

# Algorithmische Geometrie

## Lineare Programme & Schnitt von Halbebenen



# Entwicklung einer Gussform

## Gießverfahren (Fertigungsverfahren)

- wiederholten Herstellung eines Objekts
- flüssiges Material wird in die Gussform gegossen
- dort verfestigt es sich und wird wieder herausgenommen



# Entwicklung einer Gussform

## Gießverfahren (Fertigungsverfahren)

- wiederholten Herstellung eines Objekts
- flüssiges Material wird in die Gussform gegossen
- dort verfestigt es sich und wird wieder herausgenommen
- Verlorene Formen: werden beim Herausnehmen zerstört
- Dauerformen: können erhalten werden



# Entwicklung einer Gussform

## Gießverfahren (Fertigungsverfahren)

- wiederholten Herstellung eines Objekts
- flüssiges Material wird in die Gussform gegossen
- dort verfestigt es sich und wird wieder herausgenommen
- Verlorene Formen: werden beim Herausnehmen zerstört
- Dauerformen: können erhalten werden



**Für welche Objekte gibt es Dauerformen?**

# Entwicklung einer Gussform

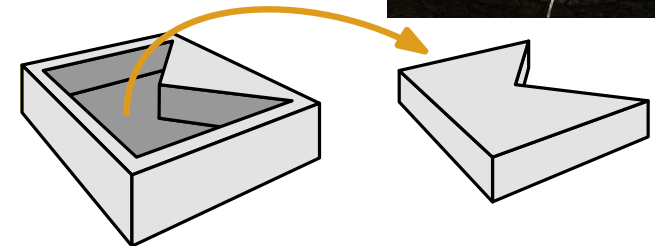
## Gießverfahren (Fertigungsverfahren)

- wiederholten Herstellung eines Objekts
- flüssiges Material wird in die Gussform gegossen
- dort verfestigt es sich und wird wieder herausgenommen
- Verlorene Formen: werden beim Herausnehmen zerstört
- Dauerformen: können erhalten werden



## Für welche Objekte gibt es Dauerformen?

- Annahme: Form besteht aus nur einem Teil



# Entwicklung einer Gussform

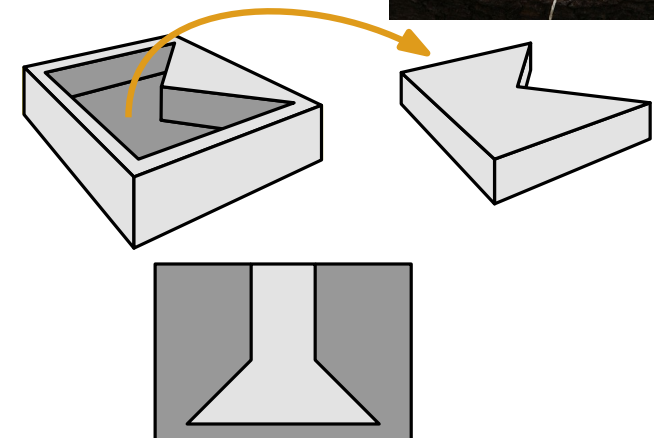
## Gießverfahren (Fertigungsverfahren)

- wiederholten Herstellung eines Objekts
- flüssiges Material wird in die Gussform gegossen
- dort verfestigt es sich und wird wieder herausgenommen
- Verlorene Formen: werden beim Herausnehmen zerstört
- Dauerformen: können erhalten werden



## Für welche Objekte gibt es Dauerformen?

- Annahme: Form besteht aus nur einem Teil
- herausnehmen des gegossenen Objekts ist nicht immer möglich



# Entwicklung einer Gussform

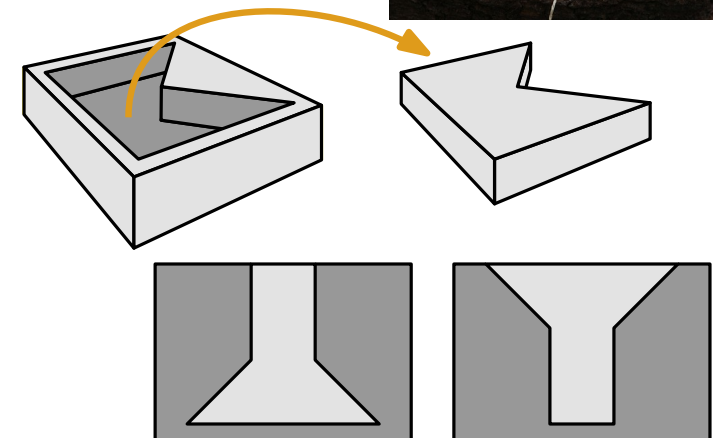
## Gießverfahren (Fertigungsverfahren)

- wiederholten Herstellung eines Objekts
- flüssiges Material wird in die Gussform gegossen
- dort verfestigt es sich und wird wieder herausgenommen
- Verlorene Formen: werden beim Herausnehmen zerstört
- Dauerformen: können erhalten werden



## Für welche Objekte gibt es Dauerformen?

- Annahme: Form besteht aus nur einem Teil
- herausnehmen des gegossenen Objekts ist nicht immer möglich
- eine andere Gussform für dasselbe Objekt funktioniert



# Entwicklung einer Gussform

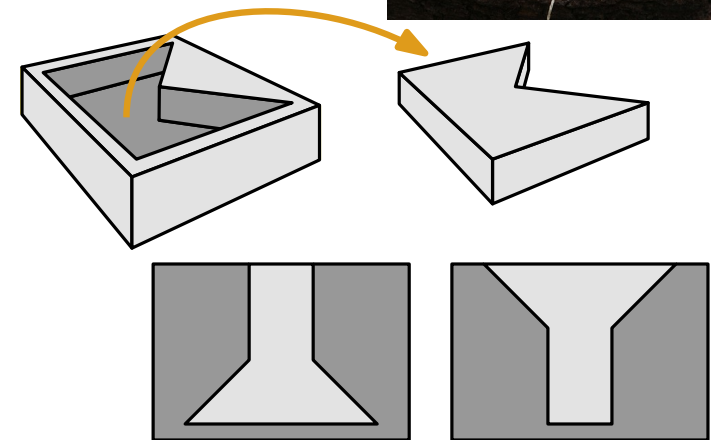
## Gießverfahren (Fertigungsverfahren)

- wiederholten Herstellung eines Objekts
- flüssiges Material wird in die Gussform gegossen
- dort verfestigt es sich und wird wieder herausgenommen
- Verlorene Formen: werden beim Herausnehmen zerstört
- Dauerformen: können erhalten werden



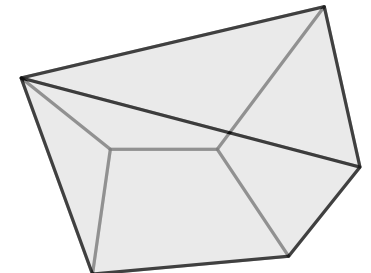
## Für welche Objekte gibt es Dauerformen?

- Annahme: Form besteht aus nur einem Teil
- herausnehmen des gegossenen Objekts ist nicht immer möglich
- eine andere Gussform für dasselbe Objekt funktioniert



### Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?

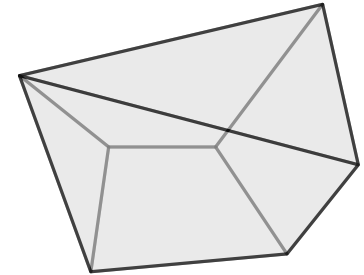




# Erste Beobachtungen

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?

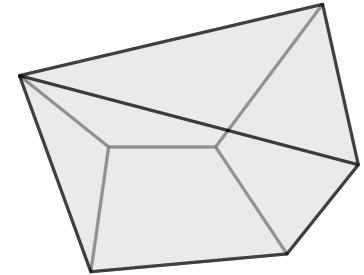


**Welche Entscheidungen haben wir zu treffen?**

# Erste Beobachtungen

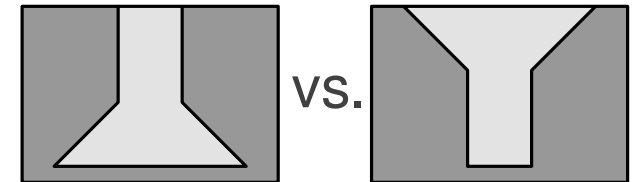
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Welche Entscheidungen haben wir zu treffen?

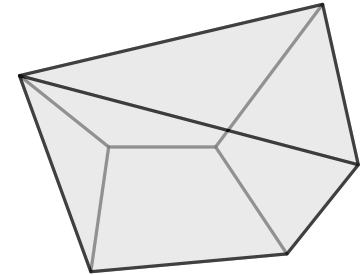
- die Wahl der oberen Facette



# Erste Beobachtungen

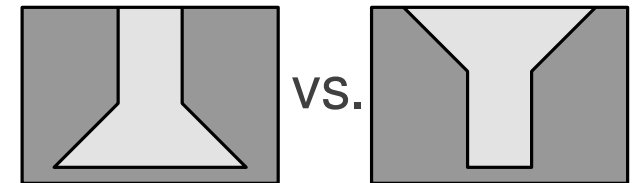
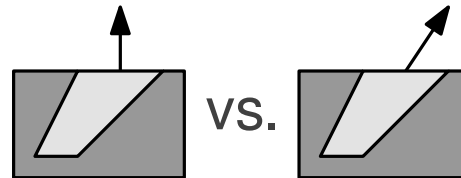
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Welche Entscheidungen haben wir zu treffen?

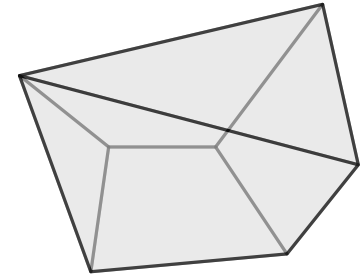
- die Wahl der oberen Facette
- die Richtung der Translation



# Erste Beobachtungen

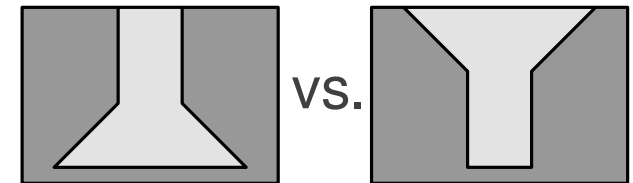
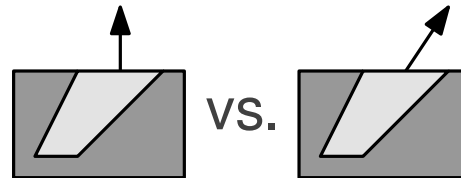
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



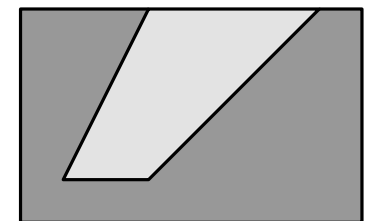
## Welche Entscheidungen haben wir zu treffen?

- die Wahl der oberen Facette
- die Richtung der Translation



## Vereinfachte Situation

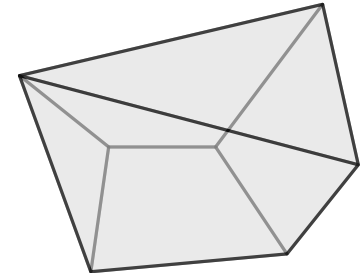
- obere Facette schon ausgewählt  $\Rightarrow$  Gussform steht fest



# Erste Beobachtungen

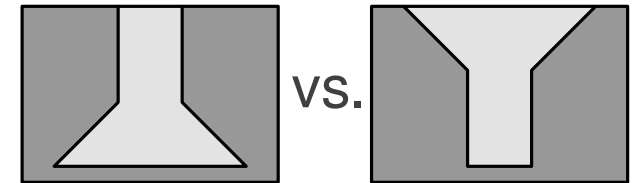
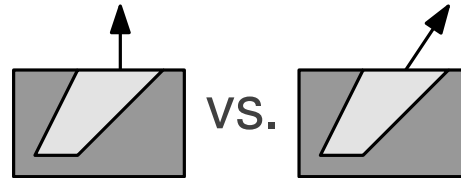
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Welche Entscheidungen haben wir zu treffen?

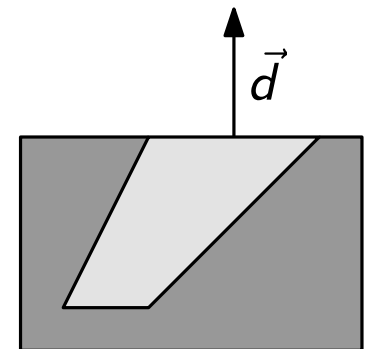
- die Wahl der oberen Facette
- die Richtung der Translation



## Vereinfachte Situation

- obere Facette schon ausgewählt  $\Rightarrow$  Gussform steht fest

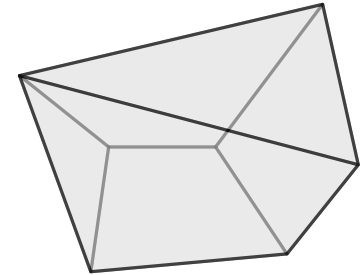
- Richtung der Translation:  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$



# Erste Beobachtungen

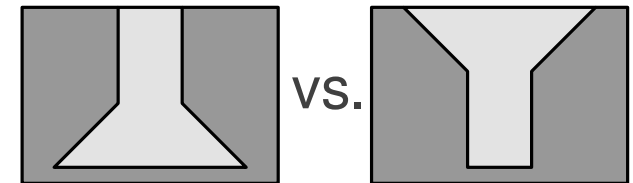
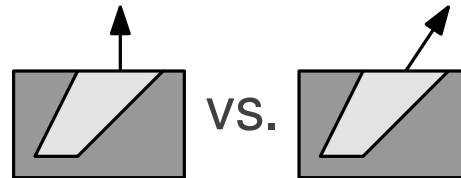
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



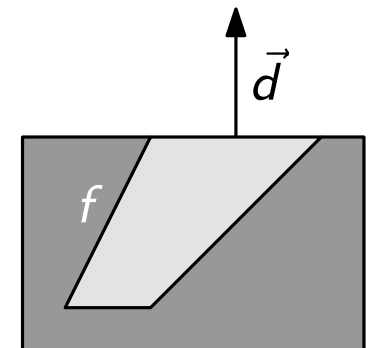
## Welche Entscheidungen haben wir zu treffen?

- die Wahl der oberen Facette
- die Richtung der Translation



## Vereinfachte Situation

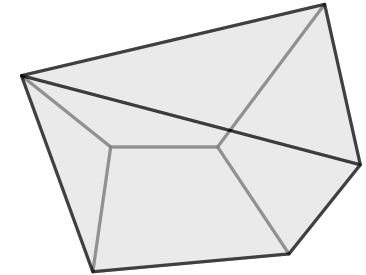
- obere Facette schon ausgewählt  $\Rightarrow$  Gussform steht fest
- Richtung der Translation:  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- betrachte eine gewöhnliche Facette  $f$  (nicht die obere)



# Erste Beobachtungen

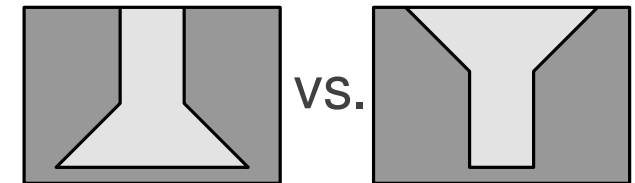
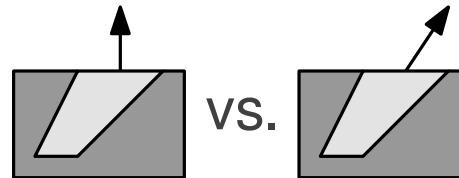
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



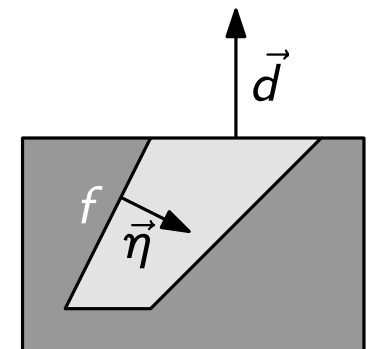
## Welche Entscheidungen haben wir zu treffen?

- die Wahl der oberen Facette
- die Richtung der Translation



## Vereinfachte Situation

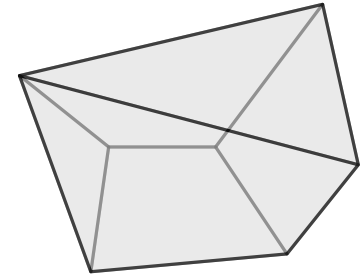
- obere Facette schon ausgewählt  $\Rightarrow$  Gussform steht fest
- Richtung der Translation:  $\vec{d} = (d_x \ d_y \ d_z)^T \in \mathbb{R}^3$
- betrachte eine gewöhnliche Facette  $f$  (nicht die obere)
- sei  $\vec{\eta}$  die innere Normale



# Erste Beobachtungen

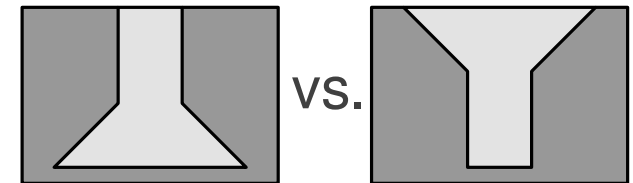
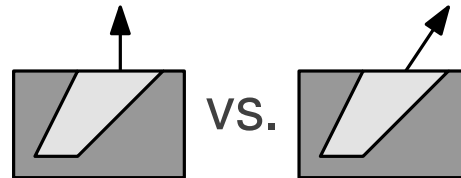
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



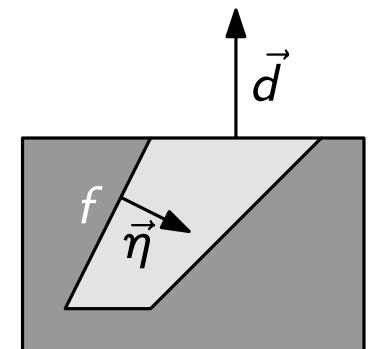
## Welche Entscheidungen haben wir zu treffen?

