Eingabe: eine Menge K von *Klauseln* mit *Literalen* aus
$\{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$
wobei Klausel k das Gewicht w_k besitzt

Ausgabe: Eine Wahrheitsbelegung die eine Klauselmenge mit
größtmöglichem Gesamtgewicht erfüllt.

Beispiel: 4. Das MAX-SAT Problem

Eingabe: eine Menge K von *Klauseln* mit *Literalen* aus $\{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$
wobei Klausel k das Gewicht w_k besitzt

Ausgabe: Eine Wahrheitsbelegung die eine Klauselmenge mit größtmöglichem Gesamtgewicht erfüllt.

ALG1: Eine 2-Approximation ist einfach. (Warum?)

Beispiel: 4. Das MAX-SAT Problem

Eingabe: eine Menge K von *Klauseln* mit *Literalen* aus $\{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$
wobei Klausel k das Gewicht w_k besitzt

Ausgabe: Eine Wahrheitsbelegung die eine Klauselmenge mit größtmöglichem Gesamtgewicht erfüllt.

ALG1: Eine 2-Approximation ist einfach. (Warum?)

ALG2: Randomisierung für lange Klauseln

ALG2: Randomisierung für lange Klauseln

FOR $i = 1$ TO n DO

- setze $x_i = \mathrm{TRUE}$ mit Wahrscheinlichkeit $1/2$,
  und $x_i = \mathrm{FALSE}$ sonst

ALG2: Randomisierung für lange Klauseln

FOR $i = 1$ TO n DO

- setze $x_i = \mathrm{TRUE}$ mit Wahrscheinlichkeit $1/2$, und $x_i = \mathrm{FALSE}$ sonst

Beobachtung 1: Eine Klausel der Länge ℓ wird mit Wahrscheinlichkeit $1 - 1/2^\ell$ erfüllt.

ALG2: Randomisierung für lange Klauseln

FOR $i = 1$ TO n DO

- setze $x_i = \mathrm{TRUE}$ mit Wahrscheinlichkeit $1/2$, und $x_i = \mathrm{FALSE}$ sonst

Beobachtung 1: Eine Klausel der Länge ℓ wird mit Wahrscheinlichkeit $1 - 1/2^\ell$ erfüllt.

Beobachtung 2: Wenn jede Klausel mindestens ℓ Literale besitzt, dann ist das *erwartete* Gewicht erfüllter Klauseln

$$\geq (1 - 1/2^\ell) \cdot W_{\mathrm{OPT}}.$$

IP-Formulierung

IP-Formulierung

Interpretation: y_i ist die Belegung von x_i;

IP-Formulierung

Interpretation: y_i ist die Belegung von x_i;

$z_k = 1$ falls Klausel k erfüllt ist, und $z_k = 0$ sonst

IP-Formulierung

Interpretation: y_i ist die Belegung von x_i;

$z_k = 1$ falls Klausel k erfüllt ist, und $z_k = 0$ sonst

Maximiere $\sum_{k \in K} w_k z_k$

IP-Formulierung

Interpretation: y_i ist die Belegung von x_i;

$z_k = 1$ falls Klausel k erfüllt ist, und $z_k = 0$ sonst

Maximiere $\sum_{k \in K} w_k z_k$

so dass $\sum_{x_i \in k} y_i + \sum_{\neg x_j \in k} (1 - y_j) \geq z_k \quad \forall k \in K$

IP-Formulierung

Interpretation: y_i ist die Belegung von x_i;

$z_k = 1$ falls Klausel k erfüllt ist, und $z_k = 0$ sonst

Maximiere $\sum_{k \in K} w_k z_k$

so dass $\sum_{x_i \in k} y_i + \sum_{\neg x_j \in k} (1 - y_j) \geq z_k \quad \forall k \in K$

$y_i, z_k \in \{0, 1\}$

IP-Formulierung

Interpretation: y_i ist die Belegung von x_i;

$z_k = 1$ falls Klausel k erfüllt ist, und $z_k = 0$ sonst

Maximiere $\sum_{k \in K} w_k z_k$

so dass $\sum_{x_i \in k} y_i + \sum_{\neg x_j \in k} (1 - y_j) \geq z_k \quad \forall k \in K$

