Lineare Programmierung in niedrigen Dimensionen

Geometrische Algorithmen

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 €

$$G(x_1, x_2) = 30x_1 + 50x_2$$

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 €

$$G(x_1, x_2) = 30x_1 + 50x_2$$

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:

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

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 €

$$G(x_1, x_2) = 30x_1 + 50x_2$$

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:

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

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

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 €

$$G(x_1, x_2) = 30x_1 + 50x_2$$

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 $880 R_1$, $150 R_2$ und $60 R_3$. Damit gilt:

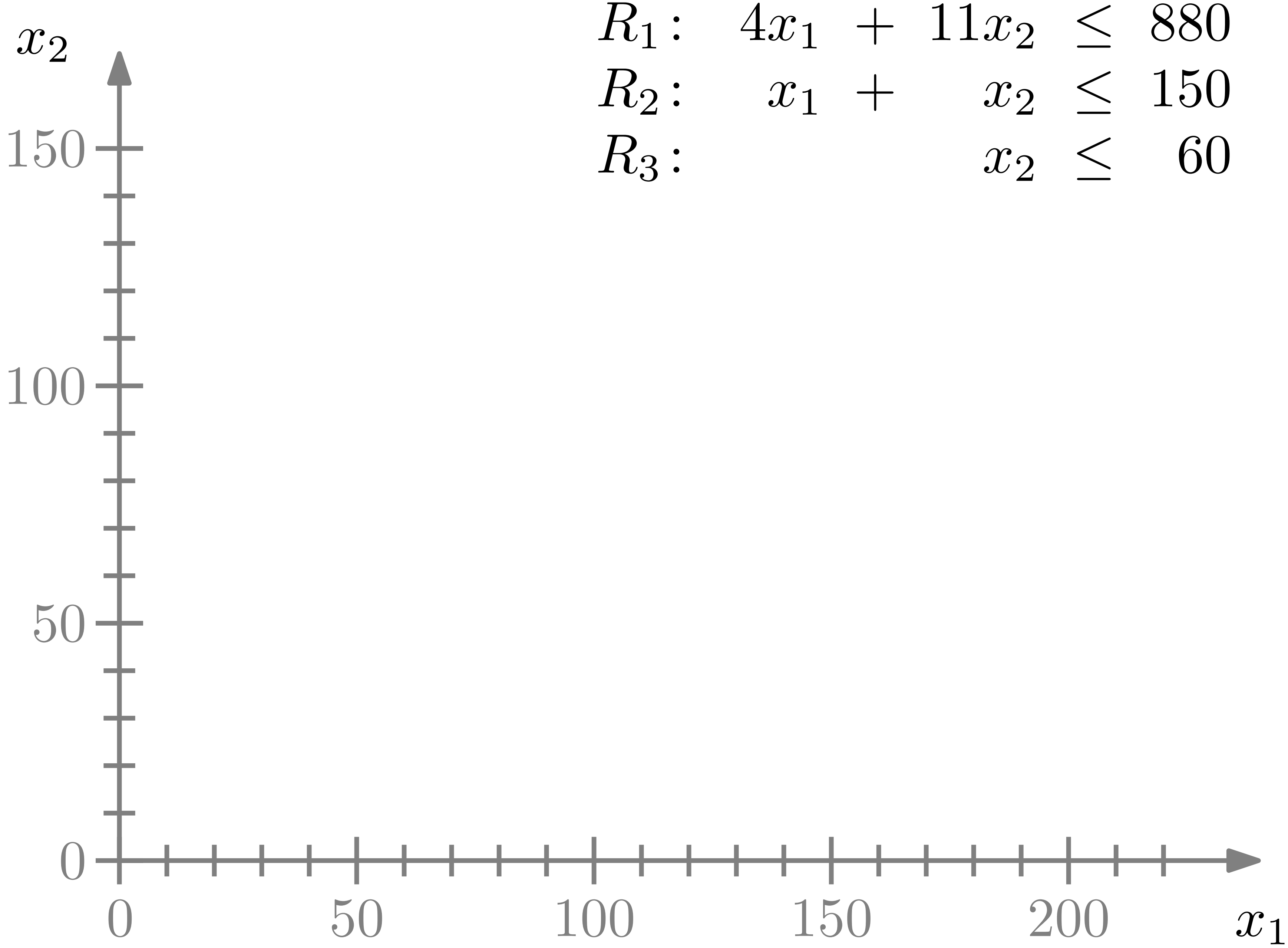
$$\begin{array}{lrcrcr} R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\ R_2: & x_1 & + & x_2 & \leq & 150 \\ R_3: & & & x_2 & \leq & 60 \end{array}$$

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

Lösung

Lineare Beschränkungen:

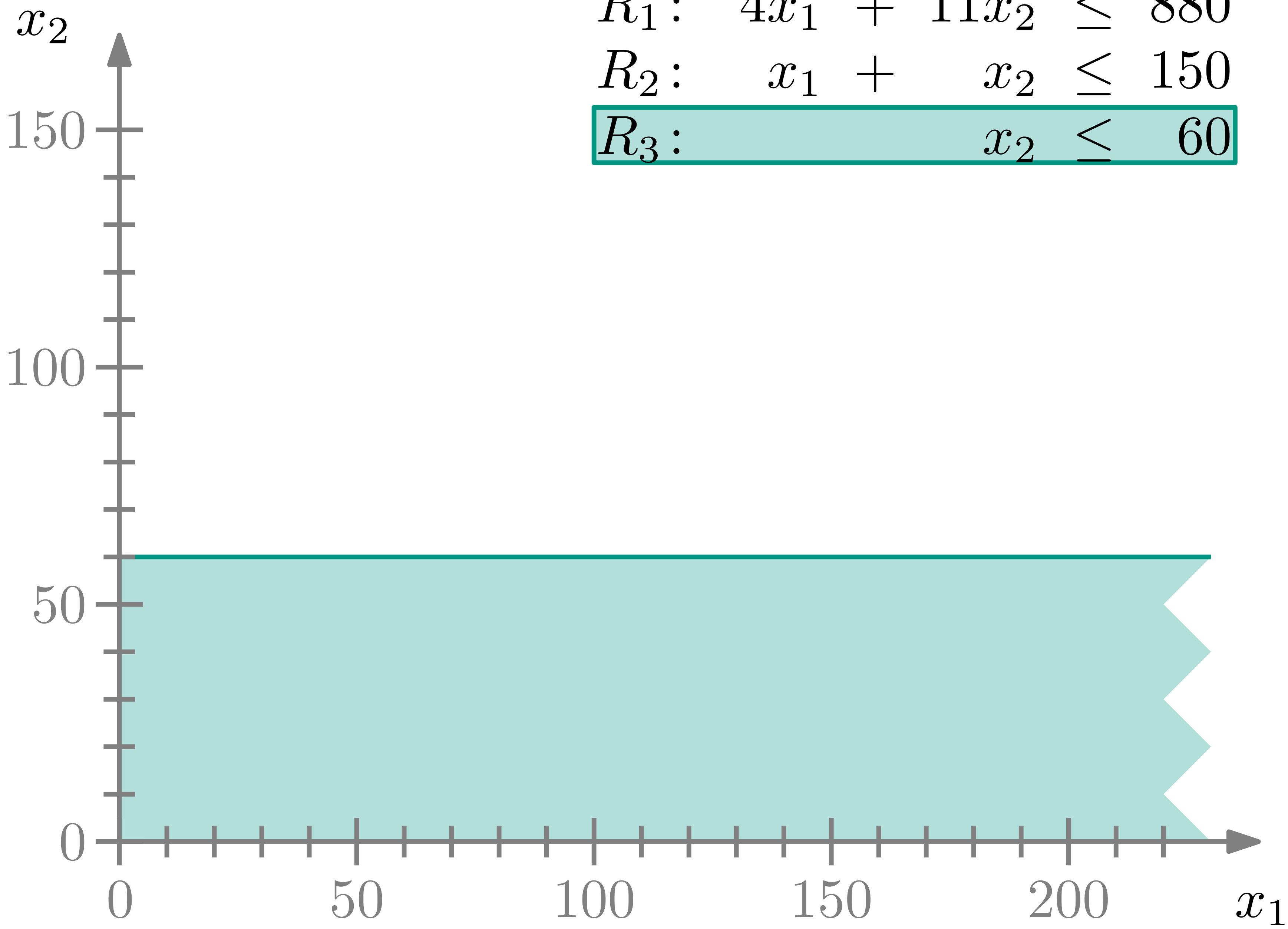
$$\begin{array}{lrcrcr} R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\ R_2: & x_1 & + & x_2 & \leq & 150 \\ R_3: & & & x_2 & \leq & 60 \end{array}$$



Lösung

Lineare Beschränkungen:

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



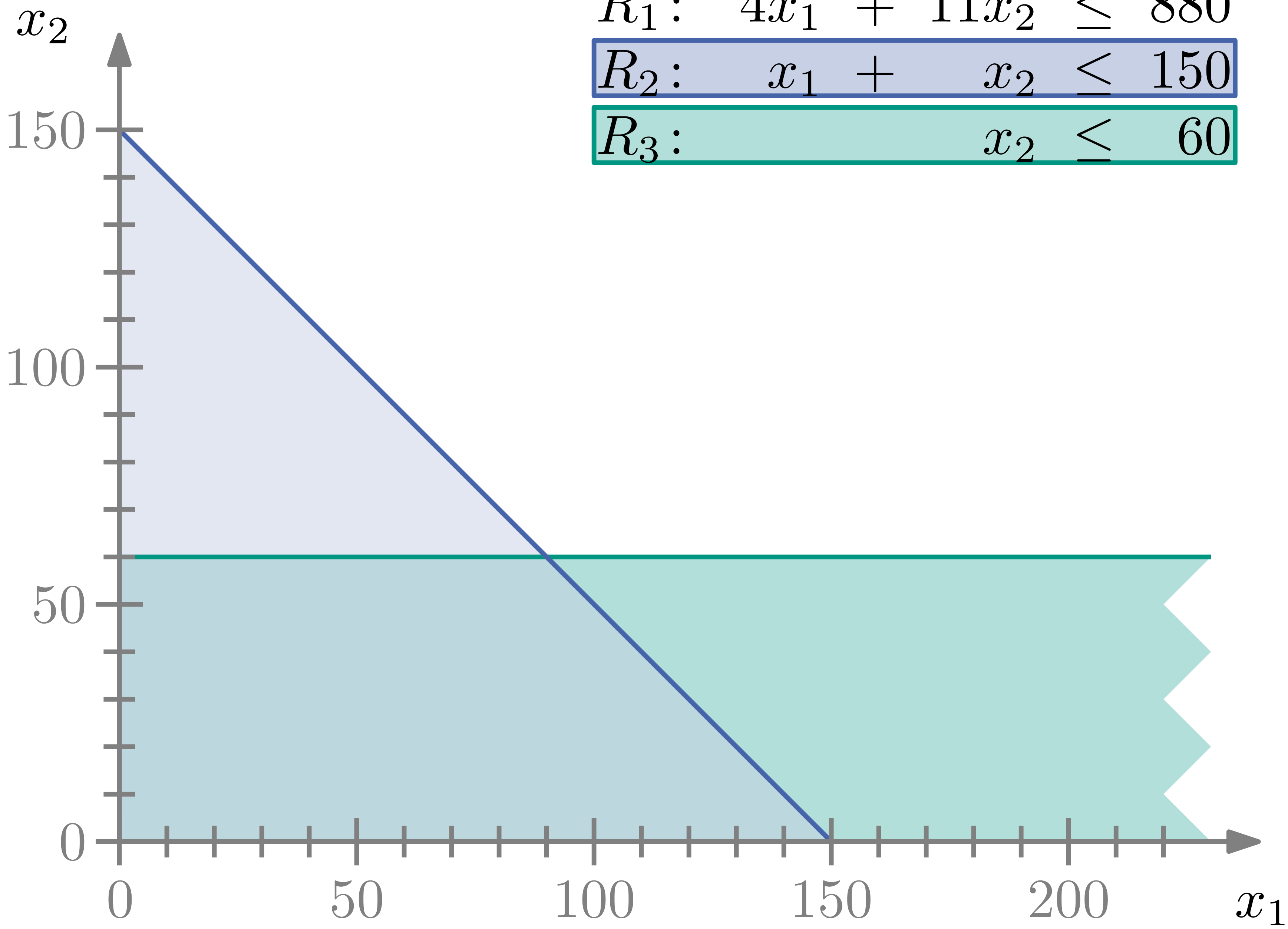
Lösung

Lineare Beschränkungen:

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

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

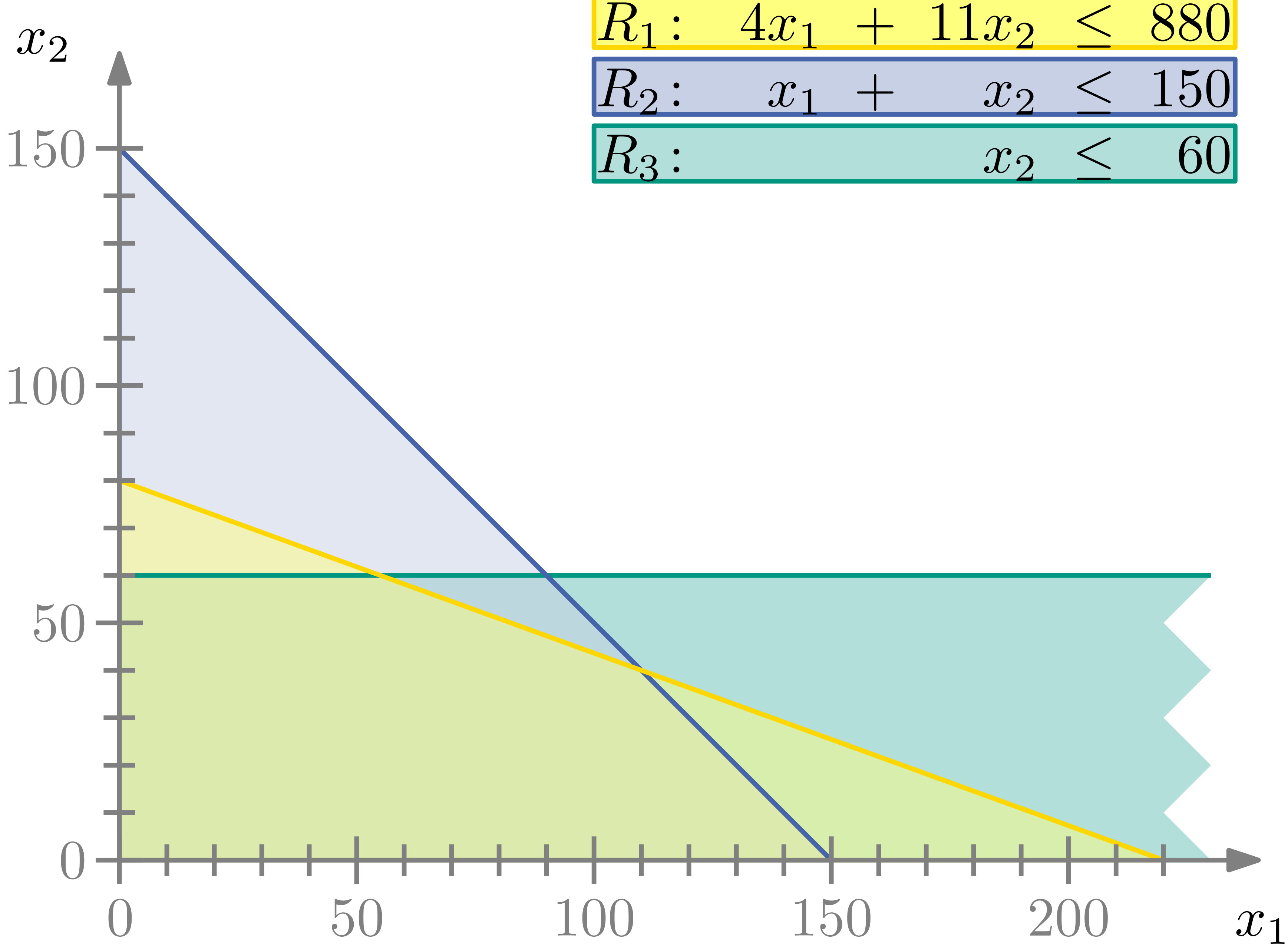
$$\boxed{R_3: \quad x_2 \leq 60}$$



Lösung

Lineare Beschränkungen:

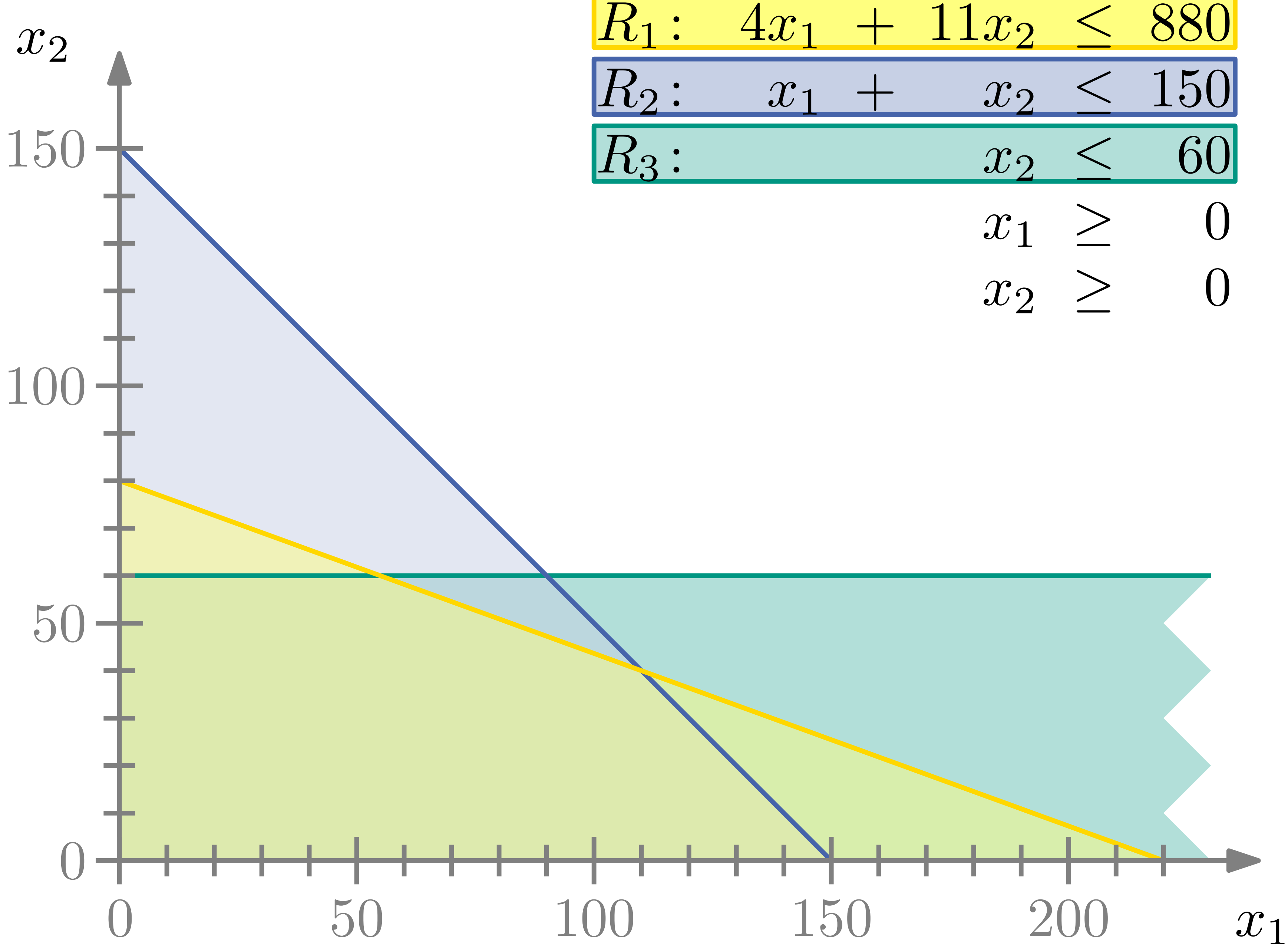
$$R_1: \quad 4x_1 + 11x_2 \leq 880$$
$$R_2: \quad x_1 + x_2 \leq 150$$
$$R_3: \quad x_2 \leq 60$$



Lösung

Lineare Beschränkungen:

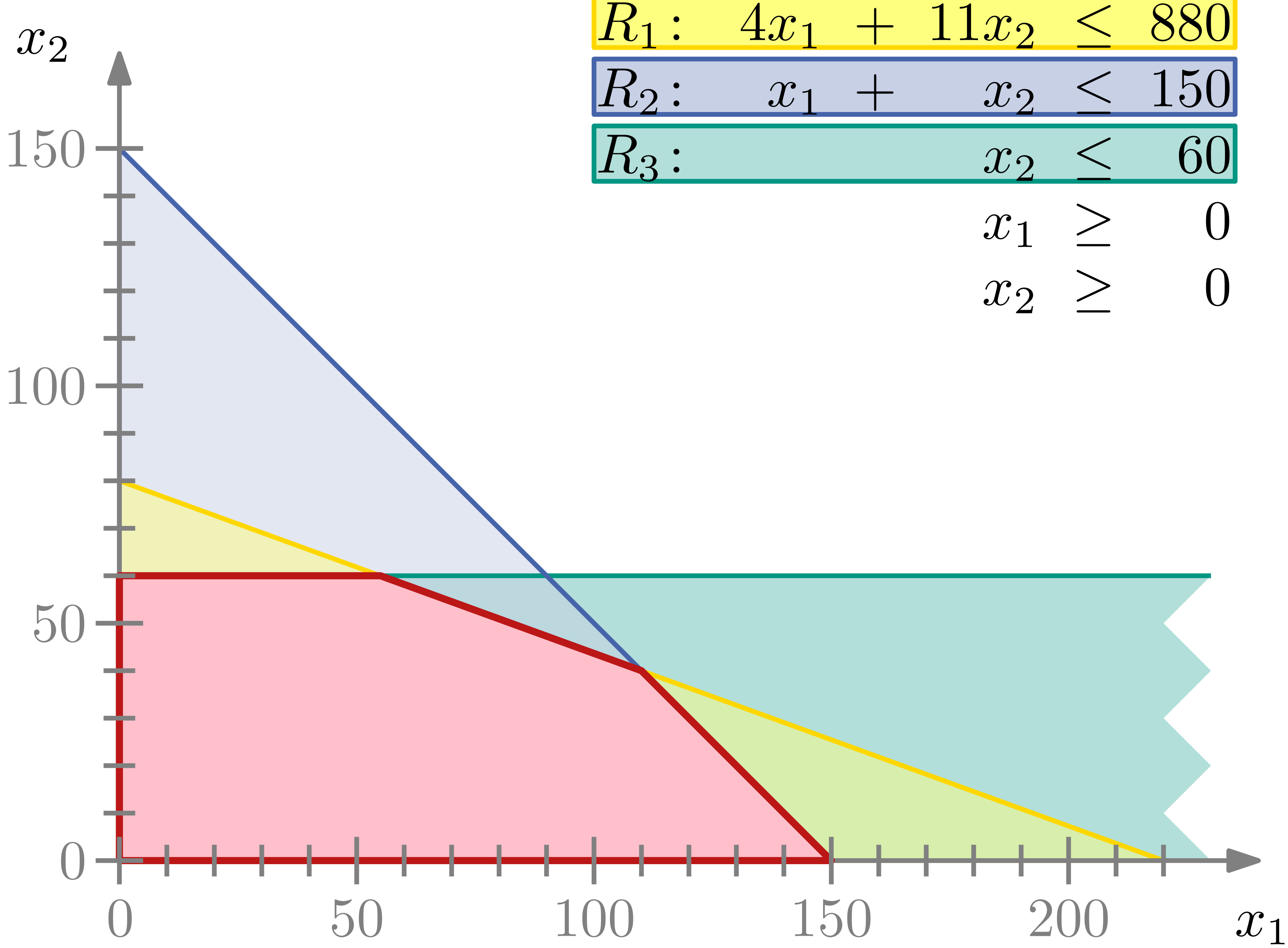
$$\begin{array}{lrcrcr} R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\ R_2: & x_1 & + & x_2 & \leq & 150 \\ R_3: & & & x_2 & \leq & 60 \\ & & & x_1 & \geq & 0 \\ & & & x_2 & \geq & 0 \end{array}$$



