

# Vorlesung Algorithmische Geometrie

## Lineares Programmieren

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Martin Nöllenburg  
08.05.2012



# Gewinnmaximierung

Sie sind Chef einer Firma, die aus drei Rohstoffen  $R_1, R_2$  und  $R_3$  zwei Produkte  $P_1$  und  $P_2$  herstellt. Produzieren Sie  $x_1$  Einheiten  $P_1$  und  $x_2$  Einheiten  $P_2$ , so beträgt Ihr Gewinn in €

$$G(x_1, x_2) = 300x_1 + 500x_2$$

Um eine Charge der Produkte herzustellen werden jeweils folgende Mengen an Rohstoffen benötigt:

$$P_1: 4R_1 + R_2$$

$$P_2: 11R_1 + R_2 + R_3$$

und in Ihrem Lager befinden sich  $880R_1, 150R_2$  und  $60R_3$ . Damit gilt:

$$R_1: 4x_1 + 11x_2 \leq 880$$

$$R_2: x_1 + x_2 \leq 150$$

$$R_3: x_2 \leq 60$$

Welche Wahl von  $(x_1, x_2)$  maximiert Ihren Gewinn?

# Lösung

## Lineare Beschränkungen:

$$R_1: 4x_1 + 11x_2 \leq 880$$

$$R_2: x_1 + x_2 \leq 150$$

$$R_3: x_2 \leq 60$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$Ax \leq b$$