- die Wahl der oberen Facette
- die Richtung der Translation



## Vereinfachte Situation

- obere Facette schon ausgewählt  $\Rightarrow$  Gussform steht fest
- Richtung der Translation:  $\vec{d} = (d_x \ d_y \ d_z)^T \in \mathbb{R}^3$
- betrachte eine gewöhnliche Facette  $f$  (nicht die obere)
- sei  $\vec{\eta}$  die innere Normale
- Problem: Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist größer  $90^\circ$

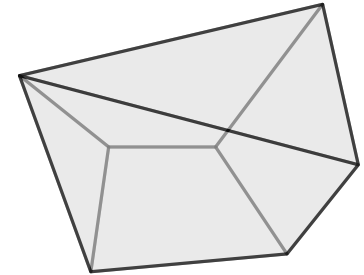




# Erste Beobachtungen

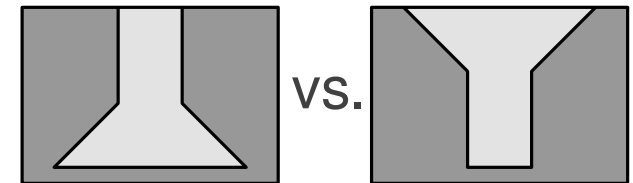
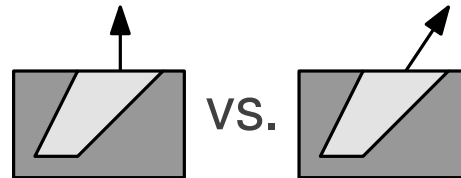
## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



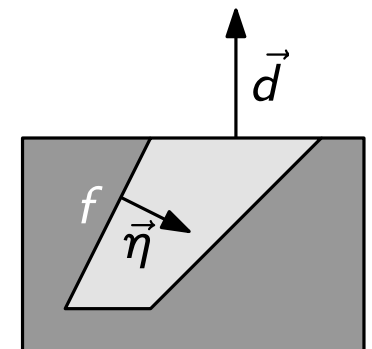
## Welche Entscheidungen haben wir zu treffen?

- die Wahl der oberen Facette
- die Richtung der Translation



## Vereinfachte Situation

- obere Facette schon ausgewählt  $\Rightarrow$  Gussform steht fest
- Richtung der Translation:  $\vec{d} = (d_x \ d_y \ d_z)^T \in \mathbb{R}^3$
- betrachte eine gewöhnliche Facette  $f$  (nicht die obere)
- sei  $\vec{\eta}$  die innere Normale
- Problem: Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist größer  $90^\circ$



## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

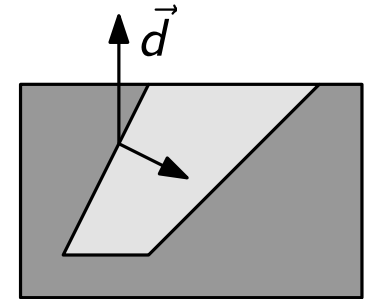
# Gute und schlechte Translationen

## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

## Beweis

- Winkel  $> 90^\circ \Rightarrow$  kann nicht entfernt werden
  
- Winkel  $\leq 90^\circ \Rightarrow$  kann entfernt werden



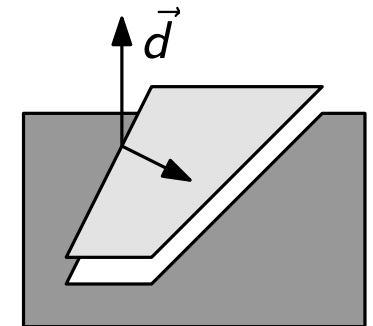
# Gute und schlechte Translationen

## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

## Beweis

- Winkel  $> 90^\circ \Rightarrow$  kann nicht entfernt werden
  - scheitert schon zu Beginn der Translation
- Winkel  $\leq 90^\circ \Rightarrow$  kann entfernt werden



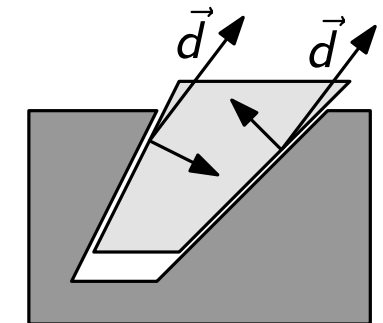
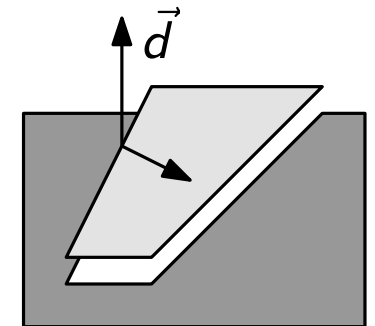
# Gute und schlechte Translationen

## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

## Beweis

- Winkel  $> 90^\circ \Rightarrow$  kann nicht entfernt werden
  - scheitert schon zu Beginn der Translation
- Winkel  $\leq 90^\circ \Rightarrow$  kann entfernt werden
  - zu Beginn geht erstmal alles gut



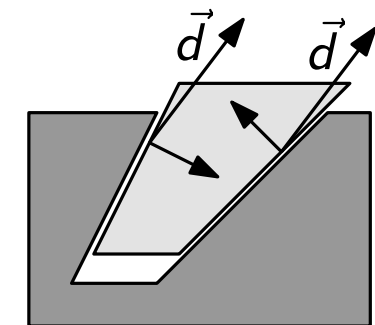
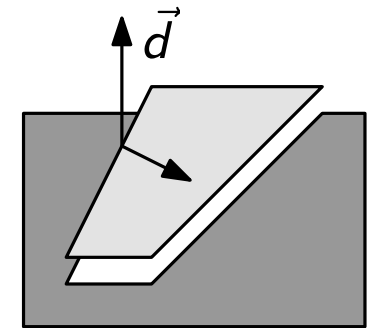
# Gute und schlechte Translationen

## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

## Beweis

- Winkel  $> 90^\circ \Rightarrow$  kann nicht entfernt werden
  - scheitert schon zu Beginn der Translation
- Winkel  $\leq 90^\circ \Rightarrow$  kann entfernt werden
  - zu Beginn geht erstmal alles gut
  - Kann das Polyeder später nochmal mit der Gussform kollidieren?



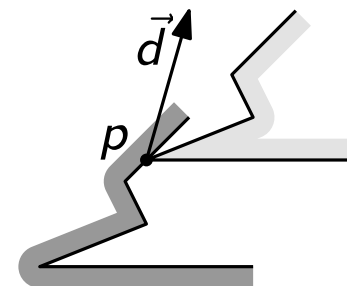
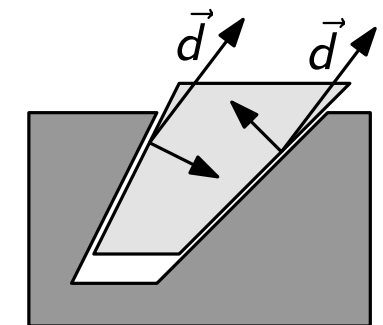
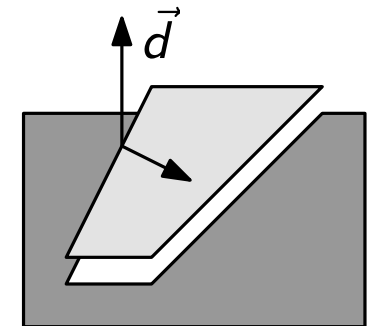
# Gute und schlechte Translationen

## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

## Beweis

- Winkel  $> 90^\circ \Rightarrow$  kann nicht entfernt werden
  - scheitert schon zu Beginn der Translation
- Winkel  $\leq 90^\circ \Rightarrow$  kann entfernt werden
  - zu Beginn geht erstmal alles gut
  - Kann das Polyeder später nochmal mit der Gussform kollidieren?
  - sei  $p$  der erste Punkt von  $P$ , der mit der Form kollidiert



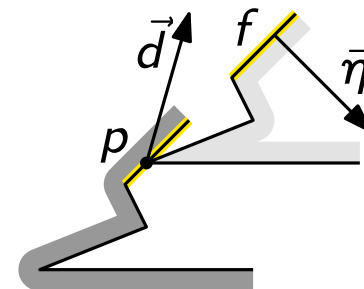
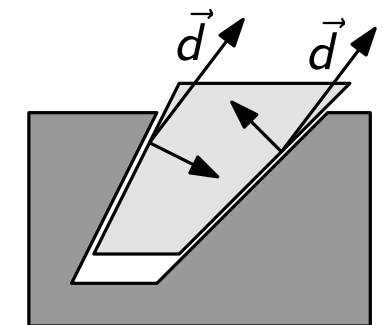
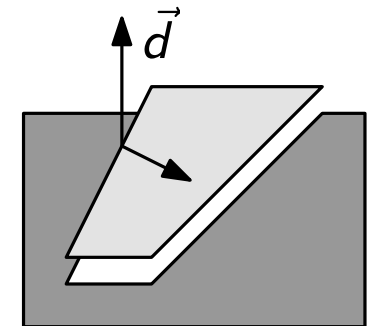
# Gute und schlechte Translationen

## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

## Beweis

- Winkel  $> 90^\circ \Rightarrow$  kann nicht entfernt werden
  - scheitert schon zu Beginn der Translation
- Winkel  $\leq 90^\circ \Rightarrow$  kann entfernt werden
  - zu Beginn geht erstmal alles gut
  - Kann das Polyeder später nochmal mit der Gussform kollidieren?
  - sei  $p$  der erste Punkt von  $P$ , der mit der Form kollidiert
  - sei  $f$  die zugehörige Facette von  $P$  mit innerer Normalen  $\vec{\eta}$



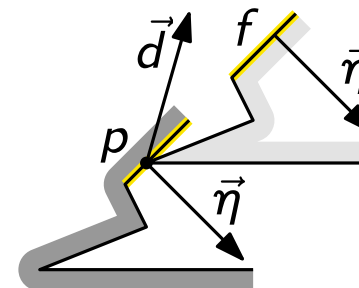
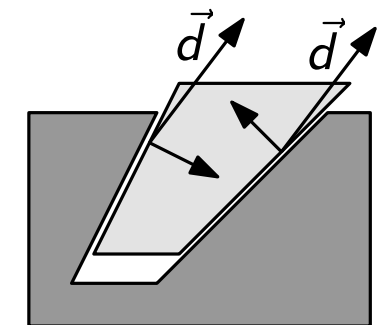
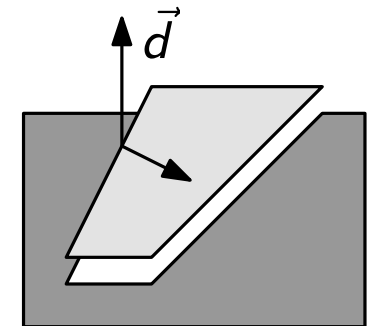
# Gute und schlechte Translationen

## Lemma

$P$  kann genau dann in Richtung  $\vec{d}$  entfernt werden, wenn der Winkel zwischen  $\vec{d}$  und der inneren Normalen jeder gewöhnlichen Facette  $\leq 90^\circ$  ist.

## Beweis

- Winkel  $> 90^\circ \Rightarrow$  kann nicht entfernt werden
  - scheitert schon zu Beginn der Translation
- Winkel  $\leq 90^\circ \Rightarrow$  kann entfernt werden
  - zu Beginn geht erstmal alles gut
  - Kann das Polyeder später nochmal mit der Gussform kollidieren?
  - sei  $p$  der erste Punkt von  $P$ , der mit der Form kollidiert
  - sei  $f$  die zugehörige Facette von  $P$  mit innerer Normalen  $\vec{\eta}$
  - Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$ :  $> 90^\circ$

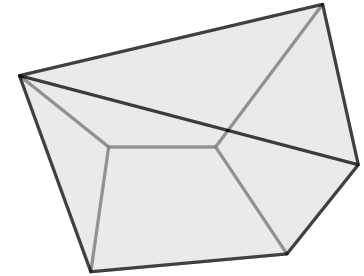




# Was wollen wir eigentlich?

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



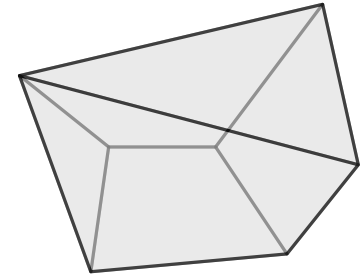
## Ziel

- wähle obere Facette für  $P$  und eine Richtung  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- sodass für jede innere Normale  $\vec{\eta}$  einer gewöhnlichen Facette gilt:  
Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist maximal  $90^\circ$

# Was wollen wir eigentlich?

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



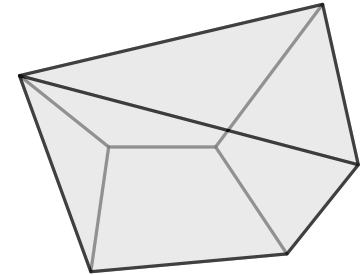
## Ziel

- wähle obere Facette für  $P$  und eine Richtung  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- sodass für jede innere Normale  $\vec{\eta}$  einer gewöhnlichen Facette gilt:  
Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist maximal  $90^\circ$
- beachte: wir können  $d_z = 1$  annehmen **Warum?**

# Was wollen wir eigentlich?

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Ziel

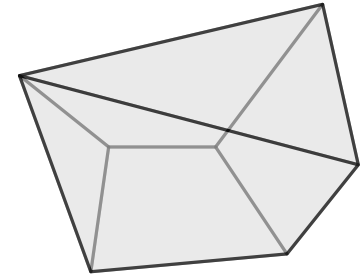
- wähle obere Facette für  $P$  und eine Richtung  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- sodass für jede innere Normale  $\vec{\eta}$  einer gewöhnlichen Facette gilt:  
Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist maximal  $90^\circ$
- beachte: wir können  $d_z = 1$  annehmen **Warum?**

**Erinnerung (Skalarprodukt):** Winkel zwischen  $\vec{d}$  und  $\vec{\eta} \leq 90^\circ \Leftrightarrow \vec{d} \cdot \vec{\eta} \geq 0$

# Was wollen wir eigentlich?

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Ziel

- wähle obere Facette für  $P$  und eine Richtung  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- sodass für jede innere Normale  $\vec{\eta}$  einer gewöhnlichen Facette gilt:  
Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist maximal  $90^\circ$
- beachte: wir können  $d_z = 1$  annehmen **Warum?**

**Erinnerung (Skalarprodukt):** Winkel zwischen  $\vec{d}$  und  $\vec{\eta} \leq 90^\circ \Leftrightarrow \vec{d} \cdot \vec{\eta} \geq 0$

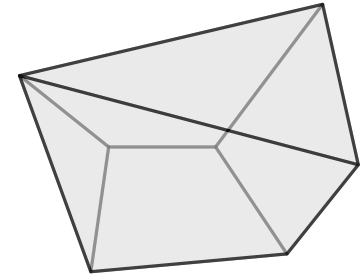
## Neuformulierung des Problems (für feste obere Facette)

- finde  $d_x$  und  $d_y$

# Was wollen wir eigentlich?

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Ziel

- wähle obere Facette für  $P$  und eine Richtung  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- sodass für jede innere Normale  $\vec{\eta}$  einer gewöhnlichen Facette gilt:  
Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist maximal  $90^\circ$
- beachte: wir können  $d_z = 1$  annehmen **Warum?**

**Erinnerung (Skalarprodukt):** Winkel zwischen  $\vec{d}$  und  $\vec{\eta} \leq 90^\circ \Leftrightarrow \vec{d} \cdot \vec{\eta} \geq 0$

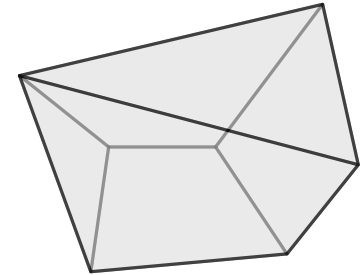
## Neuformulierung des Problems (für feste obere Facette)

- finde  $d_x$  und  $d_y$
- sodass für jede gewöhnliche Facette gilt:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$

# Was wollen wir eigentlich?

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Ziel

- wähle obere Facette für  $P$  und eine Richtung  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- sodass für jede innere Normale  $\vec{\eta}$  einer gewöhnlichen Facette gilt:  
Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist maximal  $90^\circ$
- beachte: wir können  $d_z = 1$  annehmen **Warum?**

**Erinnerung (Skalarprodukt):** Winkel zwischen  $\vec{d}$  und  $\vec{\eta} \leq 90^\circ \Leftrightarrow \vec{d} \cdot \vec{\eta} \geq 0$

## Neuformulierung des Problems (für feste obere Facette)

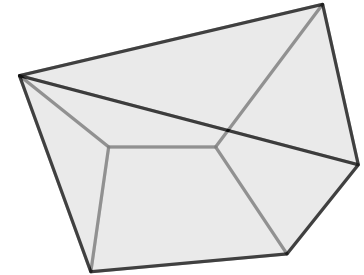
- finde  $d_x$  und  $d_y$
- sodass für jede gewöhnliche Facette gilt:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$

→ das ist ein lineares Programm (LP)

# Was wollen wir eigentlich?

## Problem: Gussform

Gegeben ein Polyeder  $P$ , gibt es eine Gussform für  $P$ , aus der  $P$  mittels einer Translation entfernt werden kann?



