Vorlesung Algorithmische Geometrie

Lineares Programmieren

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

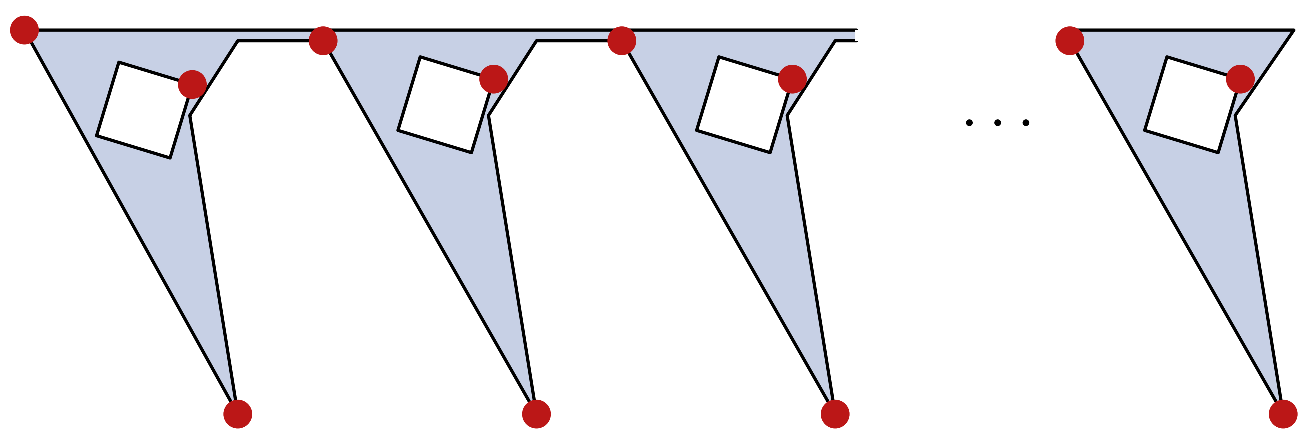
Martin Nöllenburg

03.05.2011

Nachtrag Art Gallery Problem

Lässt sich der Triangulierungs-Algorithmus auch auf Polygone mit Löchern erweitern?

- Triangulierung: ja
- Aber reichen weiterhin $\lfloor n/3 \rfloor$ Kameras aus?
  Nein, eine Verallgemeinerung des Art-Gallery-Theorems besagt, dass manchmal $\lfloor (n+h)/3 \rfloor$ Kameras nötig, aber immer ausreichend sind, wobei h die Anzahl der Löcher ist. [Hoffmann et al., 91]



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:

$$\begin{aligned} P_1: &\quad 4R_1 + R_2 \\ P_2: &\quad 11R_1 + R_2 + R_3 \end{aligned}$$

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

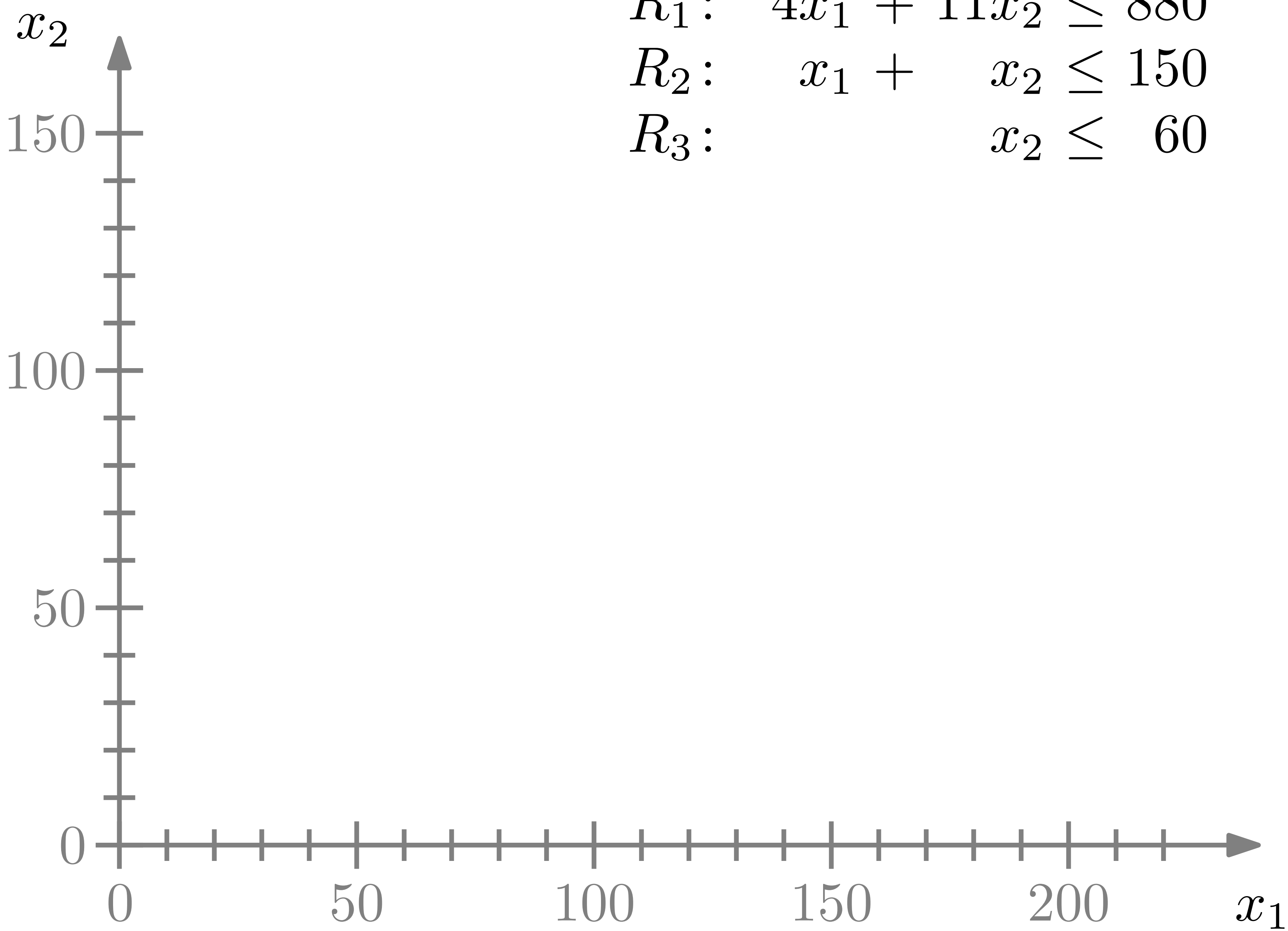
$$\begin{aligned} R_1: &\quad 4x_1 + 11x_2 \leq 880 \\ R_2: &\quad x_1 + x_2 \leq 150 \\ R_3: &\quad x_2 \leq 60 \end{aligned}$$

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

Lösung

Lineare Beschränkungen:

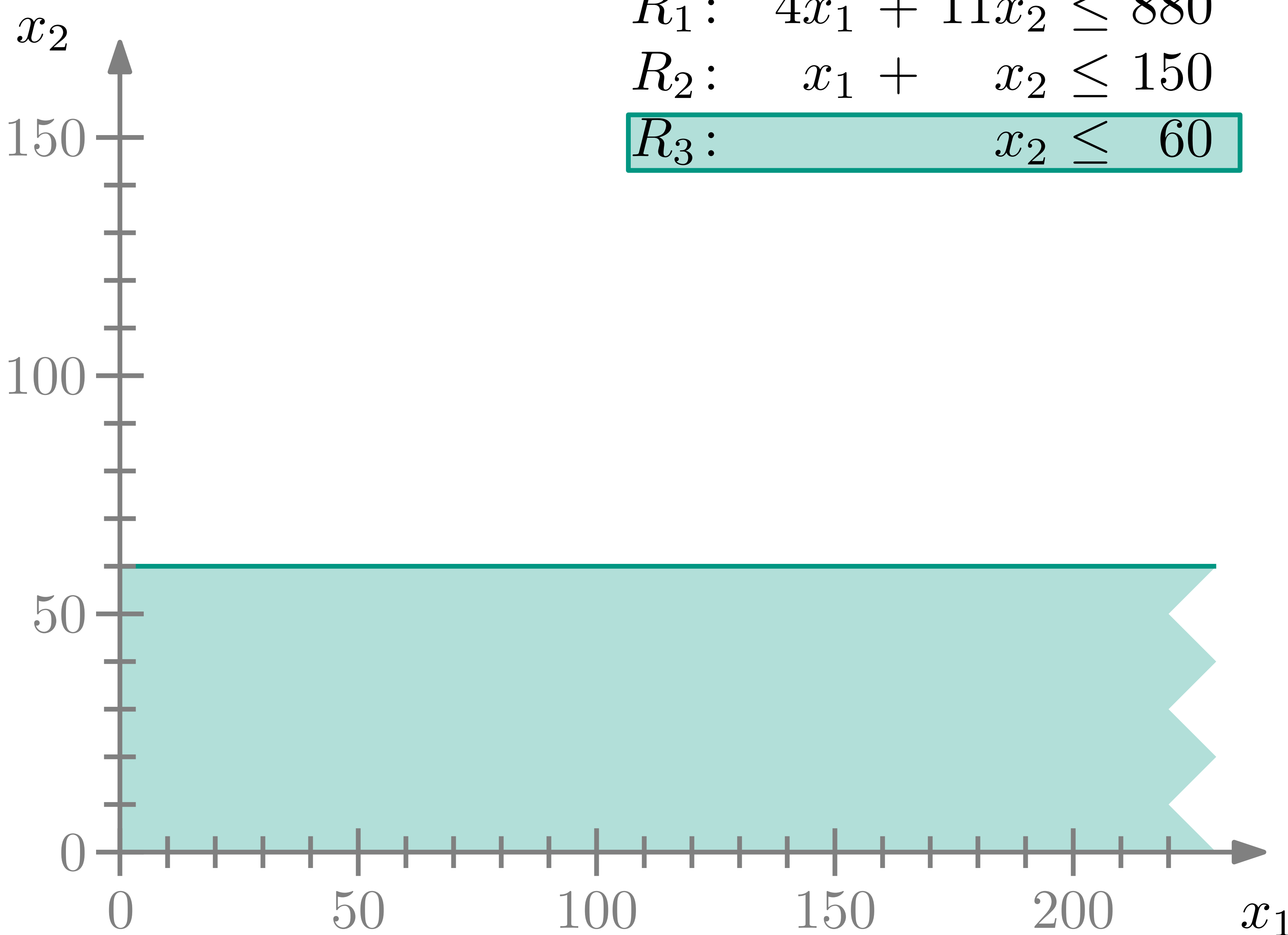
$$\begin{aligned} R_1: \quad 4x_1 + 11x_2 &\leq 880 \\ R_2: \quad x_1 + x_2 &\leq 150 \\ R_3: \quad x_2 &\leq 60 \end{aligned}$$



Lösung

Lineare Beschränkungen:

$$\begin{aligned} R_1: &\quad 4x_1 + 11x_2 \leq 880 \\ R_2: &\quad x_1 + x_2 \leq 150 \\ R_3: &\quad \boxed{x_2 \leq 60} \end{aligned}$$



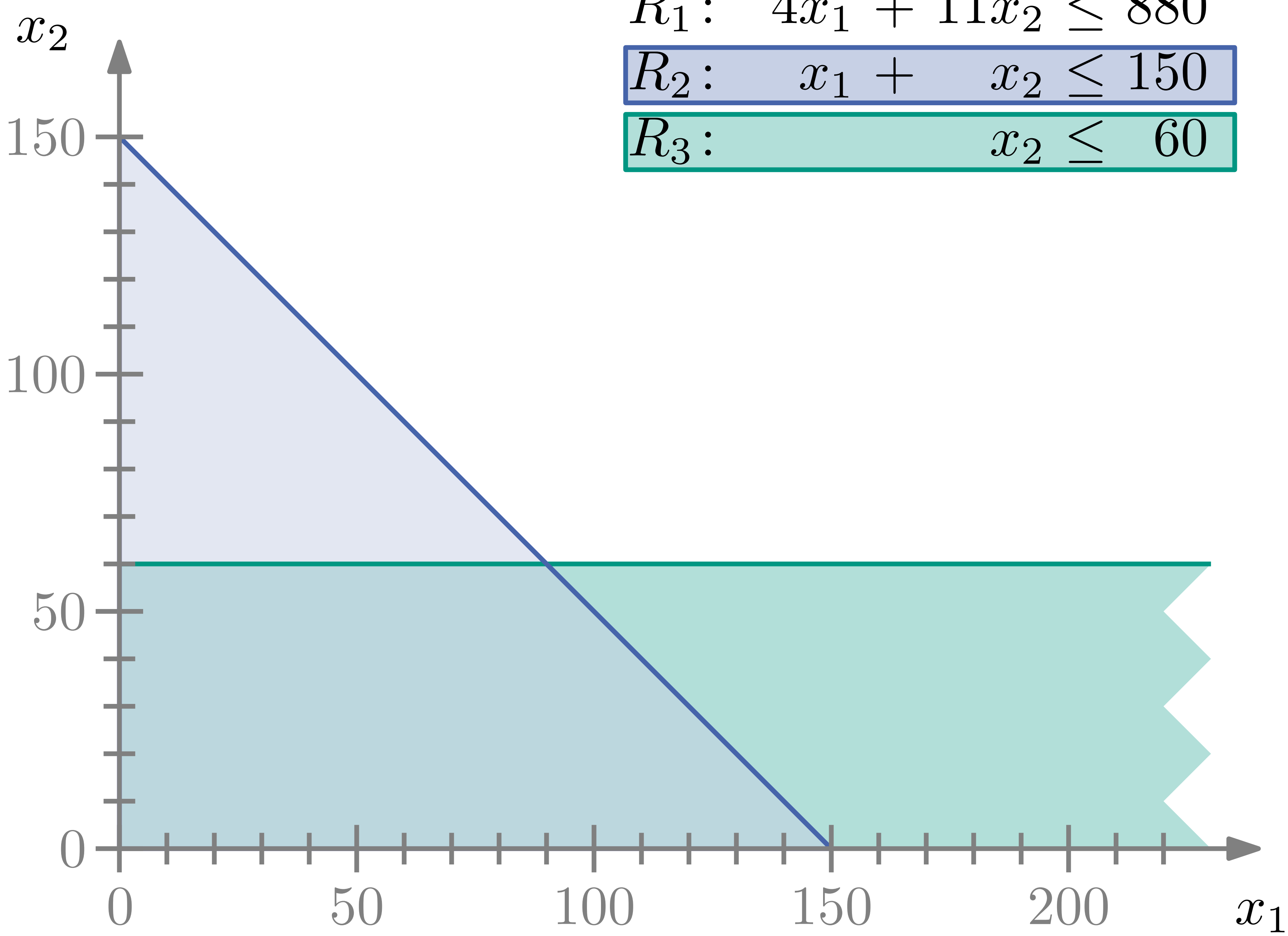
Lösung

Lineare Beschränkungen:

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

$$\boxed{R_2: \quad x_1 + x_2 \leq 150}$$

$$\boxed{R_3: \quad x_2 \leq 60}$$



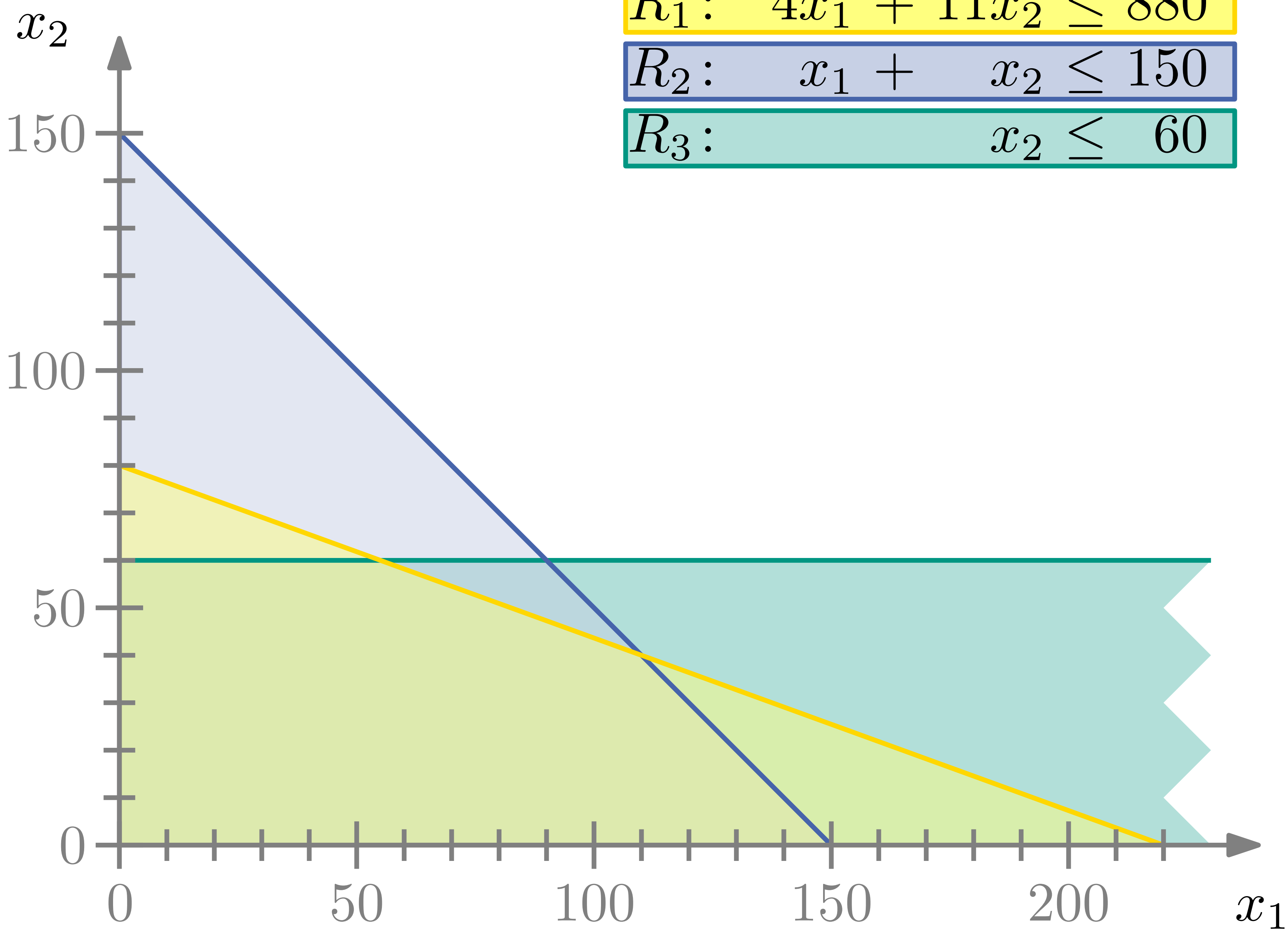
Lösung

Lineare Beschränkungen:

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

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

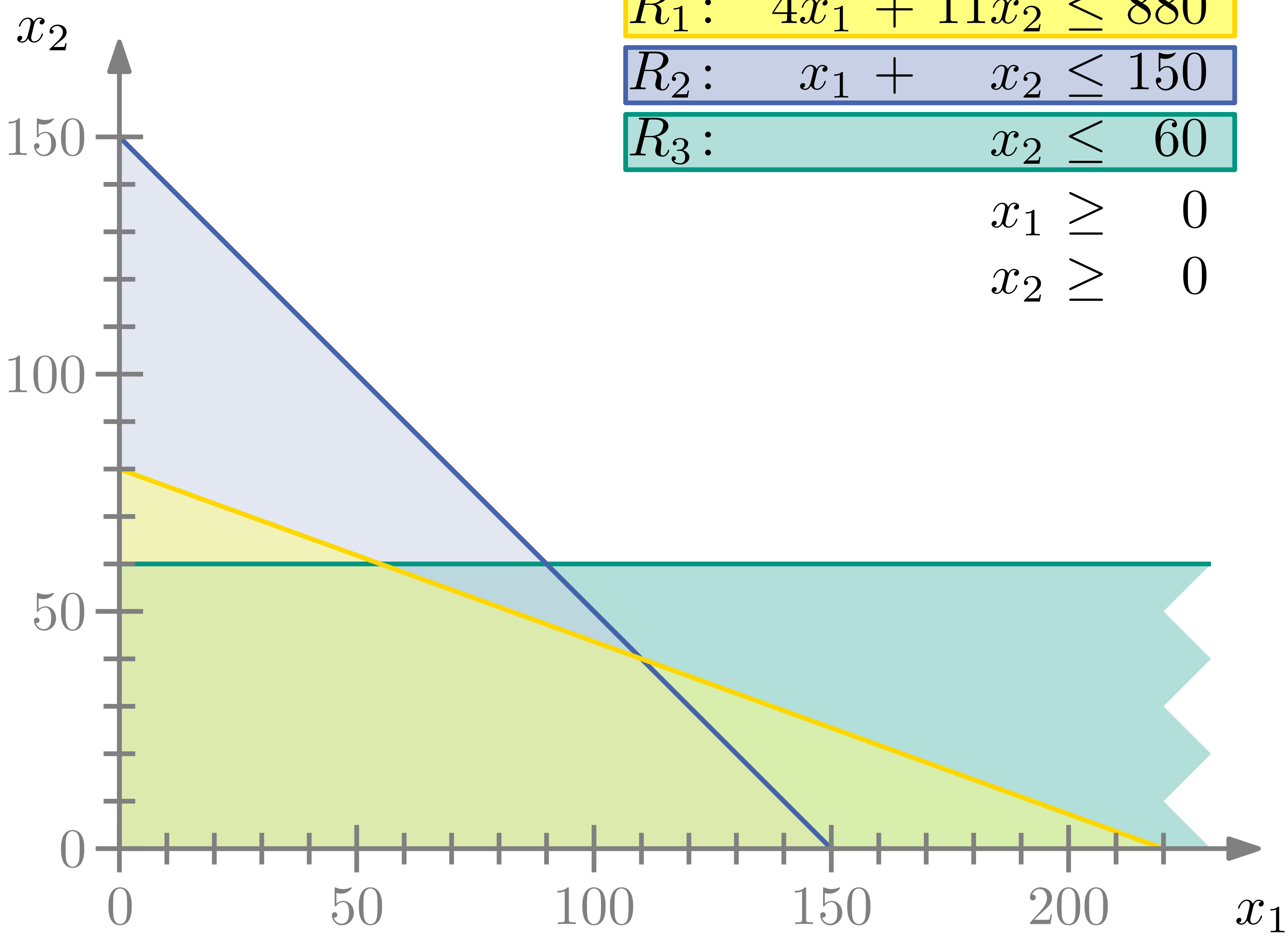
$R_3: \quad x_2 \leq 60$



Lösung

Lineare Beschränkungen:

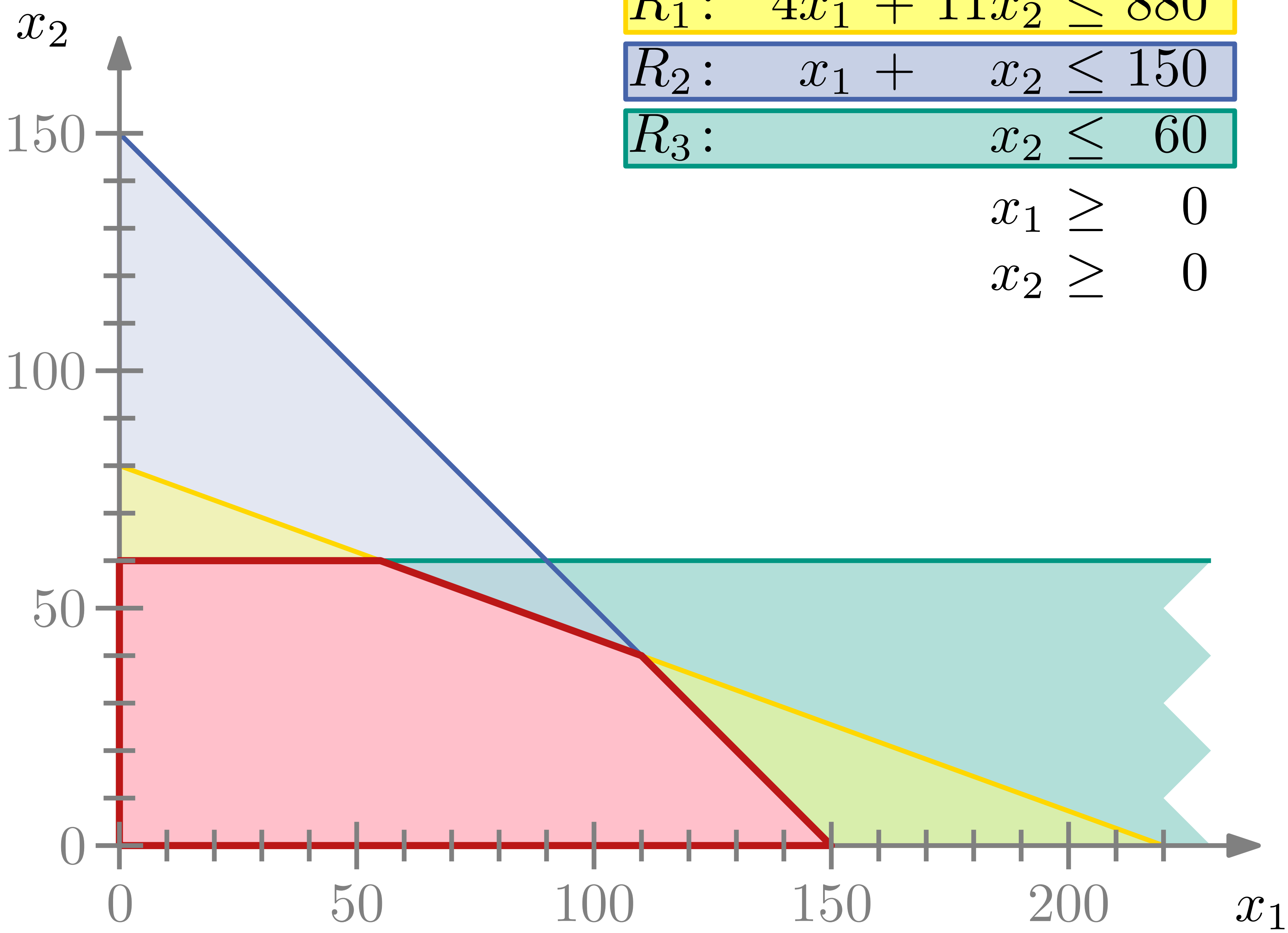
$$\begin{aligned} R_1: \quad 4x_1 + 11x_2 &\leq 880 \\ R_2: \quad x_1 + x_2 &\leq 150 \\ R_3: \quad x_2 &\leq 60 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$



Lösung

Lineare Beschränkungen:

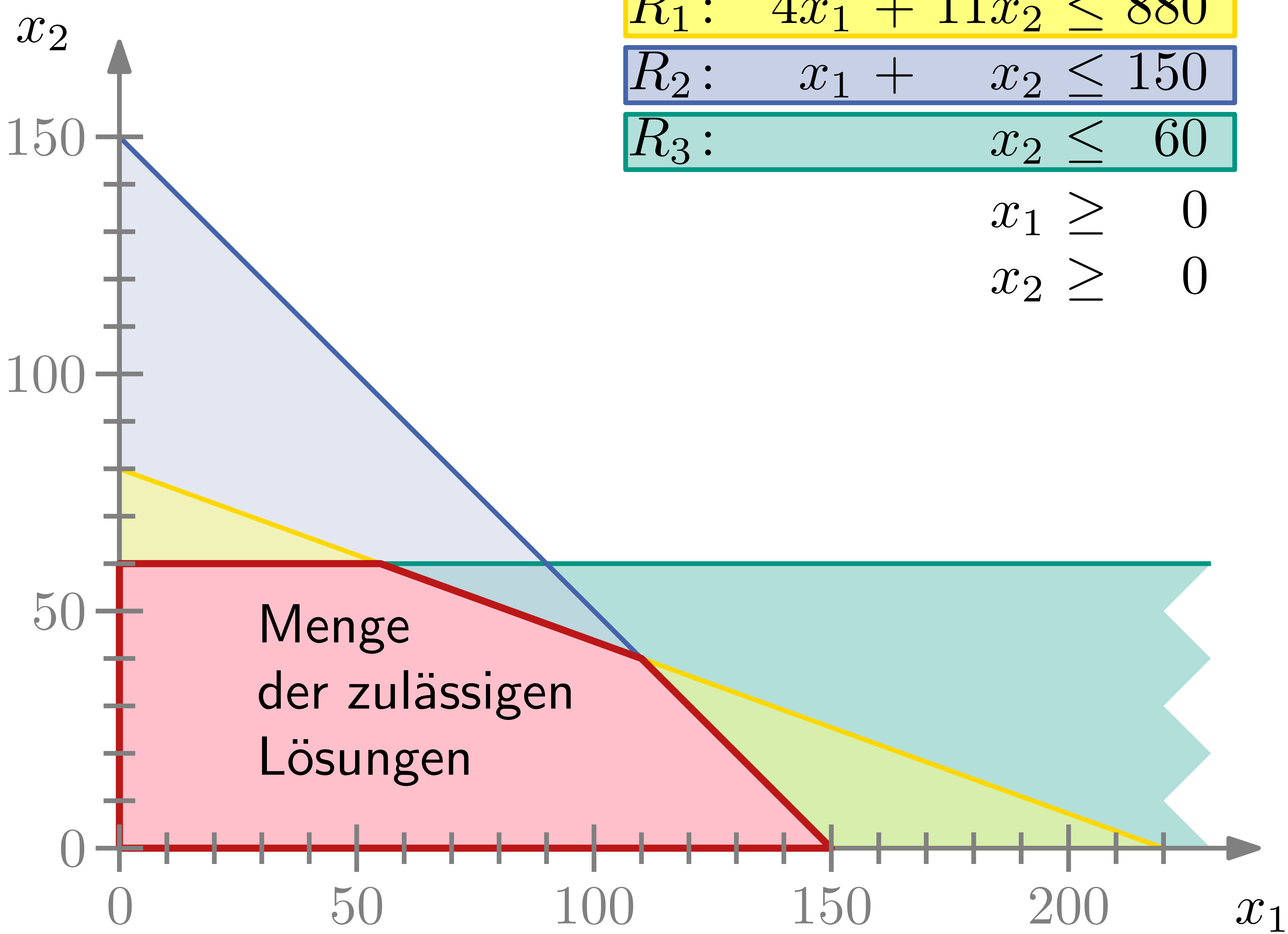
$$
\begin{aligned}
R_1: \quad 4x_1 + 11x_2 &\le 880\\
R_2: \quad x_1 + x_2 &\le 150\\
R_3: \quad x_2 &\le 60\\
x_1 &\ge 0\\
x_2 &\ge 0
\end{aligned}
$$



Lösung

Lineare Beschränkungen:

$$R_1: \quad 4x_1 + 11x_2 \leq 880$$
$$R_2: \quad x_1 + x_2 \leq 150$$
$$R_3: \quad x_2 \leq 60$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$



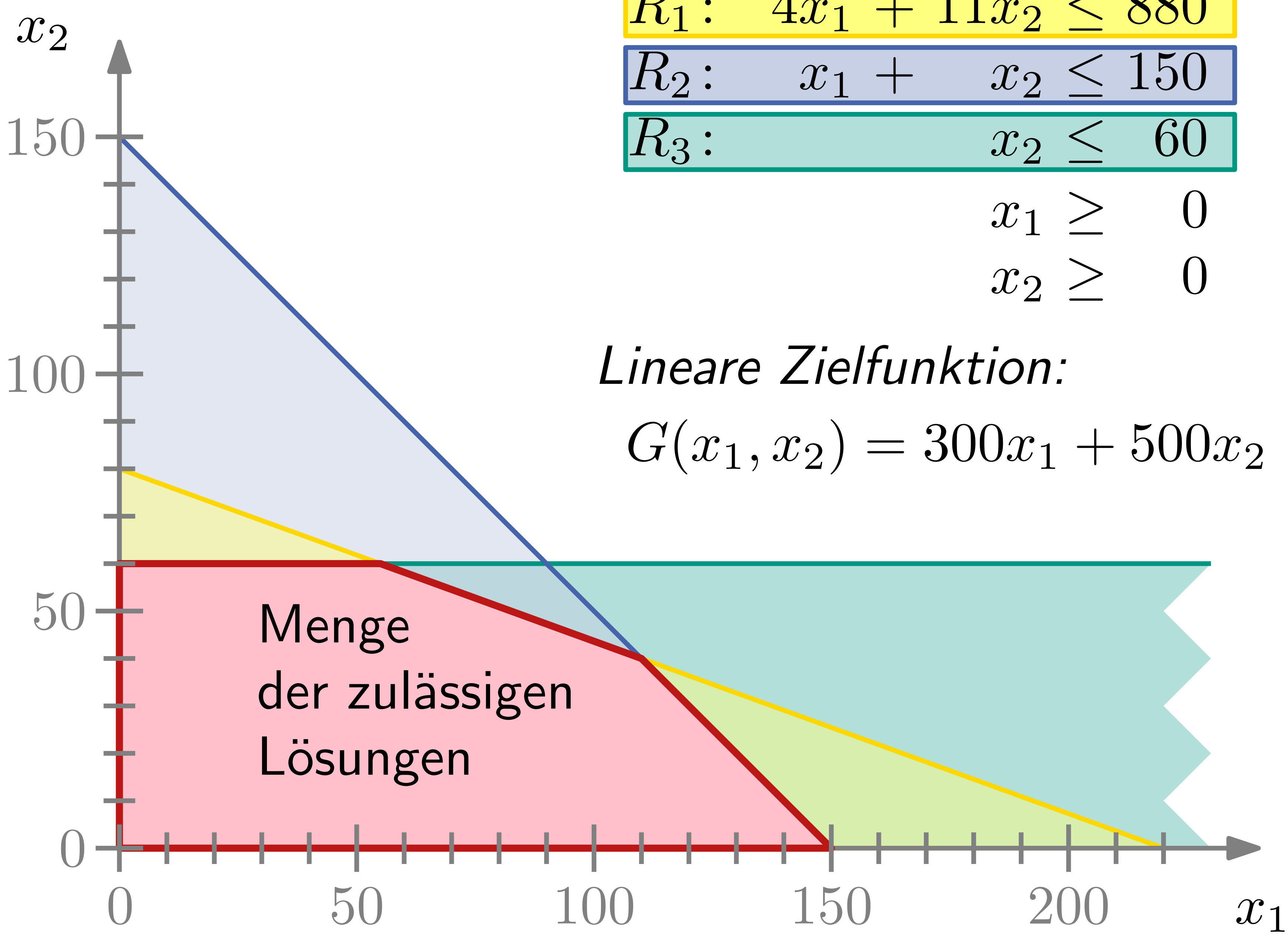
Lösung

Lineare Beschränkungen:

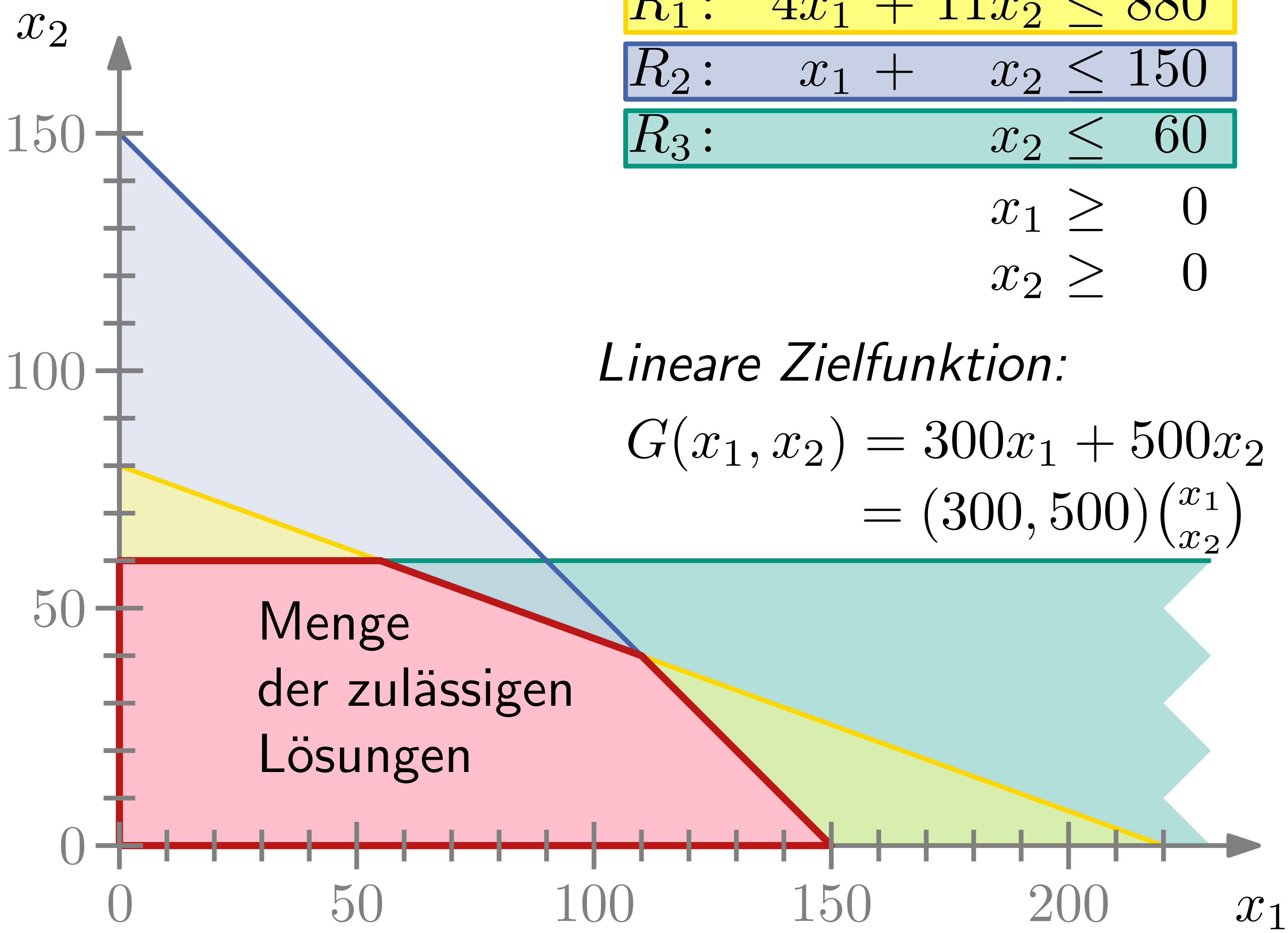
$$
\begin{aligned}
R_1: \quad 4x_1 + 11x_2 &\leq 880 \\
R_2: \quad x_1 + x_2 &\leq 150 \\
R_3: \quad x_2 &\leq 60 \\
x_1 &\geq 0 \\
x_2 &\geq 0
\end{aligned}
$$

Lineare Zielfunktion:

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



Lösung



Lineare Beschränkungen:

$$\begin{aligned} R_1: \quad 4x_1 + 11x_2 &\le 880 \\ R_2: \quad x_1 + x_2 &\le 150 \\ R_3: \quad x_2 &\le 60 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

Lineare Zielfunktion:

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

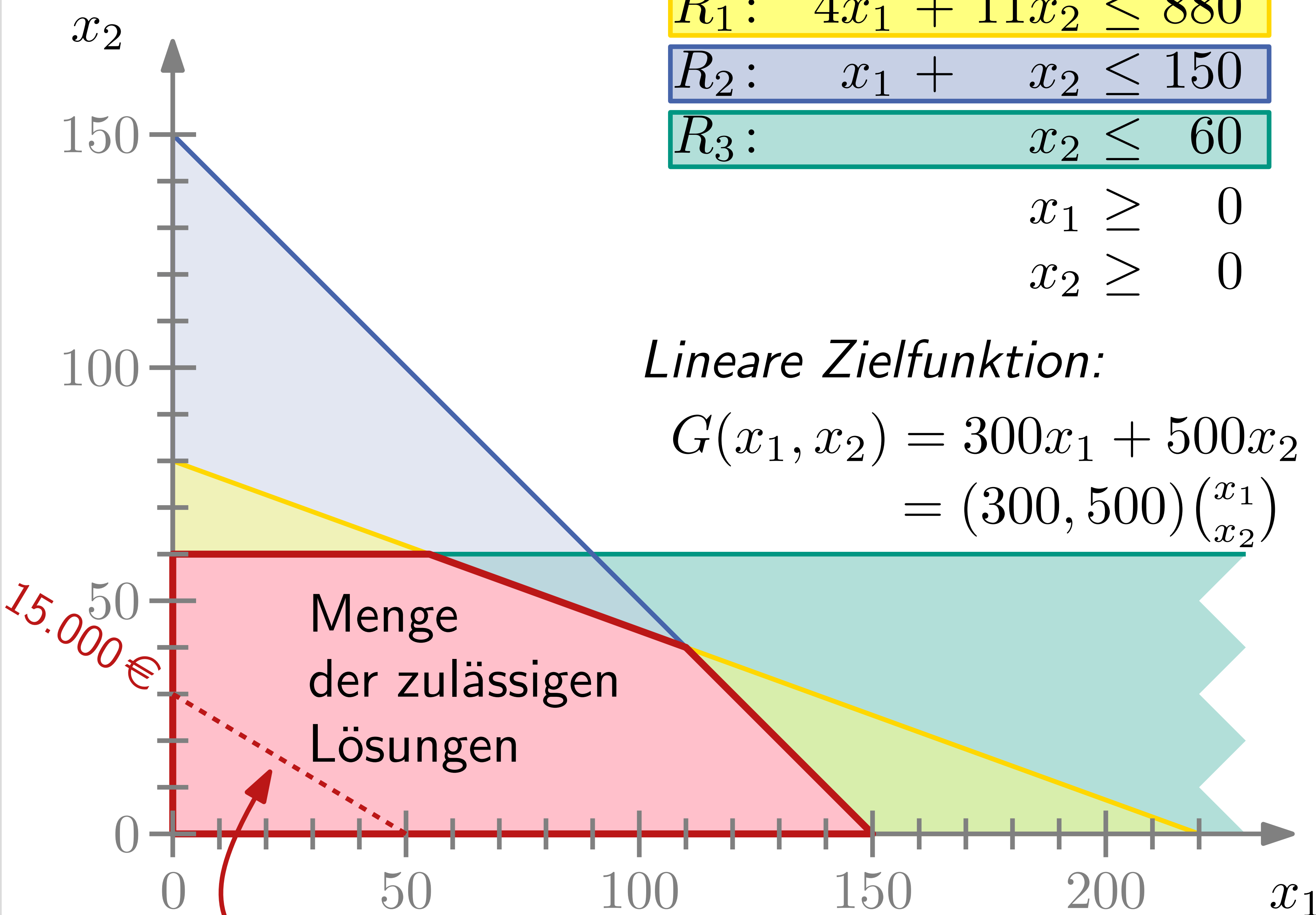
Lösung

Lineare Beschränkungen:

$$\begin{aligned} R_1:&\quad 4x_1 + 11x_2 \leq 880 \\ R_2:&\quad x_1 + x_2 \leq 150 \\ R_3:&\quad x_2 \leq 60 \\ &\quad x_1 \geq 0 \\ &\quad x_2 \geq 0 \end{aligned}$$

Lineare Zielfunktion:

$$G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}$$



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

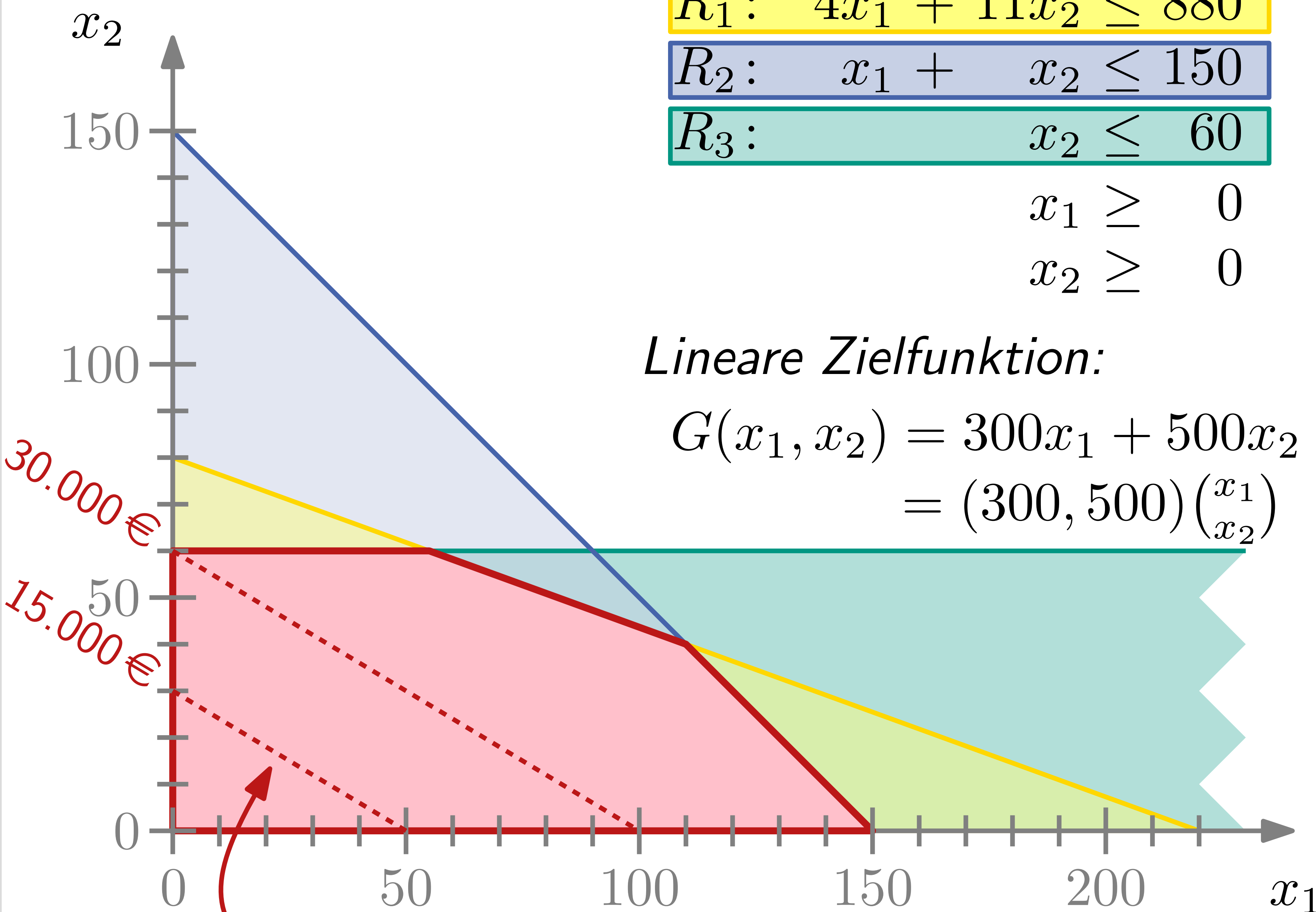
Lösung

Lineare Beschränkungen:

$$
\begin{aligned}
R_1:&\quad 4x_1 + 11x_2 \le 880\\
R_2:&\quad x_1 + x_2 \le 150\\
R_3:&\quad x_2 \le 60\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0
\end{aligned}
$$

Lineare Zielfunktion:

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



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

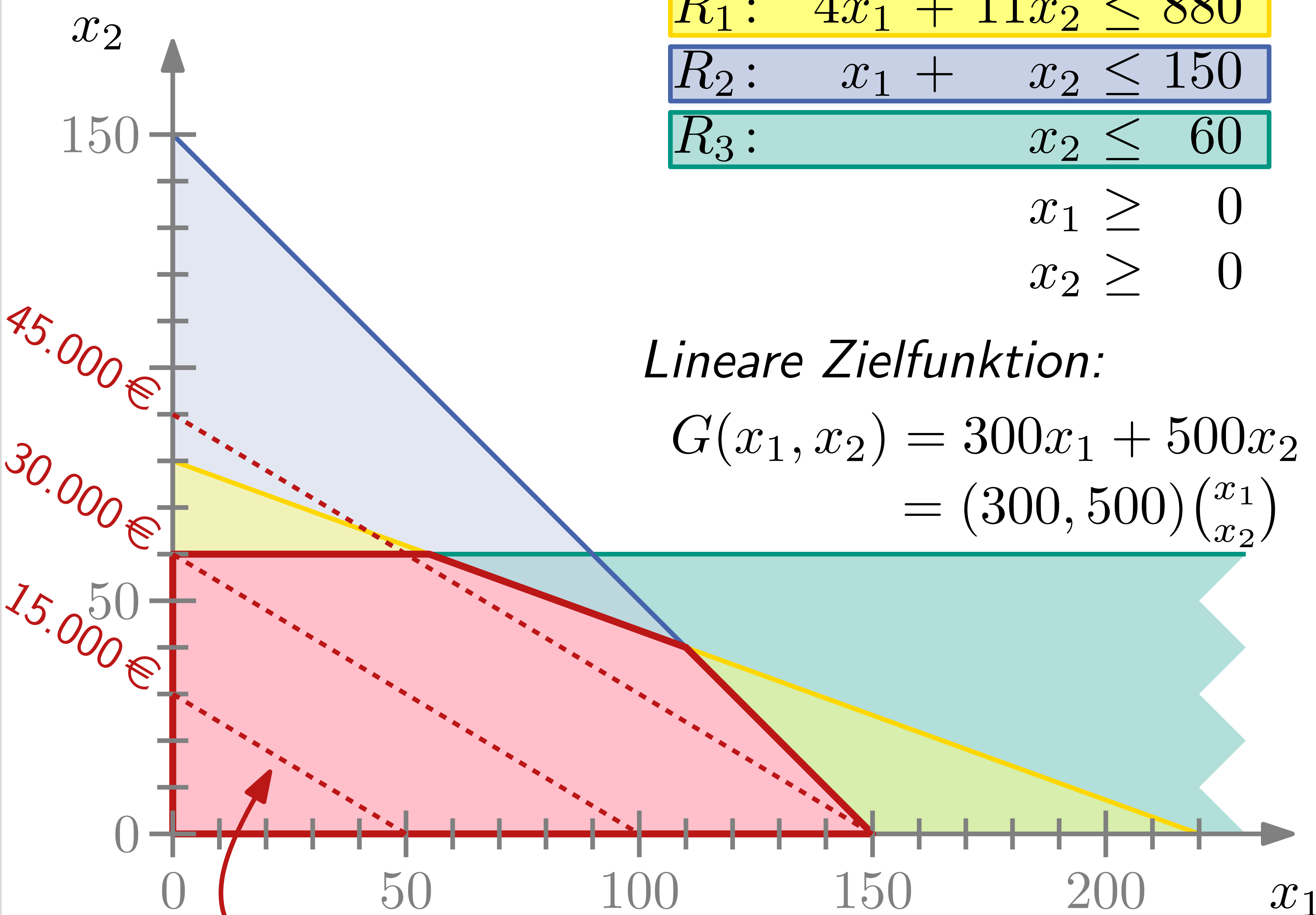
Lösung

Lineare Beschränkungen:

$$
\begin{aligned}
R_1:&\quad 4x_1 + 11x_2 \leq 880\\
R_2:&\quad x_1 + x_2 \leq 150\\
R_3:&\quad x_2 \leq 60\\
&\quad x_1 \geq 0\\
&\quad x_2 \geq 0
\end{aligned}
$$

Lineare Zielfunktion:

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



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

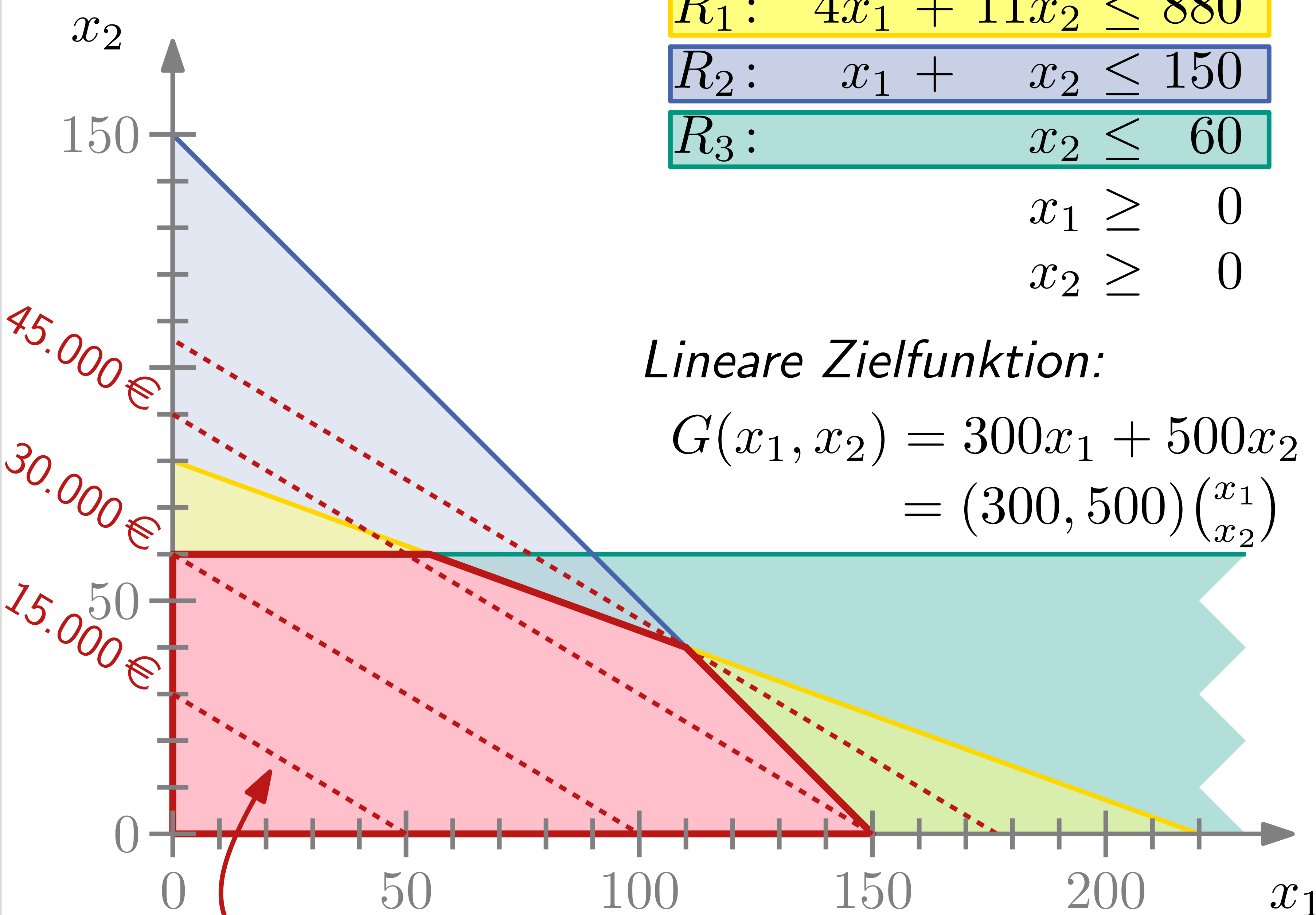
Lösung

Lineare Beschränkungen:

$$\begin{aligned} R_1: \quad 4x_1 + 11x_2 &\le 880 \\ R_2: \quad x_1 + x_2 &\le 150 \\ R_3: \quad x_2 &\le 60 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

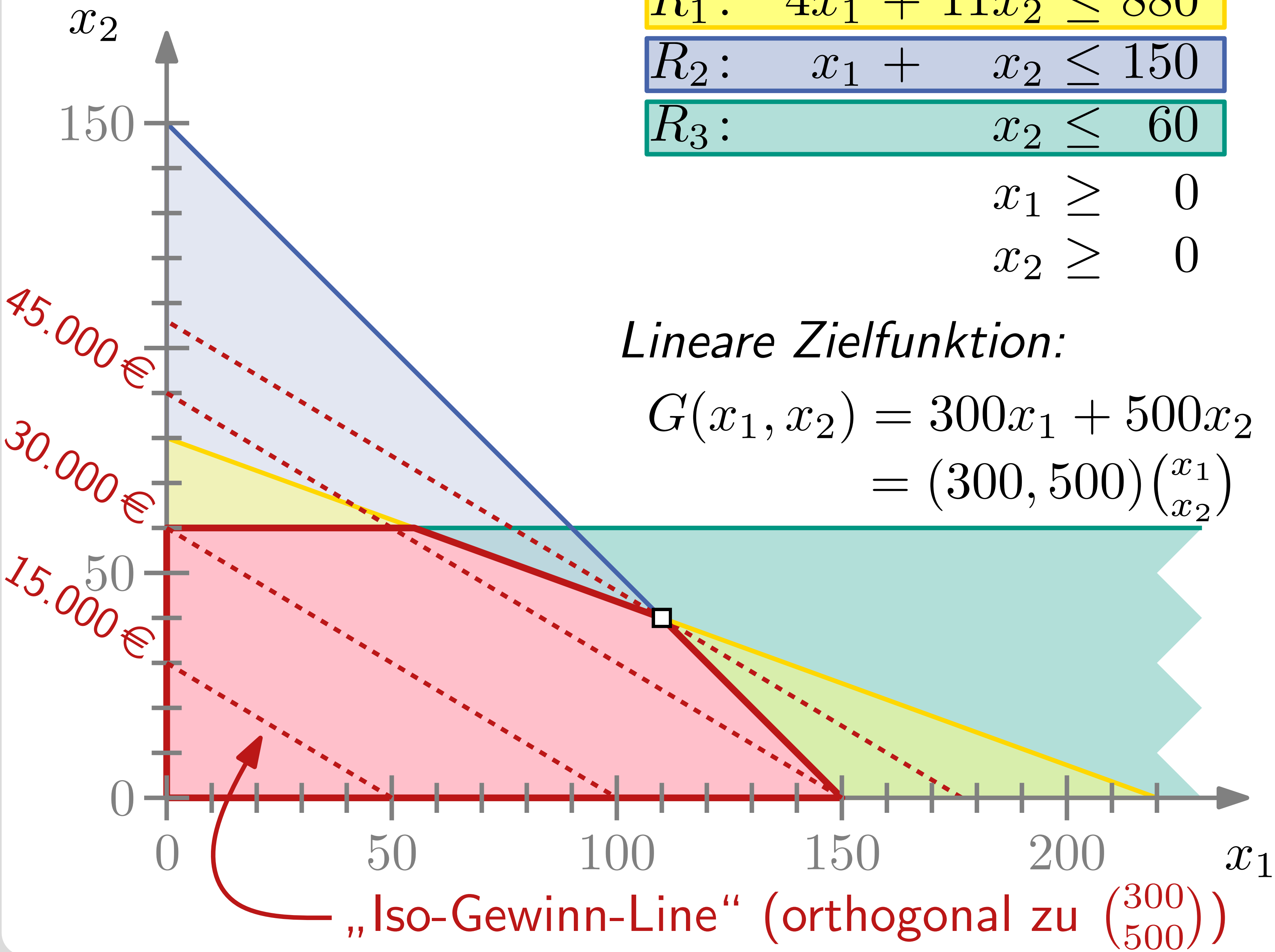
Lineare Zielfunktion:

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



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

Lösung



Lineare Beschränkungen:

$$
\begin{array}{lrcl}
R_1: & 4x_1 + 11x_2 & \leq & 880 \\
R_2: & x_1 + x_2 & \leq & 150 \\
R_3: & x_2 & \leq & 60 \\
 & x_1 & \geq & 0 \\
 & x_2 & \geq & 0
\end{array}
$$

Lineare Zielfunktion:

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

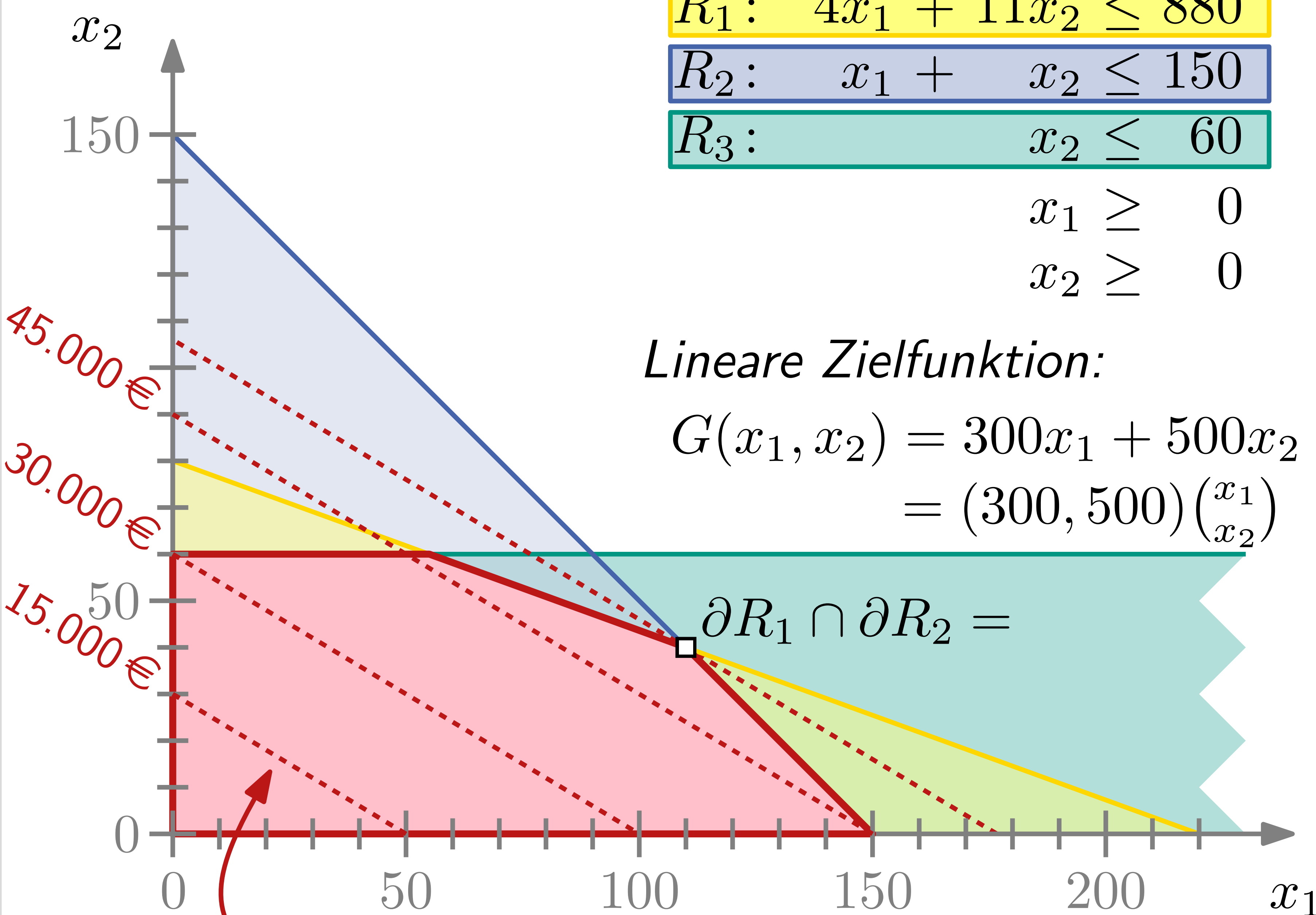
Lösung

Lineare Beschränkungen:

$$
\begin{aligned}
R_1:&\quad 4x_1 + 11x_2 \le 880\\
R_2:&\quad x_1 + x_2 \le 150\\
R_3:&\quad x_2 \le 60\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0
\end{aligned}
$$

Lineare Zielfunktion:

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



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

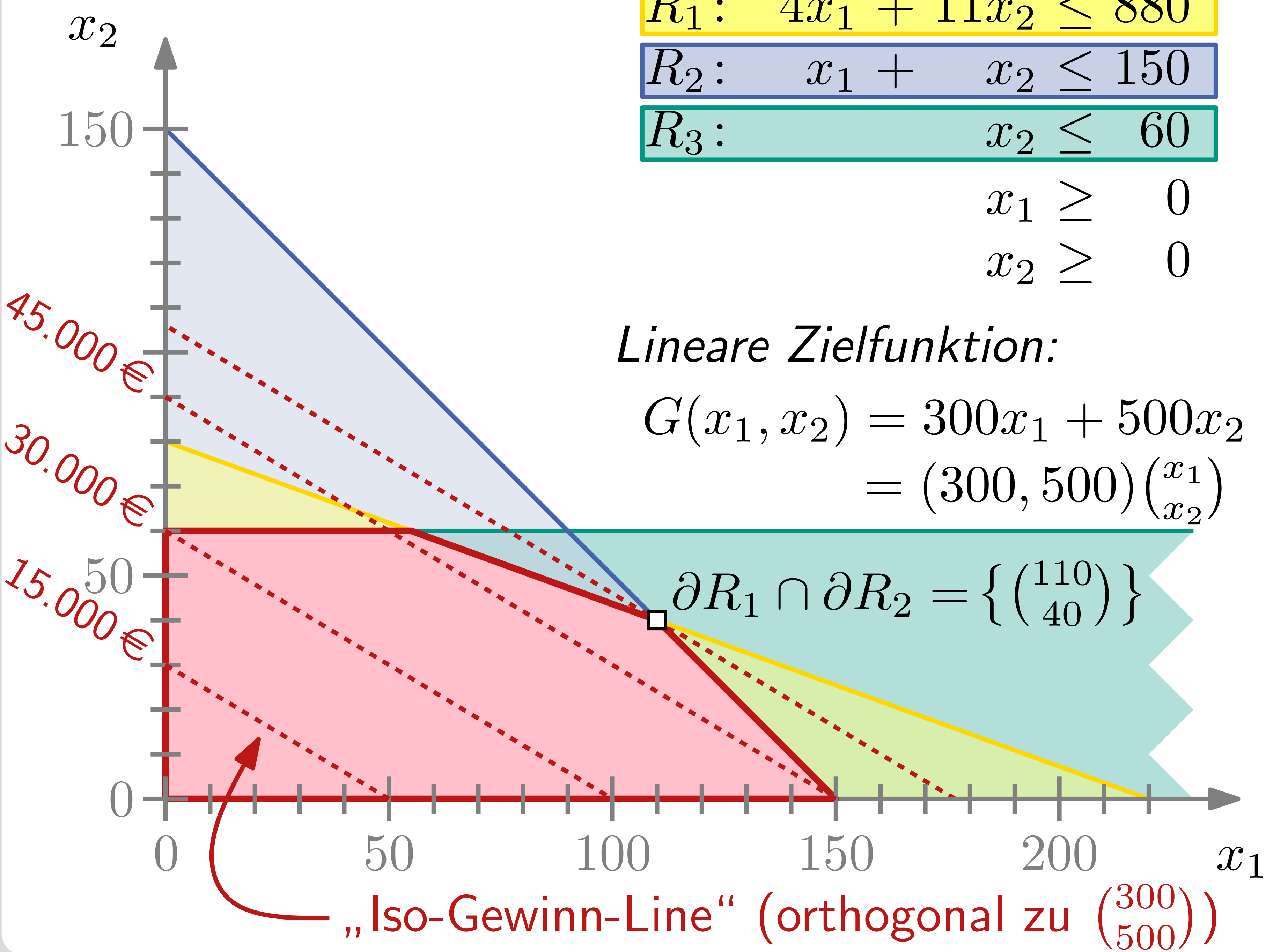
Lösung

Lineare Beschränkungen:

$$\begin{aligned} R_1: \quad 4x_1 + 11x_2 &\le 880 \\ R_2: \quad x_1 + x_2 &\le 150 \\ R_3: \quad x_2 &\le 60 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

Lineare Zielfunktion:

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



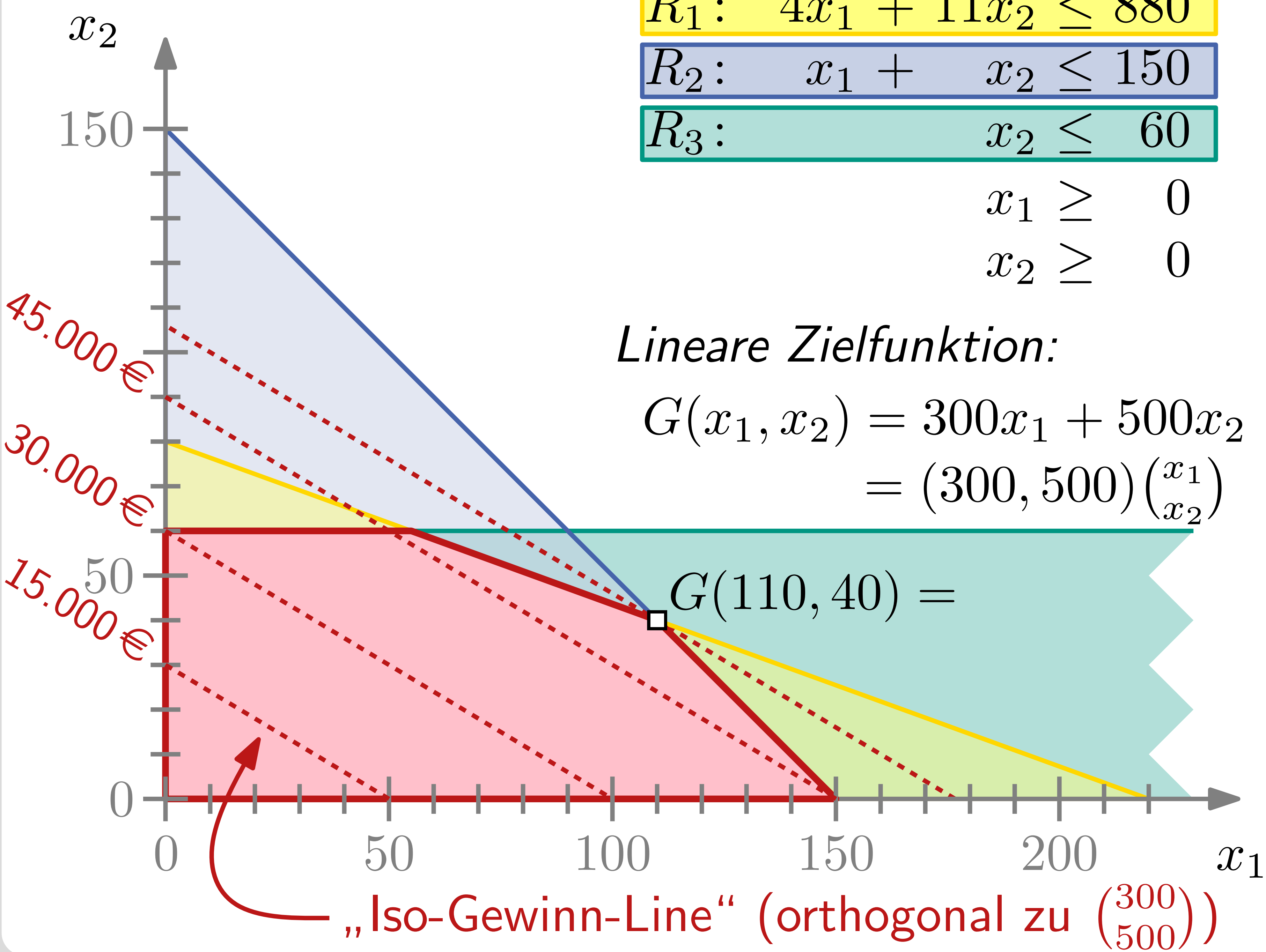
Lösung

Lineare Beschränkungen:

$$\begin{aligned} R_1: \quad 4x_1 + 11x_2 &\le 880 \\ R_2: \quad x_1 + x_2 &\le 150 \\ R_3: \quad x_2 &\le 60 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