$y_i, z_k \in \{0, 1\}$

ALG3: Randomisiertes Runden für kurze Klauseln

ALG3: Randomisiertes Runden für kurze Klauseln

– löse die LP-Relaxierung $\quad 0 \leq z_k \leq 1 \quad 0 \leq y_i \leq 1\,;$

sei (y^*, z^*) eine optimale fraktionale Lösung

ALG3: Randomisiertes Runden für kurze Klauseln

– löse die LP-Relaxierung $\quad 0 \leq z_k \leq 1 \quad 0 \leq y_i \leq 1\,;$

sei (y^*, z^*) eine optimale fraktionale Lösung

– setze $x_i = \mathrm{TRUE}$ mit Wahrscheinlichkeit y_i^*

ALG3: Randomisiertes Runden für kurze Klauseln

- löse die LP-Relaxierung $\quad 0 \leq z_k \leq 1 \quad 0 \leq y_i \leq 1\,;$

  sei (y^*, z^*) eine optimale fraktionale Lösung

- setze $x_i = \mathrm{TRUE}$ mit Wahrscheinlichkeit y_i^*

ALG4: Kombiniere ALG2 mit ALG3

ALG4: Kombiniere ALG2 mit ALG3

- Bestimme eine Wahrheitsbelegung mit ALG2

ALG4: Kombiniere ALG2 mit ALG3

- Bestimme eine Wahrheitsbelegung mit ALG2
- Bestimme eine Wahrheitsbelegung mit ALG3

ALG4: Kombiniere ALG2 mit ALG3

- Bestimme eine Wahrheitsbelegung mit ALG2
- Bestimme eine Wahrheitsbelegung mit ALG3
- gib die bessere Lösung aus!

ALG4: Kombiniere ALG2 mit ALG3

- Bestimme eine Wahrheitsbelegung mit ALG2
- Bestimme eine Wahrheitsbelegung mit ALG3
- gib die bessere Lösung aus!

Theorem: Das *erwartete* Gewicht belegter Klauseln ist

$$\geq \frac{3}{4} \sum_{k \in K} w_k z_k^* \geq \frac{3}{4} \mathrm{OPT}.$$

ALG4: Kombiniere ALG2 mit ALG3

- Bestimme eine Wahrheitsbelegung mit ALG2
- Bestimme eine Wahrheitsbelegung mit ALG3
- gib die bessere Lösung aus!

Theorem: Das *erwartete* Gewicht belegter Klauseln ist

$$\geq \frac{3}{4} \sum_{k \in K} w_k z_k^* \geq \frac{3}{4} \mathrm{OPT}.$$

Übrigens: die Integralitätslücke ist $\geq 4/3$.

Die Integralitätslücke

Die Integralitätslücke

... ist der größtmögliche Quotient des ganzzahligen und des fraktionalen Optimums für ein gegebenes Problem.

Die Integralitätslücke

... ist der größtmögliche Quotient des ganzzahligen und des fraktionalen Optimums für ein gegebenes Problem.

Sei I eine konkrete Instanz eines Minimierungsproblems

Die Integralitätslücke

... ist der größtmögliche Quotient des ganzzahligen und des fraktionalen Optimums für ein gegebenes Problem.

Sei I eine konkrete Instanz eines Minimierungsproblems

- sei $OPT(I)$ das (ganzzahlige) Optimum

Die Integralitätslücke

... ist der größtmögliche Quotient des ganzzahligen und des fraktionalen Optimums für ein gegebenes Problem.

Sei I eine konkrete Instanz eines Minimierungsproblems

- sei $OPT(I)$ das (ganzzahlige) Optimum
- sei $OPT_{\mathrm{f}rac}(I)$ das (fraktionale) Optimum der LP-Relaxierung

Die Integralitätslücke

... ist der größtmögliche Quotient des ganzzahligen und des fraktionalen Optimums für ein gegebenes Problem.

Sei I eine konkrete Instanz eines Minimierungsproblems

- sei $OPT(I)$ das (ganzzahlige) Optimum
- sei $OPT_{\mathrm{f}rac}(I)$ das (fraktionale) Optimum der LP-Relaxierung
- für Minimierungsprobleme gilt $\quad OPT_{\mathrm{f}rac}(I) \leq OPT(I)$

Die Integralitätslücke

... ist der größtmögliche Quotient des ganzzahligen und des fraktionalen Optimums für ein gegebenes Problem.