## Ziel

- wähle obere Facette für  $P$  und eine Richtung  $\vec{d} = (d_x \quad d_y \quad d_z)^T \in \mathbb{R}^3$
- sodass für jede innere Normale  $\vec{\eta}$  einer gewöhnlichen Facette gilt:  
Winkel zwischen  $\vec{d}$  und  $\vec{\eta}$  ist maximal  $90^\circ$
- beachte: wir können  $d_z = 1$  annehmen **Warum?**

**Erinnerung (Skalarprodukt):** Winkel zwischen  $\vec{d}$  und  $\vec{\eta} \leq 90^\circ \Leftrightarrow \vec{d} \cdot \vec{\eta} \geq 0$

## Neuformulierung des Problems (für feste obere Facette)

- finde  $d_x$  und  $d_y$
- sodass für jede gewöhnliche Facette gilt:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$

→ das ist ein lineares Programm (LP) **Ist die Ungleichung wirklich linear?**

# Lineare Programme



# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen

# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs

# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs
- Theorie: polynomiell lösbar
- Praxis: effiziente Algorithmen (Simplex)

# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs
- Theorie: polynomiell lösbar
- Praxis: effiziente Algorithmen (Simplex)

## Unser konkretes LP

# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs
- Theorie: polynomiell lösbar
- Praxis: effiziente Algorithmen (Simplex)

## Unser konkretes LP

- Variablen  $d_x, d_y \rightarrow$  Dimension ist 2
- Nebenbedingung für jede Facette:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$

# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs
- Theorie: polynomiell lösbar
- Praxis: effiziente Algorithmen (Simplex)

## Unser konkretes LP

- Variablen  $d_x, d_y \rightarrow$  Dimension ist 2
- Nebenbedingung für jede Facette:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$
- keine Zielfunktion

# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs
- Theorie: polynomiell lösbar
- Praxis: effiziente Algorithmen (Simplex)

## Unser konkretes LP

- Variablen  $d_x, d_y \rightarrow$  Dimension ist 2
- Nebenbedingung für jede Facette:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$
- keine Zielfunktion

## Algorithmus für das Gussform-Problem

# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs
- Theorie: polynomiell lösbar
- Praxis: effiziente Algorithmen (Simplex)

## Unser konkretes LP

- Variablen  $d_x, d_y \rightarrow$  Dimension ist 2
- Nebenbedingung für jede Facette:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$
- keine Zielfunktion

## Algorithmus für das Gussform-Problem

- wähle jede der  $n$  Facette einmal als obere Facette
- für jede obere Facette löse ein 2-dim. LP mit  $n - 1$  Nebenbedingungen



# Lineare Programme

## Allgemeines Form eines LPs

maximiere  $c_1x_1 + c_2x_2 + \dots + c_dx_d$

sodass  $a_{1,1}x_1 + \dots + a_{1,d}x_d \leq b_1$

$a_{2,1}x_1 + \dots + a_{2,d}x_d \leq b_2$

$\vdots$

$a_{n,1}x_1 + \dots + a_{n,d}x_d \leq b_n$

- Variablen  $x_1, \dots, x_d$
- $a_{i,j}, b_i, c_i$  sind Konstanten
- eine Zielfunktion
- $n$  Nebenbedingungen
- $d$  ist die Dimension des LPs
- Theorie: polynomiell lösbar
- Praxis: effiziente Algorithmen (Simplex)

## Unser konkretes LP

- Variablen  $d_x, d_y \rightarrow$  Dimension ist 2
- Nebenbedingung für jede Facette:  $\eta_x \cdot d_x + \eta_y \cdot d_y + \eta_z \geq 0$
- keine Zielfunktion

## Algorithmus für das Gussform-Problem

- wähle jede der  $n$  Facette einmal als obere Facette
- für jede obere Facette löse ein 2-dim. LP mit  $n - 1$  Nebenbedingungen

## Ziel im Folgenden

- schnellerer Algo für das Lösen 2-dimensionaler LPs

# Finde die optimale Lösung

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$  (P)

$x_2 \geq 0$  (A)

$x_2 - x_1 \leq 1$  (U)

$x_1 + 6x_2 \leq 15$  (S)

$4x_1 - x_2 \leq 10$  (E)

## 2D LPs

### Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## 2D LPs

### Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

### Geometrische Interpretation

# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

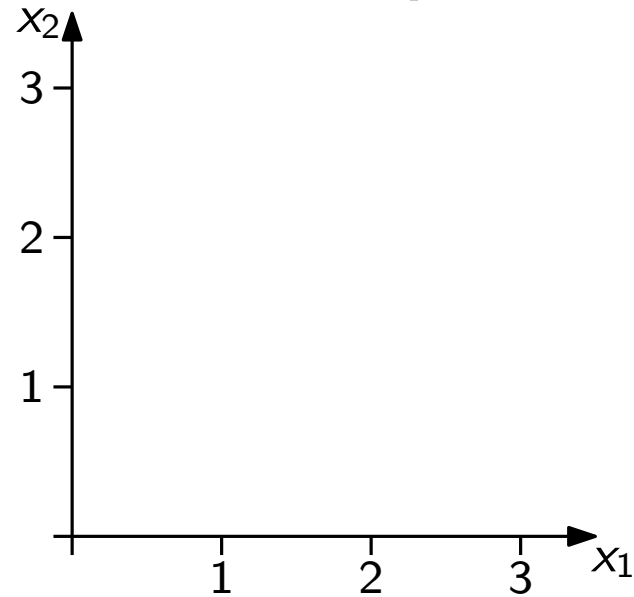
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

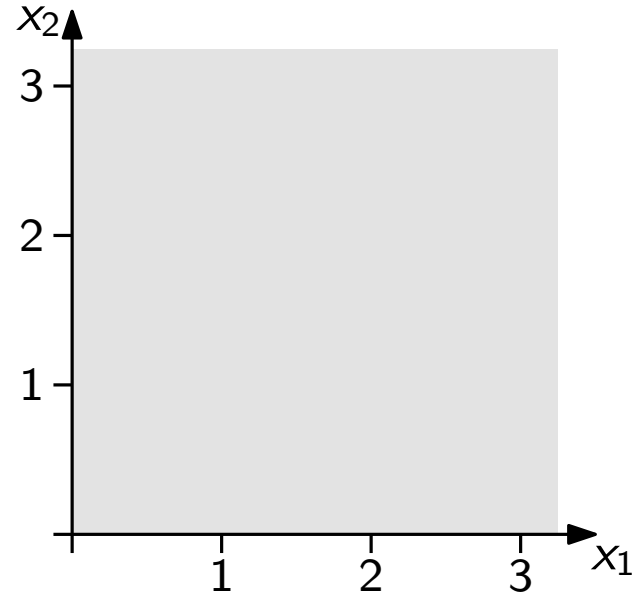
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

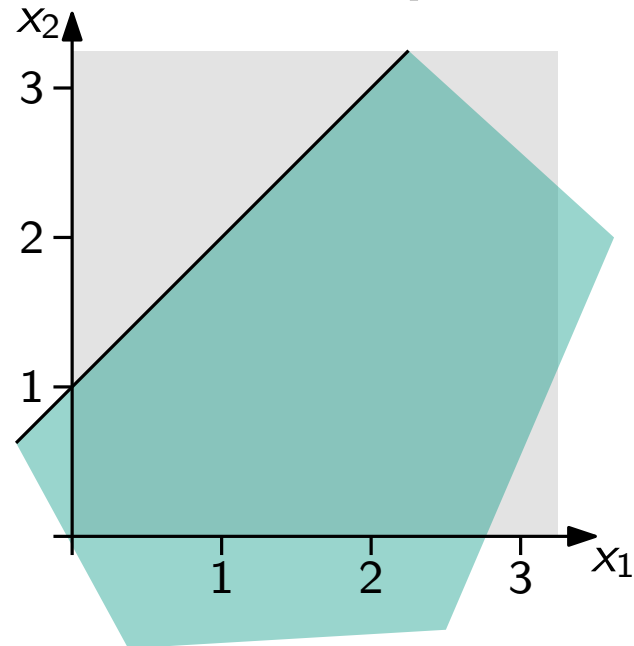
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

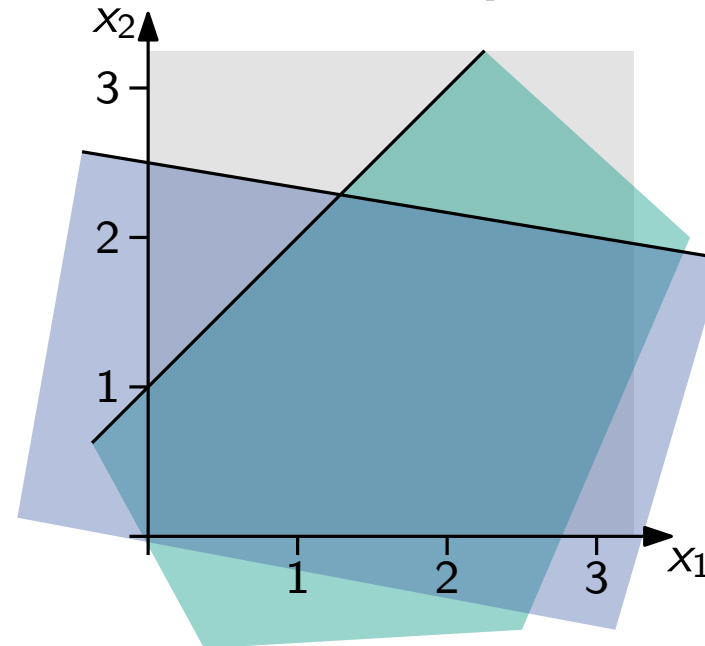
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation





# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

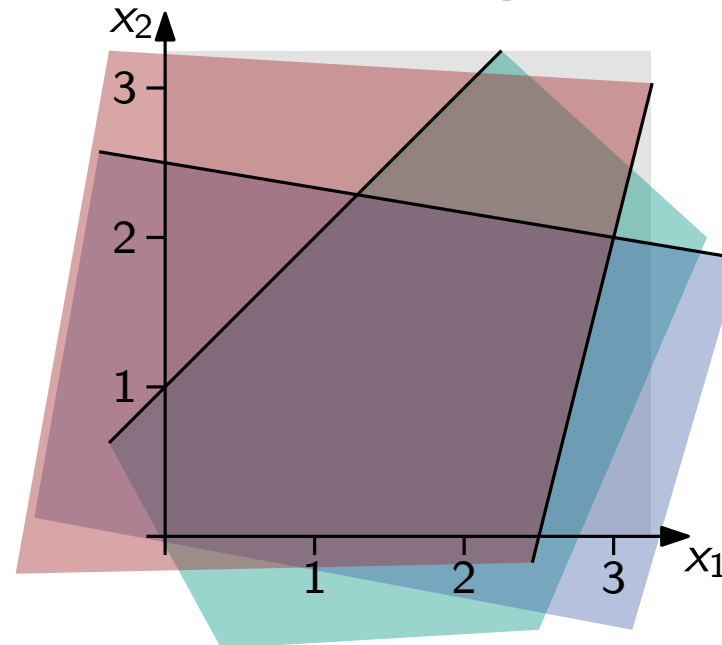
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

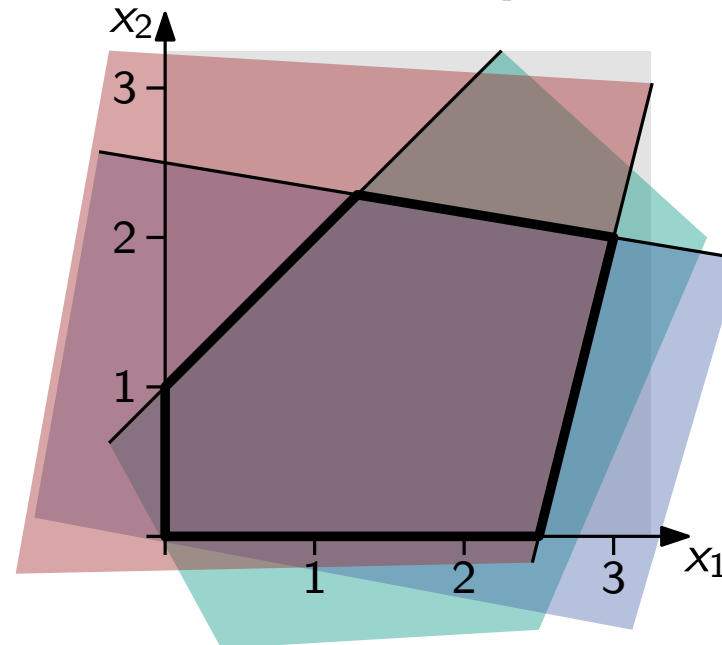
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

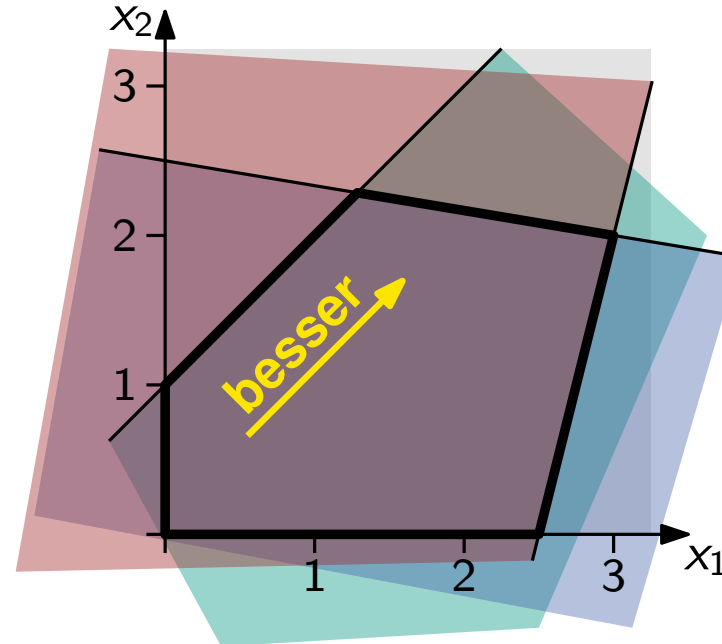
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

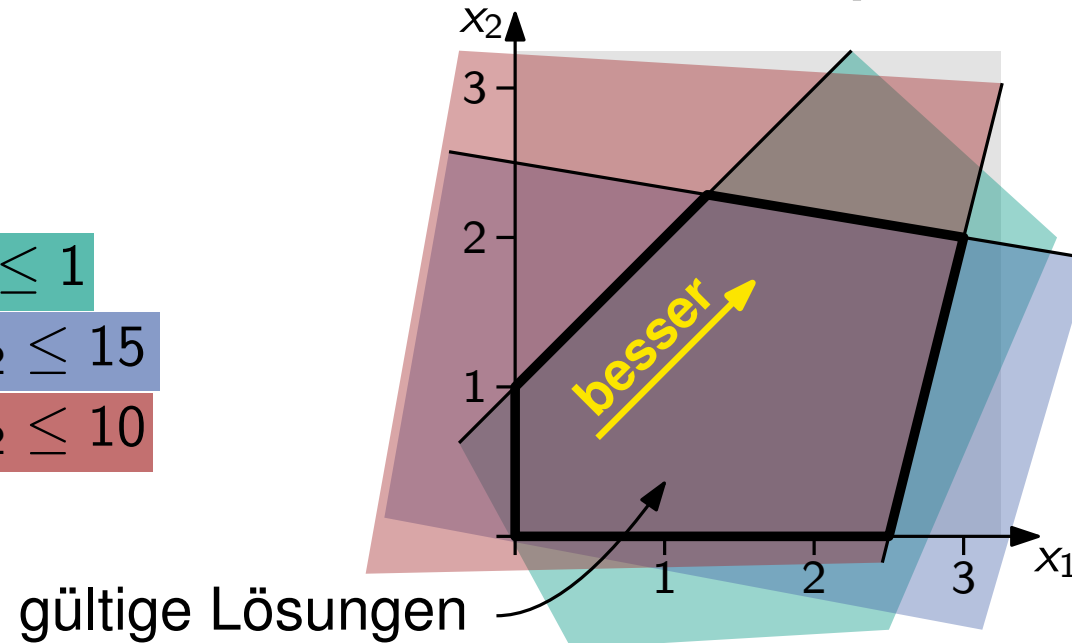
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

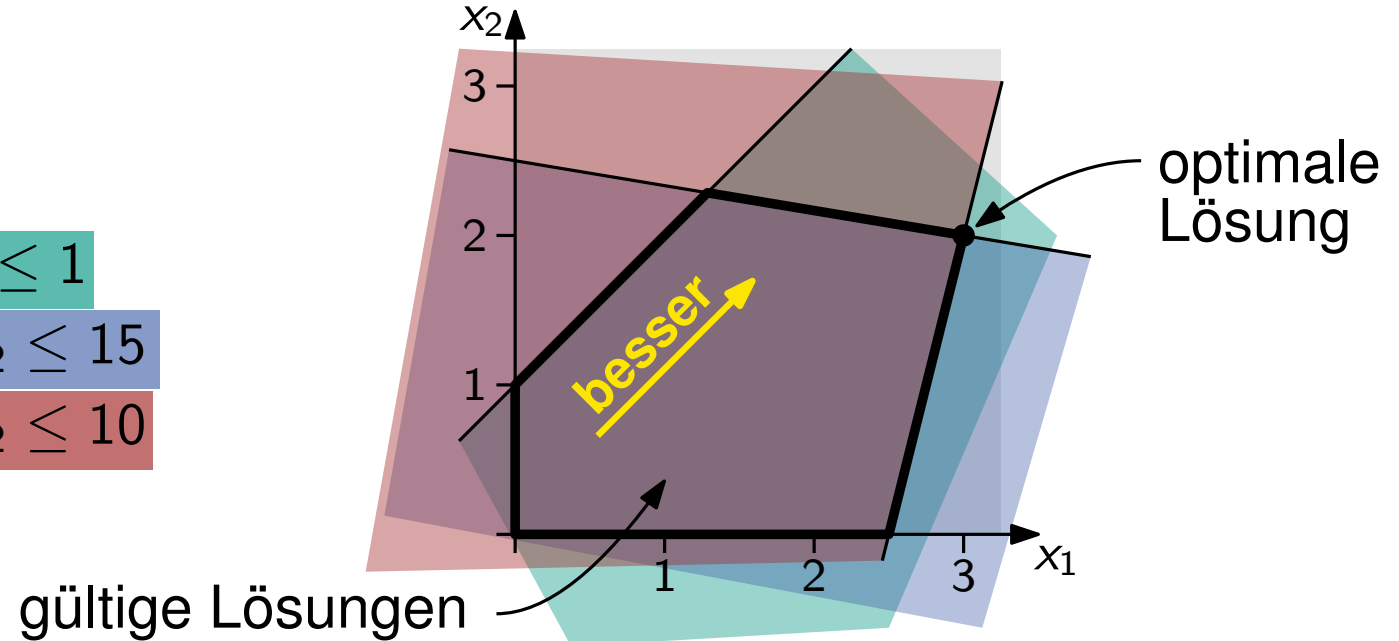
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