Lineare Zielfunktion:

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



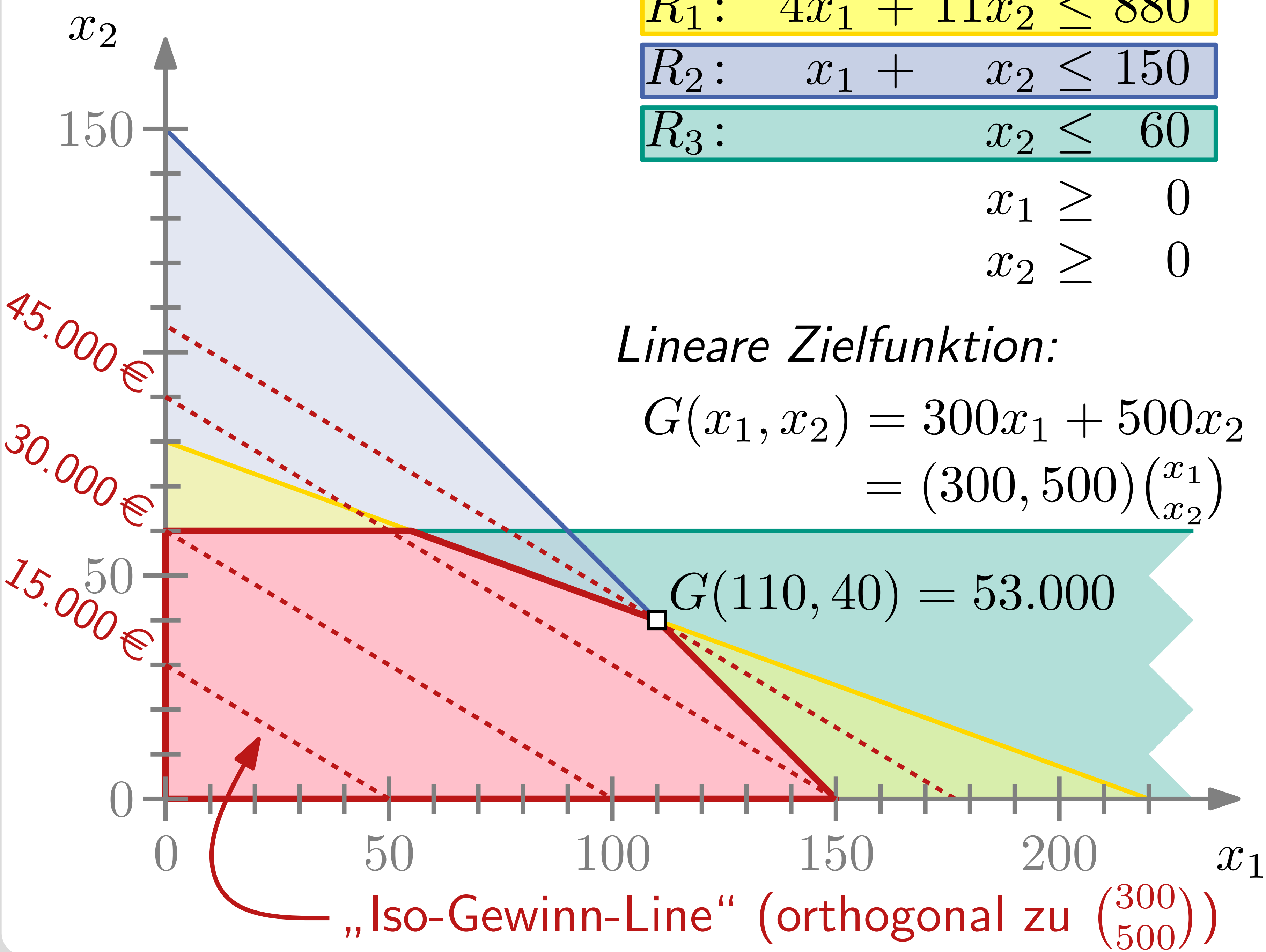
Lösung

Lineare Beschränkungen:

$$
\begin{aligned}
R_1:&\quad 4x_1 + 11x_2 \le 880\\
R_2:&\quad x_1 + x_2 \le 150\\
R_3:&\quad x_2 \le 60\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0
\end{aligned}
$$

Lineare Zielfunktion:

$$
G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}
$$



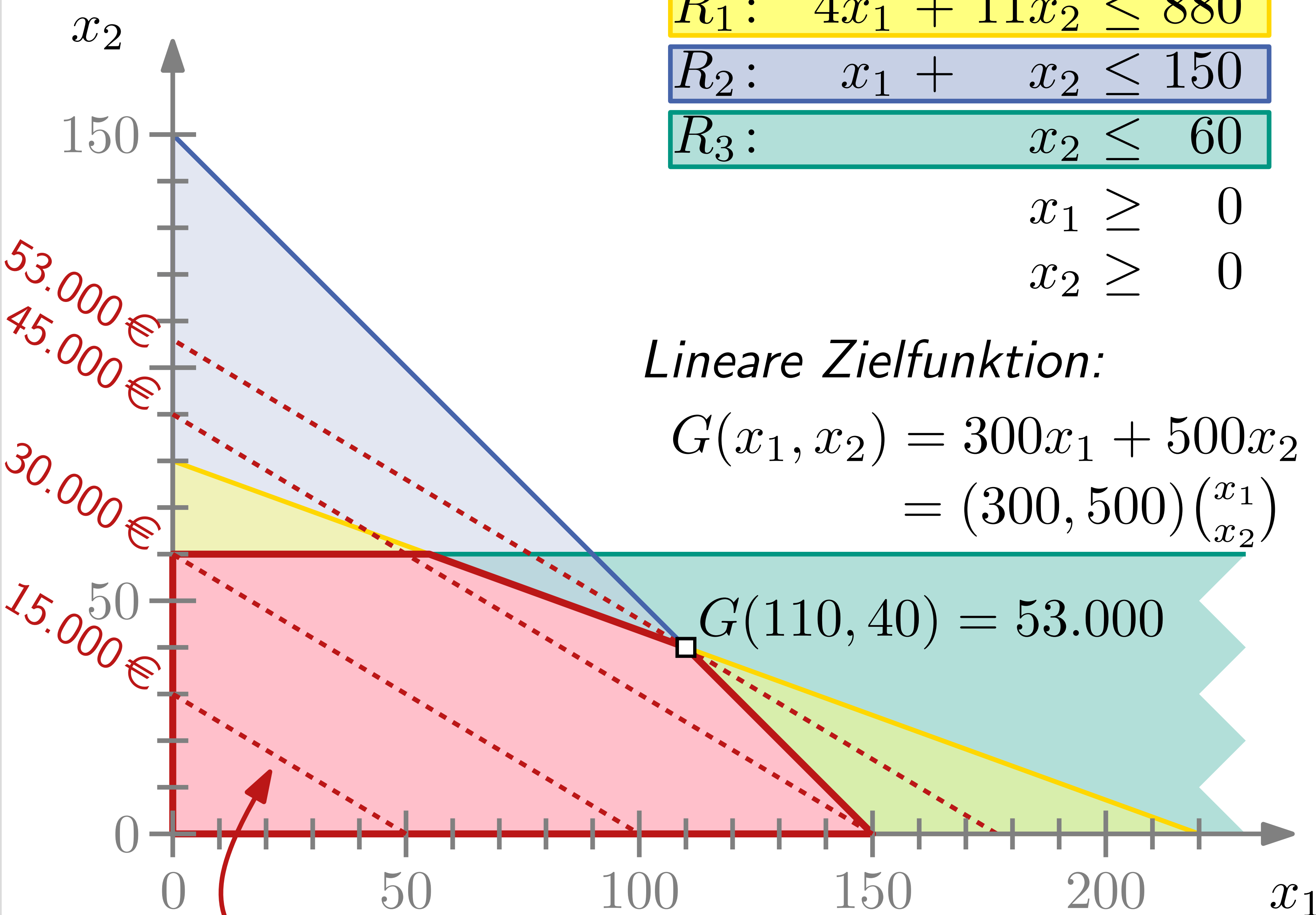
Lösung

Lineare Beschränkungen:

$$R_1: \quad 4x_1 + 11x_2 \leq 880$$
$$R_2: \quad x_1 + x_2 \leq 150$$
$$R_3: \quad x_2 \leq 60$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

Lineare Zielfunktion:

$$G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}$$



„Iso-Gewinn-Line" (orthogonal zu $\binom{300}{500}$)

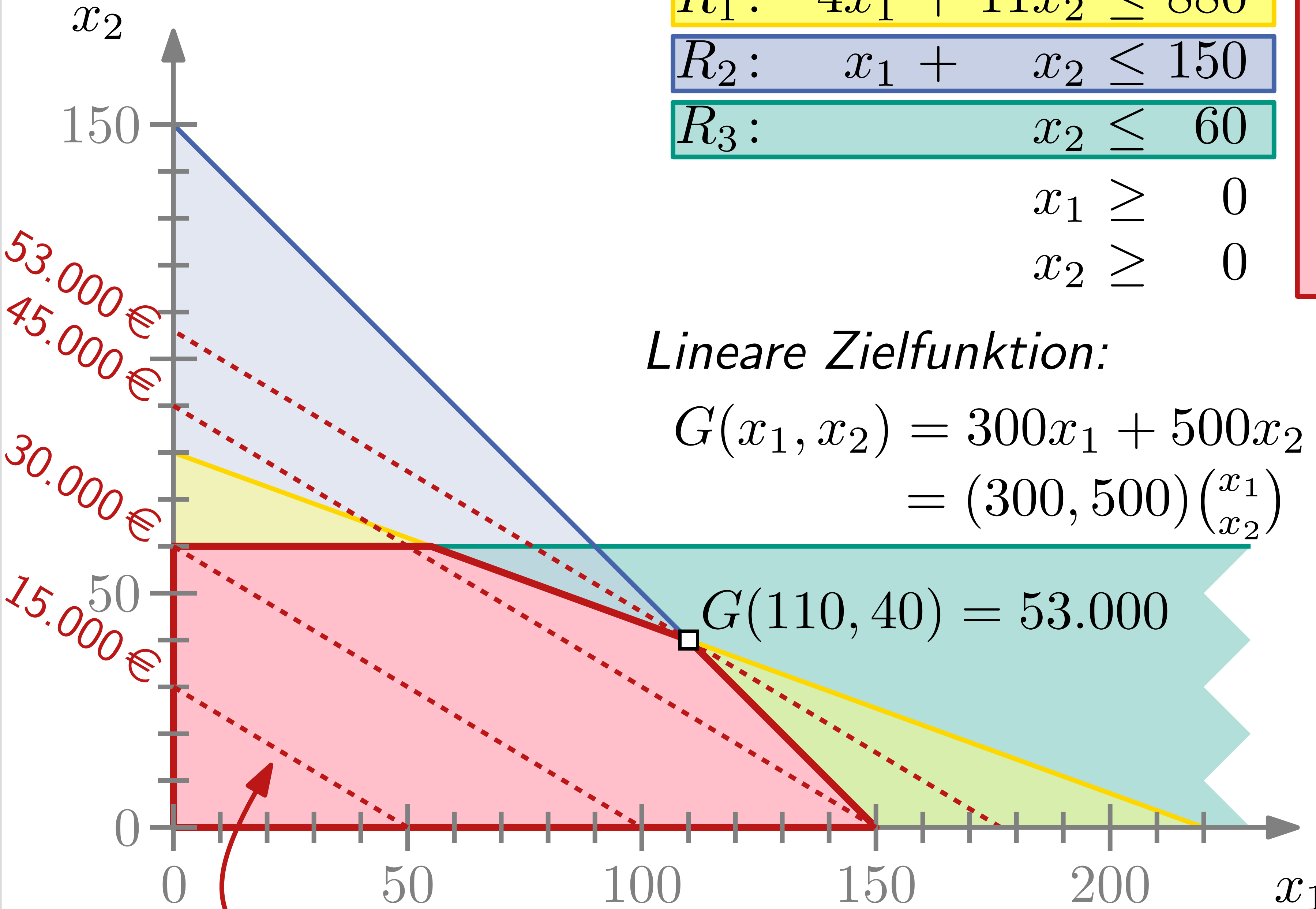
Lösung

Lineare Beschränkungen:

$$\begin{aligned}
R_1:&\quad 4x_1 + 11x_2 \le 880\\
R_2:&\quad x_1 + x_2 \le 150\\
R_3:&\quad x_2 \le 60\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0
\end{aligned}$$

Lineare Zielfunktion:

$$G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}$$



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

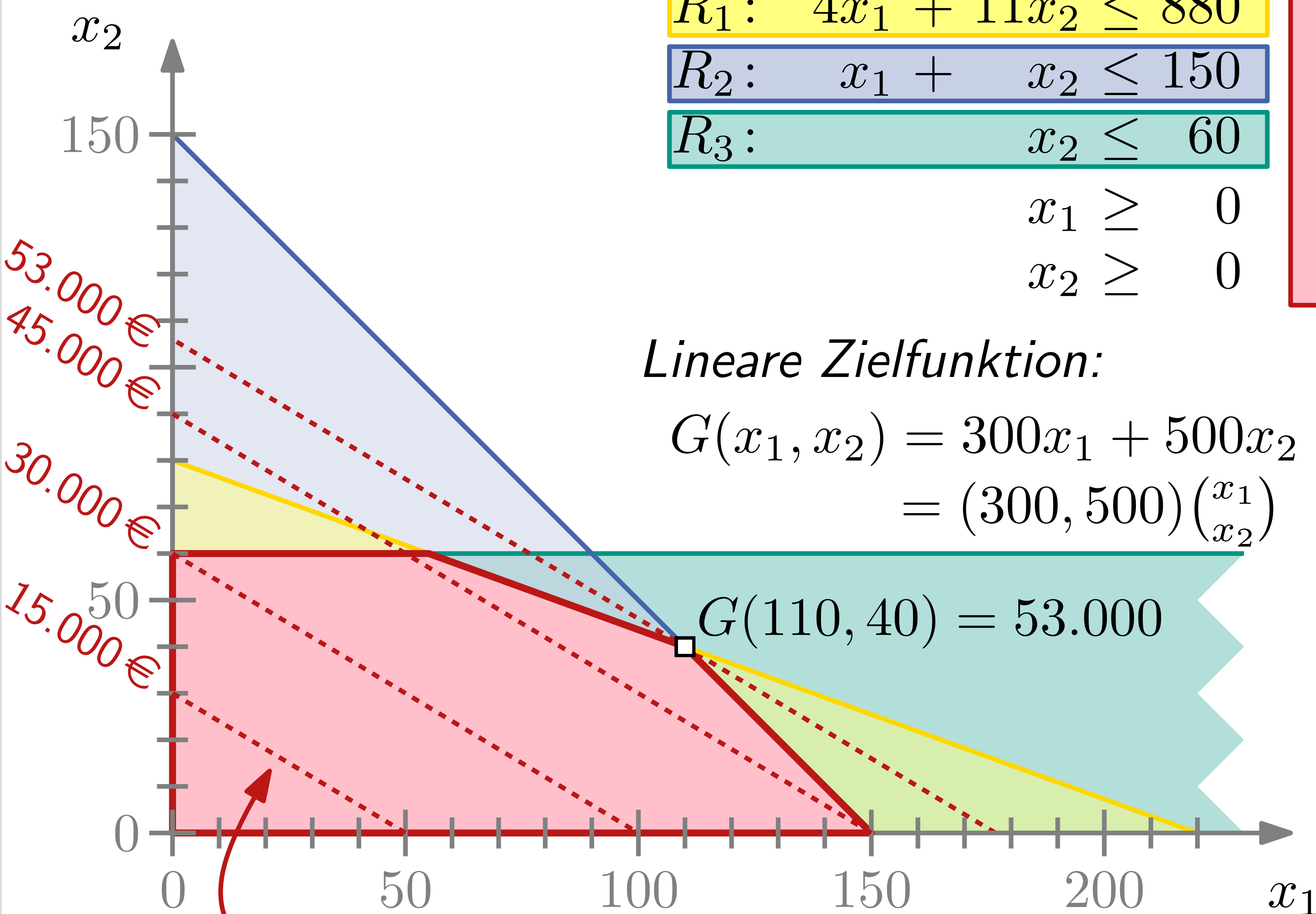
Lösung

Lineare Beschränkungen:

$$
\begin{aligned}
R_1: \quad 4x_1 + 11x_2 &\le 880\\
R_2: \quad x_1 + x_2 &\le 150\\
R_3: \quad x_2 &\le 60\\
x_1 &\ge 0\\
x_2 &\ge 0
\end{aligned}
\qquad Ax \le b
$$

Lineare Zielfunktion:

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



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

Lösung

Lineare Beschränkungen:

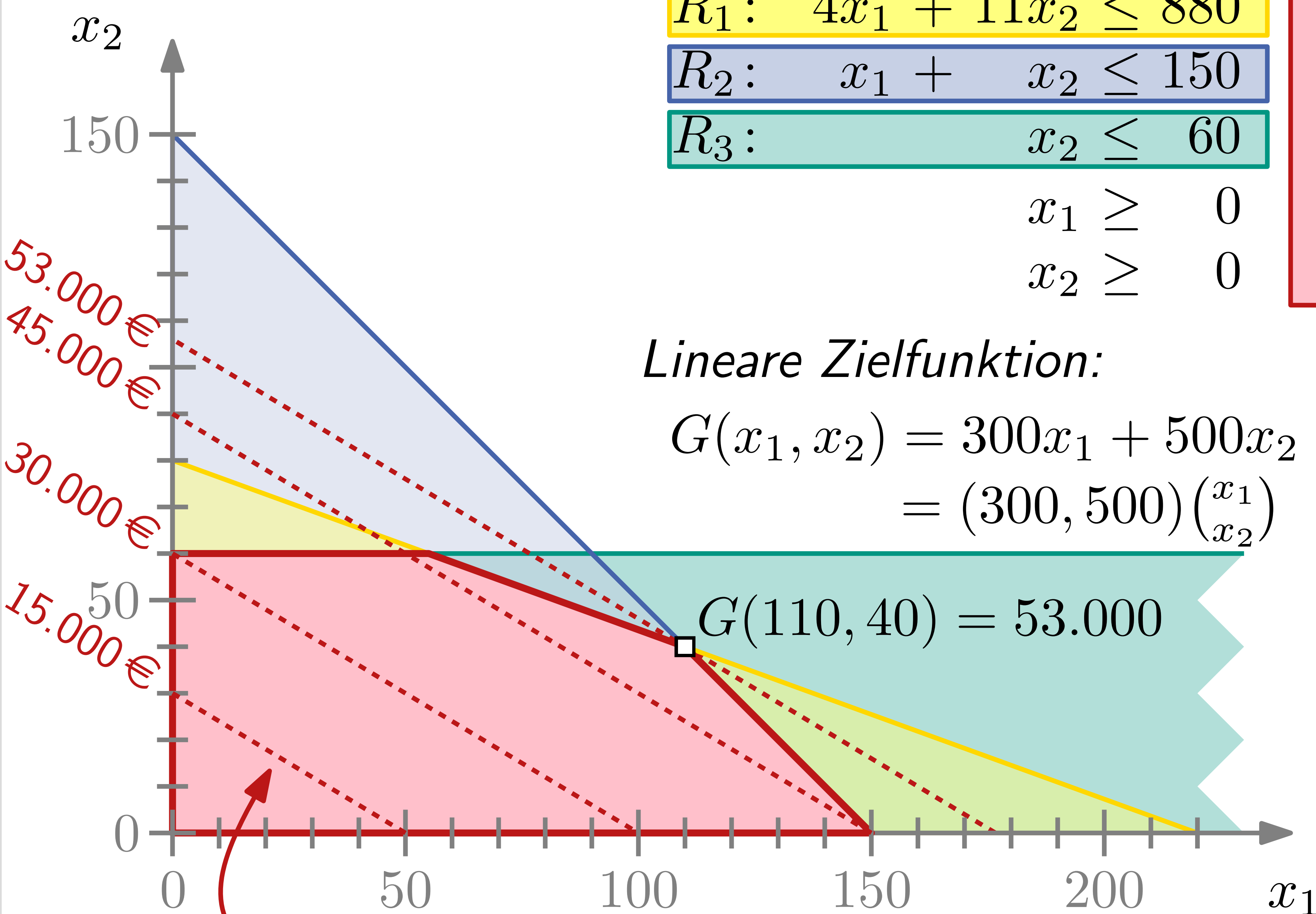
$$\begin{aligned}
R_1:&\quad 4x_1 + 11x_2 \le 880\\
R_2:&\quad x_1 + x_2 \le 150\\
R_3:&\quad x_2 \le 60\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0
\end{aligned}$$

$$Ax \le b$$

$$x \ge 0$$

Lineare Zielfunktion:

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



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

Lösung

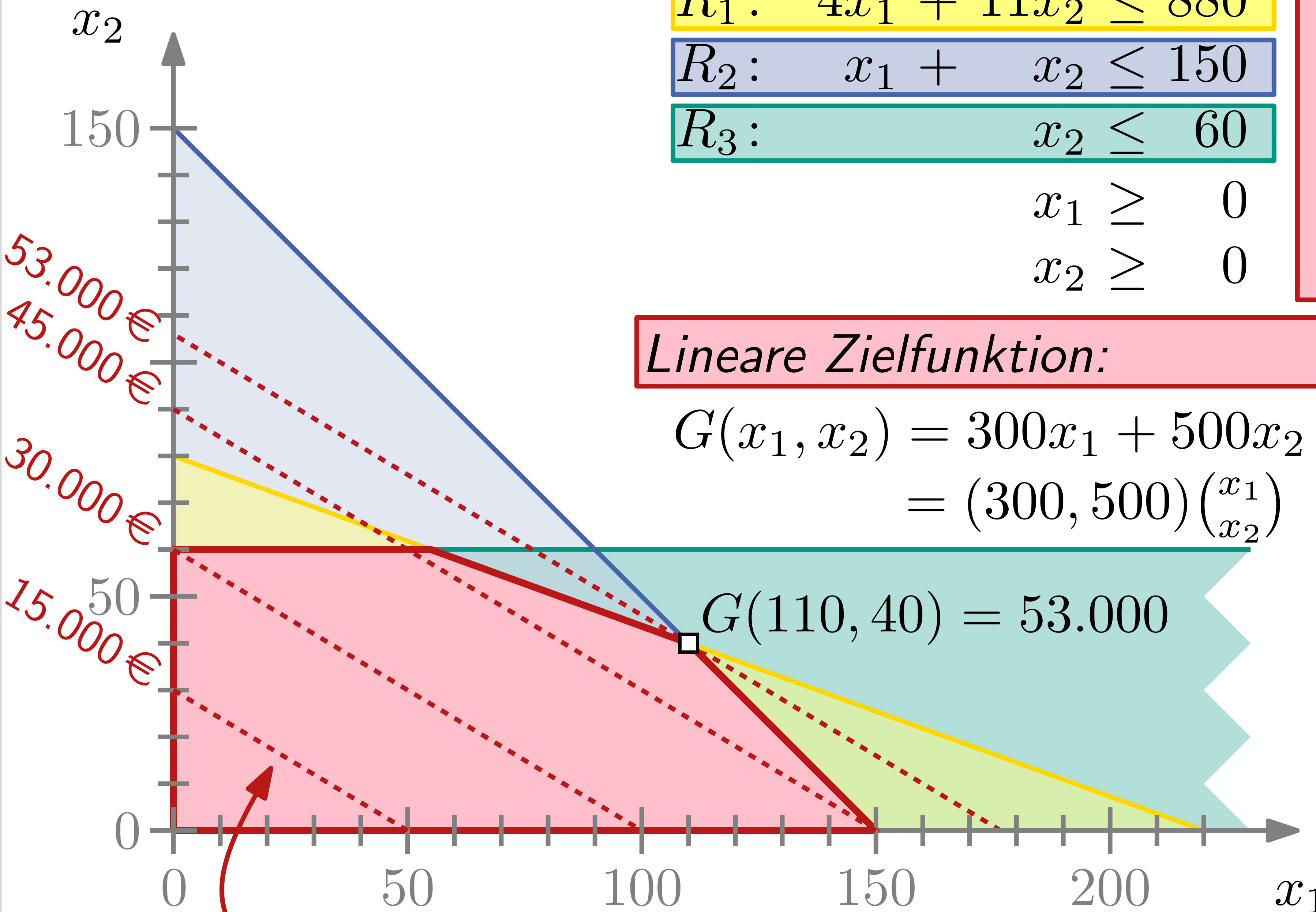
Lineare Beschränkungen:

$$R_1: \quad 4x_1 + 11x_2 \leq 880$$
$$R_2: \quad x_1 + x_2 \leq 150$$
$$R_3: \quad x_2 \leq 60$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

$$Ax \leq b$$
$$x \geq 0$$

Lineare Zielfunktion:

$$G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}$$



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

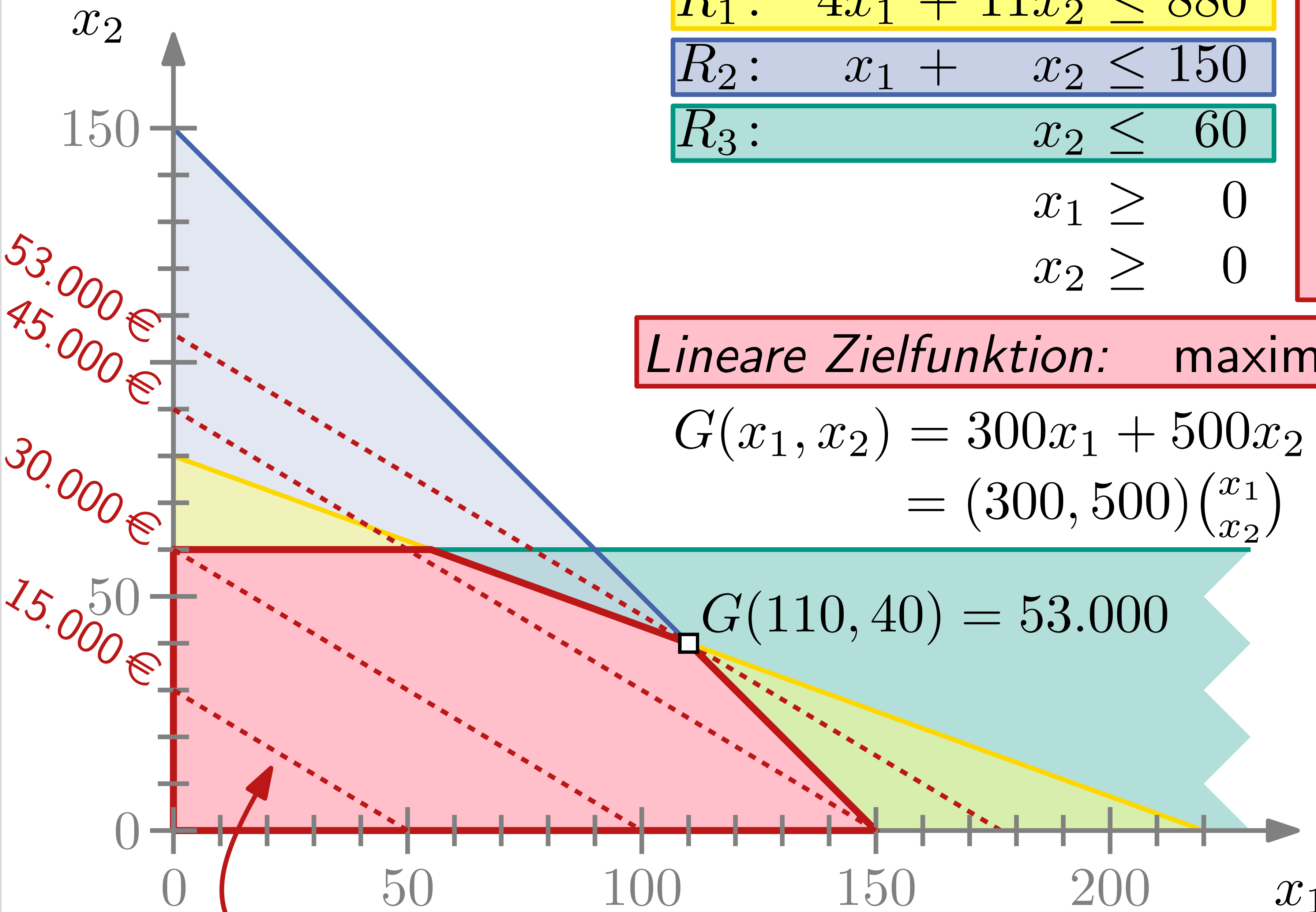
Lösung

Lineare Beschränkungen:

$$
\begin{aligned}
R_1:&\quad 4x_1 + 11x_2 \le 880\\
R_2:&\quad x_1 + x_2 \le 150\\
R_3:&\quad x_2 \le 60\\
&\quad x_1 \ge 0\\
&\quad x_2 \ge 0
\end{aligned}
\qquad
\begin{aligned}
Ax &\le b\\
x &\ge 0
\end{aligned}
$$

Lineare Zielfunktion: maximiere $c^{\mathrm{T}}x$

$$
G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}
$$



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

Lösung

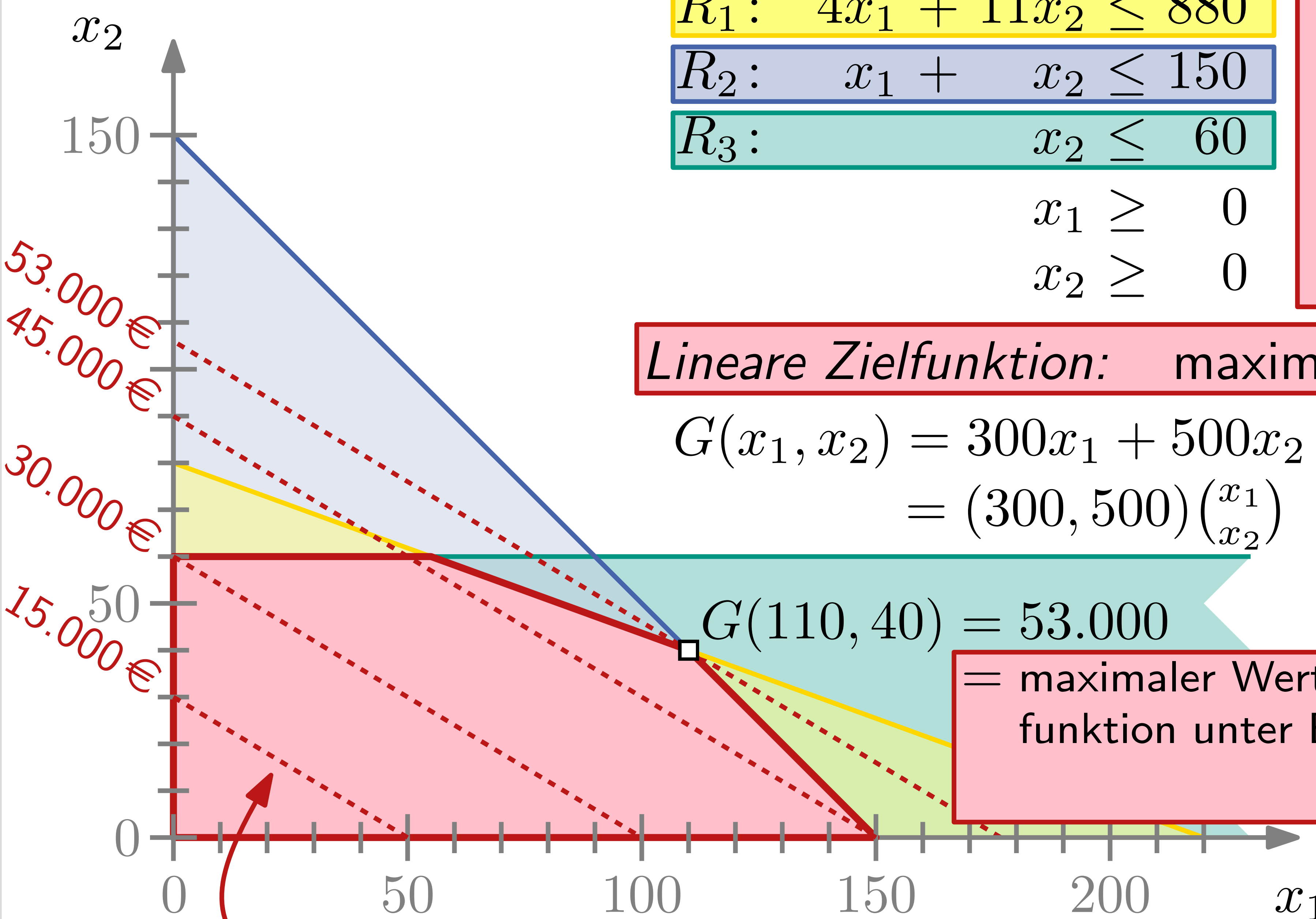
Lineare Beschränkungen:

$$R_1: \quad 4x_1 + 11x_2 \leq 880$$
$$R_2: \quad x_1 + x_2 \leq 150$$
$$R_3: \quad x_2 \leq 60$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

$$Ax \leq b$$
$$x \geq 0$$

Lineare Zielfunktion: maximiere $c^{\mathrm{T}}x$

$$G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}$$



= maximaler Wert der Zielfunktion unter Beschränk.