Sei I eine konkrete Instanz eines Minimierungsproblems

- sei $OPT(I)$ das (ganzzahlige) Optimum
- sei $OPT_{\mathrm{f}rac}(I)$ das (fraktionale) Optimum der LP-Relaxierung
- für Minimierungsprobleme gilt $OPT_{\mathrm{f}rac}(I) \leq OPT(I)$

Definition: Die *Integralitätslücke (integrality gap)* ist der größte Quotient über alle Instanzen

$$\sup_I \frac{OPT(I)}{OPT_{\mathrm{f}rac}(I)}$$

Die Integralitätslücke

... ist der größtmögliche Quotient des ganzzahligen und des fraktionalen Optimums für ein gegebenes Problem.

Sei I eine konkrete Instanz eines Minimierungsproblems

- sei $OPT(I)$ das (ganzzahlige) Optimum
- sei $OPT_{\mathrm{f}rac}(I)$ das (fraktionale) Optimum der LP-Relaxierung
- für Minimierungsprobleme gilt $OPT_{\mathrm{f}rac}(I) \leq OPT(I)$

Definition: Die *Integralitätslücke (integrality gap)* ist der größte Quotient über alle Instanzen

$$\sup_I \frac{OPT(I)}{OPT_{\mathrm{f}rac}(I)}$$

(bei Maximierungsproblemen

$$\sup_I \frac{OPT_{\mathrm{f}rac}(I)}{OPT(I)})$$

Beispiel 5: Das SET COVER Problem

Beispiel 5: Das SET COVER Problem

Eingabe: n Teilmengen $S_j \subseteq \{1, 2, \ldots, m\}$
mit Gewichten $w_j \quad j = 1, 2, \ldots n$

Beispiel 5: Das SET COVER Problem

Eingabe: n Teilmengen $S_j \subseteq \{1, 2, \ldots, m\}$
mit Gewichten $w_j \quad j = 1, 2, \ldots n$

Ausgabe: eine Überdeckung der $\{1, 2, \ldots, m\}$ mit einer leichtesten
Auswahl $\{S_j \mid j \in C\}$ der Teilmengen.
($C \subset \{1, \ldots n\}$ sind die Indizes der ausgewählten Mengen S_j)

Beispiel 5: Das SET COVER Problem

Eingabe: n Teilmengen $S_j \subseteq \{1, 2, \ldots, m\}$
mit Gewichten $w_j \quad j = 1, 2, \ldots n$

Ausgabe: eine Überdeckung der $\{1, 2, \ldots, m\}$ mit einer leichtesten
Auswahl $\{S_j \mid j \in C\}$ der Teilmengen.
($C \subset \{1, \ldots n\}$ sind die Indizes der ausgewählten Mengen S_j)

IP-Formulierung

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$
- so dass $\sum_{j: i \in S_j} x_j \geq 1 \quad$ für $i = 1, 2, \ldots m$

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$
- so dass $\sum_{j: i \in S_j} x_j \geq 1 \quad$ für $i = 1, 2, \ldots m$
- und $x_j \in \{0, 1\}$

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$
- so dass $\sum_{j:i \in S_j} x_j \geq 1 \quad$ für $i = 1, 2, \ldots m$
- und $x_j \in \{0, 1\}$

LP-Relaxierung: $\quad 0 \leq x_j \, (\leq 1)$

Beobachtung

Beobachtung

– SET COVER ist eine *Verallgemeinerung* von VERTEX COVER, wobei die Mengen S_j den Knoten entsprechen.

Beobachtung

- SET COVER ist eine *Verallgemeinerung* von VERTEX COVER, wobei die Mengen S_j den Knoten entsprechen.
- Insbesondere ist jedes 'Element' (Kante) in höchstens zwei 'Mengen' (Knoten) enthalten, und deterministisches Runden ergibt eine 2-Approximation.

Beobachtung

- SET COVER ist eine *Verallgemeinerung* von VERTEX COVER, wobei die Mengen S_j den Knoten entsprechen.
- Insbesondere ist jedes 'Element' (Kante) in höchstens zwei 'Mengen' (Knoten) enthalten, und deterministisches Runden ergibt eine 2-Approximation.

Deterministisches Runden

Deterministisches Runden

Falls jedes Element in höchstens ℓ Teilmengen enthalten ist...

Deterministisches Runden

Falls jedes Element in höchstens ℓ Teilmengen enthalten ist...

– sei x^* eine optimale fraktionale Lösung

Deterministisches Runden

Falls jedes Element in höchstens ℓ Teilmengen enthalten ist...