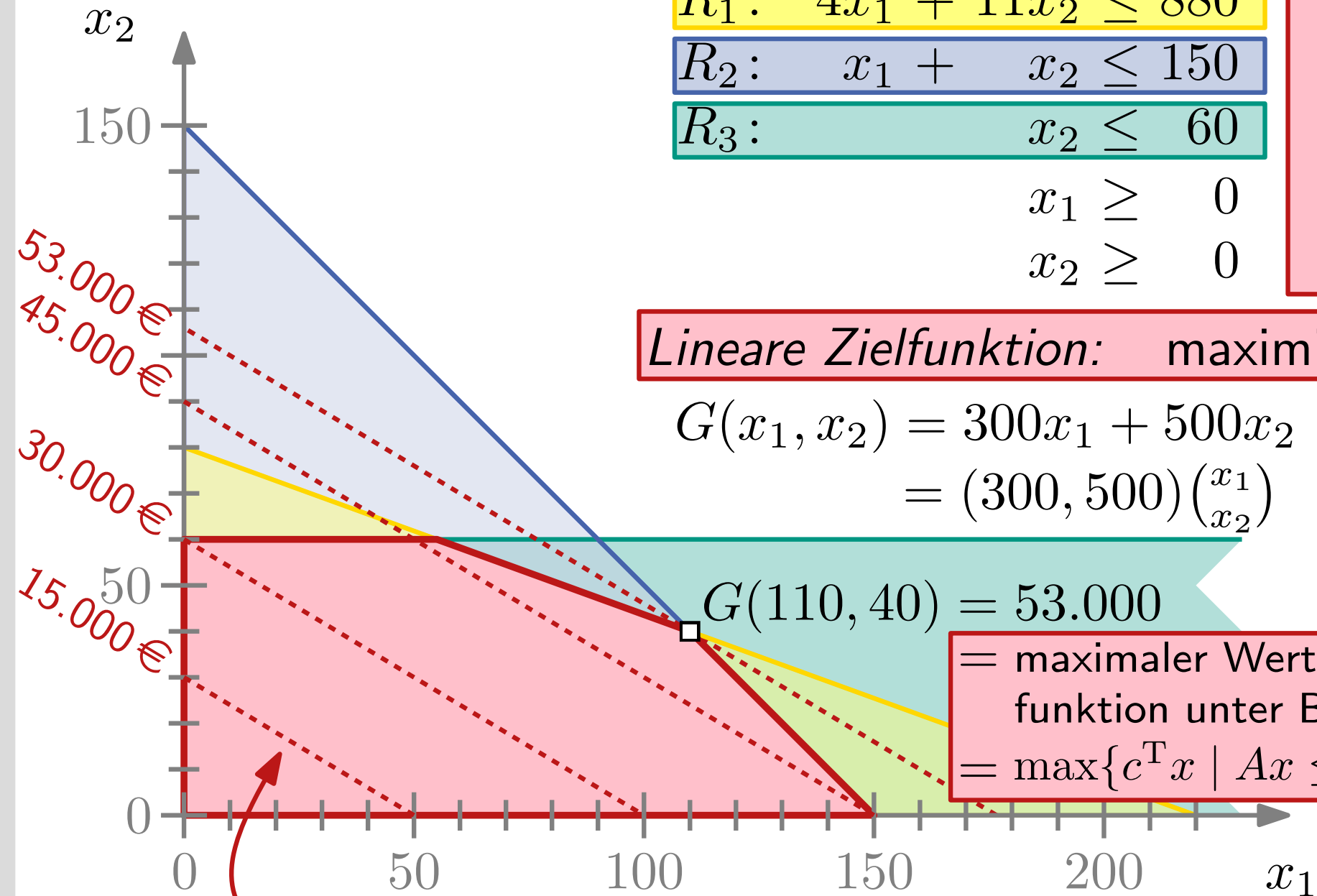
$$x \geq 0$$

## Lineare Zielfunktion: maximiere $c^T x$

$$G(x_1, x_2) = 300x_1 + 500x_2 \\ = (300, 500) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$G(110, 40) = 53.000$$

= maximaler Wert der Zielfunktion unter Beschränk.  
=  $\max\{c^T x \mid Ax \leq b, x \geq 0\}$



„Iso-Gewinn-Line“ (orthogonal zu  $\begin{pmatrix} 300 \\ 500 \end{pmatrix}$ )

**Definition:** Eine Menge linearer Nebenbedingungen  $H$  mit einer linearen Zielfunktion  $c$  in  $\mathbb{R}^d$  bilden ein **lineares Programm (LP)**:

$$\begin{array}{ll} \text{maximiere} & c_1x_1 + c_2x_2 + \cdots + c_dx_d \\ \text{unter den NB} & \left. \begin{array}{l} a_{1,1}x_1 + \cdots + a_{1,d}x_d \leq b_1 \\ a_{2,1}x_1 + \cdots + a_{2,d}x_d \leq b_2 \\ \vdots \\ a_{n,1}x_1 + \cdots + a_{n,d}x_d \leq b_n \end{array} \right\} H \end{array}$$

- $H$  entspricht einer Menge von Halbebenen in  $\mathbb{R}^d$ .
- Gesucht ist ein Punkt  $x \in \bigcap_{h \in H} h$ , der  $c^T x$  maximiert, also  $\max\{c^T x \mid Ax \leq b, x \geq 0\}$ .
- LP ist ein zentrales Optimierungsverfahren im Operations Research.

# Algorithmen für LPs

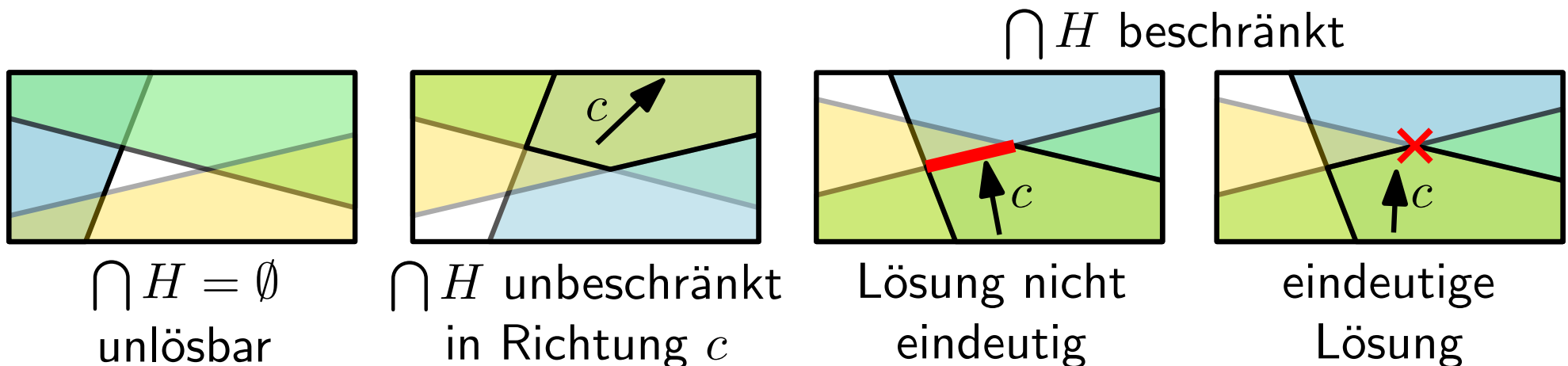
Viele Algorithmen zum Lösen von LPs in der Praxis existieren:

- Simplex-Algorithmus [Dantzig, 1947]
- Ellipsoid-Methode [Khatchiyan, 1979]
- Innere-Punkt-Methode [Karmarkar, 1979]

Funktionieren gut, besonders für große Werte von  $n$  (Anzahl Nebenbedingungen) und  $d$  (Anzahl Variablen).

**Heute:** Spezialfall  $d = 2$

Möglichkeiten für den Lösungsraum



# Erste Variante

**Idee:** Berechne den zulässigen Bereich  $\bigcap H$  und suche nach der Ecke  $p$ , die  $c^T p$  maximiert.

- Halbebenen sind konvex
- Versuche einfachen Divide-and-Conquer Algorithmus

IntersectHalfplanes( $H$ )

**if**  $|H| = 1$  **then**

$C \leftarrow H$

**else**

$(H_1, H_2) \leftarrow \text{SplitInHalves}(H)$

$C_1 \leftarrow \text{IntersectHalfplanes}(H_1)$

$C_2 \leftarrow \text{IntersectHalfplanes}(H_2)$

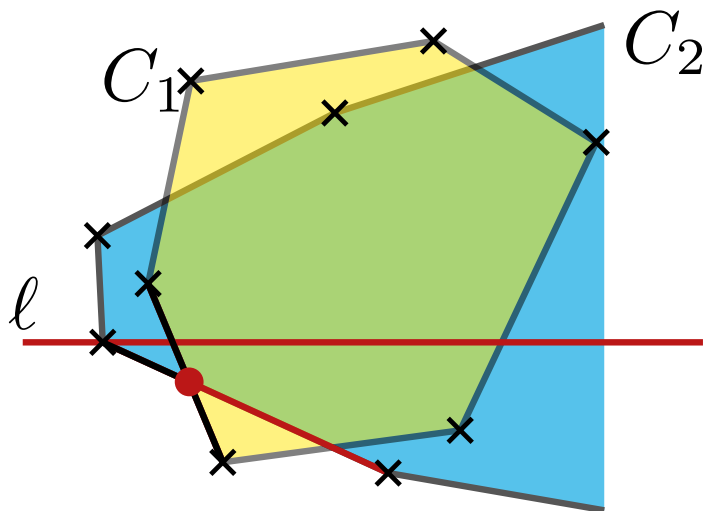
$C \leftarrow \text{IntersectConvexRegions}(C_1, C_2)$

**return**  $C$

# Schnitt konvexer Regionen

Methode  $\text{IntersectConvexRegions}(C_1, C_2)$  kann mit Sweep-Line Verfahren implementiert werden:

- betrachte jeweils linke und rechte Grenze von  $C_1$  und  $C_2$
- bewege sweep line  $\ell$  von oben nach unten und speichere die  $\leq 4$  schneidenden Kanten
- Knoten in  $C_1 \cup C_2$  definieren Events, Behandlung je nach Typ der beginnenden Kante in  $O(1)$  Zeit



## Satz 1:

Der Schnitt zweier konvexer Polygone mit  $n_1 + n_2 = n$  Knoten kann in  $O(n)$  Zeit berechnet werden.