Lösung

Lineare Beschränkungen:

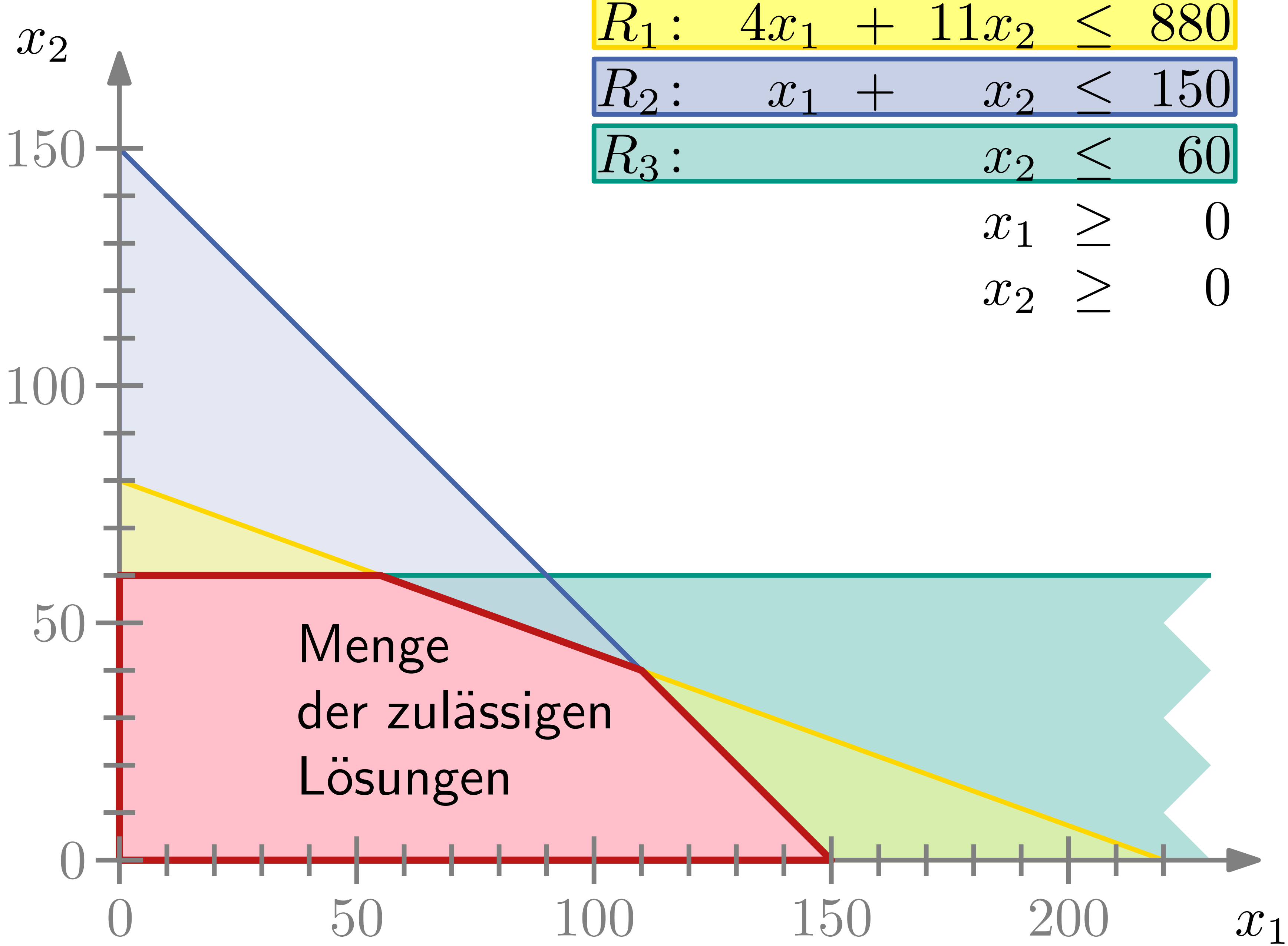
$$\begin{array}{lrcrcr} R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\ R_2: & x_1 & + & x_2 & \leq & 150 \\ R_3: & & & x_2 & \leq & 60 \\ & & & x_1 & \geq & 0 \\ & & & x_2 & \geq & 0 \end{array}$$



Lösung

Lineare Beschränkungen:

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



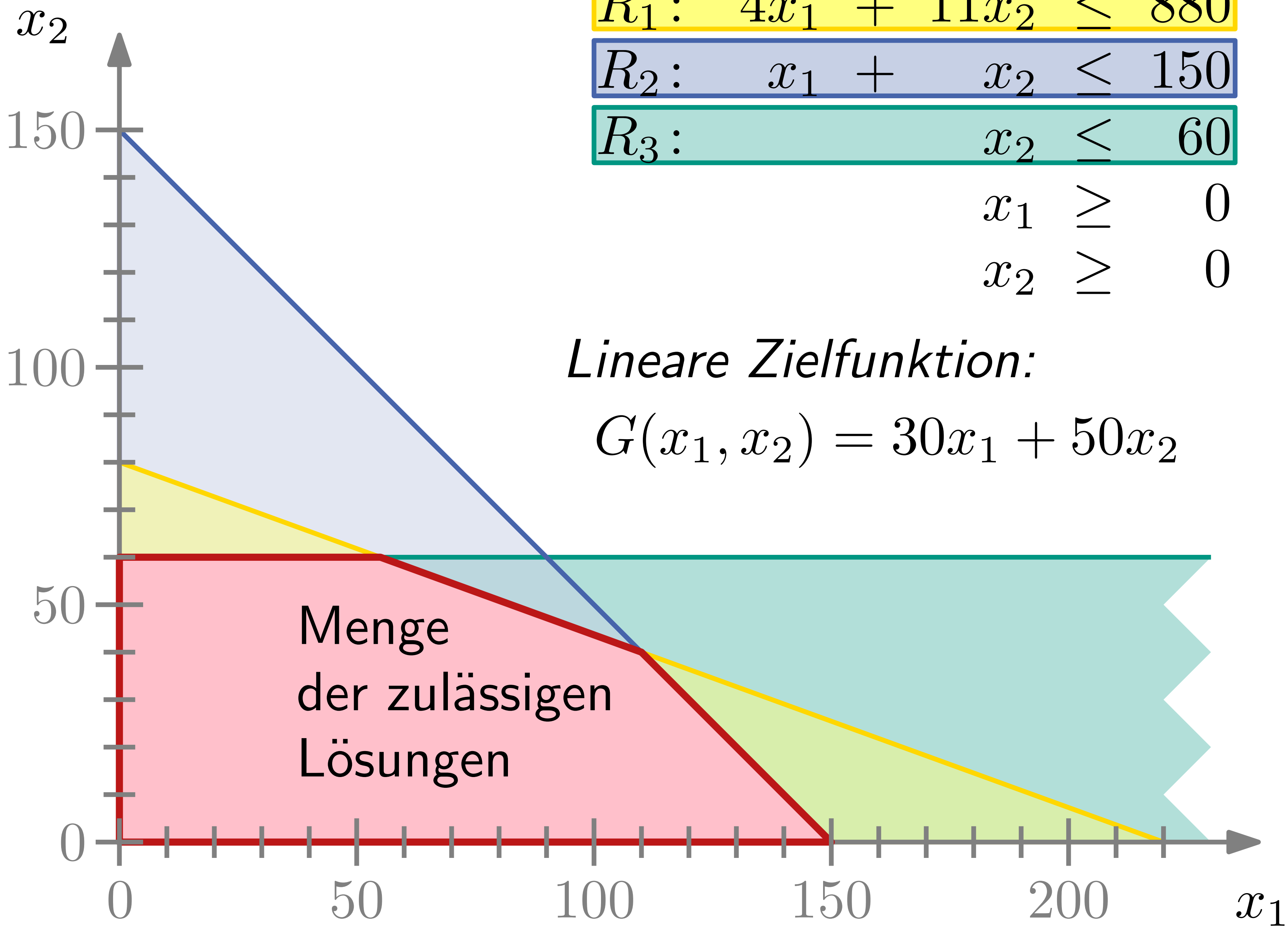
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$G(x_1, x_2) = 30x_1 + 50x_2$$



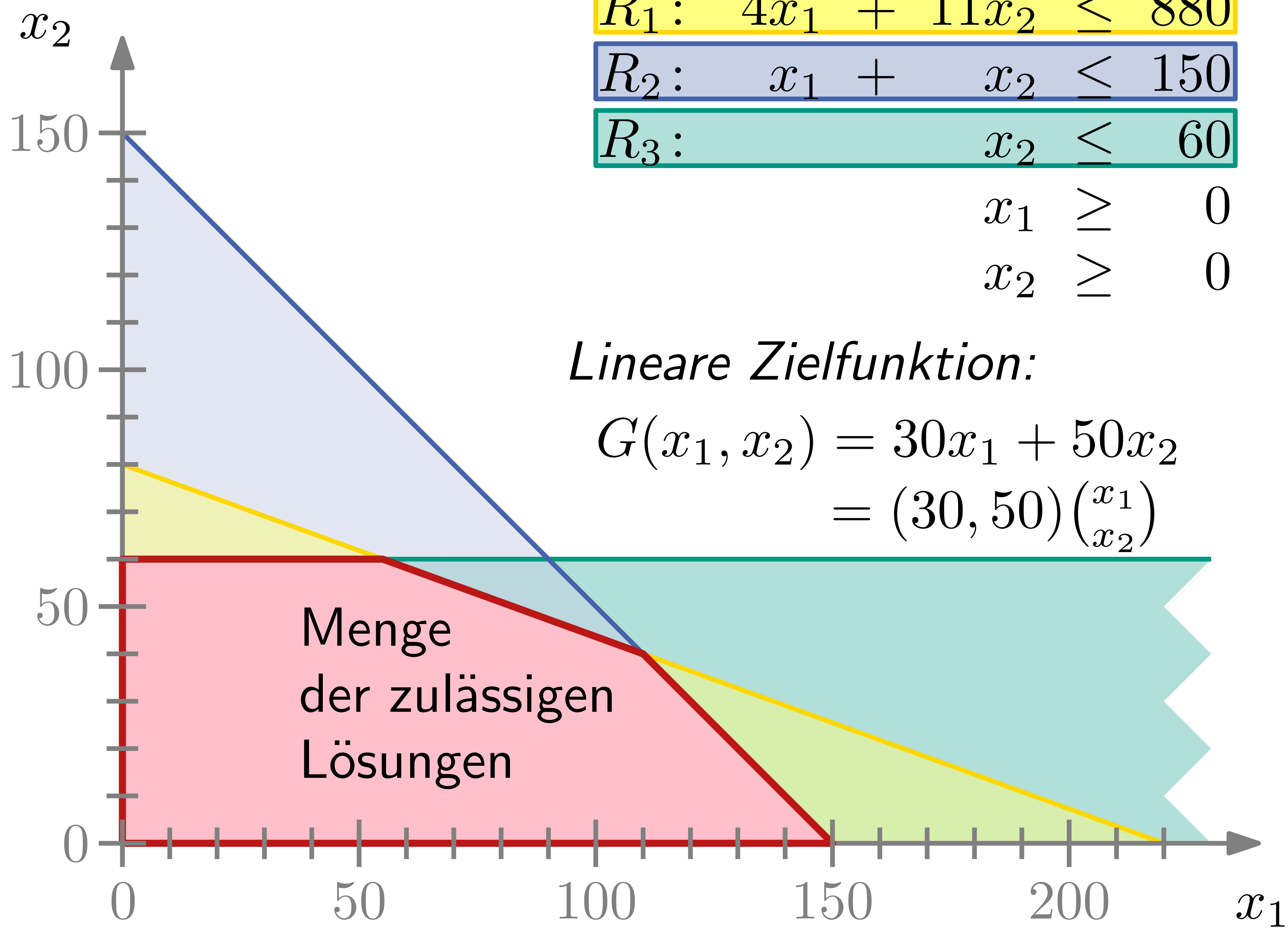
Lösung

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:

$$\begin{aligned} G(x_1, x_2) &= 30x_1 + 50x_2 \\ &= (30, 50)\binom{x_1}{x_2} \end{aligned}$$