- sei x^* eine optimale fraktionale Lösung
- Nehme S_i in die Überdeckung, wenn $x_i^* \geq 1/\ell$

Deterministisches Runden

Falls jedes Element in höchstens ℓ Teilmengen enthalten ist...

- sei x^* eine optimale fraktionale Lösung
- Nehme S_i in die Überdeckung, wenn $x_i^* \geq 1/\ell$

dies ergibt eine ℓ-approximative Mengenüberdeckung.

Deterministisches Runden

Falls jedes Element in höchstens ℓ Teilmengen enthalten ist...

- sei x^* eine optimale fraktionale Lösung
- Nehme S_i in die Überdeckung, wenn $x_i^* \geq 1/\ell$

dies ergibt eine ℓ-approximative Mengenüberdeckung.

Greedy Algorithmus für SET COVER

Greedy Algorithmus für SET COVER

Die Variante für ungewichtete Teilmengen:

- Setze $C = \emptyset$;
- WHILE es nicht überdeckte Elemente gibt, DO
  - Setze $C = D \cup \{j\}$ falls S_j die meisten neuen Elemente überdeckt.

Greedy Algorithmus für SET COVER

Die Variante für ungewichtete Teilmengen:

- Setze $C = \emptyset$;
- WHILE es nicht überdeckte Elemente gibt, DO
  - Setze $C = D \cup \{j\}$ falls S_j die meisten neuen Elemente überdeckt.

Die formale Version für gewichtete Teilmengen: (U ist die Menge bereits überdeckter Elemente)

Greedy Algorithmus für SET COVER

Die Variante für ungewichtete Teilmengen:

- Setze $C = \emptyset$;
- WHILE es nicht überdeckte Elemente gibt, DO
  - Setze $C = D \cup \{j\}$ falls S_j die meisten neuen Elemente überdeckt.

Die formale Version für gewichtete Teilmengen: (U ist die Menge bereits überdeckter Elemente)

- $C := \emptyset \quad U := \emptyset$
- WHILE $U \neq \{1, 2, \ldots, m\}$ DO
  - bestimme eine Menge S_k mit
  $$\frac{S_k \setminus U}{w_k} = \max_{j \notin C} \frac{S_j \setminus U}{w_j}$$
  - $U := U \cup S_k \qquad C := C \cup \{k\}$

Greedy Algorithmus für SET COVER

Die Variante für ungewichtete Teilmengen:

- Setze $C = \emptyset$;
- WHILE es nicht überdeckte Elemente gibt, DO
  - Setze $C = D \cup \{j\}$ falls S_j die meisten neuen Elemente überdeckt.

Die formale Version für gewichtete Teilmengen: (U ist die Menge bereits überdeckter Elemente)

- $C := \emptyset \quad U := \emptyset$
- WHILE $U \neq \{1, 2, \ldots, m\}$ DO
  - bestimme eine Menge S_k mit
$$\frac{S_k \setminus U}{w_k} = \max_{j \notin C} \frac{S_j \setminus U}{w_j}$$
  - $U := U \cup S_k \qquad C := C \cup \{k\}$

Theorem: Der Greedy Algorithmus für SET COVER ist $(1 + \ln m)$-approximativ.

Definition: die Klasse $f(n)$-$\mathcal{A}PX$

Definition: die Klasse $f(n)$-$\mathcal{APX}$

$f(n)$-$\mathcal{APX}$ ist die Klasse von Problemen mit einem effizienten $\mathcal{O}(f(n))$-approximativen Algorithmus (für Eingabelänge n).

Definition: die Klasse $f(n)$-$\mathcal{A}PX$

$f(n)$-$\mathcal{APX}$ ist die Klasse von Problemen mit einem effizienten $\mathcal{O}(f(n))$-approximativen Algorithmus (für Eingabelänge n).

poly-$\mathcal{APX}$ ist die Klasse von allen Problemen aus $q(n)$-$\mathcal{APX}$ für irgendein Polynom $q(n)$.

Definition: die Klasse $f(n)$-$\mathcal{A}PX$

$f(n)$-$\mathcal{APX}$ ist die Klasse von Problemen mit einem effizienten $\mathcal{O}(f(n))$-approximativen Algorithmus (für Eingabelänge n).

poly-$\mathcal{APX}$ ist die Klasse von allen Problemen aus $q(n)$-$\mathcal{APX}$ für irgendein Polynom $q(n)$.