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

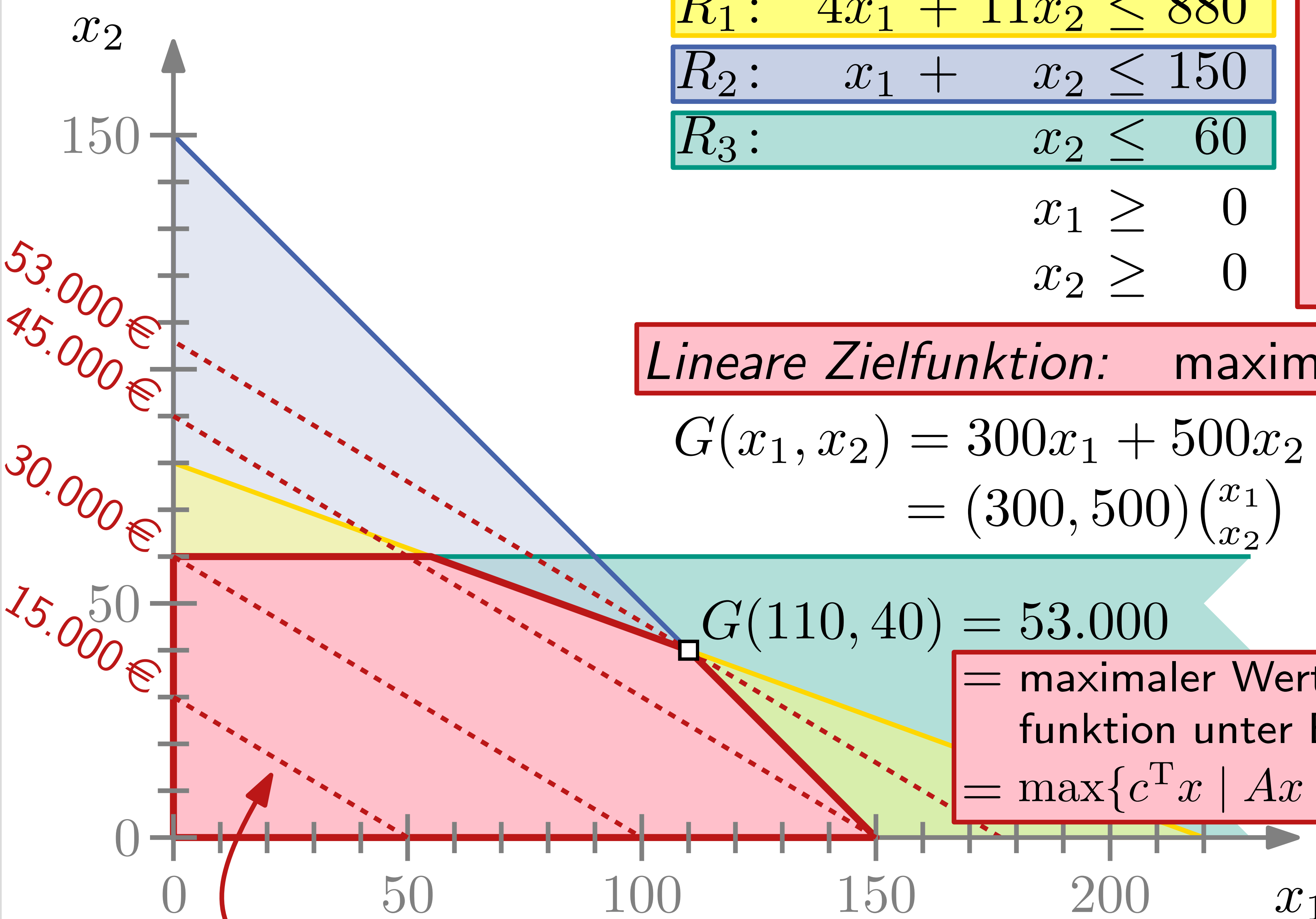
Lösung

Lineare Beschränkungen:

$$\begin{aligned} R_1: \quad 4x_1 + 11x_2 &\le 880 \\ R_2: \quad x_1 + x_2 &\le 150 \\ R_3: \quad x_2 &\le 60 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned} \qquad \begin{aligned} Ax &\le b \\ x &\ge 0 \end{aligned}$$

Lineare Zielfunktion: maximiere $c^{\mathrm{T}}x$

$$G(x_1, x_2) = 300x_1 + 500x_2 = (300, 500)\binom{x_1}{x_2}$$



$G(110, 40) = 53.000$

= maximaler Wert der Zielfunktion unter Beschränk.

$= \max\{c^{\mathrm{T}}x \mid Ax \le b, x \ge 0\}$

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

Lineares Programmieren

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

$$\text{maximiere} \quad c_1x_1 + c_2x_2 + \cdots + c_dx_d$$

$$\begin{array}{rrcl} \text{unter den NB} & a_{1,1}x_1 + \cdots + a_{1,d}x_d & \leq & b_1 \\ & a_{2,1}x_1 + \cdots + a_{2,d}x_d & \leq & b_2 \\ & & \vdots & \\ & a_{n,1}x_1 + \cdots + a_{n,d}x_d & \leq & b_n \end{array}$$

Lineares Programmieren

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

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

- 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 $\max\{c^{\mathrm{T}}x \mid Ax \leq b, x \geq 0\}$.
- 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).

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 $d = 2$

Algorithmen für LPs

Viele Algorithmen zum Lösen von LPs in der Praxis existieren:

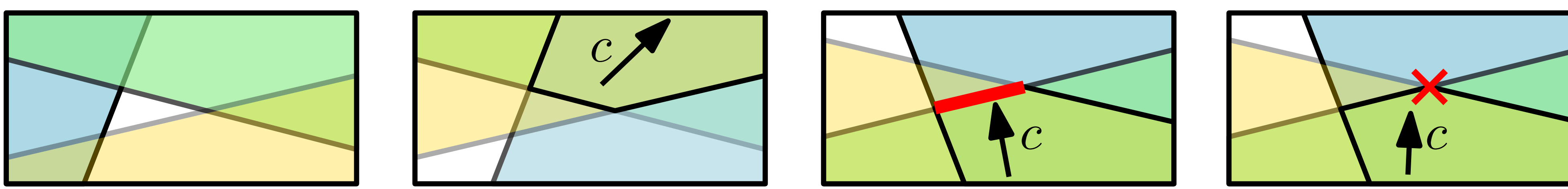
- Simplex-Algorithmus [Dantzig, 1947]
- Ellipsoid-Methode [Khatchiyan, 1979]
- Innere-Punkt-Methode [Karmarkar, 1979]

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

Heute: Spezialfall $d = 2$

Möglichkeiten für den Lösungsraum

$\bigcap H$ beschränkt



$\bigcap H = \emptyset$ unlösbar	$\bigcap H$ unbeschränkt in Richtung c	Lösung nicht eindeutig	eindeutige Lösung

Erste Variante

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

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

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
  $(H_1, H_2) \leftarrow$ SplitInHalves(H)
  $C_1 \leftarrow$ IntersectHalfplanes(H_1)
  $C_2 \leftarrow$ IntersectHalfplanes(H_2)
  $C \leftarrow$ IntersectConvexRegions(C_1, C_2)
return C

Schnitt konvexer Regionen

Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

Schnitt konvexer Regionen

Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

- betrachte jeweils linke und rechte Grenze von C_1 und C_2

Schnitt konvexer Regionen

Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten

Schnitt konvexer Regionen

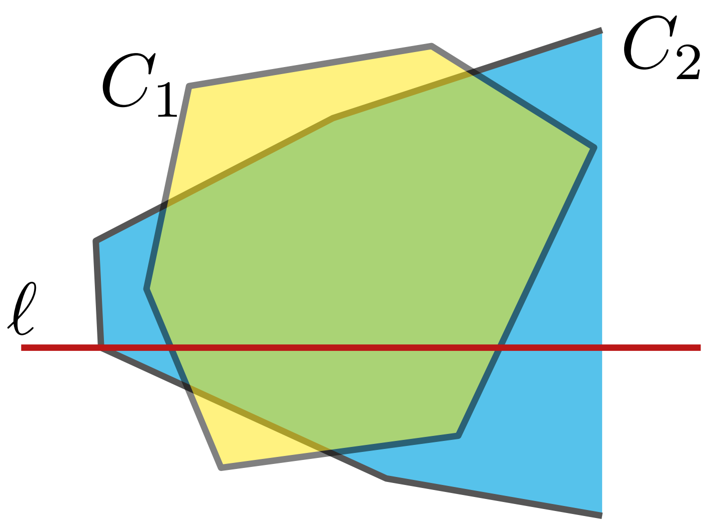
Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

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

Schnitt konvexer Regionen

Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

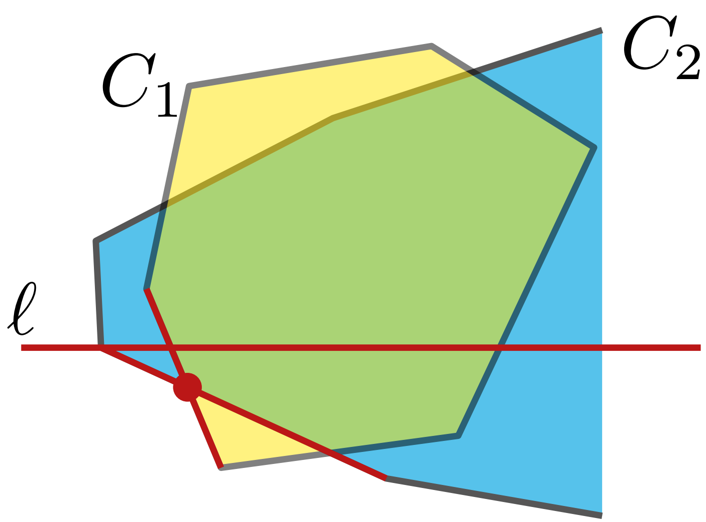
- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten
- Knoten in $C_1 \cup C_2$ definieren Events, Behandlung je nach Typ der beginnenden Kante in $O(1)$ Zeit



Schnitt konvexer Regionen

Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

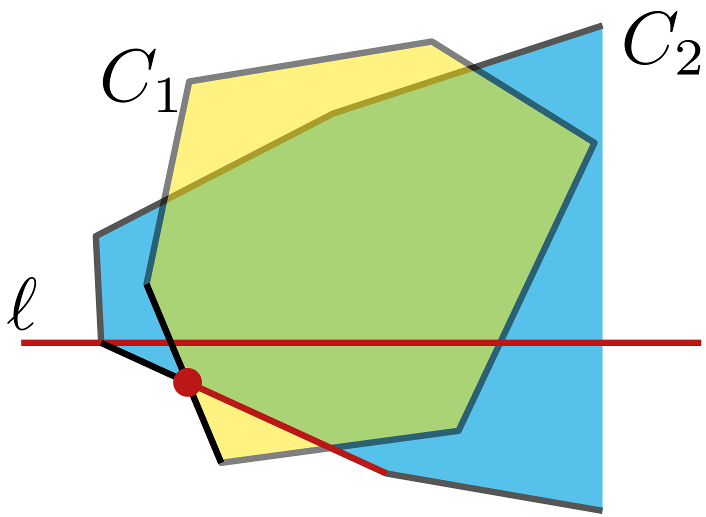
- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten
- Knoten in $C_1 \cup C_2$ definieren Events, Behandlung je nach Typ der beginnenden Kante in $O(1)$ Zeit



Schnitt konvexer Regionen

Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

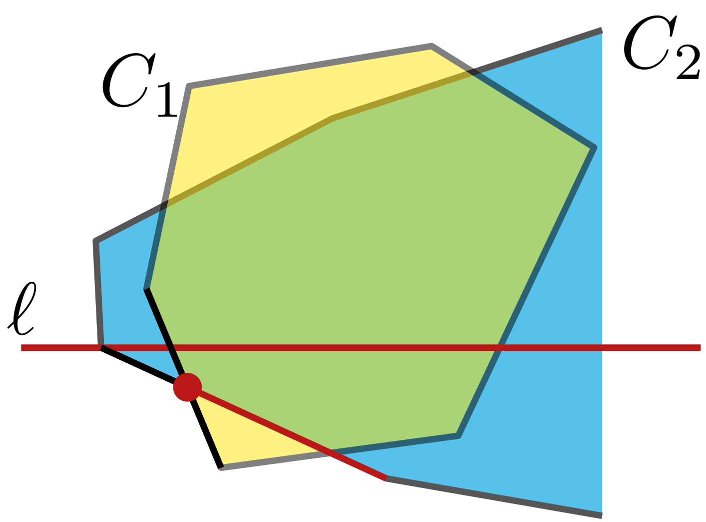
- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten
- Knoten in $C_1 \cup C_2$ definieren Events, Behandlung je nach Typ der beginnenden Kante in $O(1)$ Zeit



Schnitt konvexer Regionen

Methode IntersectConvexRegions kann mit Sweep-Line Verfahren implementiert werden:

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



Satz:
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
  $(H_1, H_2) \leftarrow$ SplitInHalves(H)
  $C_1 \leftarrow$ IntersectHalfplanes(H_1)
  $C_2 \leftarrow$ IntersectHalfplanes(H_2)
  $C \leftarrow$ IntersectConvexRegions(C_1, C_2)
return C

Aufgabe: Welche Laufzeit hat IntersectHalfplanes(H)?

Laufzeit von IntersectHalfplanes(H)

IntersectHalfplanes(H)

if $|H| = 1$ **then**
  $C \leftarrow H$
else
  $(H_1, H_2) \leftarrow$ SplitInHalves(H)
  $C_1 \leftarrow$ IntersectHalfplanes(H_1)
  $C_2 \leftarrow$ IntersectHalfplanes(H_2)
  $C \leftarrow$ IntersectConvexRegions(C_1, C_2)
return C

Aufgabe: Welche Laufzeit hat IntersectHalfplanes(H)?

Rekurrenzgleichung

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

Laufzeit von IntersectHalfplanes(H)

IntersectHalfplanes(H)

if $|H| = 1$ **then**
  $C \leftarrow H$
else
  $(H_1, H_2) \leftarrow$ SplitInHalves(H)
  $C_1 \leftarrow$ IntersectHalfplanes(H_1)
  $C_2 \leftarrow$ IntersectHalfplanes(H_2)
  $C \leftarrow$ 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 $O(n \log n)$

Laufzeit von IntersectHalfplanes(H)

IntersectHalfplanes(H)

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

- zulässiger Bereich $\bigcap H$ lässt sich in $O(n \log n)$ Zeit berechnen
- $\bigcap H$ 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.

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

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?

Beschränkte LPs

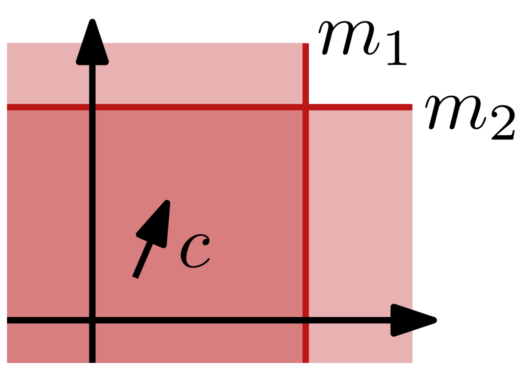
Idee: Statt gesamtes zulässiges Polygon zu berechnen suche inkrementell nach optimaler Ecke.

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

Wie kann man unbeschränkte zulässige Gebiete umgehen?

Für einen ausreichend großen Wert M definiere Halbebenen

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



Beschränkte LPs

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

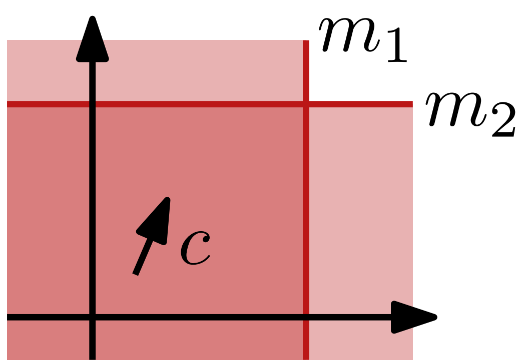
Invariante: aktuell beste Lösung ist eindeutige Ecke des zulässigen aktuellen Polygons

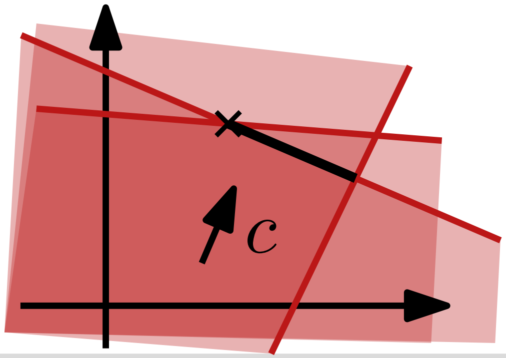
Wie kann man unbeschränkte zulässige Gebiete umgehen?

Gibt es mehrere Optima, wähle lexikographisch kleinsten Punkt!

Für einen ausreichend großen Wert M definiere Halbebenen

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





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

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

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

Eigenschaften

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

Eigenschaften

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

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

Eigenschaften

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

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

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

(i) Falls $v_{i-1} \in h_i$ gilt $v_i = v_{i-1}$,

(ii) sonst ist entweder $C_i = \emptyset$ oder $v_i \in \ell_i$.

Eigenschaften

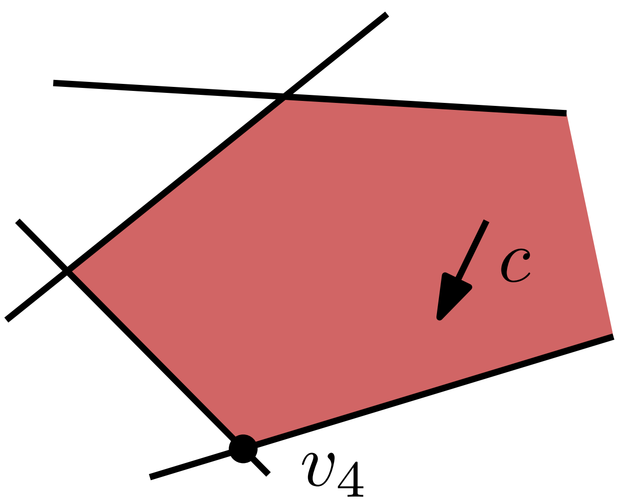
- jede Region C_i hat eine eindeutige optimale Ecke v_i
- es gilt die Inklusionsbeziehung $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

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

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

(i) Falls $v_{i-1} \in h_i$ gilt $v_i = v_{i-1}$,

(ii) sonst ist entweder $C_i = \emptyset$ oder $v_i \in \ell_i$.