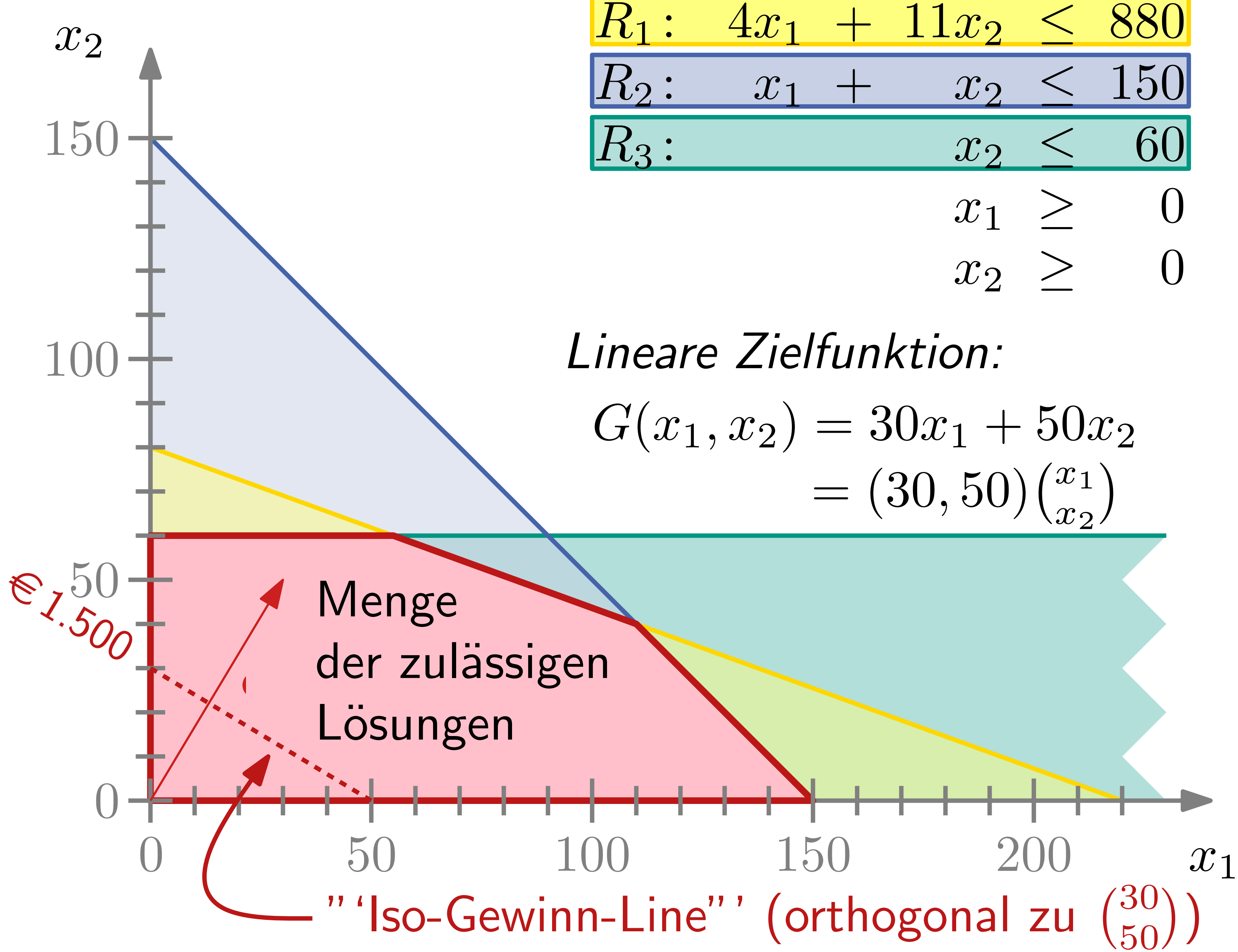
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$
\begin{aligned}
G(x_1, x_2) &= 30x_1 + 50x_2 \\
&= (30, 50)\binom{x_1}{x_2}
\end{aligned}
$$