Zusammenfassung/Nicht-Approximierbarkeit

Zusammenfassung/Nicht-Approximierbarkeit

Wir kennen Probleme aus den folgenden Problemklassen gemäß effizienter Approximierbarkeit (angenommen $\mathcal{P} \neq \mathcal{NP}$):

Zusammenfassung/Nicht-Approximierbarkeit

Wir kennen Probleme aus den folgenden Problemklassen gemäß effizienter Approximierbarkeit (angenommen $\mathcal{P} \neq \mathcal{NP}$):

$\mathcal{NPO}$ das allgemeine TRAVELLING-SALESMAN-PROBLEM ist nicht effizient approximierbar (z.B. nicht 2^n-approximierbar)

Zusammenfassung/Nicht-Approximierbarkeit

Wir kennen Probleme aus den folgenden Problemklassen gemäß effizienter Approximierbarkeit (angenommen $\mathcal{P} \neq \mathcal{NP}$):

$\mathcal{NPO}$ das allgemeine TRAVELLING-SALESMAN-PROBLEM ist nicht effizient approximierbar (z.B. nicht 2^n-approximierbar)

(wir haben dies durch eine Reduktion von HAMILTONSCHER-KREIS nachgewiesen)

Zusammenfassung/Nicht-Approximierbarkeit

Wir kennen Probleme aus den folgenden Problemklassen gemäß effizienter Approximierbarkeit (angenommen $\mathcal{P} \neq \mathcal{NP}$):

$\mathcal{NPO}$ das allgemeine TRAVELLING-SALESMAN-PROBLEM ist nicht effizient approximierbar (z.B. nicht 2^n-approximierbar)

(wir haben dies durch eine Reduktion von HAMILTONSCHER-KREIS nachgewiesen)

$\Rightarrow$ 0-1 PROGRAMMIERUNG ist nicht approximierbar.

Zusammenfassung/Nicht-Approximierbarkeit

Wir kennen Probleme aus den folgenden Problemklassen gemäß effizienter Approximierbarkeit (angenommen $\mathcal{P} \neq \mathcal{NP}$):

$\mathcal{NPO}$ das allgemeine TRAVELLING-SALESMAN-PROBLEM ist nicht effizient approximierbar (z.B. nicht 2^n-approximierbar)

(wir haben dies durch eine Reduktion von HAMILTONSCHER-KREIS nachgewiesen)

$\Rightarrow$ 0-1 PROGRAMMIERUNG ist nicht approximierbar.

n-$\mathcal{APX}$ max-CLIQUE und max-INDEPENDENT-SET sind nicht polynomiell approximierbar mit Faktor $\leq n^{1-\epsilon}$

(diese Probleme lassen sich leicht ineinander transformieren)

Zusammenfassung/Nicht-Approximierbarkeit

Wir kennen Probleme aus den folgenden Problemklassen gemäß effizienter Approximierbarkeit (angenommen $\mathcal{P} \neq \mathcal{NP}$):

$\mathcal{NPO}$ das allgemeine TRAVELLING-SALESMAN-PROBLEM ist nicht effizient approximierbar (z.B. nicht 2^n-approximierbar)

(wir haben dies durch eine Reduktion von HAMILTONSCHER-KREIS nachgewiesen)

$\Rightarrow$ 0-1 PROGRAMMIERUNG ist nicht approximierbar.

n-$\mathcal{APX}$ max-CLIQUE und max-INDEPENDENT-SET sind nicht polynomiell approximierbar mit Faktor $\leq n^{1-\epsilon}$

(diese Probleme lassen sich leicht ineinander transformieren)

log-$\mathcal{APX}$ (sehr wahrscheinlich) min-SET-COVER ist nicht polynomiell approximierbar mit Faktor $\leq (1-\varepsilon)\ln m$
(m ist die Anzahl der Elemente)

Zusammenfassung/Nicht-Approximierbarkeit

Wir kennen Probleme aus den folgenden Problemklassen gemäß effizienter Approximierbarkeit (angenommen $\mathcal{P} \neq \mathcal{NP}$):

$\mathcal{NPO}$ das allgemeine TRAVELLING-SALESMAN-PROBLEM ist nicht effizient approximierbar (z.B. nicht 2^n-approximierbar)

(wir haben dies durch eine Reduktion von HAMILTONSCHER-KREIS nachgewiesen)

$\Rightarrow$ 0-1 PROGRAMMIERUNG ist nicht approximierbar.

n-$\mathcal{APX}$ max-CLIQUE und max-INDEPENDENT-SET sind nicht polynomiell approximierbar mit Faktor $\leq n^{1-\epsilon}$

(diese Probleme lassen sich leicht ineinander transformieren)

log-$\mathcal{APX}$ (sehr wahrscheinlich) min-SET-COVER ist nicht polynomiell approximierbar mit Faktor $\leq (1-\varepsilon)\ln m$
(m ist die Anzahl der Elemente)

($\Rightarrow$ der Greedy Algorithmus ist optimal!)