# Laufzeit von IntersectHalfplanes( $H$ )

IntersectHalfplanes( $H$ )

**if**  $|H| = 1$  **then**

$C \leftarrow H$

**else**

$(H_1, H_2) \leftarrow \text{SplitInHalves}(H)$

$C_1 \leftarrow \text{IntersectHalfplanes}(H_1)$

$C_2 \leftarrow \text{IntersectHalfplanes}(H_2)$

$C \leftarrow \text{IntersectConvexRegions}(C_1, C_2)$

**return**  $C$

**Aufgabe:** Welche Laufzeit hat IntersectHalfplanes( $H$ )?

Rekurrenzgleichung

$$T(n) = \begin{cases} O(1) & \text{falls } n = 1 \\ O(n) + 2T(n/2) & \text{falls } n > 1 \end{cases}$$

Master Theorem  $\Rightarrow$   
Laufzeit  $O(n \log n)$



## IntersectHalfplanes( $H$ )

if  $|H| = 1$  then

- zulässiger Bereich  $\cap H$  lässt sich in  $O(n \log n)$  Zeit berechnen
- $\cap H$  hat Komplexität  $O(n)$
- Knoten  $p$  der  $c^T p$  maximiert kann in  $O(n \log n)$  Zeit gefunden werden

**Geht es besser?**

**Aufgabe:** Welche Laufzeit hat IntersectHalfplanes( $H$ )?

Rekurrenzgleichung

$$T(n) = \begin{cases} O(1) & \text{falls } n = 1 \\ O(n) + 2T(n/2) & \text{falls } n > 1 \end{cases}$$

Master Theorem  $\Rightarrow$   
Laufzeit  $O(n \log n)$

# Beschränkte LPs

**Idee:** Statt gesamtes zulässiges Polygon zu berechnen suche inkrementell nach optimaler Ecke.

**Invariante:** aktuell beste Lösung ist eindeutige Ecke des zulässigen aktuellen Polygons

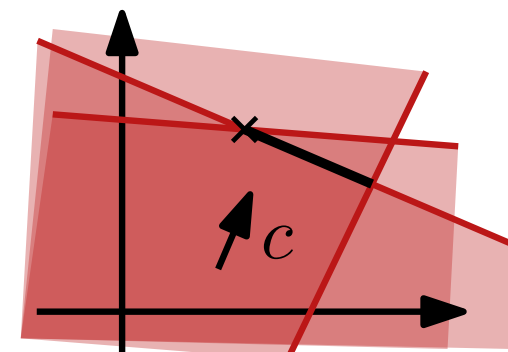
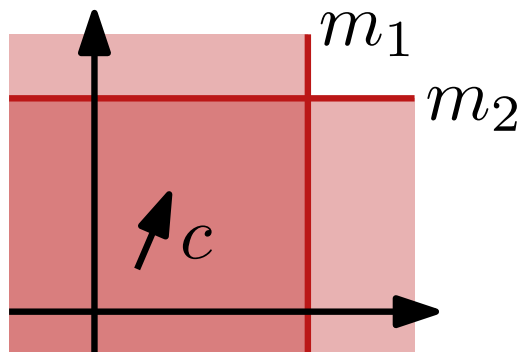
Wie kann man unbeschränkte zulässige Gebiete umgehen?

Gibt es mehrere Optima, wähle lexikographisch kleinsten Punkt!

Für einen ausreichend großen Wert  $M$  definiere Halbebenen

$$m_1 = \begin{cases} x \leq M & \text{falls } c_x > 0 \\ -x \leq M & \text{sonst} \end{cases}$$

$$m_2 = \begin{cases} y \leq M & \text{falls } c_y > 0 \\ -y \leq M & \text{sonst} \end{cases}$$



**Idee:** Statt gesamtes zulässiges Polygon zu berechnen suche inkrementell nach optimaler Ecke.

**Invariante:** aktuell beste Lösung ist eindeutige Ecke des zulässigen aktuellen Polygons

Wie kann man unbeschränkte zulässige Gebiete umgehen?

Gibt es mehrere Optima, wähle lexikographisch kleinsten Punkt!

