The Simplex Algorithm

Algorithm 4: Simplex Algorithm

Input: $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$

Output: $\tilde{x} \in \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$ maximizing $c^t x$ or the message that $\max\{c^t x \mid Ax = b, x \geq 0\}$ is unbounded or infeasible

1 Compute a feasible basis B;

2 If no such basis exists, stop with the message “INFEASIBLE”;

3 Set $N = \{1, \ldots, n\} \setminus B$ and compute the feasible basic solution x for B;

4 Compute the simplex tableau $\begin{array}{ccccc} x_B & = & p & + & Qx_N \\ \hline z & = & z_0 & + & r^t x_N \end{array}$ for B;

5 **if** $r \leq 0$ **then**

  return $\tilde{x} = x$;

6 Choose $\alpha \in N$ with $r_\alpha > 0$;

7 **if** $q_{i\alpha} \geq 0$ **for all** $i \in B$ **then**

  return “UNBOUNDED”;

8 Choose $\beta \in B$ with $q_{\beta\alpha} < 0$ and $\frac{p_\beta}{q_{\beta\alpha}} = \max\{\frac{p_i}{q_{i\alpha}} \mid q_{i\alpha} < 0, i \in B\}$;

9 Set $B = (B \setminus \{\beta\}) \cup \{\alpha\}$;

10 GOTO line 3;

Klee-Minty-Würfel

$$
\begin{aligned}
\max x_n \\
-x_1 &\leq 0 \\
x_1 &\leq 1 \\
\epsilon x_{j-1} - x_j &\leq 0 \quad \text{for } j \in \{2, \ldots, n\} \\
\epsilon x_{j-1} + x_j &\leq 1 \quad \text{for } j \in \{2, \ldots, n\}
\end{aligned}
$$

Der Lösungsraum dieses LPs heißt **Klee-Minty-Würfel**.

Mit der Bland-Regel betrachtet der SIMPLEX-ALGORITHMUS (beginnend mit der Ecke $(0, \ldots, 0)$) 2^n Basen.