Diese weiter enthalten die Approximations-Klassen:

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING,

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING, k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET),

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING, k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET), FACILITY LOCATION, k-MEDIAN

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING, k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET), FACILITY LOCATION, k-MEDIAN

$\Rightarrow$ sie alle besitzen kein PTAS.

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING, k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET), FACILITY LOCATION, k-MEDIAN

$\Rightarrow$ sie alle besitzen kein PTAS.

$\mathcal{PTAS}$ min-SCHEDULING, EUKLIDSCHES TSP, BIN-PACKING (asymptotisch)

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING, k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET), FACILITY LOCATION, k-MEDIAN

$\Rightarrow$ sie alle besitzen kein PTAS.

$\mathcal{PTAS}$ min-SCHEDULING, EUKLIDSCHES TSP, BIN-PACKING (asymptotisch)

$\mathcal{FPTAS}$ min-SCHEDULING-m, RUCKSACK

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING, k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET), FACILITY LOCATION, k-MEDIAN

$\Rightarrow$ sie alle besitzen kein PTAS.

$\mathcal{PTAS}$ min-SCHEDULING, EUKLIDSCHES TSP, BIN-PACKING (asymptotisch)

$\mathcal{FPTAS}$ min-SCHEDULING-m, RUCKSACK

$\mathcal{PO}$ LINEARE PROGRAMMIERUNG, max-MATCHING, min-SPANNBÄUME KÜRZESTE WEGE, min-PRÄFIX-CODE

Diese weiter enthalten die Approximations-Klassen:

$\mathcal{APX}$ Es ist jeweils eine $\alpha > \beta > 1$ bekannt, so dass die folgenden α-approximierbar aber nicht β-approximierbar sind:

min-VERTEX-COVER ($\beta \approx 1.36$; (VERMUTUNG: $\beta = 2 - \varepsilon$));
Δ-TSP ($\beta = 220/219$), BIN-PACKING, k-CENTER ($\beta = 2 - \varepsilon$, Reduktion von DOMINATING SET), FACILITY LOCATION, k-MEDIAN

$\Rightarrow$ sie alle besitzen kein PTAS.

$\mathcal{PTAS}$ min-SCHEDULING, EUKLIDSCHES TSP, BIN-PACKING (asymptotisch)

$\mathcal{FPTAS}$ min-SCHEDULING-m, RUCKSACK

$\mathcal{PO}$ LINEARE PROGRAMMIERUNG, max-MATCHING, min-SPANNBÄUME KÜRZESTE WEGE, min-PRÄFIX-CODE

Beispiel 5: Das SET COVER Problem

Beispiel 5: Das SET COVER Problem

Eingabe: n Teilmengen $S_j \subseteq \{1, 2, \ldots, m\}$
mit Gewichten $w_j \quad j = 1, 2, \ldots n$

Beispiel 5: Das SET COVER Problem

Eingabe: n Teilmengen $S_j \subseteq \{1, 2, \ldots, m\}$
mit Gewichten $w_j \quad j = 1, 2, \ldots n$

Ausgabe: eine Überdeckung der $\{1, 2, \ldots, m\}$ mit einer leichtesten
Auswahl $\{S_j \mid j \in C\}$ der Teilmengen.
($C \subset \{1, \ldots n\}$ sind die Indizes der ausgewählten Mengen S_j)

Beispiel 5: Das SET COVER Problem

Eingabe: n Teilmengen $S_j \subseteq \{1, 2, \ldots, m\}$
mit Gewichten $w_j \quad j = 1, 2, \ldots n$

Ausgabe: eine Überdeckung der $\{1, 2, \ldots, m\}$ mit einer leichtesten
Auswahl $\{S_j \mid j \in C\}$ der Teilmengen.
($C \subset \{1, \ldots n\}$ sind die Indizes der ausgewählten Mengen S_j)