Für einen ausreichend großen Wert  $M$  definiere Halbebenen

$$m_1 = \begin{cases} x \leq M & \text{falls } c_x > 0 \\ -x \leq M & \text{sonst} \end{cases} \quad m_2 = \begin{cases} y \leq M & \text{falls } c_y > 0 \\ -y \leq M & \text{sonst} \end{cases}$$

Für ein LP  $(H, c)$  mit  $H = \{h_1, \dots, h_n\}$ ,  $c = (c_x, c_y)$ , zulässigem Polygon  $C$  und  $1 \leq i \leq n$  definiere

$$H_i = \{m_1, m_2, h_1, \dots, h_i\}, \quad C_i = m_1 \cap m_2 \cap h_1 \cap \dots \cap h_i$$

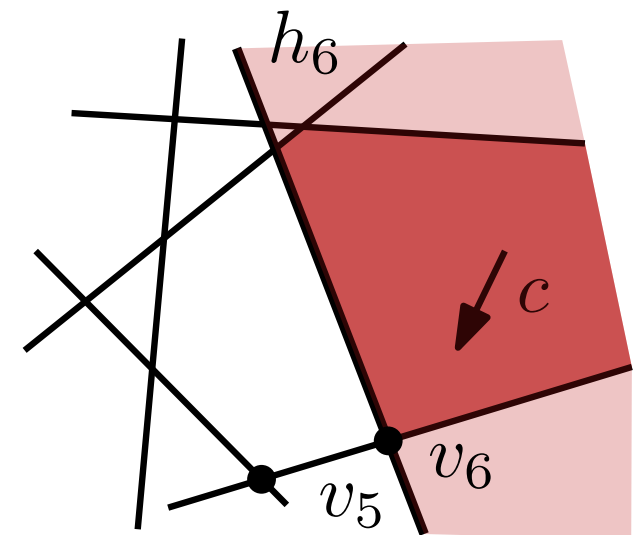
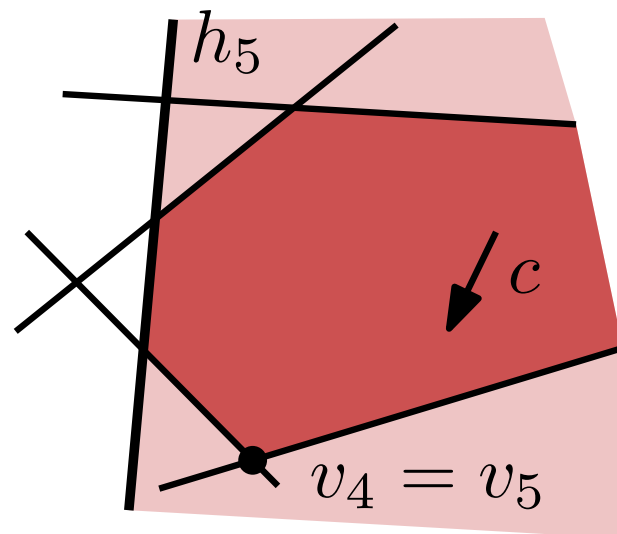
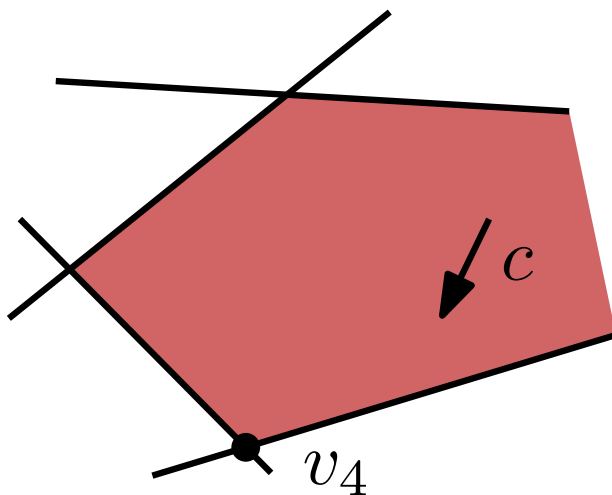
# Eigenschaften

- jede Region  $C_i$  hat eine eindeutige optimale Ecke  $v_i$
- es gilt die Inklusionsbeziehung  $C_0 \supseteq C_1 \supseteq \dots \supseteq C_n = C$

Wie ändert sich die optimale Ecke  $v_{i-1}$  wenn man  $h_i$  hinzufügt?