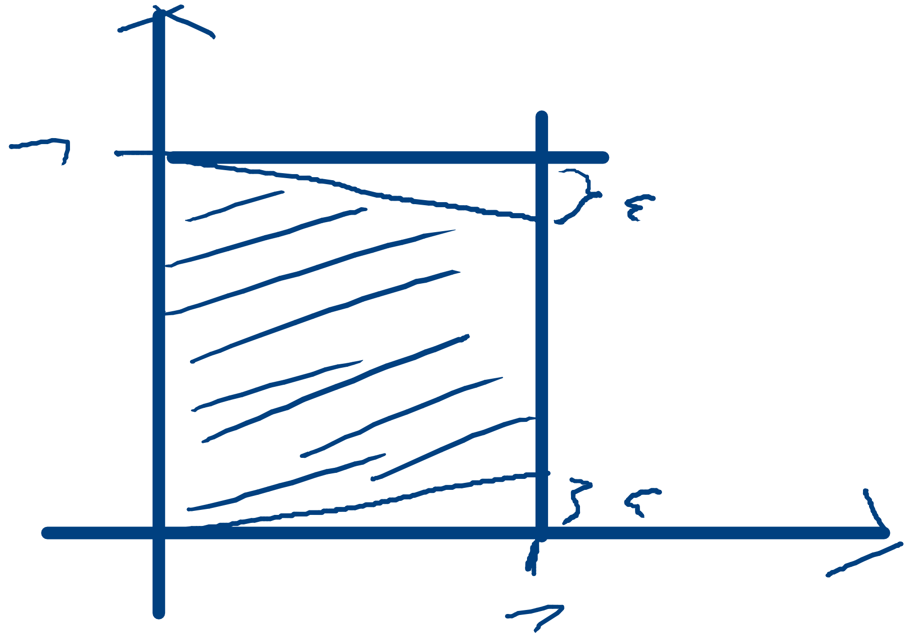
$\max \; x_2$

$x_1 \geq 0$

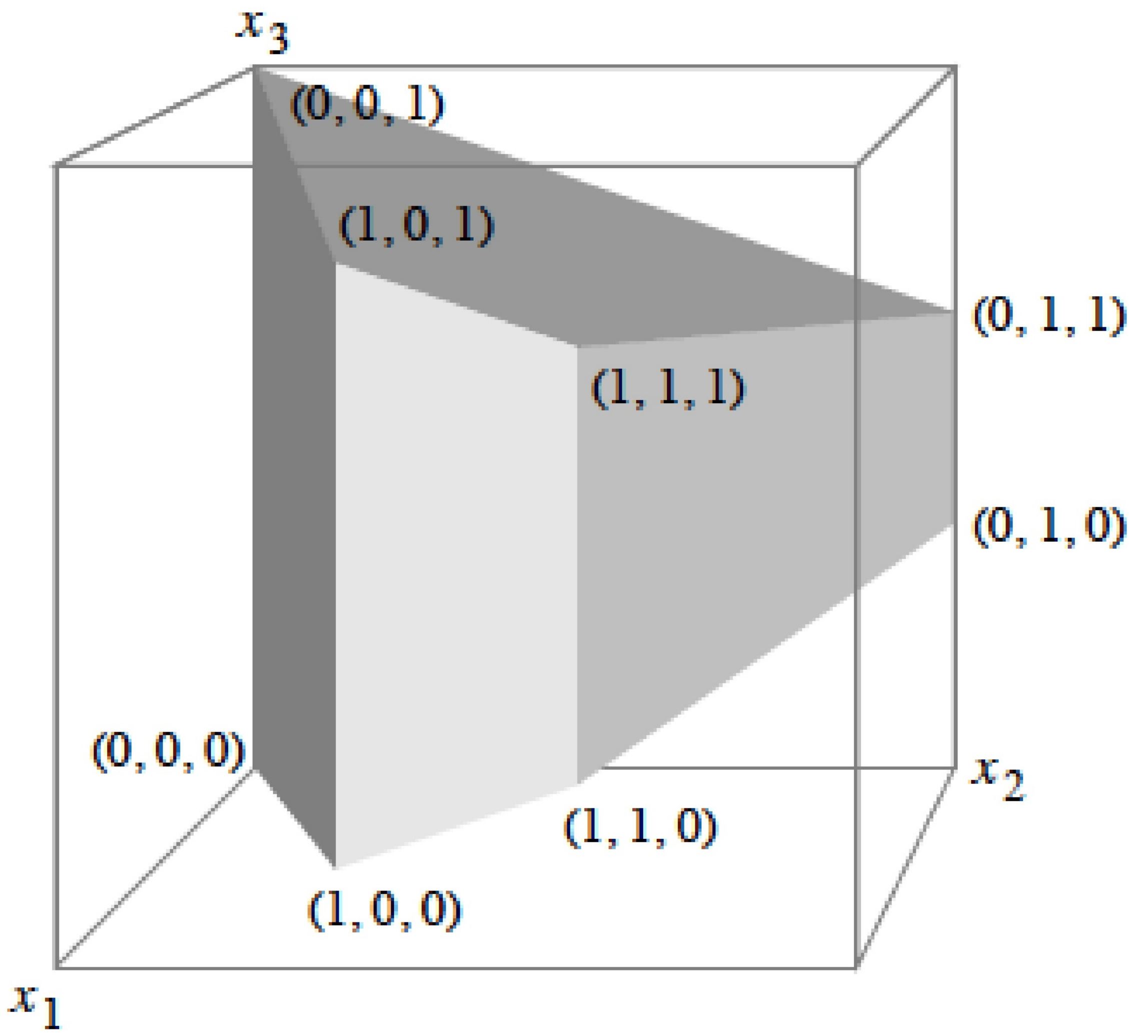
$x_1 \leq 1$

$\varepsilon x_1 \leq x_2$

$x_2 \leq 1 - \varepsilon x_1$



Klee-Minty-Würfel



Effizienz des SIMPLEX-ALGORITHMUS

Schlechte Nachricht:

Für jede bisher angegebene Pivotregel konnten Beispiel gefunden werden, auf denen der SIMPLEX-ALGORITHMUS mit dieser Regel keine polynomielle Laufzeit hat.