IP-Formulierung

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$
- so dass $\sum_{j:i \in S_j} x_j \geq 1 \quad$ für $i = 1, 2, \ldots m$

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$
- so dass $\sum_{j:i \in S_j} x_j \geq 1 \quad$ für $i = 1, 2, \ldots m$
- und $x_j \in \{0, 1\}$

IP-Formulierung

IP-Formulierung:

(Interpretation: $x_j = 1$ falls $j \in C$ und $x_j = 0$ sonst)

- Minimiere $\sum_{j=1}^{n} x_j w_j$
- so dass $\sum_{j:i \in S_j} x_j \geq 1 \quad$ für $i = 1, 2, \ldots m$
- und $x_j \in \{0, 1\}$

LP-Relaxierung: $0 \leq x_j \, (\leq 1)$

Randomisiertes Runden für SET COVER

Randomisiertes Runden für SET COVER

1. sei $x = (x_1, x_2, \ldots, x_n)$ eine optimale fraktionale Lösung

Randomisiertes Runden für SET COVER

1. sei $x = (x_1, x_2, \ldots, x_n)$ eine optimale fraktionale Lösung
2. FOR $t = 1$ to T DO

Randomisiertes Runden für SET COVER

1. sei $x = (x_1, x_2, \ldots, x_n)$ eine optimale fraktionale Lösung

2. FOR $t = 1$ to T DO
   - FOR $j = 1$ to n DO:

     nimm S_j mit Wahrscheinlichkeit x_j in die Mengenüberdeckung C_t

Randomisiertes Runden für SET COVER

1. sei $x = (x_1, x_2, \ldots, x_n)$ eine optimale fraktionale Lösung
2. FOR $t = 1$ to T DO
   - FOR $j = 1$ to n DO:

     nimm S_j mit Wahrscheinlichkeit x_j in die Mengenüberdeckung C_t
3. Gib $C = \bigcup_{t=1}^{T} C_t$ als Überdeckung aus.

Randomisiertes Runden für SET COVER

1. sei $x = (x_1, x_2, \ldots, x_n)$ eine optimale fraktionale Lösung

2. FOR $t = 1$ to T DO
   - FOR $j = 1$ to n DO:
     nimm S_j mit Wahrscheinlichkeit x_j in die Mengenüberdeckung C_t

3. Gib $C = \bigcup_{t=1}^{T} C_t$ als Überdeckung aus.

Theorem: Für $T = \ln m + 1$ erhalten wir eine vollständige Überdeckung mit Wahrscheinlichkeit $> 1/2$.

Die Überdeckung hat *erwartetes* Gewicht $\leq W_{\mathrm{OPT}} \cdot (\ln m + 1)$.

Vollständig unimodulare Programme

Beispiel 6: max-MATCHING

Beispiel 6: max-MATCHING

Für $G(V, E)$ mit Kantengewichten w_e,

finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)

mit maximalem Gewicht

Beispiel 6: max-MATCHING

Für $G(V,E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)
mit maximalem Gewicht

LP-Relaxierung:

Beispiel 6: max-MATCHING

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)
mit maximalem Gewicht

LP-Relaxierung:

Beabsichtigte Bedeutung der Variablen x_e ist

Beispiel 6: max-MATCHING

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)
mit maximalem Gewicht

LP-Relaxierung:

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

Beispiel 6: max-MATCHING

Für $G(V, E)$ mit Kantengewichten w_e,

finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)

mit maximalem Gewicht

LP-Relaxierung:

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

- Maximiere $\sum_{e \in E} x_e w_e$

Beispiel 6: max-MATCHING

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)
mit maximalem Gewicht

LP-Relaxierung:

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

- Maximiere $\sum_{e \in E} x_e w_e$
- so dass

$$\sum_{u, \{u,v\} \in E} x_{\{u,v\}} \leq 1 \qquad \forall v \in V$$

Beispiel 6: max-MATCHING

Für $G(V,E)$ mit Kantengewichten w_e,

finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)

mit maximalem Gewicht

LP-Relaxierung:

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

- Maximiere $\sum_{e \in E} x_e w_e$
- so dass
$$\sum_{u,\{u,v\} \in E} x_{\{u,v\}} \leq 1 \qquad \forall v \in V$$
- und $0 \leq x_e \qquad \forall e \in E$

Beispiel 6: max-MATCHING

Für $G(V, E)$ mit Kantengewichten w_e,
finde ein Matching $M \subseteq E$ (Kanten ohne gemeinsame Endknoten)
mit maximalem Gewicht

LP-Relaxierung:

Beabsichtigte Bedeutung der Variablen x_e ist

$x_e = 1$ wenn $e \in M$ und $x_e = 0$ sonst.