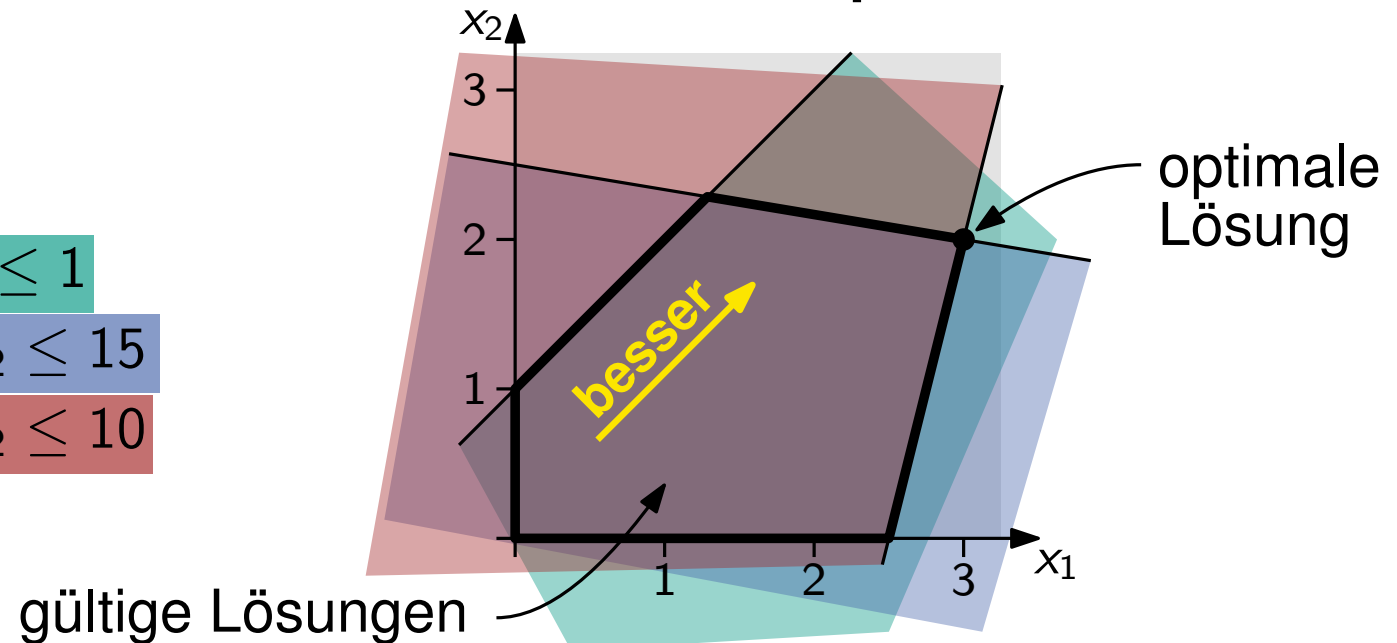
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



## Lösbarkeit

- LP ist **unlösbar (infeasible)**, wenn es keine gültige Lösung gibt

# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

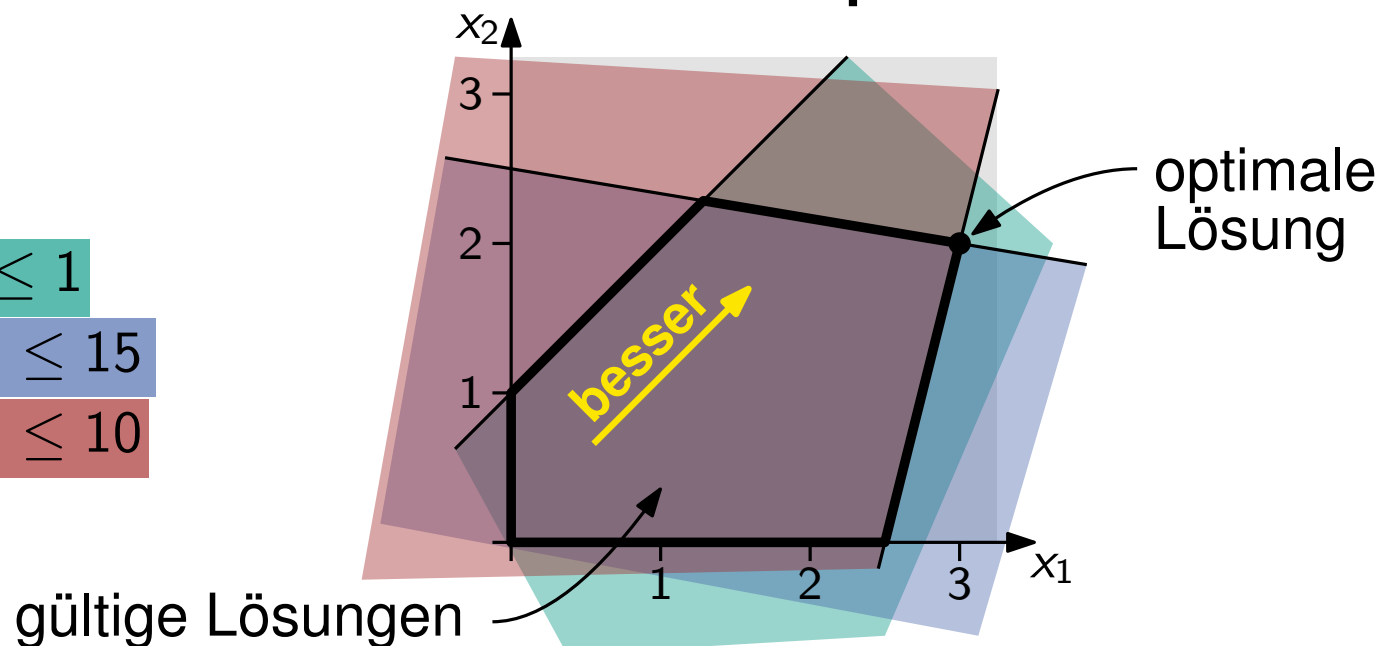
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



## Lösbarkeit

- LP ist **unlösbar (infeasible)**, wenn es keine gültige Lösung gibt
- LP ist **unbeschränkt (unbounded)**, wenn es beliebig gute gültige Lösungen gibt (Optimierungsfunktion wird beliebig groß)

# 2D LPs

## Beispiel

maximiere:  $x_1 + x_2$

sodass:  $x_1 \geq 0$

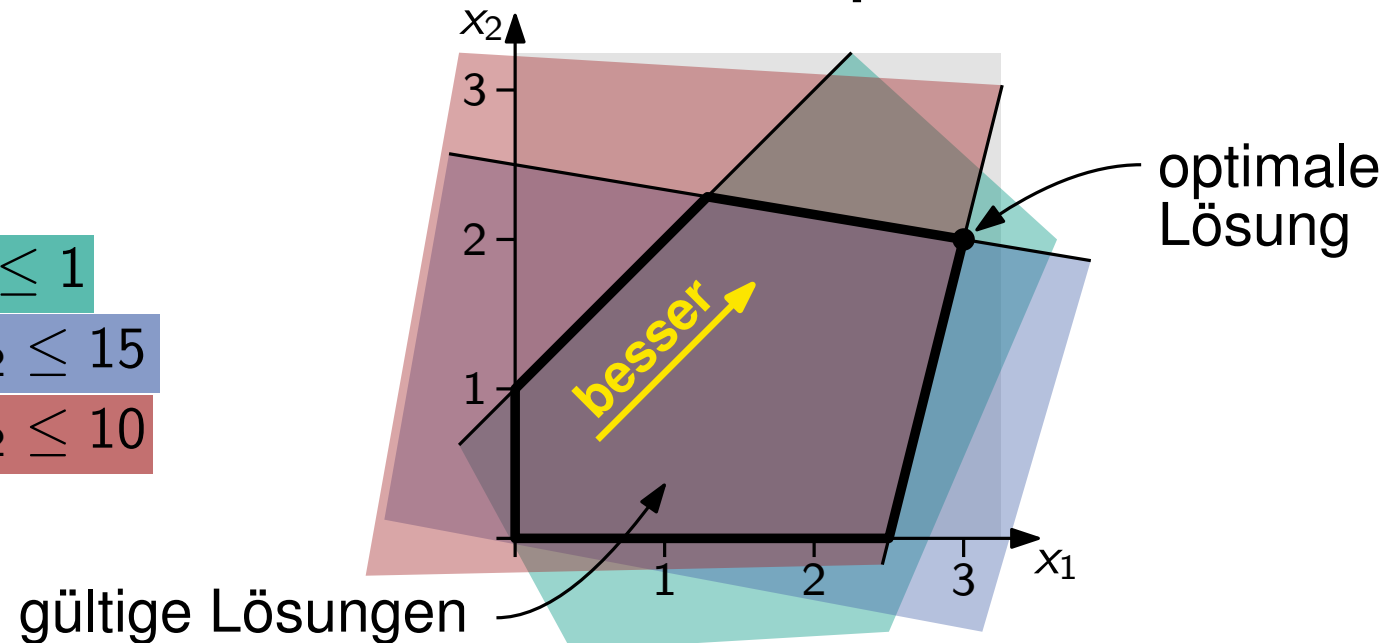
$x_2 \geq 0$

$x_2 - x_1 \leq 1$

$x_1 + 6x_2 \leq 15$

$4x_1 - x_2 \leq 10$

## Geometrische Interpretation



## Lösbarkeit

- LP ist **unlösbar (infeasible)**, wenn es keine gültige Lösung gibt
- LP ist **unbeschränkt (unbounded)**, wenn es beliebig gute gültige Lösungen gibt (Optimierungsfunktion wird beliebig groß)

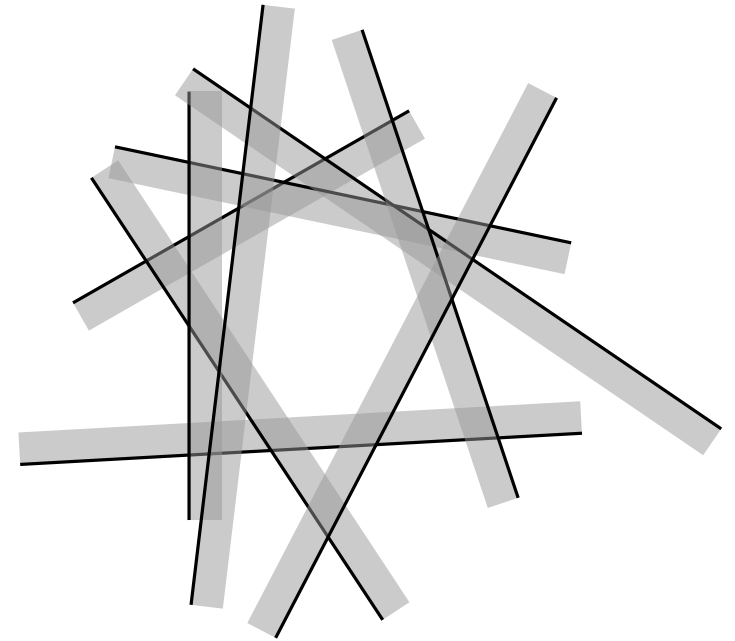
## Problem: Schnitt von Halbebenen

Gegeben  $n$  Halbebenen, berechne den Schnitt dieser Halbebenen.



# Schnitt von Halbebenen

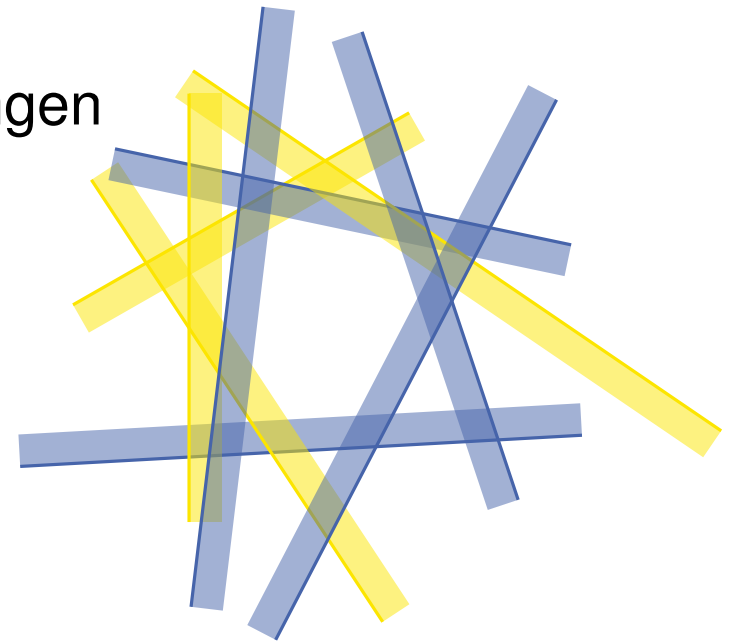
Plan: Teile und herrsche



# Schnitt von Halbebenen

## Plan: Teile und herrsche

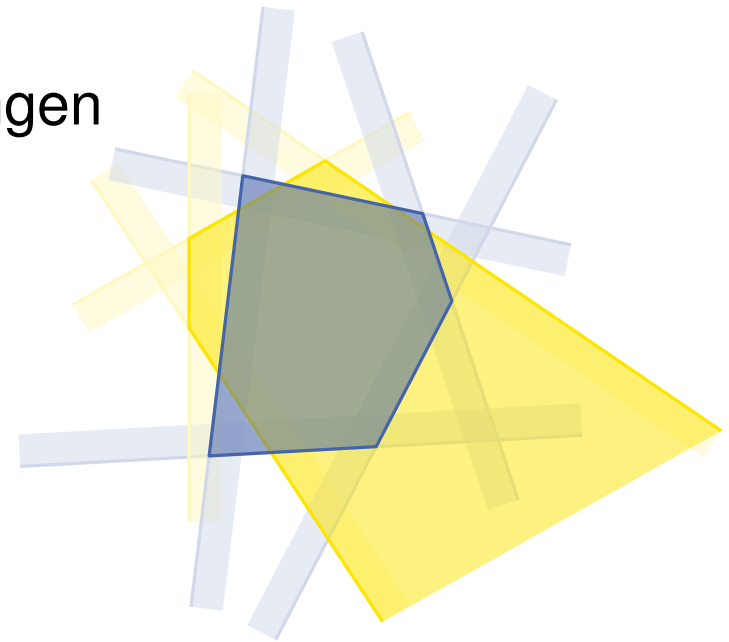
- teile Halbebenen in etwa gleich große Teilmengen



# Schnitt von Halbebenen

## Plan: Teile und herrsche

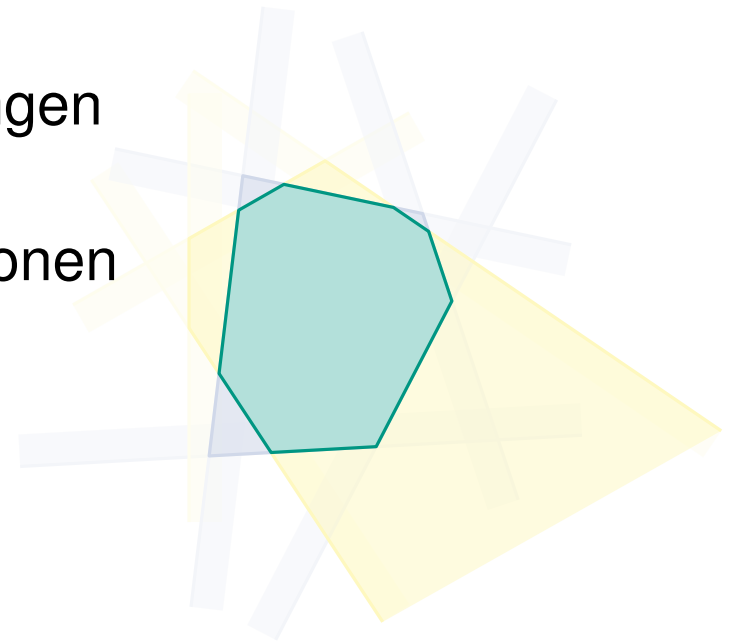
- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen



# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

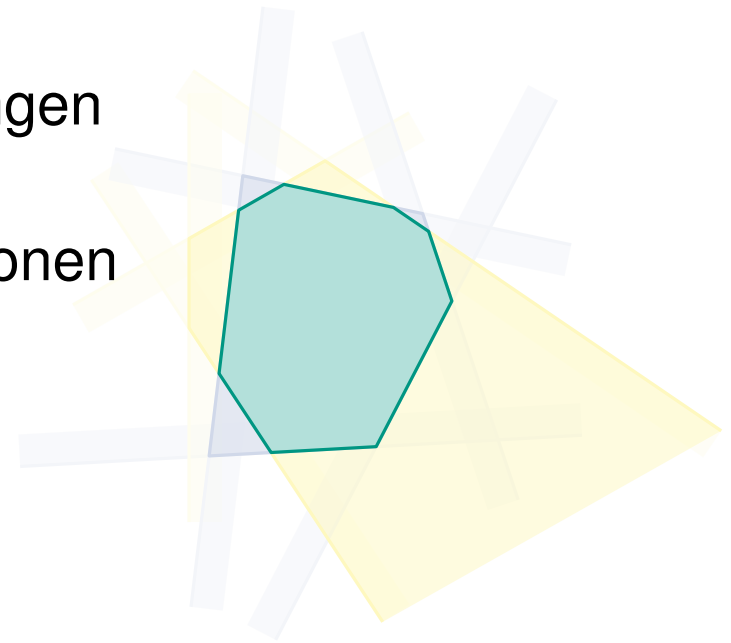


# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts



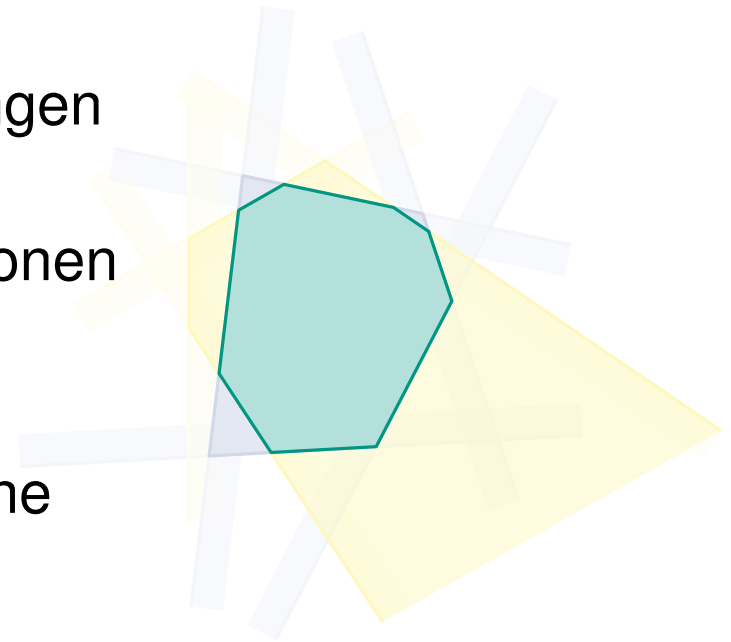
# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone



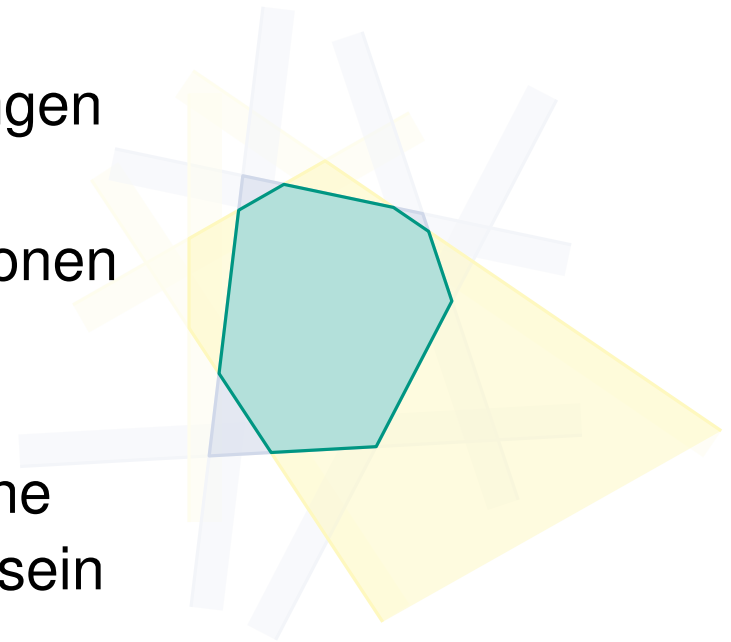
# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein



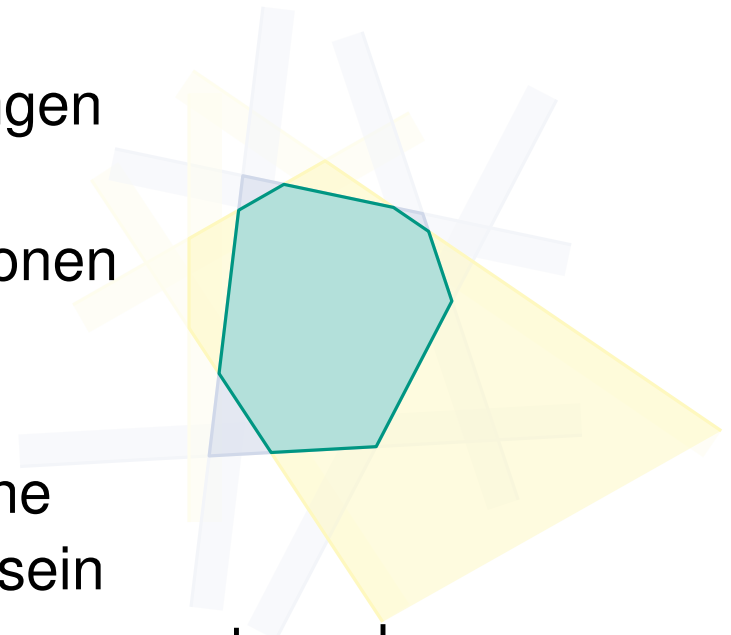
# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein
- Sweep-Line Algorithmus kann entsprechend angepasst werden  
→ Laufzeit  $O(n \log n)$





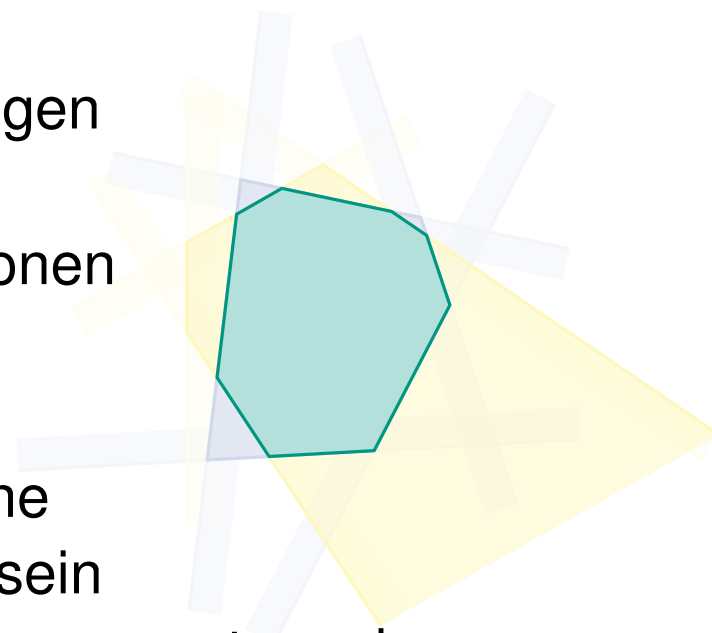
# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein
- Sweep-Line Algorithmus kann entsprechend angepasst werden
  - Laufzeit  $O(n \log n)$  **Warum nicht  $O((n + k) \log n)$ ?**



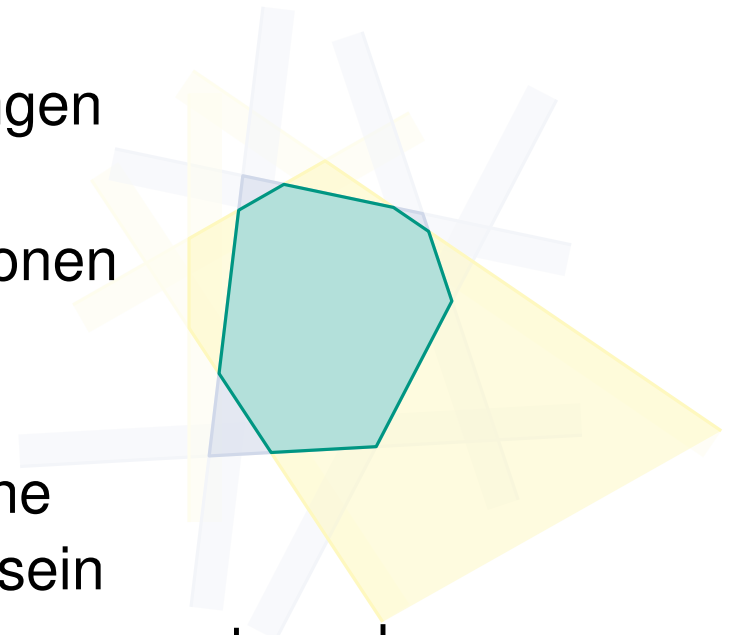
# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein
- Sweep-Line Algorithmus kann entsprechend angepasst werden
  - Laufzeit  $O(n \log n)$  **Warum nicht  $O((n + k) \log n)$ ?**
- Ausnutzung der Konvexität beim Schnitt → Laufzeit  $O(n)$



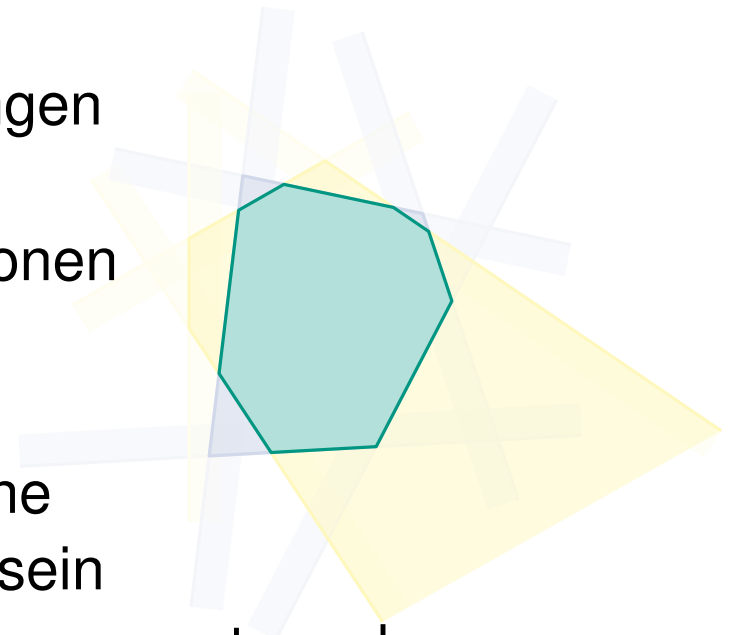
# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein
- Sweep-Line Algorithmus kann entsprechend angepasst werden
  - Laufzeit  $O(n \log n)$  **Warum nicht  $O((n + k) \log n)$ ?**
- Ausnutzung der Konvexität beim Schnitt → Laufzeit  $O(n)$



## Gesamtlaufzeit:

# Schnitt von Halbebenen

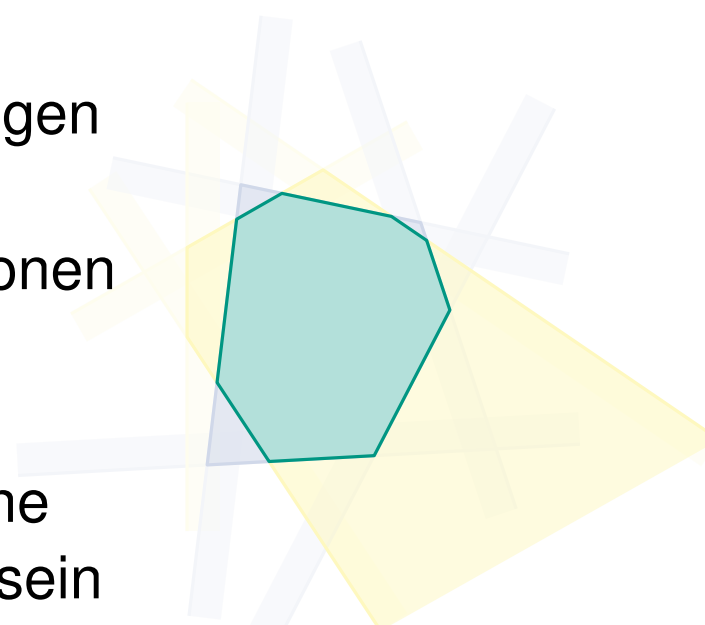
## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein
- Sweep-Line Algorithmus kann entsprechend angepasst werden
  - Laufzeit  $O(n \log n)$  **Warum nicht  $O((n + k) \log n)$ ?**
- Ausnutzung der Konvexität beim Schnitt → Laufzeit  $O(n)$

**Gesamtlaufzeit:**  $T(n) = O(n) + 2T(n/2)$



# Schnitt von Halbebenen

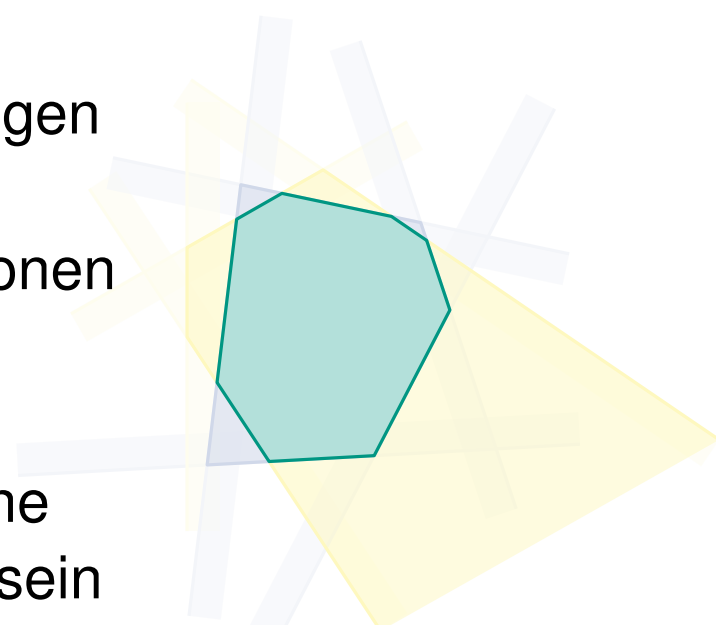
## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein
- Sweep-Line Algorithmus kann entsprechend angepasst werden
  - Laufzeit  $O(n \log n)$  **Warum nicht  $O((n + k) \log n)$ ?**
- Ausnutzung der Konvexität beim Schnitt → Laufzeit  $O(n)$

**Gesamtlaufzeit:**  $T(n) = O(n) + 2T(n/2) \Rightarrow O(n \log n)$



# Schnitt von Halbebenen

## Plan: Teile und herrsche

- teile Halbebenen in etwa gleich große Teilmengen
- berechne Schnitt für jede der Teilmengen
- berechne Schnitt der zwei resultierenden Regionen

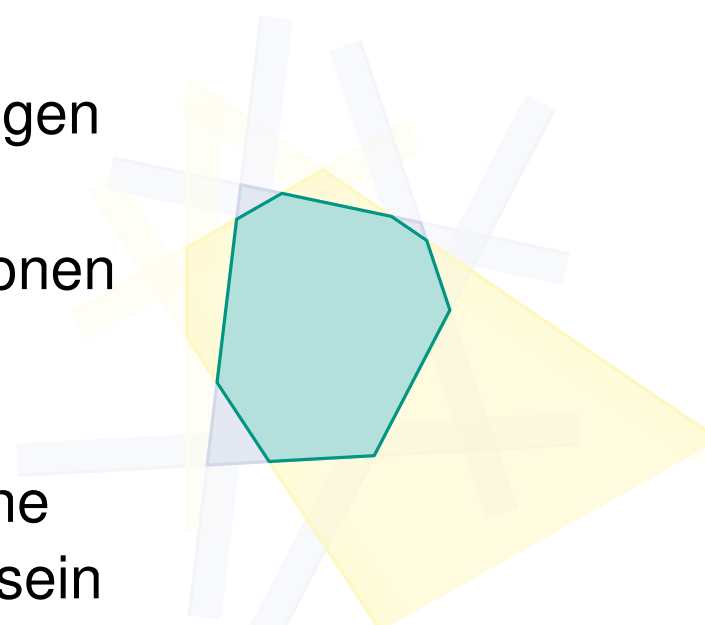
## Berechnung des Schnitts

- im Prinzip der Schnitt zweier konvexer Polygone
- Unterschied: Regionen können unbeschränkt sein
- Sweep-Line Algorithmus kann entsprechend angepasst werden
  - Laufzeit  $O(n \log n)$  **Warum nicht  $O((n + k) \log n)$ ?**
- Ausnutzung der Konvexität beim Schnitt → Laufzeit  $O(n)$

**Gesamtlaufzeit:**  $T(n) = O(n) + 2T(n/2) \Rightarrow O(n \log n)$

## Geht es besser?

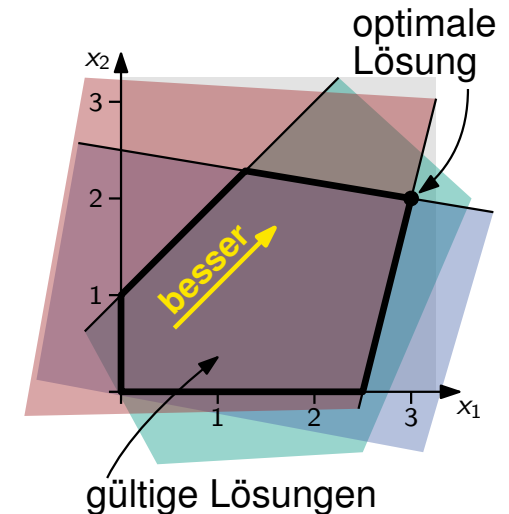
- verwandt zur Berechnung der Konvexen Hülle (via Dualität)
- untere Schranke:  $\Omega(n \log n)$



# Inkrementeller Algorithmus für 2D LPs

## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert



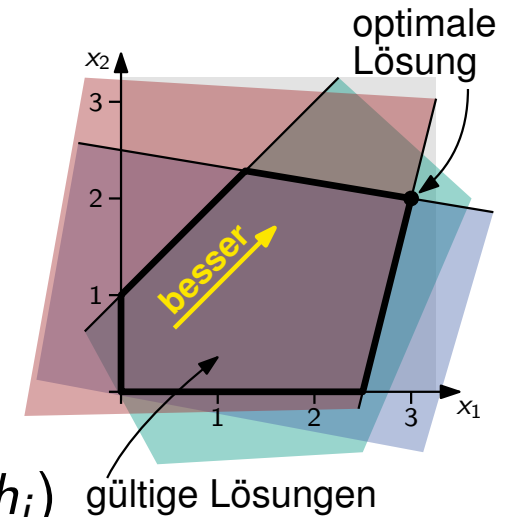
# Inkrementeller Algorithmus für 2D LPs

## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert

## Inkrementeller Ansatz

- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_j = h_1 \cap h_2 \cap \dots \cap h_j$  (gültige Region bzgl.  $h_1, \dots, h_j$ )





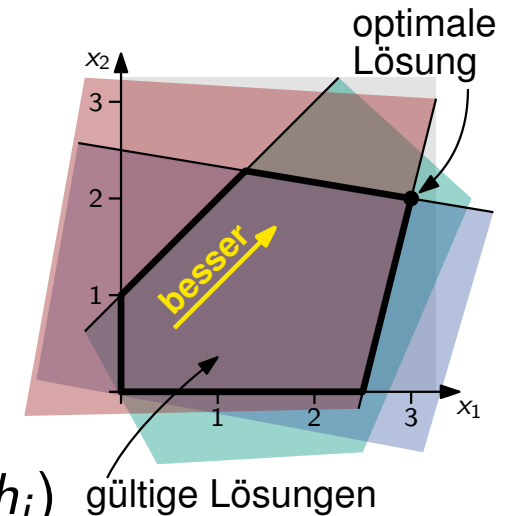
# Inkrementeller Algorithmus für 2D LPs

## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert

## Inkrementeller Ansatz

- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_i = h_1 \cap h_2 \cap \dots \cap h_i$  (gültige Region bzgl.  $h_1, \dots, h_i$ )
- Annahme: wir kennen einen optimalen Punkt  $v_{i-1} \in C_{i-1}$
- Ziel: finde optimalen Punkt  $v_i \in C_i$