Effizienz des SIMPLEX-ALGORITHMUS

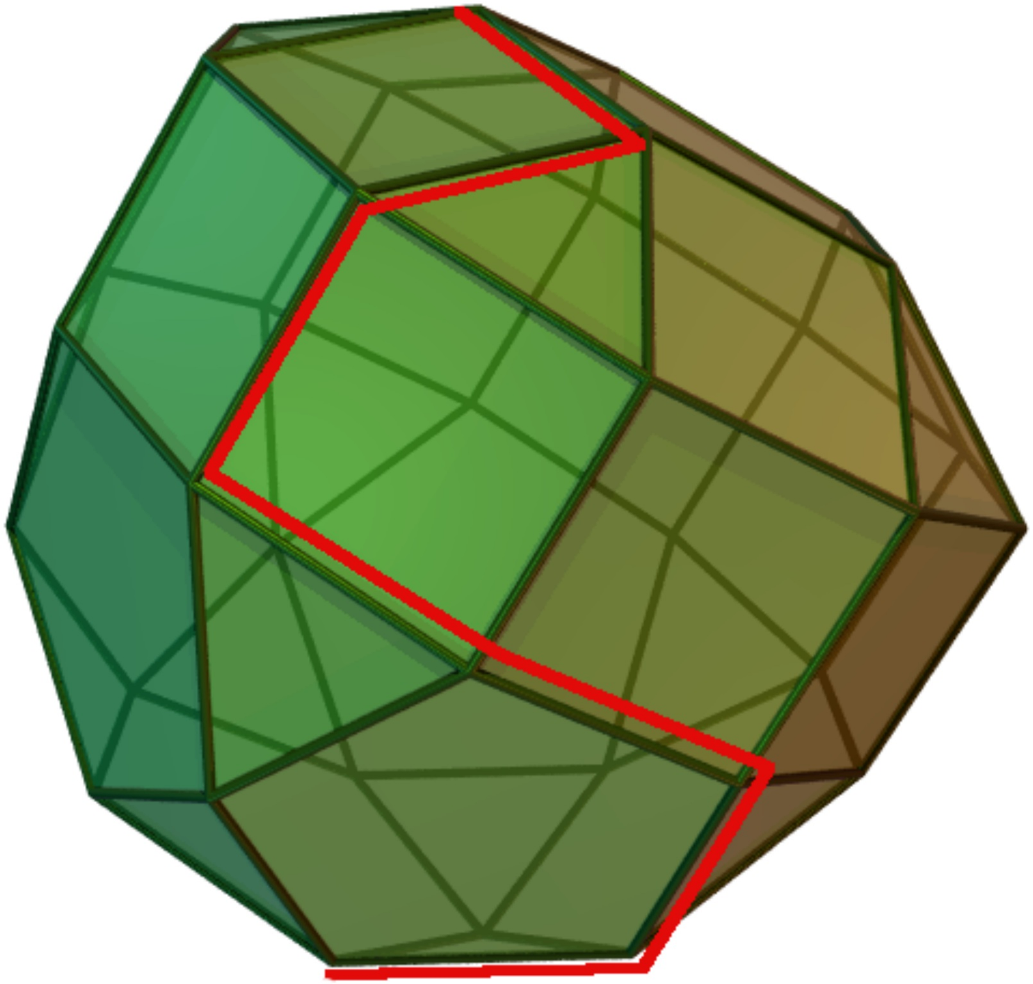
Kann man vielleicht allgemein zeigen, dass der Algorithmus nicht polynomiell ist?