**Lemma 1:** Für  $1 \leq i \leq n$  und Grenzgerade  $\ell_i$  von  $h_i$  gilt:

- Falls  $v_{i-1} \in h_i$  gilt  $v_i = v_{i-1}$ ,
- sonst ist entweder  $C_i = \emptyset$  oder  $v_i \in \ell_i$ .



# Eindimensionales LP

Im Fall (ii) von Lemma 1 suchen wir den besten Punkt auf der Strecke  $\ell_i \cap C_{i-1}$ :

- parametrisiere  $\ell_i : y = ax + b$
- definiere neue Zielfunktion  $f_c^i(x) = c^T \begin{pmatrix} x \\ ax+b \end{pmatrix}$
- für  $j \leq i - 1$  sei  $\sigma_x(\ell_j, \ell_i)$   $x$ -Koordinate von  $\ell_j \cap \ell_i$

Damit ergibt sich folgendes eindimensionales LP:

$$\text{maximiere } f_c^i(x) = c_x x + c_y (ax + b)$$

$$\begin{array}{ll} \text{mit NB } x \leq \sigma_x(\ell_j, \ell_i) & \text{falls } \ell_i \cap h_j \text{ nach rechts beschr.} \\ x \geq \sigma_x(\ell_j, \ell_i) & \text{falls } \ell_i \cap h_j \text{ nach links beschr.} \end{array}$$

**Lemma 2:** Ein eindimensionales LP kann in linearer Zeit gelöst werden, d.h. in Fall (ii) kann man in  $O(i)$  Zeit die neue Ecke  $v_i$  bestimmen, bzw.  $C_i = \emptyset$  feststellen.

# Inkrementeller Algorithmus

$2d\text{BoundedLP}(H, c, m_1, m_2)$

$C_0 \leftarrow m_1 \cap m_2$

$v_0 \leftarrow$  eindeutige Ecke von  $C_0$

**for**  $i \leftarrow 1$  **to**  $n$  **do**

**if**  $v_{i-1} \in h_i$  **then**

$v_i \leftarrow v_{i-1}$   $O(1)$

**else**

$v_i \leftarrow 1d\text{BoundedLP}(\sigma(H_{i-1}), f_c^i)$   $O(i)$

**if**  $v_i = \text{nil}$  **then**  
            **return** unlösbar

**return**  $v_n$

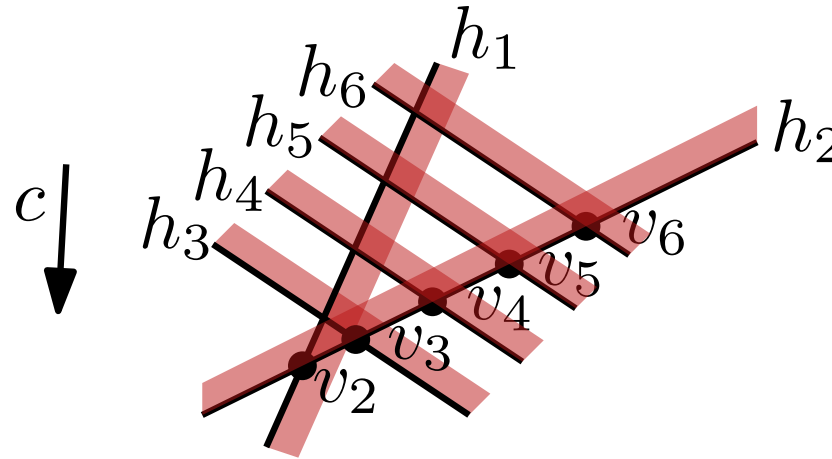
worst-case Laufzeit:

$$T(n) = \sum_{i=1}^n O(i) = O(n^2)$$

**Lemma 3:** Algorithmus  $2d\text{BoundedLP}$  benötigt  $\Theta(n^2)$  Laufzeit um ein LP mit  $n$  Nebenbed. und 2 Variablen zu lösen.