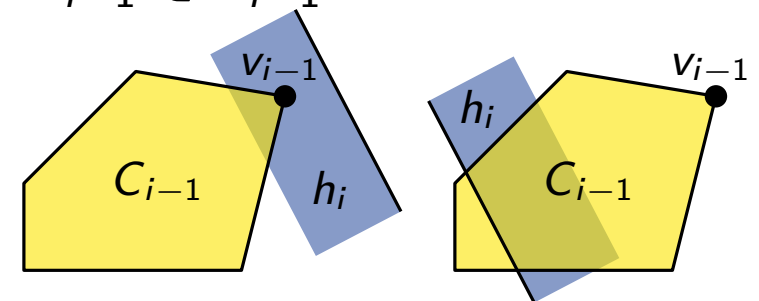
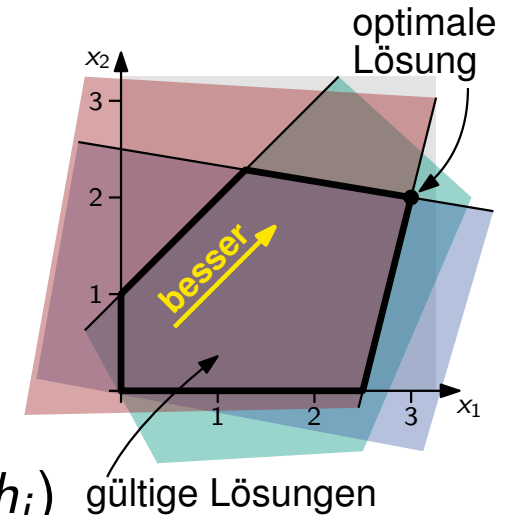
# Inkrementeller Algorithmus für 2D LPs

## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert

## Inkrementeller Ansatz

- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_i = h_1 \cap h_2 \cap \dots \cap h_i$  (gültige Region bzgl.  $h_1, \dots, h_i$ )
- Annahme: wir kennen einen optimalen Punkt  $v_{i-1} \in C_{i-1}$
- Ziel: finde optimalen Punkt  $v_i \in C_i$
- Fall 1:  $v_{i-1} \in C_i$



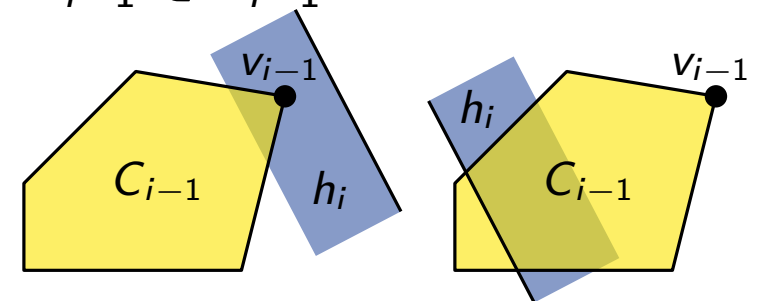
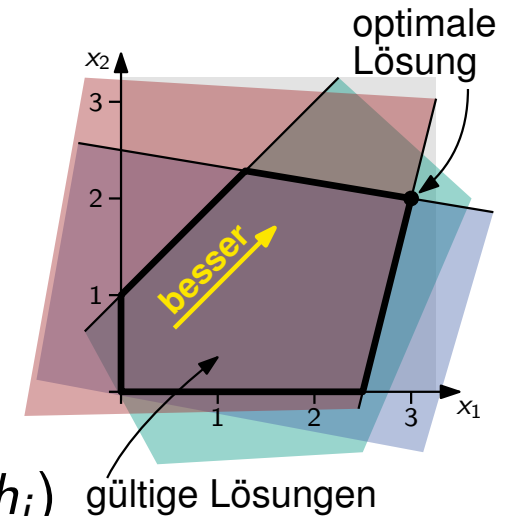
# Inkrementeller Algorithmus für 2D LPs

## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert

## Inkrementeller Ansatz

- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_i = h_1 \cap h_2 \cap \dots \cap h_i$  (gültige Region bzgl.  $h_1, \dots, h_i$ )
- Annahme: wir kennen einen optimalen Punkt  $v_{i-1} \in C_{i-1}$
- Ziel: finde optimalen Punkt  $v_i \in C_i$
- Fall 1:  $v_{i-1} \in C_i$ 
  - $v_{i-1}$  ist auch optimal in  $C_i$  (da  $C_i \subseteq C_{i-1}$ )



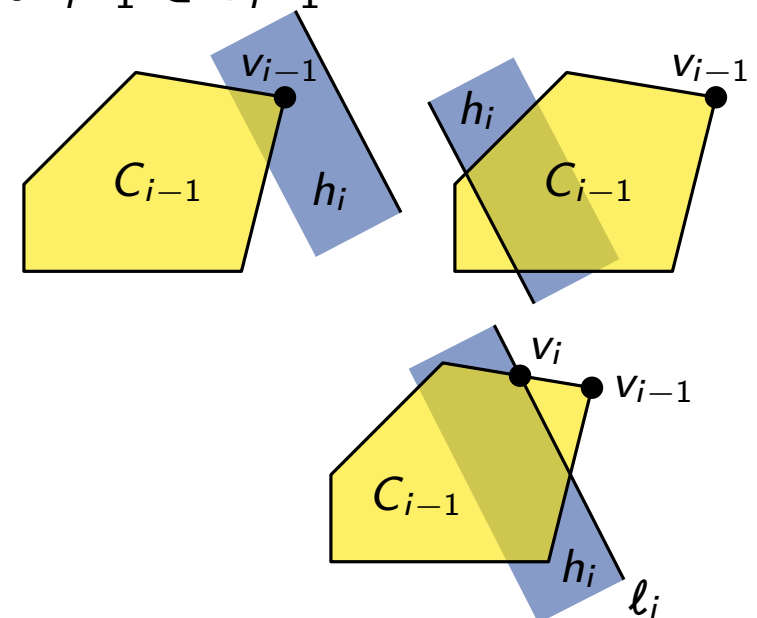
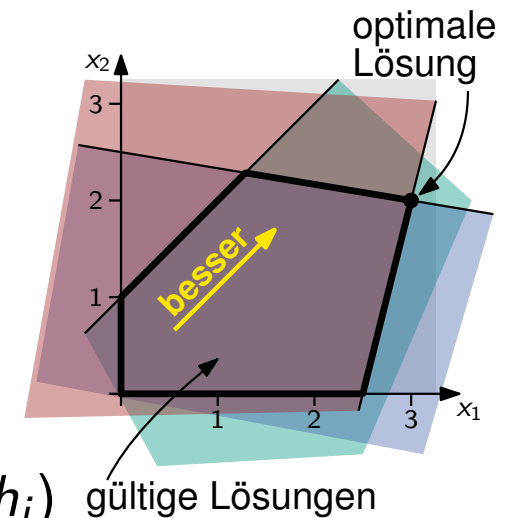
# Inkrementeller Algorithmus für 2D LPs

## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert

## Inkrementeller Ansatz

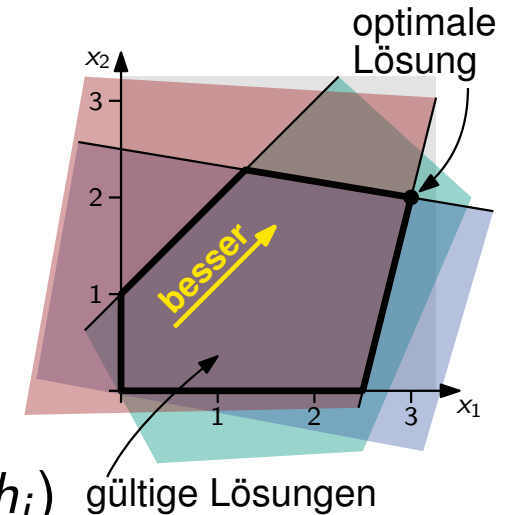
- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_i = h_1 \cap h_2 \cap \dots \cap h_i$  (gültige Region bzgl.  $h_1, \dots, h_i$ )
- Annahme: wir kennen einen optimalen Punkt  $v_{i-1} \in C_{i-1}$
- Ziel: finde optimalen Punkt  $v_i \in C_i$
- Fall 1:  $v_{i-1} \in C_i$ 
  - $v_{i-1}$  ist auch optimal in  $C_i$  (da  $C_i \subseteq C_{i-1}$ )
- Fall 2:  $v_{i-1} \notin C_i$



# Inkrementeller Algorithmus für 2D LPs

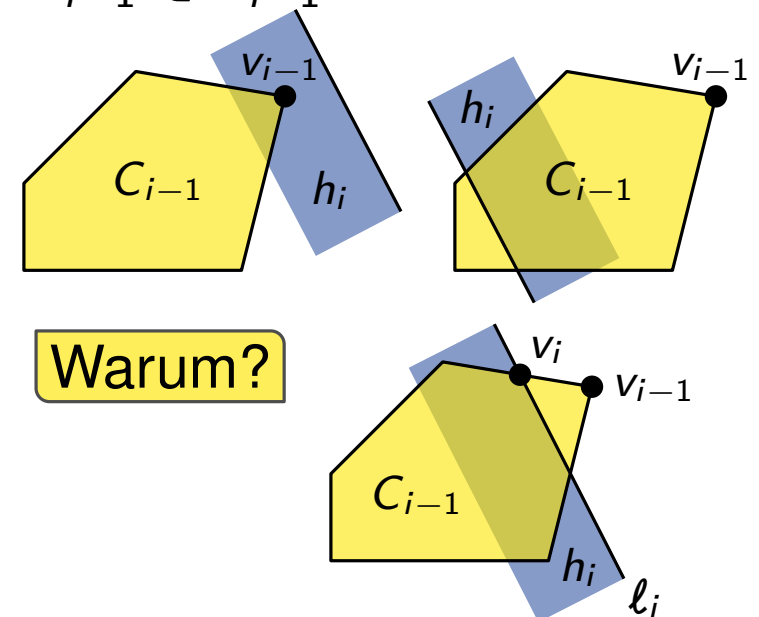
## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert



## Inkrementeller Ansatz

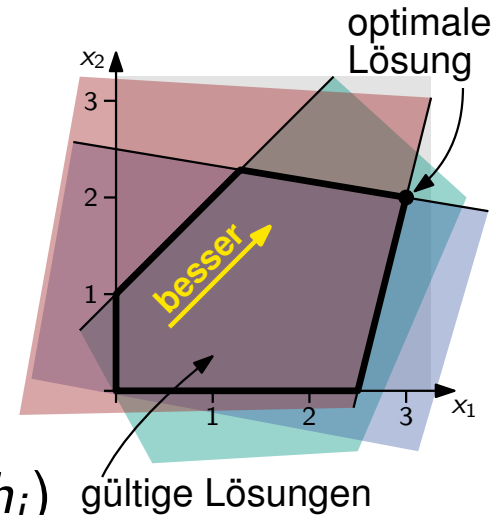
- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_i = h_1 \cap h_2 \cap \dots \cap h_i$  (gültige Region bzgl.  $h_1, \dots, h_i$ )
- Annahme: wir kennen einen optimalen Punkt  $v_{i-1} \in C_{i-1}$
- Ziel: finde optimalen Punkt  $v_i \in C_i$
- Fall 1:  $v_{i-1} \in C_i$ 
  - $v_{i-1}$  ist auch optimal in  $C_i$  (da  $C_i \subseteq C_{i-1}$ )
- Fall 2:  $v_{i-1} \notin C_i$ 
  - $v_i$  liegt auf der Begrenzungslinie  $\ell_i$  von  $h_i$



# Inkrementeller Algorithmus für 2D LPs

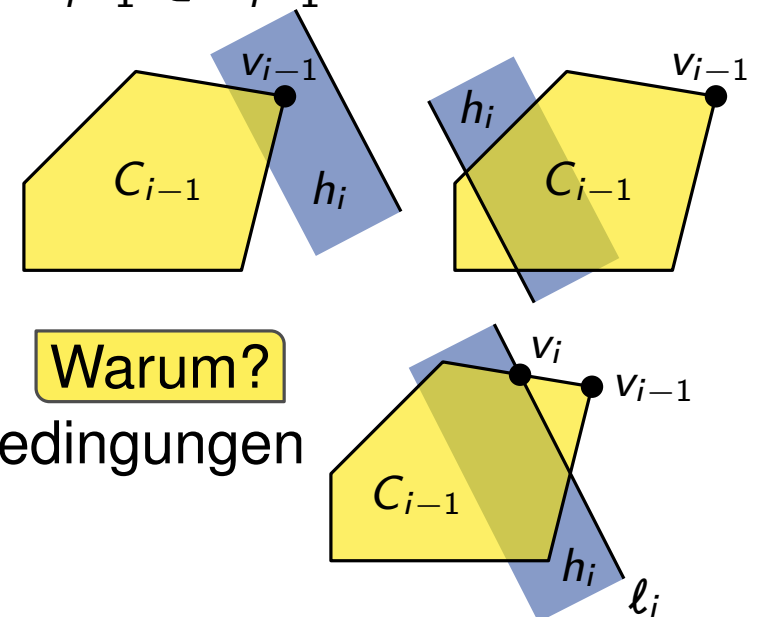
## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert



## Inkrementeller Ansatz

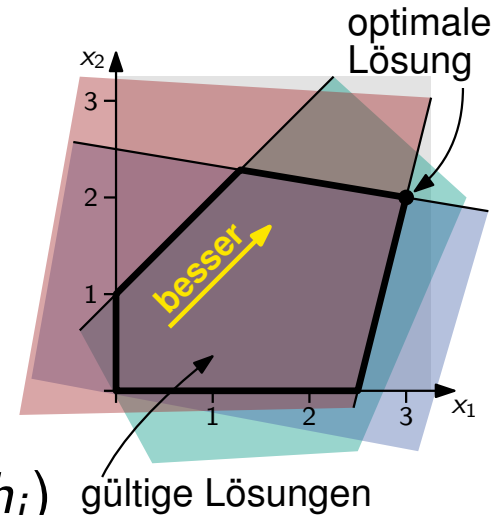
- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_i = h_1 \cap h_2 \cap \dots \cap h_i$  (gültige Region bzgl.  $h_1, \dots, h_i$ )
- Annahme: wir kennen einen optimalen Punkt  $v_{i-1} \in C_{i-1}$
- Ziel: finde optimalen Punkt  $v_i \in C_i$
- Fall 1:  $v_{i-1} \in C_i$ 
  - $v_{i-1}$  ist auch optimal in  $C_i$  (da  $C_i \subseteq C_{i-1}$ )
- Fall 2:  $v_{i-1} \notin C_i$ 
  - $v_i$  liegt auf der Begrenzungslinie  $\ell_i$  von  $h_i$  **Warum?**
  - das ist im Prinzip ein 1D LP mit  $i$  Nebenbedingungen



# Inkrementeller Algorithmus für 2D LPs

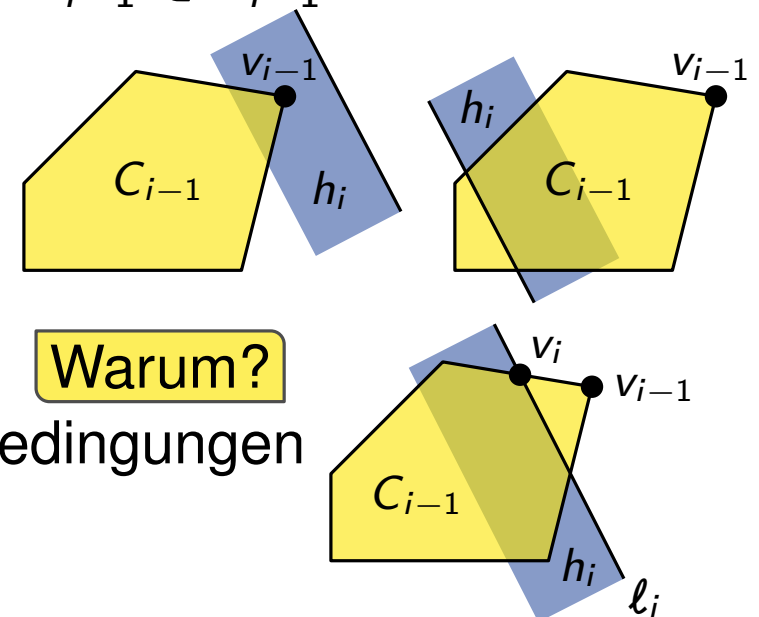
## Beobachtung

- wir müssen die gültige Region nicht explizit berechnen
- es genügt ein gültiger Punkt, der die Zielfunktion maximiert



## Inkrementeller Ansatz

- seien  $h_1, \dots, h_n$  die Halbebenen (Nebenbedingungen)
- sei  $C_i = h_1 \cap h_2 \cap \dots \cap h_i$  (gültige Region bzgl.  $h_1, \dots, h_i$ )
- Annahme: wir kennen einen optimalen Punkt  $v_{i-1} \in C_{i-1}$
- Ziel: finde optimalen Punkt  $v_i \in C_i$
- Fall 1:  $v_{i-1} \in C_i$ 
  - $v_{i-1}$  ist auch optimal in  $C_i$  (da  $C_i \subseteq C_{i-1}$ )
- Fall 2:  $v_{i-1} \notin C_i$ 
  - $v_i$  liegt auf der Begrenzungslinie  $\ell_i$  von  $h_i$  **Warum?**
  - das ist im Prinzip ein 1D LP mit  $i$  Nebenbedingungen
  - können wir in  $O(i)$  Zeit lösen **Wie?**



# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?



# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht

# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt → nicht nur zu Beginn problematisch

# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt → nicht nur zu Beginn problematisch
- Lösung: siehe Übung

# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

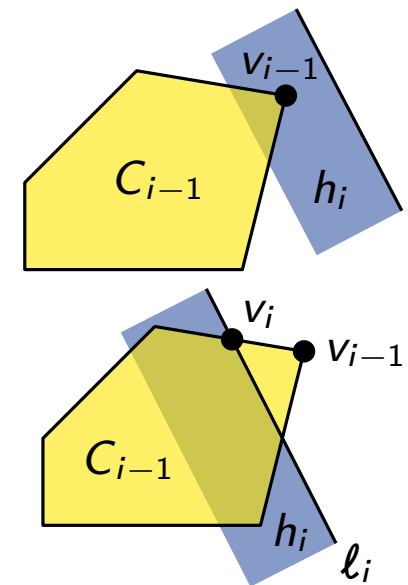
- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt  $\rightarrow$  nicht nur zu Beginn problematisch
- Lösung: siehe Übung

## Was interessiert uns ein $O(n^2)$ Algorithmus?

- Fall  $v_{i-1} \in C_i$  ist billig (setze einfach  $v_i = v_{i-1}$ )
- Fall  $v_{i-1} \notin C_i$  ist teuer ( $O(i)$  zur Berechnung von  $v_i$ )



# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

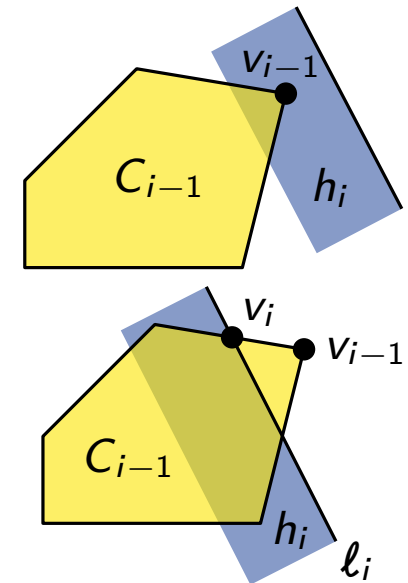
- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt → nicht nur zu Beginn problematisch
- Lösung: siehe Übung

## Was interessiert uns ein $O(n^2)$ Algorithmus?

- Fall  $v_{i-1} \in C_i$  ist billig (setze einfach  $v_i = v_{i-1}$ )
- Fall  $v_{i-1} \notin C_i$  ist teuer ( $O(i)$  zur Berechnung von  $v_i$ )
- Hoffnung:  $v_{i-1} \notin C_i$  tritt nur selten ein



# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

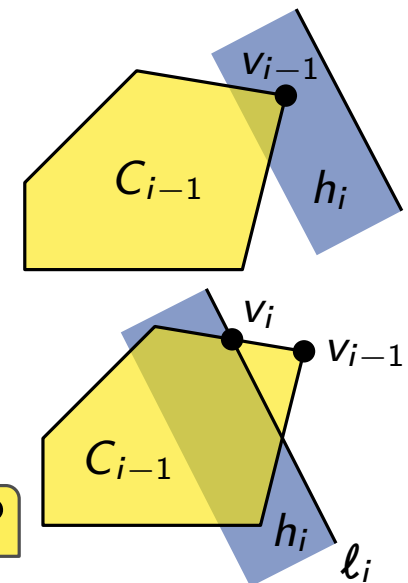
## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt → nicht nur zu Beginn problematisch
- Lösung: siehe Übung

## Was interessiert uns ein $O(n^2)$ Algorithmus?

- Fall  $v_{i-1} \in C_i$  ist billig (setze einfach  $v_i = v_{i-1}$ )
- Fall  $v_{i-1} \notin C_i$  ist teuer ( $O(i)$  zur Berechnung von  $v_i$ )
- Hoffnung:  $v_{i-1} \notin C_i$  tritt nur selten ein
- es gibt Ordnung  $h_1, \dots, h_n$ , sodass  $v_{i-1} \in C_i$  für  $i \geq 3$

Warum?



# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

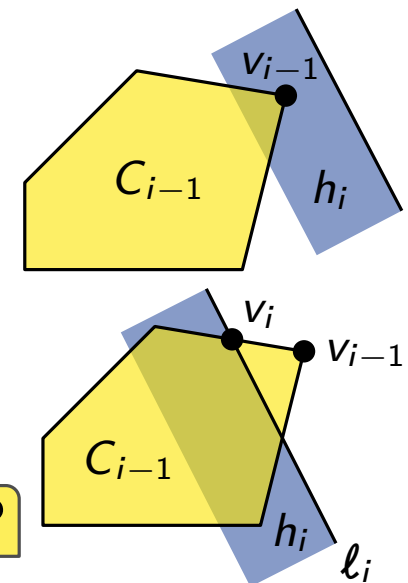
## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt → nicht nur zu Beginn problematisch
- Lösung: siehe Übung

## Was interessiert uns ein $O(n^2)$ Algorithmus?

- Fall  $v_{i-1} \in C_i$  ist billig (setze einfach  $v_i = v_{i-1}$ )
- Fall  $v_{i-1} \notin C_i$  ist teuer ( $O(i)$  zur Berechnung von  $v_i$ )
- Hoffnung:  $v_{i-1} \notin C_i$  tritt nur selten ein
- es gibt Ordnung  $h_1, \dots, h_n$ , sodass  $v_{i-1} \in C_i$  für  $i \geq 3$
- diese Ordnung finden ist nicht so einfach

Warum?



# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

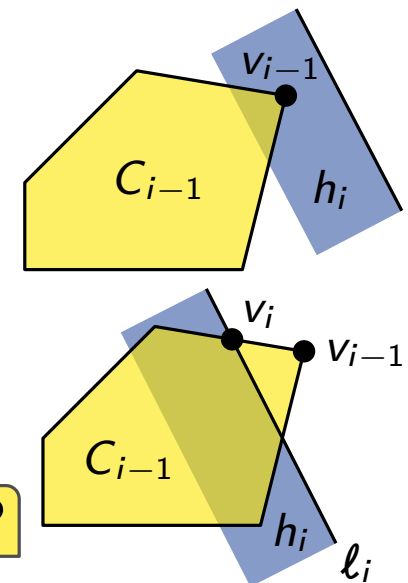
## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt → nicht nur zu Beginn problematisch
- Lösung: siehe Übung

## Was interessiert uns ein $O(n^2)$ Algorithmus?

- Fall  $v_{i-1} \in C_i$  ist billig (setze einfach  $v_i = v_{i-1}$ )
- Fall  $v_{i-1} \notin C_i$  ist teuer ( $O(i)$  zur Berechnung von  $v_i$ )
- Hoffnung:  $v_{i-1} \notin C_i$  tritt nur selten ein
- es gibt Ordnung  $h_1, \dots, h_n$ , sodass  $v_{i-1} \in C_i$  für  $i \geq 3$
- diese Ordnung finden ist nicht so einfach
- aber: die meisten Ordnungen sind gut

Warum?





# Inkrementeller Algorithmus für 2D LPs

## Offene Fragen

- Wie fangen wir an?
- Was interessiert uns ein  $O(n^2)$  Algorithmus?

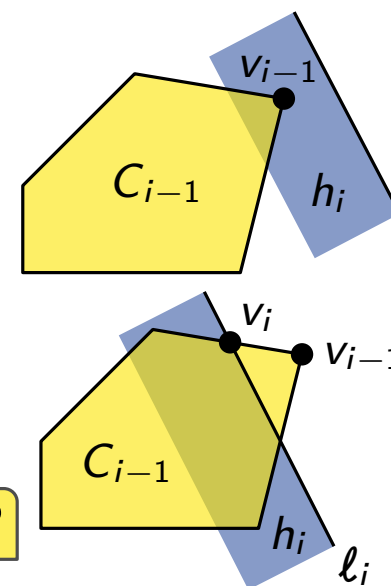
## Wie fangen wir an?

- Problem: zu Beginn ist die gültige Region  $C_0$  unbeschränkt
- einen „optimalen Punkt“ gibt es nicht
- ggf. ist sogar  $C_n$  unbeschränkt → nicht nur zu Beginn problematisch
- Lösung: siehe Übung

## Was interessiert uns ein $O(n^2)$ Algorithmus?

- Fall  $v_{i-1} \in C_i$  ist billig (setze einfach  $v_i = v_{i-1}$ )
- Fall  $v_{i-1} \notin C_i$  ist teuer ( $O(i)$  zur Berechnung von  $v_i$ )
- Hoffnung:  $v_{i-1} \notin C_i$  tritt nur selten ein
- es gibt Ordnung  $h_1, \dots, h_n$ , sodass  $v_{i-1} \in C_i$  für  $i \geq 3$
- diese Ordnung finden ist nicht so einfach
- aber: die meisten Ordnungen sind gut → zufällige Ordnung

Warum?



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)

# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i$$

# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right]$$

# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$

■ dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

**Erwartete Zeit in Iteration  $i$ :**  $\mathbb{E}[X_i] =$

# Randomisierter inkrementeller Algorithmus

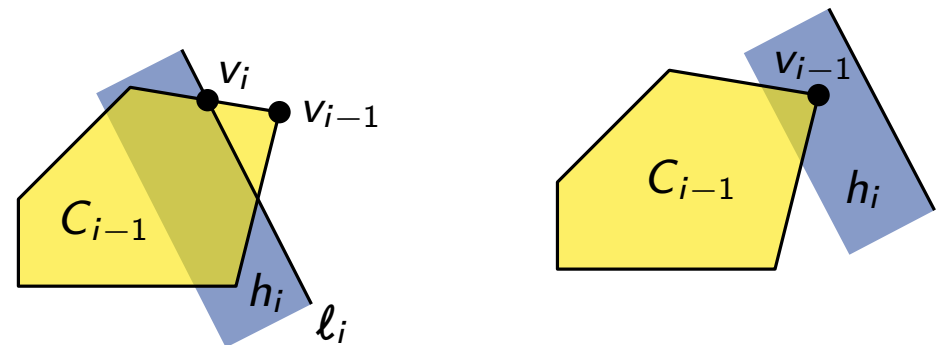
## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$

■ dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

Erwartete Zeit in Iteration  $i$ :  $\mathbb{E}[X_i] =$



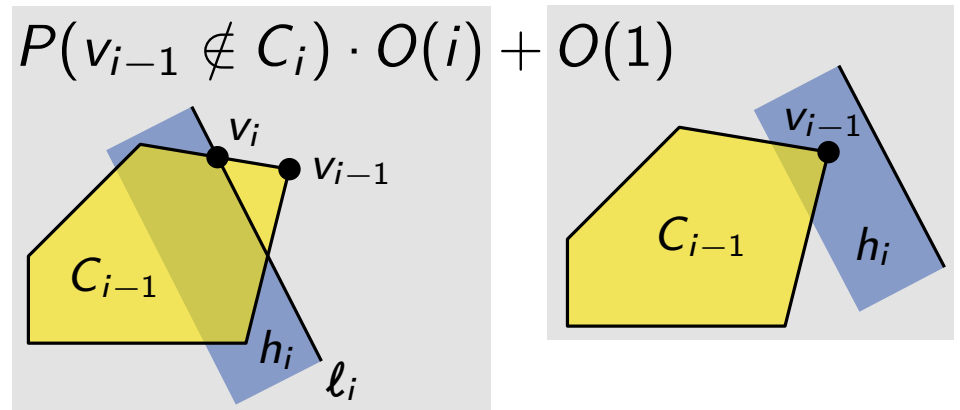
# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

**Erwartete Zeit in Iteration  $i$ :**  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$





# Randomisierter inkrementeller Algorithmus

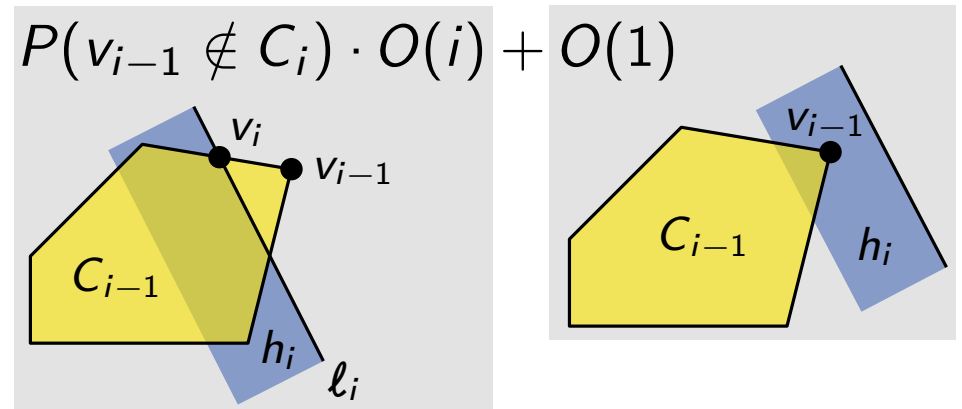
## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

Erwartete Zeit in Iteration  $i$ :  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

Was ist  $P(v_{i-1} \notin C_i)$ ?



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

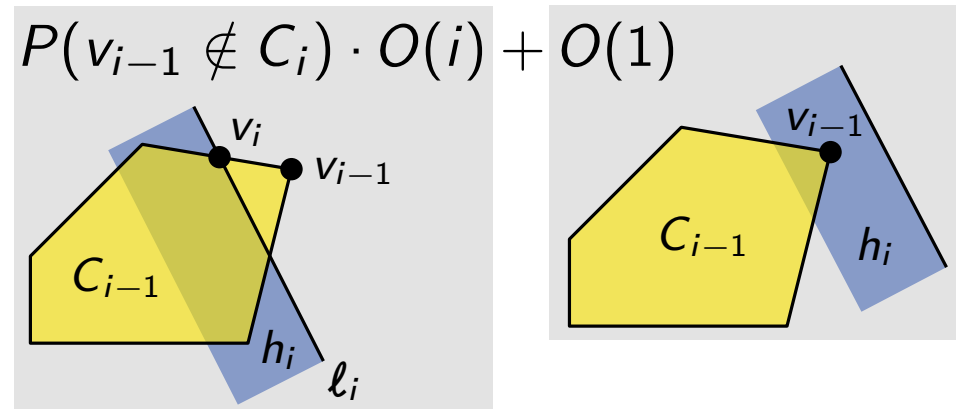
- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

Erwartete Zeit in Iteration  $i$ :  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

Was ist  $P(v_{i-1} \notin C_i)$ ?

- es gilt:  $v_{i-1} \notin C_i \Leftrightarrow v_i \neq v_{i-1}$



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

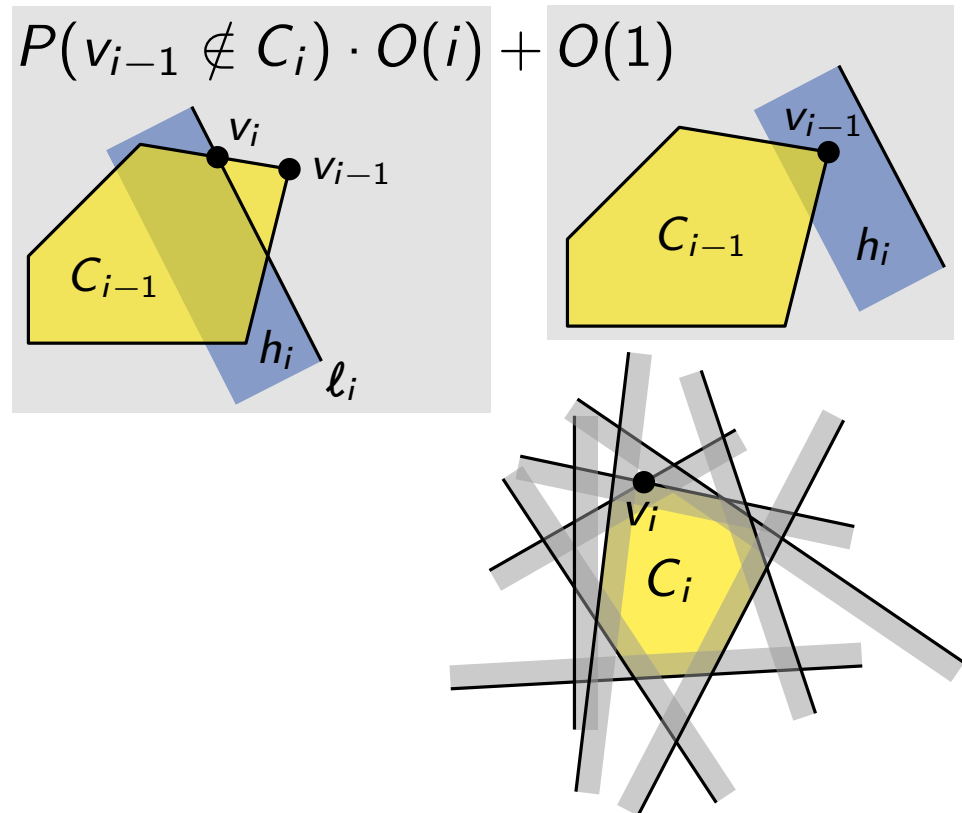
- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

Erwartete Zeit in Iteration  $i$ :  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

## Was ist $P(v_{i-1} \notin C_i)$ ?

- es gilt:  $v_{i-1} \notin C_i \Leftrightarrow v_i \neq v_{i-1}$
- umgekehrte Betrachtungsweise
  - gehe von  $C_i$  zu  $C_{i-1}$  (entferne  $h_i$ )



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

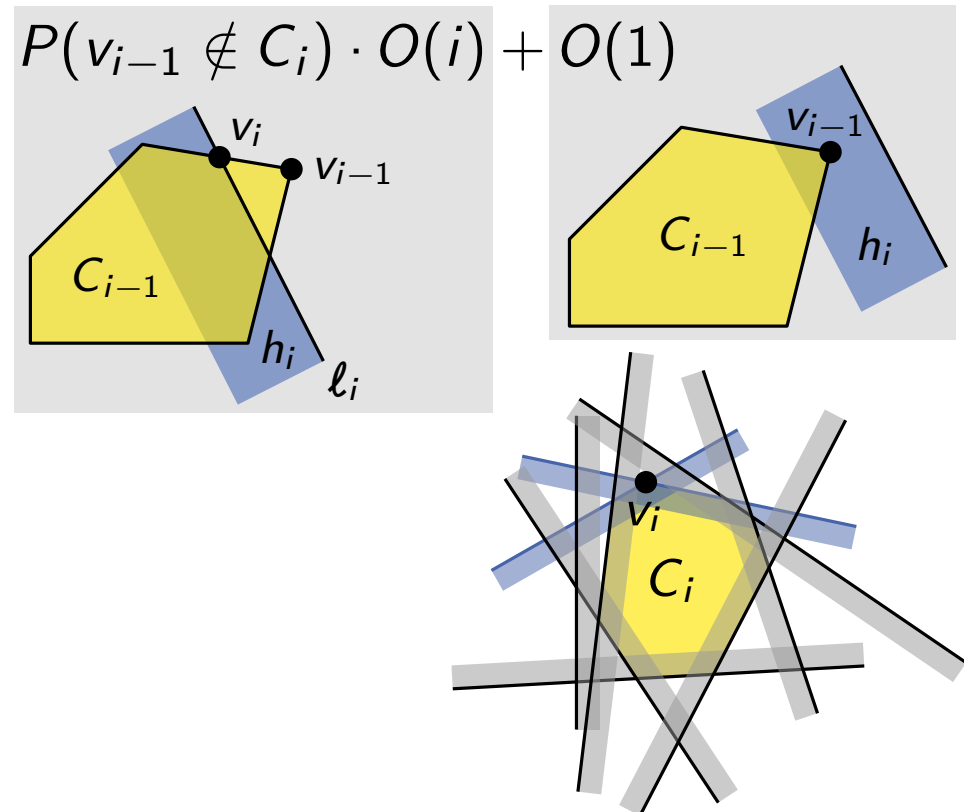
- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