Kann man vielleicht allgemein zeigen, dass der Algorithmus nicht polynomiell ist?

Definition

Der **kombinatorische Durchmesser** eines spitzen Polyeders P ist der Durchmesser (d.h. der größte Abstand zwischen zwei Knoten) des ungerichteten Graphen G_P, wobei $V(G_P)$ die Menge der Ecken von P sei und zwei Knoten $v, w \in V(G_P)$ genau dann durch eine Kante in G_P verbunden sind, wenn es eine Fläche der Dimension 1 gibt, die v und w enthält.

Wege in G_p



Quelle: `https://de.wikipedia.org/`

Effizienz des Simplex-Algorithmus

Ohne Annahmen über die Startlösung ist der kombinatorische Durchmesser eine untere Schranke für die Laufzeit des SIMPLEX-ALGORITHMUS.

Effizienz des Simplex-Algorithmus

Ohne Annahmen über die Startlösung ist der kombinatorische Durchmesser eine untere Schranke für die Laufzeit des SIMPLEX-ALGORITHMUS.

Aber: Es ist unbekannt, ob der der Durchmesser polynomiell (oder sogar linear) beschränkt in der Eingabegröße ist.

Revidierter SIMPLEX-ALGORITHMUS

$$\begin{array}{cccccc} x_B & = & p & + & Qx_N \\ \hline z & = & z_0 & + & r^t x_N \end{array}$$

$Q = -A_B^{-1} A_N$

- Die explizite Berechnung des Simplex-Tableaus ist recht zeitaufwendig und kann vermieden werden.
- Die $m \times (n - m)$-matrix Q muss nicht vollständig gespeichert werden. Es reicht, die Spalte von Q für den Index $\alpha \in N$ mit $r_\alpha > 0$, den man ausählt, zu berechnen (Spaltenerzeugung/ column generation).
- Und die Matrix A_B^{-1} muss nicht explizit bestimmt werden. Es reicht, Gleichungssysteme der Form $A_B y = d$ zu lösen. Dazu kann man z.B. eine LU-Zerlegung von A_B verwalten und geeignet aktualisieren.
- Der SIMPLEX-ALGORITHMUS mit diesen Anpassungen heißt REVIDIERTER SIMPLEX-ALGORITHMUS.

Dualer SIMPLEX-ALGORITHMUS

Simplex-Tableau $T(B)$: $\begin{array}{cccc} x_B & = & p & + & Qx_N \\ \hline z & = & z_0 & + & r^t x_N \end{array}$

- Invariante während des (primalen) SIMPLEX-ALGORITHMUS: $p \geq 0$ (äquivalent zur (primalen) Zulässigkeit)
- Sobald $r \leq 0$ gilt, ist die Lösung optimal. Dann ist $\tilde{y} = A_B^{-t} c_B$ eine optimale Lösung des dualen LPs $\min\{b^t y \mid A^t y \geq c\}$.
- $r \leq 0$ ist äquivalent zur dualen Zulässigkeit.
- Im dualen SIMPLEX-ALGORITHMUS bleibt duale Zulässigkeit (also $r \leq 0$) stets garantiert, während primale Zulässigkeit (also $p \geq 0$) erst am Ende erreicht wird.
- Der duale Algorithmus hat dieselbe Effizienz wie der primale Algorithmus.
- Details: Siehe Übung.

Möglicher Vorteil des dualen SIMPLEX-ALGORITHMUS: Wenn neue Nebenbedingungen eingefügt werden, bleibt die duale Lösung zulässig und wir können mit ihr weiterrechnen.

$$\tilde{y} = A_B^{-t} c_B$$

$$\begin{pmatrix} A_B & A_N \end{pmatrix}$$

z.z.: $\tilde{y}^t A \geq c^t$

$$\tilde{y}^t A_B = c_B$$

$$\tilde{y}^t A_N = c_B^t A_B^{-1} A_N \underset{\text{I}}{\geq} c_N$$

$$r = c_N - (c_B^t A_B^{-1} A_N)^t \leq 0$$

Aus komplementärem Schlupf folgt, dass $\tilde{y}$ optimal ist.