# Gibt es einen Ausweg?

**Beob.:** Nicht die Halbebenen  $H$  sind problematisch für die Laufzeit, sondern die Reihenfolge der Abarbeitung.



Wie findet man (schnell) eine gute Reihenfolge?



$2d\text{RandomizedBoundedLP}(H, c, m_1, m_2)$

$C_0 \leftarrow m_1 \cap m_2$

$v_0 \leftarrow$  eindeutige Ecke von  $C_0$

$H \leftarrow \text{RandomPermutation}(H)$

**for**  $i \leftarrow 1$  **to**  $n$  **do**

**if**  $v_{i-1} \in h_i$  **then**

$v_i \leftarrow v_{i-1}$

**else**

$v_i \leftarrow 1d\text{BoundedLP}(\sigma(H_{i-1}), f_c^i)$

**if**  $v_i = \text{nil}$  **then**

**return** unlösbar

**return**  $v_n$



RandomPermutation( $A$ )

**Input:** Array  $A[1 \dots n]$

**Output:** Array  $A$ , zufällig gleichverteilt permutiert

**for**  $k \leftarrow n$  **to** 2 **do**

$r \leftarrow \text{Random}(k)$   
    tausche  $A[r]$  und  $A[k]$

Zufallszahl zw. 1 und  $k$

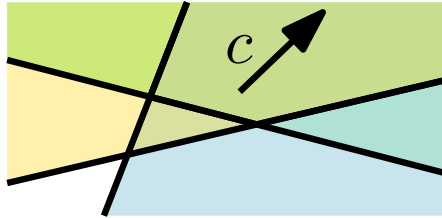
**Beob.:** Die Laufzeit von 2dRandomizedBoundedLP hängt jetzt von der zufälligen Permutation ab. Betrachte daher die **erwartete Laufzeit**.

**Satz 2:** Ein beschränktes zweidimensionales LP mit  $n$  Halbebenen kann in erwarteter  $O(n)$  Laufzeit gelöst werden.

# Unbeschränkte LPs

**Bisher:** künstliche Beschränkung von  $C$  durch  $m_1$  und  $m_2$

**Jetzt:** erkenne und behandle unbeschränkte LPs



$\cap H$  unbeschränkt in Richtung  $c$

**Def.:** Ein LP  $(H, c)$  heißt **unbeschränkt**, wenn es einen Strahl  $\rho = \{p + \lambda d \mid \lambda > 0\}$  in  $C = \cap H$  gibt, so dass die Zielfunktion  $f_c$  beliebig große Werte entlang  $\rho$  annimmt.

Es muss gelten:

- $\langle d, c \rangle > 0$
- $\langle d, \eta(h) \rangle \geq 0$  für alle  $h \in H$  wobei  $\eta(h)$  Normalenvektor auf zulässiger Seite von  $h$  ist

**Lemma 4:** Ein LP  $(H, c)$  ist unbeschränkt genau dann wenn es ein  $d \in \mathbb{R}^2$  gibt mit

- $\langle d, c \rangle > 0$
- $\langle d, \eta(h) \rangle \geq 0$  für alle  $h \in H$
- LP  $(H', c)$  mit  $H' = \{h \in H \mid \langle d, \eta(h) \rangle = 0\}$  ist lösbar.

Teste Unbeschränktheit durch eindimensionales LP:

## Schritt 1:

- rotiere Koordinatensystem bis  $c = (0, 1)$
- normalisiere Vektor  $d$  mit  $\langle d, c \rangle > 0$  zu  $d = (d_x, 1)$
- für Normalenvektor  $\eta(h) = (\eta_x, \eta_y)$  gilt  
$$\langle d, \eta(h) \rangle = d_x \eta_x + \eta_y \geq 0$$
- prüfe dieses 1d-LP auf Lösbarkeit