- Maximiere $\sum_{e \in E} x_e w_e$
- so dass
$$\sum_{u, \{u,v\} \in E} x_{\{u,v\}} \leq 1 \qquad \forall v \in V$$
- und $0 \leq x_e \qquad \forall e \in E$

Vollständig unimodulare Matrizen (ohne Beweis)

Vollständig unimodulare Matrizen (ohne Beweis)

Definition: Die Matrix A ist *vollständig unimodular*, wenn $\det B \in \{-1, 0, 1\}$ für *jede* quadratische *Teilmatrix* B von A gilt.

Vollständig unimodulare Matrizen (ohne Beweis)

Definition: Die Matrix A ist *vollständig unimodular*, wenn $\det B \in \{-1, 0, 1\}$ für *jede* quadratische *Teilmatrix* B von A gilt.

(insb. sind alle Einträge 1, -1 oder 0)

Vollständig unimodulare Matrizen (ohne Beweis)

Definition: Die Matrix A ist *vollständig unimodular*, wenn $\det B \in \{-1, 0, 1\}$ für *jede* quadratische *Teilmatrix* B von A gilt.

(insb. sind alle Einträge 1, -1 oder 0)

Theorem: Wenn A vollständig unimodular, und b ganzzahlig ist, dann haben die Ecken der Lösungspolyeder
$\{x \mid A \cdot x = b, x \geq 0\}$ bzw. $\{x \mid A \cdot x \geq b\}$
nur ganzzahlige Koordinaten.

Vollständig unimodulare Matrizen (ohne Beweis)

Definition: Die Matrix A ist *vollständig unimodular*, wenn $\det B \in \{-1, 0, 1\}$ für *jede* quadratische *Teilmatrix* B von A gilt.

(insb. sind alle Einträge 1, -1 oder 0)

Theorem: Wenn A vollständig unimodular, und b ganzzahlig ist, dann haben die Ecken der Lösungspolyeder $\{x \mid A \cdot x = b, x \geq 0\}$ bzw. $\{x \mid A \cdot x \geq b\}$ nur ganzzahlige Koordinaten.

Behauptung: Die Inzidenzmatrix für einen Graphen ist vollständig unimodular dann und nur dann, wenn der Graph bipartit ist.

Vollständig unimodulare Matrizen (ohne Beweis)

Definition: Die Matrix A ist *vollständig unimodular*, wenn $\det B \in \{-1, 0, 1\}$ für *jede* quadratische *Teilmatrix* B von A gilt.

(insb. sind alle Einträge 1, -1 oder 0)

Theorem: Wenn A vollständig unimodular, und b ganzzahlig ist, dann haben die Ecken der Lösungspolyeder $\{x \mid A \cdot x = b, x \geq 0\}$ bzw. $\{x \mid A \cdot x \geq b\}$ nur ganzzahlige Koordinaten.

Behauptung: Die Inzidenzmatrix für einen Graphen ist vollständig unimodular dann und nur dann, wenn der Graph bipartit ist.

Korollar: Das max-MATCHING Problem auf bipartiten Graphen ist (auch) mit linearer Programmierung exakt lösbar.

Vollständig unimodulare Matrizen (ohne Beweis)

Definition: Die Matrix A ist *vollständig unimodular*, wenn $\det B \in \{-1, 0, 1\}$ für *jede* quadratische *Teilmatrix* B von A gilt.

(insb. sind alle Einträge 1, -1 oder 0)

Theorem: Wenn A vollständig unimodular, und b ganzzahlig ist, dann haben die Ecken der Lösungspolyeder
$\{x \mid A \cdot x = b, x \geq 0\}$ bzw. $\{x \mid A \cdot x \geq b\}$
nur ganzzahlige Koordinaten.

Behauptung: Die Inzidenzmatrix für einen Graphen ist vollständig unimodular dann und nur dann, wenn der Graph bipartit ist.

Korollar: Das max-MATCHING Problem auf bipartiten Graphen ist (auch) mit linearer Programmierung exakt lösbar.