**Erwartete Zeit in Iteration  $i$ :**  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

## Was ist $P(v_{i-1} \notin C_i)$ ?

- es gilt:  $v_{i-1} \notin C_i \Leftrightarrow v_i \neq v_{i-1}$
- umgekehrte Betrachtungsweise
  - gehe von  $C_i$  zu  $C_{i-1}$  (entferne  $h_i$ )
  - dann gilt:  $v_{i-1} \neq v_i \Rightarrow \ell_i$  ist eine der zwei Geraden, die  $v_i$  enthält



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

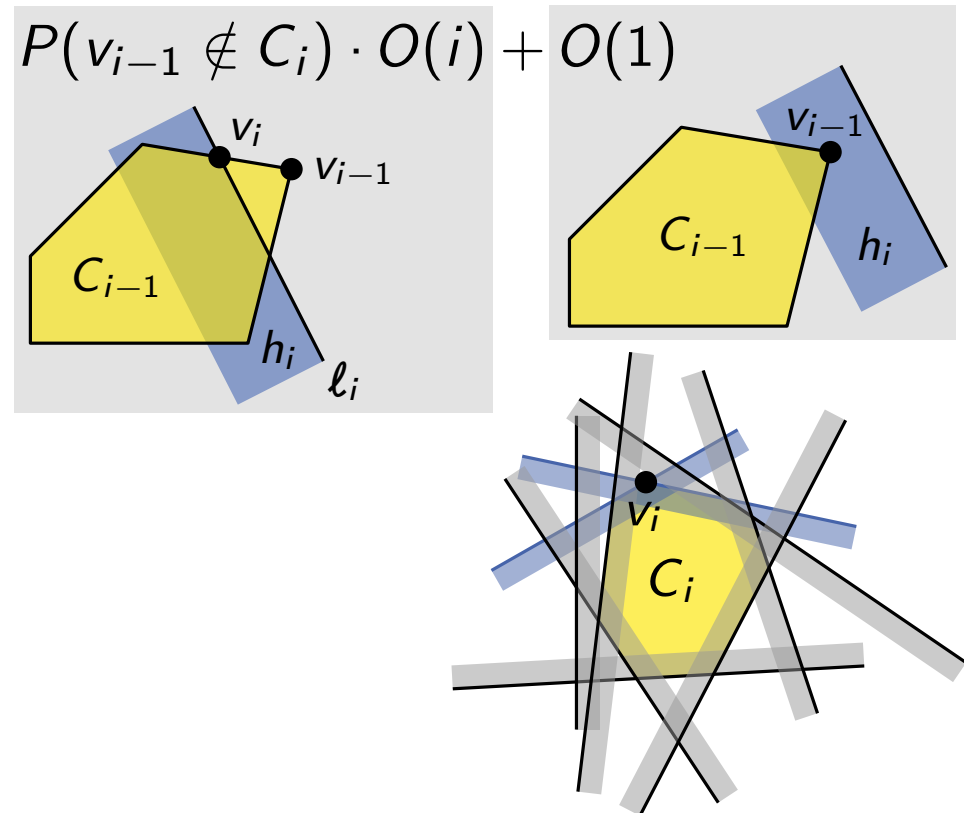
- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

**Erwartete Zeit in Iteration  $i$ :**  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

## Was ist $P(v_{i-1} \notin C_i)$ ?

- es gilt:  $v_{i-1} \notin C_i \Leftrightarrow v_i \neq v_{i-1}$
- umgekehrte Betrachtungsweise
  - gehe von  $C_i$  zu  $C_{i-1}$  (entferne  $h_i$ )
  - dann gilt:  $v_{i-1} \neq v_i \Rightarrow \ell_i$  ist eine der zwei Geraden, die  $v_i$  enthält
  - Wahrscheinlichkeit dafür:  $\frac{2}{i}$



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

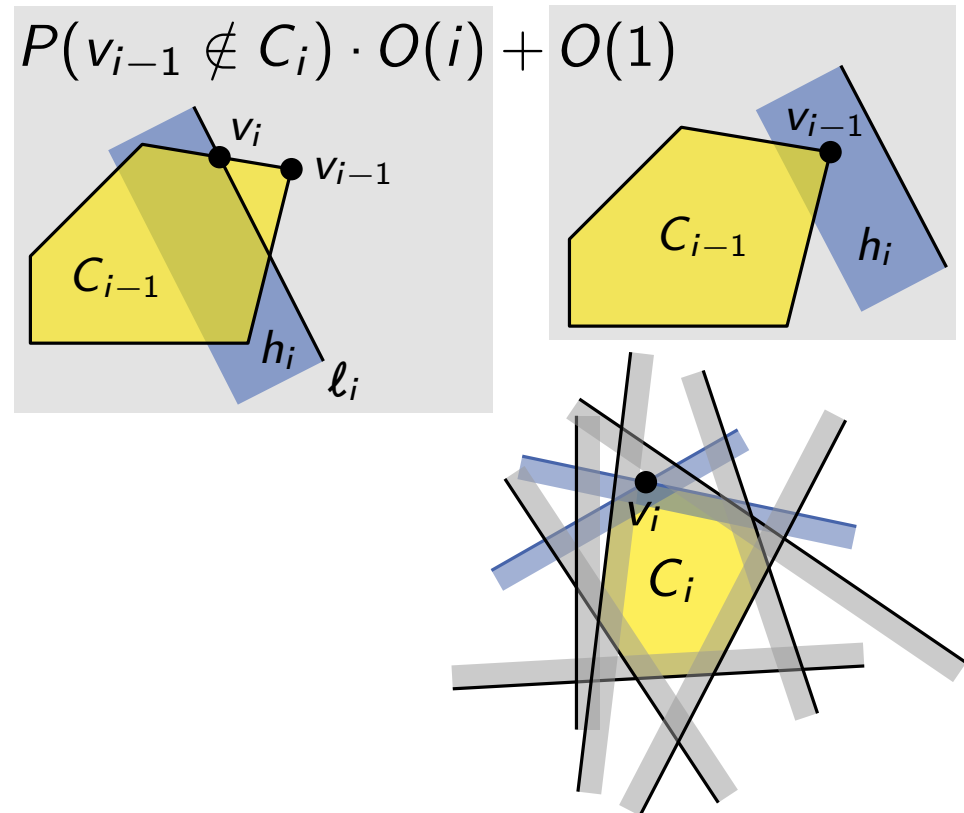
- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

Erwartete Zeit in Iteration  $i$ :  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

## Was ist $P(v_{i-1} \notin C_i)$ ?

- es gilt:  $v_{i-1} \notin C_i \Leftrightarrow v_i \neq v_{i-1}$
- umgekehrte Betrachtungsweise
  - gehe von  $C_i$  zu  $C_{i-1}$  (entferne  $h_i$ )
  - dann gilt:  $v_{i-1} \neq v_i \Rightarrow \ell_i$  ist eine der zwei Geraden, die  $v_i$  enthält
  - Wahrscheinlichkeit dafür:  $\frac{2}{i}$
- also:  $P(v_i \neq v_{i-1}) \leq \frac{2}{i}$



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

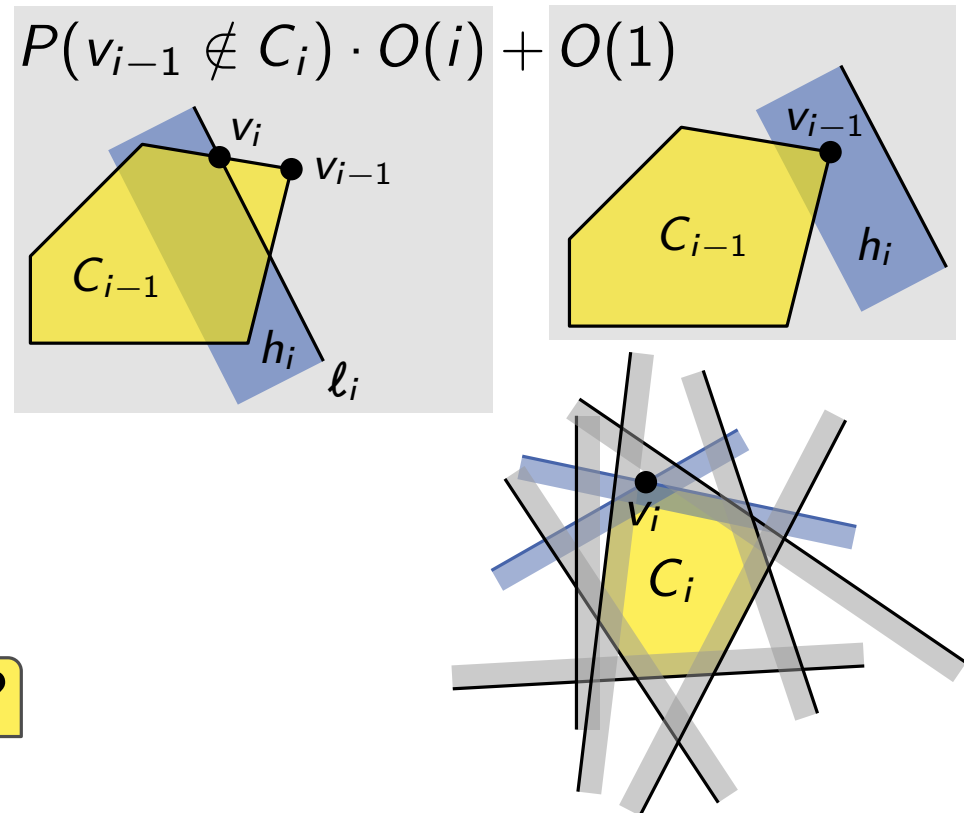
$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

Erwartete Zeit in Iteration  $i$ :  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

## Was ist $P(v_{i-1} \notin C_i)$ ?

- es gilt:  $v_{i-1} \notin C_i \Leftrightarrow v_i \neq v_{i-1}$
- umgekehrte Betrachtungsweise
  - gehe von  $C_i$  zu  $C_{i-1}$  (entferne  $h_i$ )
  - dann gilt:  $v_{i-1} \neq v_i \Rightarrow \ell_i$  ist eine der zwei Geraden, die  $v_i$  enthält
  - Wahrscheinlichkeit dafür:  $\frac{2}{i}$
- also:  $P(v_i \neq v_{i-1}) \leq \frac{2}{i}$

Warum  $\leq$ ?



# Randomisierter inkrementeller Algorithmus

## Erwartete Laufzeit

- die Laufzeit ist eine Zufallsvariable (im Folgenden mit  $X$  bezeichnet)
- weitere Zufallsvariablen:  $X_i$  ist die Laufzeit in Iteration  $i$
- dann gilt:

$$X = \sum_{i=1}^n X_i \Rightarrow \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i]$$

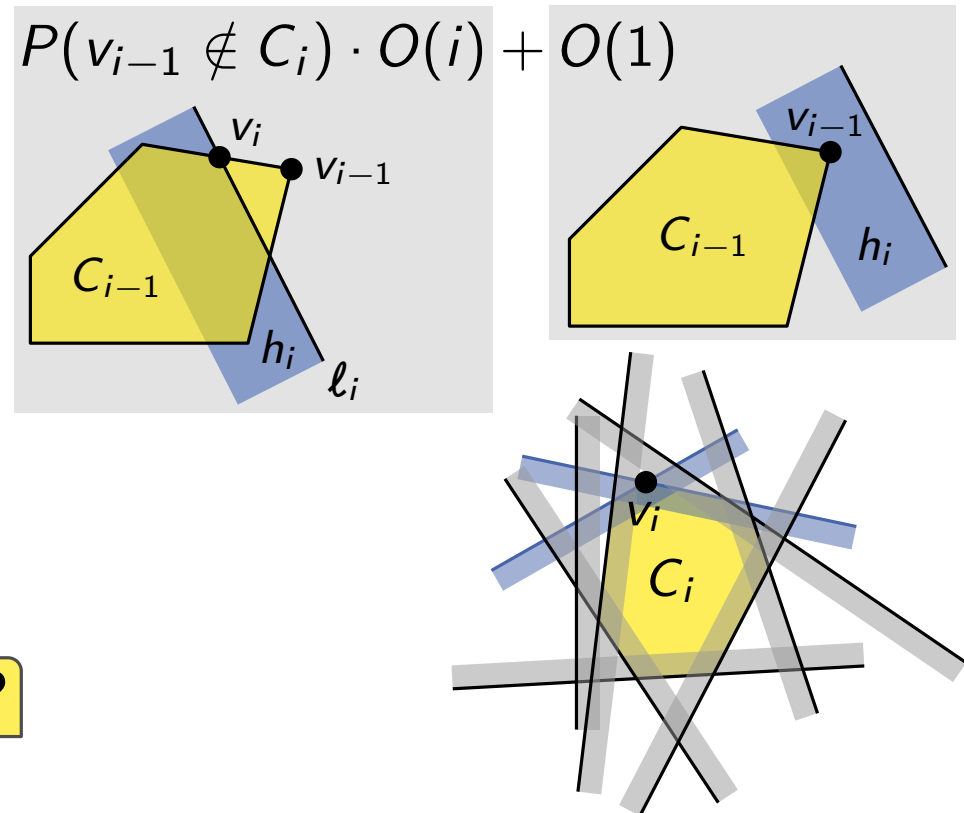
Erwartete Zeit in Iteration  $i$ :  $\mathbb{E}[X_i] = P(v_{i-1} \notin C_i) \cdot O(i) + O(1)$

## Was ist $P(v_{i-1} \notin C_i)$ ?

- es gilt:  $v_{i-1} \notin C_i \Leftrightarrow v_i \neq v_{i-1}$
- umgekehrte Betrachtungsweise
  - gehe von  $C_i$  zu  $C_{i-1}$  (entferne  $h_i$ )
  - dann gilt:  $v_{i-1} \neq v_i \Rightarrow \ell_i$  ist eine der zwei Geraden, die  $v_i$  enthält
  - Wahrscheinlichkeit dafür:  $\frac{2}{i}$

■ also:  $P(v_i \neq v_{i-1}) \leq \frac{2}{i}$  **Warum  $\leq$ ?**

$\Rightarrow \mathbb{E}[X_i] \in O(1) \Rightarrow \mathbb{E}[X] \in O(n)$





# Zusammenfassung

## Heute gesehen

- Berechnung einer 3D-Gussform  $\rightarrow$  Formulierung als 2D-LP

# Zusammenfassung

## Heute gesehen

- Berechnung einer 3D-Gussform  $\rightarrow$  Formulierung als 2D-LP
- 2D-LP  $\rightarrow$  Formulierung als Schnitt von Halbebenen

# Zusammenfassung

## Heute gesehen

- Berechnung einer 3D-Gussform  $\rightarrow$  Formulierung als 2D-LP
- 2D-LP  $\rightarrow$  Formulierung als Schnitt von Halbebenen
- Berechnung des Schnitts von Halbebenen:  $O(n \log n)$ -Algorithmus

# Zusammenfassung

## Heute gesehen

- Berechnung einer 3D-Gussform  $\rightarrow$  Formulierung als 2D-LP
- 2D-LP  $\rightarrow$  Formulierung als Schnitt von Halbebenen
- Berechnung des Schnitts von Halbebenen:  $O(n \log n)$ -Algorithmus
- Lösung eines 2D-LPs: randomisierter Algo mit erwarteter Laufzeit  $O(n)$

# Zusammenfassung

## Heute gesehen

- Berechnung einer 3D-Gussform → Formulierung als 2D-LP
- 2D-LP → Formulierung als Schnitt von Halbebenen
- Berechnung des Schnitts von Halbebenen:  $O(n \log n)$ -Algorithmus
- Lösung eines 2D-LPs: randomisierter Algo mit erwarteter Laufzeit  $O(n)$

## Was gibt es sonst noch?

- randomisierte Algo für 2D-LPs funktioniert auch für höhere Dimensionen

# Zusammenfassung

## Heute gesehen

- Berechnung einer 3D-Gussform → Formulierung als 2D-LP
- 2D-LP → Formulierung als Schnitt von Halbebenen
- Berechnung des Schnitts von Halbebenen:  $O(n \log n)$ -Algorithmus
- Lösung eines 2D-LPs: randomisierter Algo mit erwarteter Laufzeit  $O(n)$

## Was gibt es sonst noch?

- randomisierte Algo für 2D-LPs funktioniert auch für höhere Dimensionen
  - Laufzeit: weiterhin erwartet  $O(n)$ , wenn  $d$  konstant
  - wächst superexponentiell in  $d$

# Zusammenfassung

## Heute gesehen

- Berechnung einer 3D-Gussform → Formulierung als 2D-LP
- 2D-LP → Formulierung als Schnitt von Halbebenen
- Berechnung des Schnitts von Halbebenen:  $O(n \log n)$ -Algorithmus
- Lösung eines 2D-LPs: randomisierter Algo mit erwarteter Laufzeit  $O(n)$

## Was gibt es sonst noch?

- randomisierte Algo für 2D-LPs funktioniert auch für höhere Dimensionen
  - Laufzeit: weiterhin erwartet  $O(n)$ , wenn  $d$  konstant
  - wächst superexponentiell in  $d$
- diese Art der Randomisierung funktioniert für viele geometrische Probleme