Netzwerk-Simplex-Algorithmus

- Idee: Wende den SIMPLEX-ALGORITHMUS an, um Min-Cost-Flow-Probleme zu lösen.
- Laufzeit wie im allgemeinen Fall: Keine polynomielle Laufzeit beweisbar, aber gut in der Praxis.
- Kann als rein kombinatorischer Algorithmus angesehen werden.

Definition

Sei G ein gerichteter Graph mit Kapaziäten $u : E(G) \to \mathbb{R}_{>0}$ und Zahlen $b : V(G) \to \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$. Ein **zulässiger** ***b*-Fluss in** (G, u, b) ist eine Abbildung $f : E(G) \to \mathbb{R}_{\geq 0}$ mit

- $f(e) \leq u(e)$ für alle $e \in E(G)$ und
- $\sum_{e \in \delta_G^+(v)} f(e) - \sum_{e \in \delta_G^-(v)} f(e) = b(v)$ für alle $v \in V(G)$.

Notation:

- $b(v)$: **Balance** von v.
- Wenn $b(v) > 0$, heißt $b(v)$ **Angebot**/**supply** von v.
- Falls $b(v) < 0$, heißt $b(v)$ **Nachfrage**/**demand** von v.
- Knoten v von G mit $b(v) > 0$ heißen **Quellen**.
- Knoten v von G mit $b(v) < 0$ heißen **Senken**.

Minimum-Cost-Flow-Problem

- **Eingabe:** Ein gerichteter Graph G, Kapazitäten $u : E(G) \to \mathbb{R}_{>0}$, Zahlen $b : V(G) \to \mathbb{R}$ mit $\sum_{v \in V(G)} b(v) = 0$, Kantenkosten $c : E(G) \to \mathbb{R}$.
- **Aufgabe:** Finde einen b-Fluss f, der $\sum_{e \in E(G)} c(e) \cdot f(e)$ minimiert.

Definition

Sei G ein gerichteter Graph.

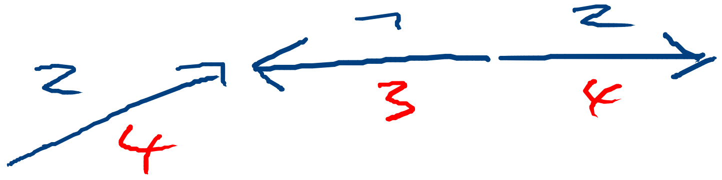
- Für $e = (v, w)$ sei $\overleftarrow{e} = (w, v)$ die **Rückwärtskante**.
- Definiere $\overleftrightarrow{G}$ durch $V(\overleftrightarrow{G}) = V(G)$ und $E(\overleftrightarrow{G}) = E(G) \dot{\cup} \{\overleftarrow{e} \mid e \in E(G)\}$.
- Kantenkosten $c : E(G) \to \mathbb{R}$ werden durch $c(\overleftarrow{e}) := -c(e)$ auf $E(\overleftrightarrow{G})$ erweitert.
- Sei (G, u, b, c) eine MINIMUM-COST-FLOW-Instanz und sei f ein b-Fluss in (G, u). Der **Residualgraph** $G_{u,f}$ ist definiert durch $V(G_{u,f}) := V(G)$ und
$$E(G_{u,f}) := \{e \in E(G) \mid f(e) < u(e)\} \quad \dot{\cup} \quad \{\overleftarrow{e} \in E(\overleftrightarrow{G}) \mid f(e) > 0\}.$$
- Für $e \in E(G)$ definieren wir die **Residualkapzität** durch $u_f(e) = u(e) - f(e)$ und durch $u_f(\overleftarrow{e}) = f(e)$.