Eigenschaften

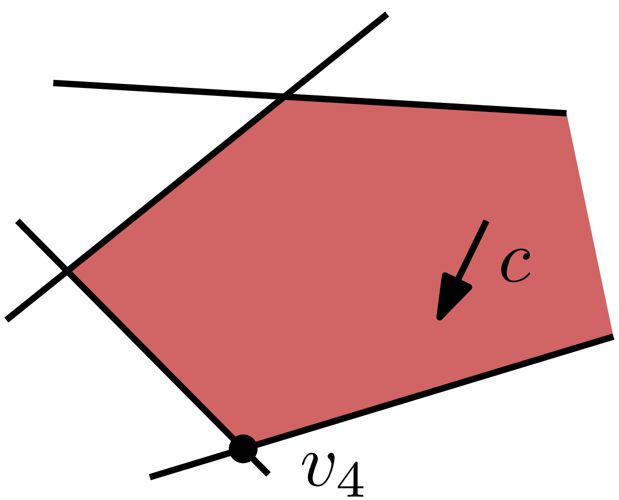
- jede Region C_i hat eine eindeutige optimale Ecke v_i
- es gilt die Inklusionsbeziehung $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

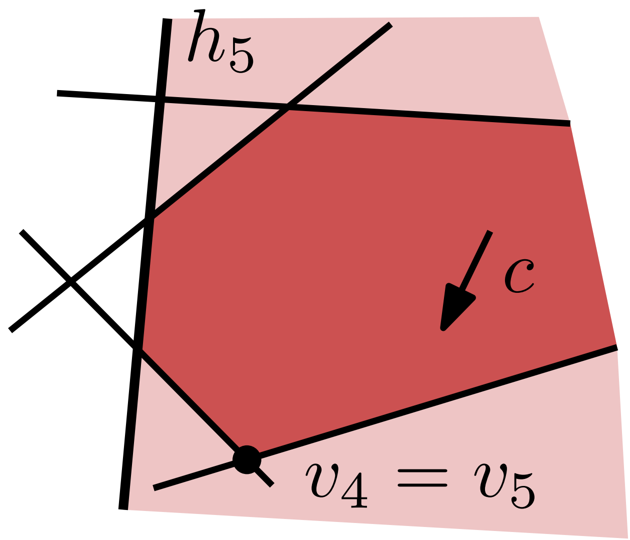
Wie ändert sich die optimale Ecke v_{i-1} wenn man h_i hinzufügt?

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

(i) Falls $v_{i-1} \in h_i$ gilt $v_i = v_{i-1}$,

(ii) sonst ist entweder $C_i = \emptyset$ oder $v_i \in \ell_i$.





Eigenschaften

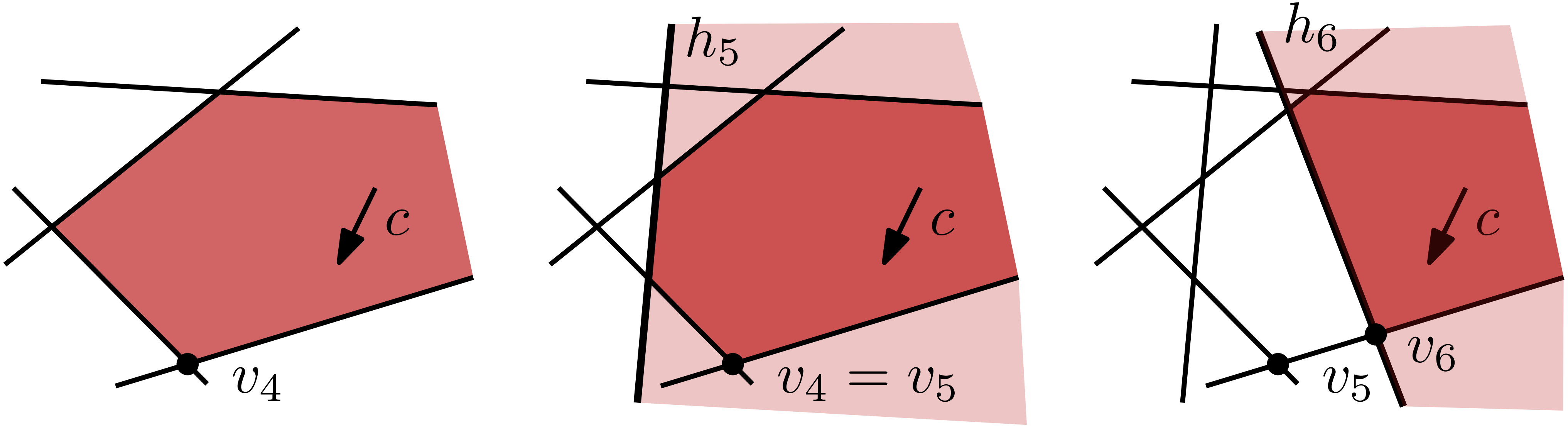
- jede Region C_i hat eine eindeutige optimale Ecke v_i
- es gilt die Inklusionsbeziehung $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

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

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

(i) Falls $v_{i-1} \in h_i$ gilt $v_i = v_{i-1}$,

(ii) sonst ist entweder $C_i = \emptyset$ oder $v_i \in \ell_i$.



Eindimensionales LP

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

- parametrisiere $\ell_i : y = ax + b$

Eindimensionales LP

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

- parametrisiere $\ell_i : y = ax + b$
- definiere neue Zielfunktion $f_c^i(x) = c^T \binom{x}{ax+b}$

Eindimensionales LP

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

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

Eindimensionales LP

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

- parametrisiere $\ell_i : y = ax + b$
- definiere neue Zielfunktion $f_c^i(x) = c^T \binom{x}{ax+b}$
- 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:

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

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

Eindimensionales LP

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

- parametrisiere $\ell_i : y = ax + b$
- definiere neue Zielfunktion $f_c^i(x) = c^T \binom{x}{ax+b}$
- 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:

maximiere $\quad f_c^i(x) = c_x x + c_y(ax + b)$

mit NB $\quad$ $x \leq \sigma_x(\ell_j, \ell_i)$ falls $\ell_i \cap h_j$ nach rechts beschr.
$x \geq \sigma_x(\ell_j, \ell_i)$ falls $\ell_i \cap h_j$ nach links beschr.

Wie löst man dieses LP? Welche Laufzeit?

Eindimensionales LP

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

- parametrisiere $\ell_i : y = ax + b$
- definiere neue Zielfunktion $f_c^i(x) = c^T \binom{x}{ax+b}$
- 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:

maximiere $\quad f_c^i(x) = c_x x + c_y(ax+b)$

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

Lemma: 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

2dBoundedLP(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**
        $v_i \leftarrow v_{i-1}$
    else
        $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
        if $v_i =$ nil **then**
            return unlösbar
return v_n

Inkrementeller Algorithmus

2dBoundedLP(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**
    $v_i \leftarrow v_{i-1}$ $\quad O(1)$
  else
    $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
    if $v_i =$ nil **then** $\quad O(i)$
      return unlösbar
return v_n

Inkrementeller Algorithmus

2dBoundedLP(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
    $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
    if $v_i =$ nil **then** $O(i)$
      return unlösbar
return v_n

worst-case Laufzeit:
$T(n) = \sum_{i=1}^{n} O(i) = O(n^2)$

Inkrementeller Algorithmus

2dBoundedLP(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
    $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
    if $v_i =$ nil **then** $O(i)$
      return unlösbar
return v_n

worst-case Laufzeit:
$T(n) = \sum_{i=1}^{n} O(i) = O(n^2)$

Frage: Kann wirklich n-mal Fall (ii) auftreten?

Inkrementeller Algorithmus

2dBoundedLP(H, c, m_1, m_2)

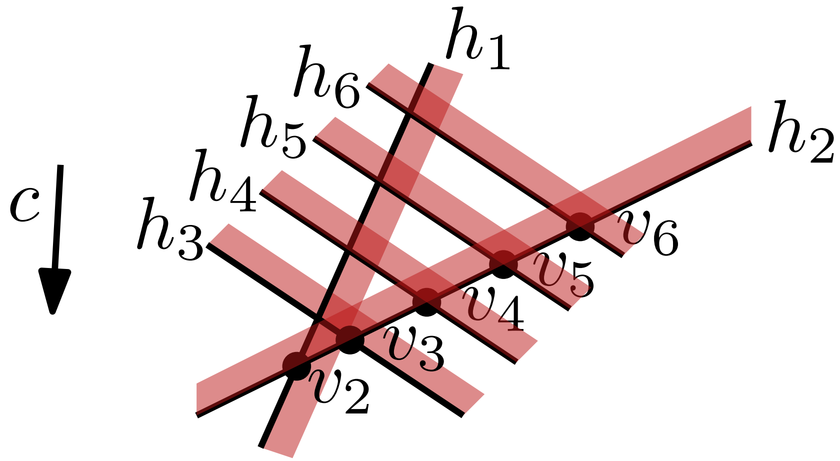
$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ eindeutige Ecke von C_0
for $i \leftarrow 1$ **to** n **do**
  if $v_{i-1} \in h_i$ **then**
    $v_i \leftarrow v_{i-1}$ $O(1)$
  else
    $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
    if $v_i =$ nil **then** $O(i)$
      return unlösbar
return v_n

worst-case Laufzeit:
$T(n) = \sum_{i=1}^{n} O(i) = O(n^2)$

Lemma: Algorithmus 2dBoundedLP benötigt $\Theta(n^2)$ Laufzeit um ein LP mit n Nebenbed. und 2 Variablen zu lösen.

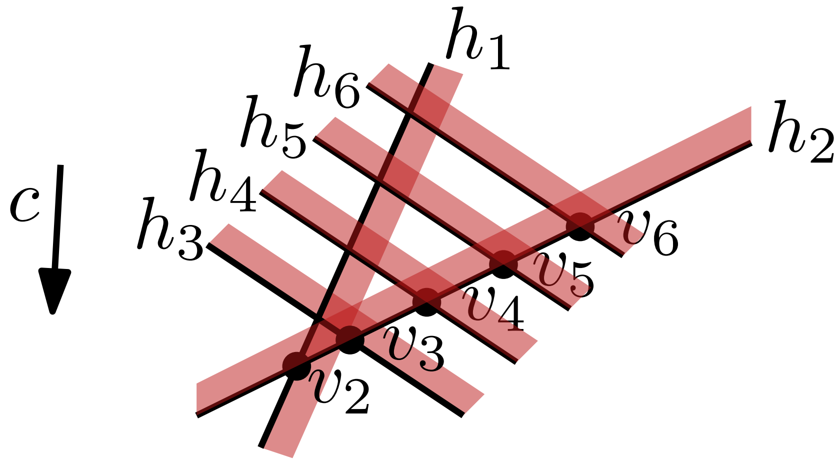
Gibt es einen Ausweg?

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



Gibt es einen Ausweg?

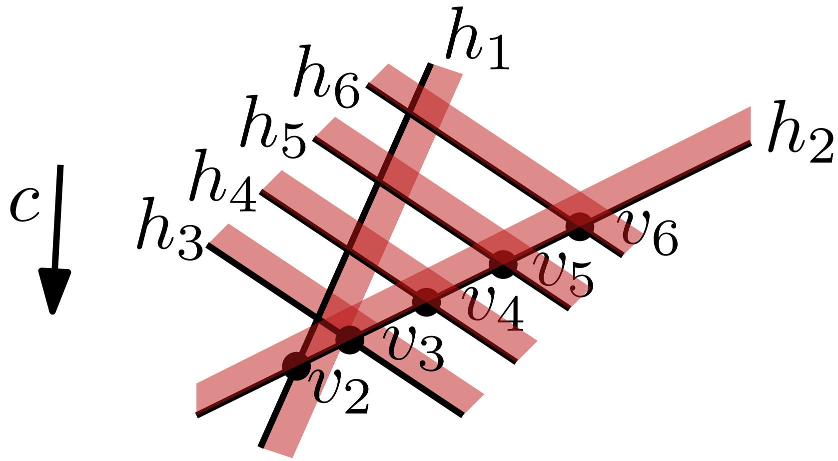
Beob.: Nicht die Halbebenen H sind problematisch für die Laufzeit, sondern die Reihenfolge der Abarbeitung.



Wie findet man (schnell) eine gute Reihenfolge?

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?

Zufällig!

Randomisierter Inkrementeller Algorithmus

2dRandomizedBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ eindeutige Ecke von C_0
$H \leftarrow$ **RandomPermutation**(H)
for $i \leftarrow 1$ **to** n **do**
    if $v_{i-1} \in h_i$ **then**
        $v_i \leftarrow v_{i-1}$
    else
        $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
        if $v_i =$ nil **then**
            return unlösbar
return v_n

Zufallspermutation

RandomPermutation(A)

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

Output: Array A, zufällig gleichverteilt permutiert

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

  $r \leftarrow$ Random(k)

  tausche $A[r]$ und $A[k]$

Zufallspermutation

RandomPermutation(A)

Input: Array $A[1 \dots n]$
Output: Array A, zufällig gleichverteilt permutiert
for $k \leftarrow n$ **to** 2 **do**
   $r \leftarrow$ Random(k)   Zufallszahl zw. 1 und k
   tausche $A[r]$ und $A[k]$

Zufallspermutation

RandomPermutation(A)

Input: Array $A[1 \dots n]$
Output: Array A, zufällig gleichverteilt permutiert
for $k \leftarrow n$ **to** 2 **do**
  $r \leftarrow$ Random(k) Zufallszahl zw. 1 und k
  tausche $A[r]$ und $A[k]$

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

Zufallspermutation

RandomPermutation(A)

Input: Array $A[1 \dots n]$
Output: Array A, zufällig gleichverteilt permutiert
for $k \leftarrow n$ **to** 2 **do**
  $r \leftarrow$ Random(k)
  tausche $A[r]$ und $A[k]$

Zufallszahl zw. 1 und k

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

Satz: Ein zweidimensionales LP mit n Halbebenen kann in erwartet $O(n)$ Laufzeit gelöst werden.

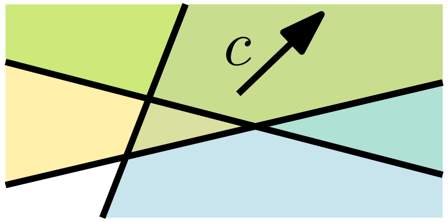
Unbeschränkte LPs

Bisher: künstliche Beschränkung von C durch m_1 und m_2

Unbeschränkte LPs

Bisher: künstliche Beschränkung von C durch m_1 und m_2

Jetzt: erkenne und behandle unbeschränkte LPs

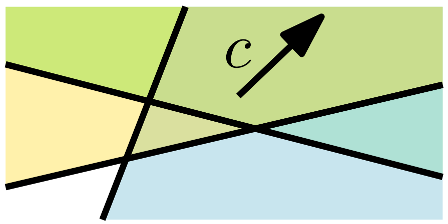


$\bigcap H$ unbeschränkt in Richtung c

Unbeschränkte LPs

Bisher: künstliche Beschränkung von C durch m_1 und m_2

Jetzt: erkenne und behandle unbeschränkte LPs



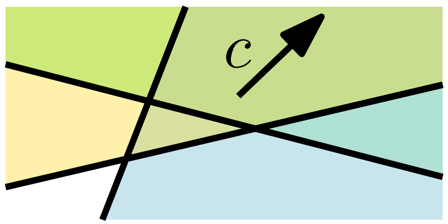
$\bigcap H$ unbeschränkt in Richtung c

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

Unbeschränkte LPs

Bisher: künstliche Beschränkung von C durch m_1 und m_2

Jetzt: erkenne und behandle unbeschränkte LPs



$\bigcap H$ unbeschränkt in Richtung c

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

Charakterisierung

Lemma: 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.

Charakterisierung

Lemma: 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

Test auf Unbeschränktheit

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

- betrachte $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- 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

Test auf Unbeschränktheit

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

- betrachte $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- 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 $\to$ 1d-LP

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

Test auf Unbeschränktheit

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

- betrachte $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- 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 ρ als Zeugen angeben.

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

Test auf Unbeschränktheit

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

- betrachte $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- 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

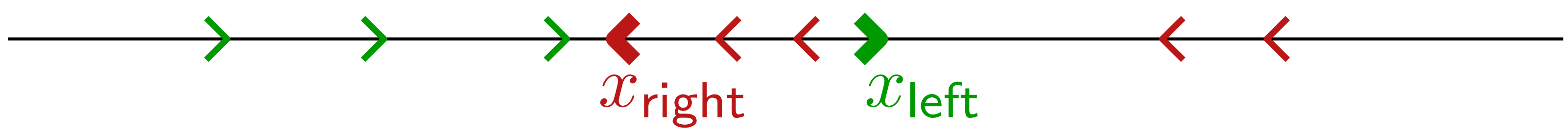
Wenn beide Schritte eine Lösung liefern ist (H, c) unbeschränkt und wir können einen Strahl ρ als Zeugen angeben.

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 beschränkt.

Zeugen für Beschränktheit

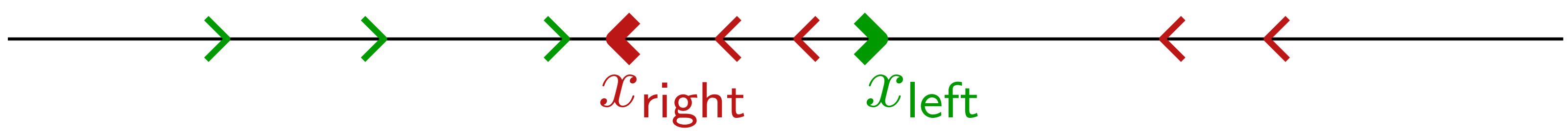
Beob.: Auch wenn LP aus Schritt 1 unlösbar ist, lassen sich die Informationen weiterverwenden!



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

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_{left} und x_{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

Algorithmus

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 ρ, unbeschränkt)

  else

    return unlösbar

else

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

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

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

Algorithmus

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 ρ, unbeschränkt)
  else
    return unlösbar
else
  $(h_1, h_2) \leftarrow$ Zeugen für Beschränktheit von (H, c)
  $\bar{H} \leftarrow H \setminus \{h_1, h_2\}$
  return 2dRandomizedBoundedLP$(\bar{H}, c, h_1, h_2)$

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

Diskussion

Lässt sich der zweidimensionale Algorithmus auch auf mehr Dimensionen verallgemeinern?

Diskussion

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.

Diskussion

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?

Diskussion

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 ρ und bei Unlösbarkeit max. drei Halbebenen mit leerem Schnitt.