**Schritt 2:** Falls es in Schritt 1 eine zulässige Lösung  $d_x^*$  gibt

- betrachte  $H' = \{h \in H \mid d_x^* \eta_x(h) + \eta_y(h) = 0\}$
- Normalen von  $H'$  sind orthogonal zu  $d = (d_x, 1) \Rightarrow$   
Halbebenen in  $H'$  sind parallel zu  $d$
- schneide Grenzgeraden in  $H'$  mit  $x$ -Achse  $\rightarrow$  1d-LP

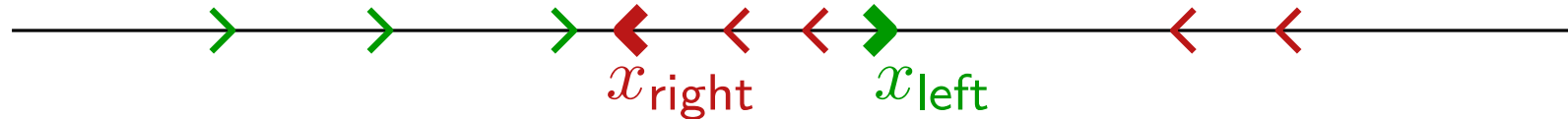
Wenn beide Schritte eine Lösung liefern ist  $(H, c)$  unbeschränkt und wir können einen Strahl  $\rho$  als Zeugen angeben.

Wenn LP in Schritt 2 unlösbar ist, dann ist insbesondere auch  $(H, c)$  unlösbar.

Wenn LP in Schritt 1 unlösbar ist, ist  $(H, c)$  nach Lemma 4 beschränkt.

# Zeugen für Beschränktheit

**Beob.:** Auch wenn LP aus Schritt 1 unlösbar ist, lassen sich die Informationen weiterverwenden!



1d-LP unlösbar  $\Leftrightarrow$  zulässiges Intervall  $[x_{\text{left}}, x_{\text{right}}] = \emptyset$

- dann ist schon  $(\{h_1, h_2\}, c)$  beschränkt ( $h_1$  und  $h_2$  Halbebenen zu  $x_{\text{left}}$  und  $x_{\text{right}}$ )
- $h_1$  und  $h_2$  sind **Zeugen** für die Beschränktheit
- verwende  $h_1$  und  $h_2$  in 2dRandomizedBoundedLP statt  $m_1$  und  $m_2$

2dRandomizedLP( $H, c$ )

$\exists?$  Vektor  $d$  mit  $\langle d, c \rangle > 0$  und  $\langle d, \eta(h) \rangle \geq 0$  für alle  $h \in H$

**if**  $d$  existiert **then**

$H' \leftarrow \{h \in H \mid \langle d, \eta(h) \rangle = 0\}$

**if**  $H'$  lösbar **then**

**return** (Strahl  $\rho$ , unbeschränkt)

**else**

**return** unlösbar

**else**

$(h_1, h_2) \leftarrow$  Zeugen für Beschränktheit von  $(H, c)$

$\overline{H} \leftarrow H \setminus \{h_1, h_2\}$

**return** 2dRandomizedBoundedLP( $\overline{H}, c, h_1, h_2$ )

**Satz 3:** Ein zweidimensionales LP mit  $n$  Halbebenen kann in erwarteter  $O(n)$  Zeit gelöst werden.

## Lässt sich der zweidimensionale Algorithmus auch auf mehr Dimensionen verallgemeinern?

Ja! So wie wir ein zweidimensionales LP inkrementell durch Randomisierung und Zurückführen auf eindimensionale LPs gelöst haben, kann man  $d$ -dimensionale LPs randomisiert inkrementell und durch Lösen von  $(d - 1)$ -dimensionalen LPs lösen. Die erwartete Laufzeit beträgt  $O(c^d d! n)$  für eine Konstante  $c$ , der Algorithmus ist also nur für kleine  $d$  sinnvoll.

## Sind die Zertifikate bei Unlösbarkeit auch sonst nützlich?

Sogenannte zertifizierende Algorithmen liefern nicht nur die Lösung, sondern auch einen Beleg zur einfachen Überprüfung der Lösung. In unserem Fall bei Unbeschränktheit den Strahl  $\rho$  und bei Unlösbarkeit max. drei Halbebenen mit leerem Schnitt.