Fluss-Augmentierung

Wenn P ein Teilgraph des Residualgraphs $G_{u,f}$ ist, dann heißt **Augmentieren f entlang P um** γ (für $\gamma > 0$), dass man f auf den Vorwärtskanten in P (also Kanten in $E(G) \cap E(P)$) um γ erhöht und auf den Rückwärtskanten in P um γ verringert.

Definition

Sei (G, u, b, c) eine Instanz des MIN-COST-FLOW-PROBLEMS. Ein b-Fluss f in (G, u) heißt **Baumlösung**, wenn der Graph $(V(G), \{e \in E(G) \mid 0 < f(e) < u(e)\})$ keinen ungerichteten Kreis enthält.



$\gamma = 2$

Lemma

Sei (G, u, b, c) eine Instanz des MIN-COST-FLOW-PROBLEMS. Ein b-Fluss f ist genau dann eine Baumlösung, wenn $\tilde{x} \in \mathbb{R}^{E(G)}$ mit $\tilde{x}_e = f(e)$ eine Ecke des Polytops

$$\left\{ x \in \mathbb{R}^{E(G)} \mid 0 \leq x_e \leq u(e) \ (e \in E(G)), \sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = b(v) \quad (v \in V(G)) \right\}.$$

ist

Beweis: „$\Longrightarrow$": Sei f eine Baumlösung und $\tilde{x} \in \mathbb{R}^{E(G)}$ mit $\tilde{x}_e = f(e)$ $\forall e \in E(G)$.

Betrachte die folgenden Nebenbedingungen:

- Für jede Kante $e \in E(G)$ mit $f(e) = 0$ betrachte die Nebenbedingung „$x_e \geq 0$".
- Für jede Kante $e \in E(G)$ mit $f(e) = u(e)$ betrachte die Nebenbedingung „$x_e = u(e)$"
- Für jede Zusammenhangskomponente von $(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$

betrachte für alle Knoten bis auf einen

die Gleichung $\sum_{e\in\delta^+(v)} x_e - \sum_{e\in\delta^-(v)} x_e = b(v)$

$\Rightarrow$ wir haben $|E(G)|$ linear unabhängig Nebenbedingungen, die von $\tilde{x}$ mit Gleichheit erfüllt werden $\Rightarrow$ $\tilde{x}$ ist Ecke des Polyeders.

„$\Leftarrow$": Sei f ein b-Fluss

Es sei $\tilde{x} \in \mathbb{R}^{E(G)}$ mit $\tilde{x}_e = f(e)$ $\forall e \in E(G)$.

Annahme: $\tilde{x}$ ist eine Ecke des Polytops

weitere Annahme: Der Graph

$(V(G), \{e \in E(G) : 0 < f(e) < u(e)\})$ enthält

einen ungerichteten Kreis C.

Wähle $\varepsilon > 0$, sodass $\varepsilon \leq \min\{\min\{f(e), u(e)-f(e)\}:$

$e \in E(C)\}$

Fixiere eine der beiden mögliche Orientierungen von C.

Eine Kante $e \in E(C)$ heiße Vorwärts-kante, wenn ihre Orientierung zu der des Kreises passt, sonst Rück-wärtskante.

Setze $x'_e = \varepsilon$ für alle Vorwärtskanten und $x'_e = -\varepsilon$ für alle Rückwärtskanten

Für alle Kanten $e \in E(G) \setminus E(C)$ sei:

$x'_e = 0$.

$\Rightarrow \hat{x} + x'$ und $\hat{x} - x'$ gehören beide

zum Polytop

Und: $\hat{x} = \frac{1}{2}((\hat{x} + x') + (\hat{x} - x'))$

Widerspruch zur Annahme, dass $\hat{x}$

eine Ecke des Polyeders ist. $\Box$