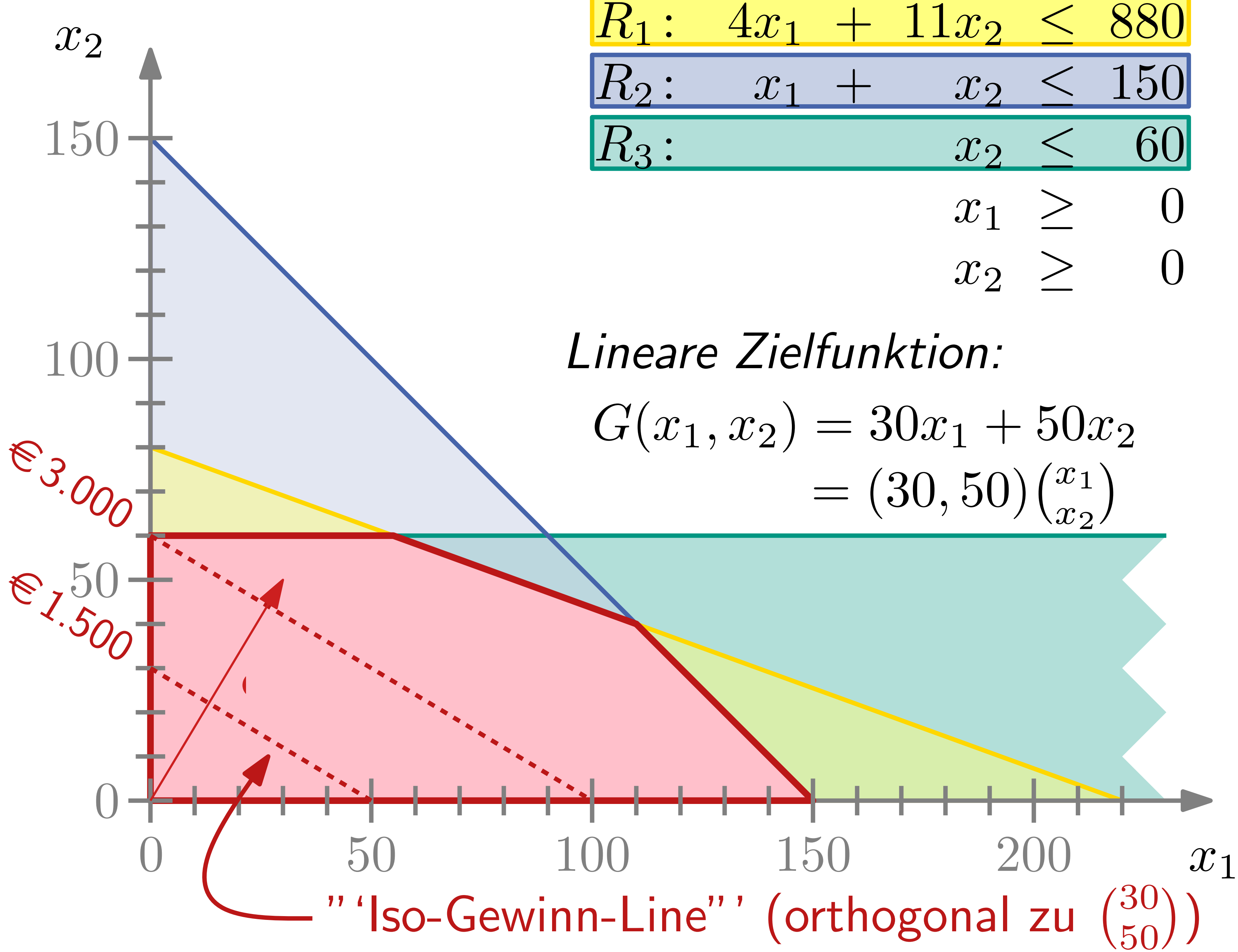
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$\begin{aligned} G(x_1, x_2) &= 30x_1 + 50x_2 \\ &= (30, 50)\binom{x_1}{x_2} \end{aligned}$$



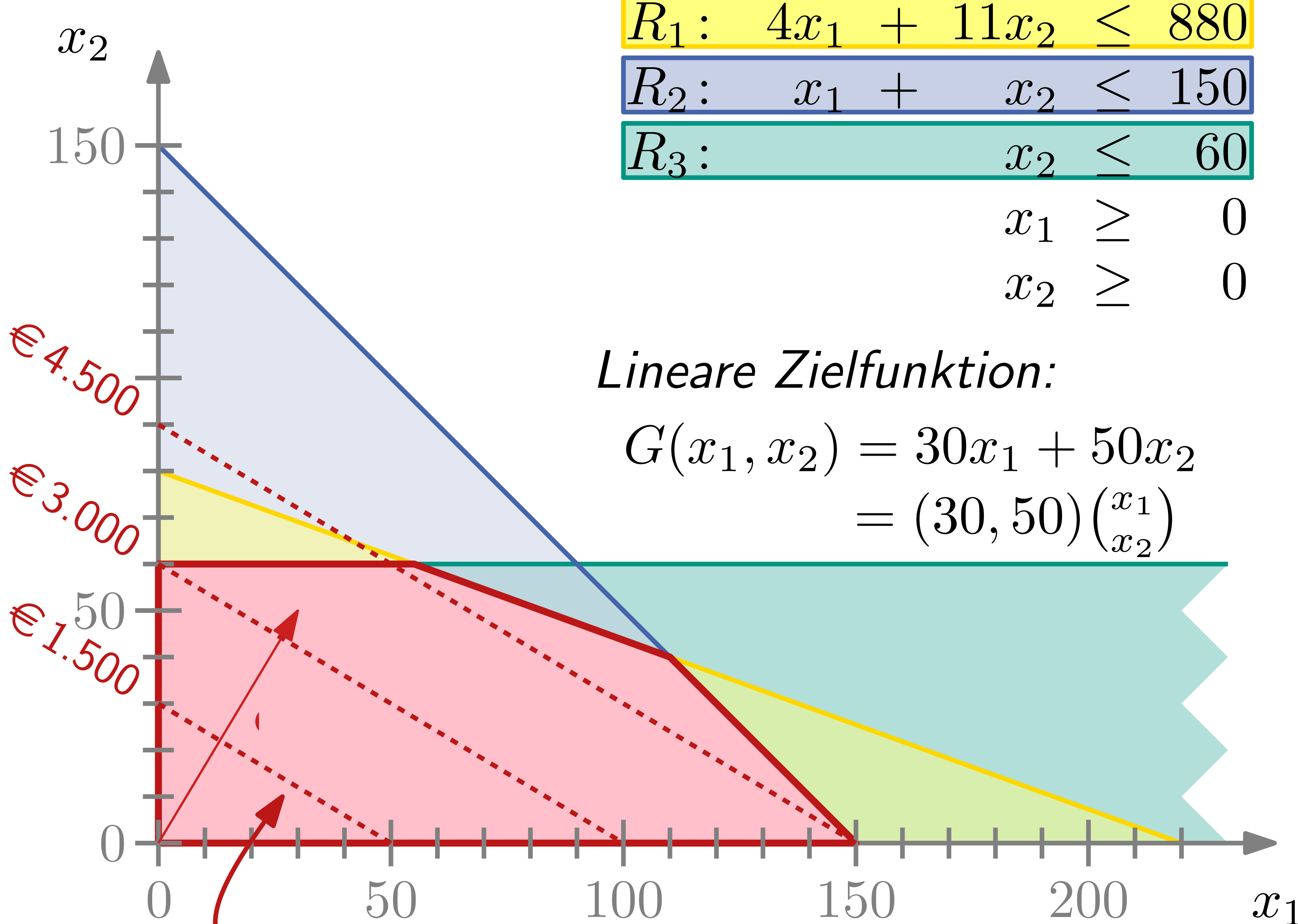
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$\begin{aligned} G(x_1, x_2) &= 30x_1 + 50x_2 \\ &= (30, 50)\binom{x_1}{x_2} \end{aligned}$$



"'Iso-Gewinn-Line"' (orthogonal zu $\binom{30}{50}$)

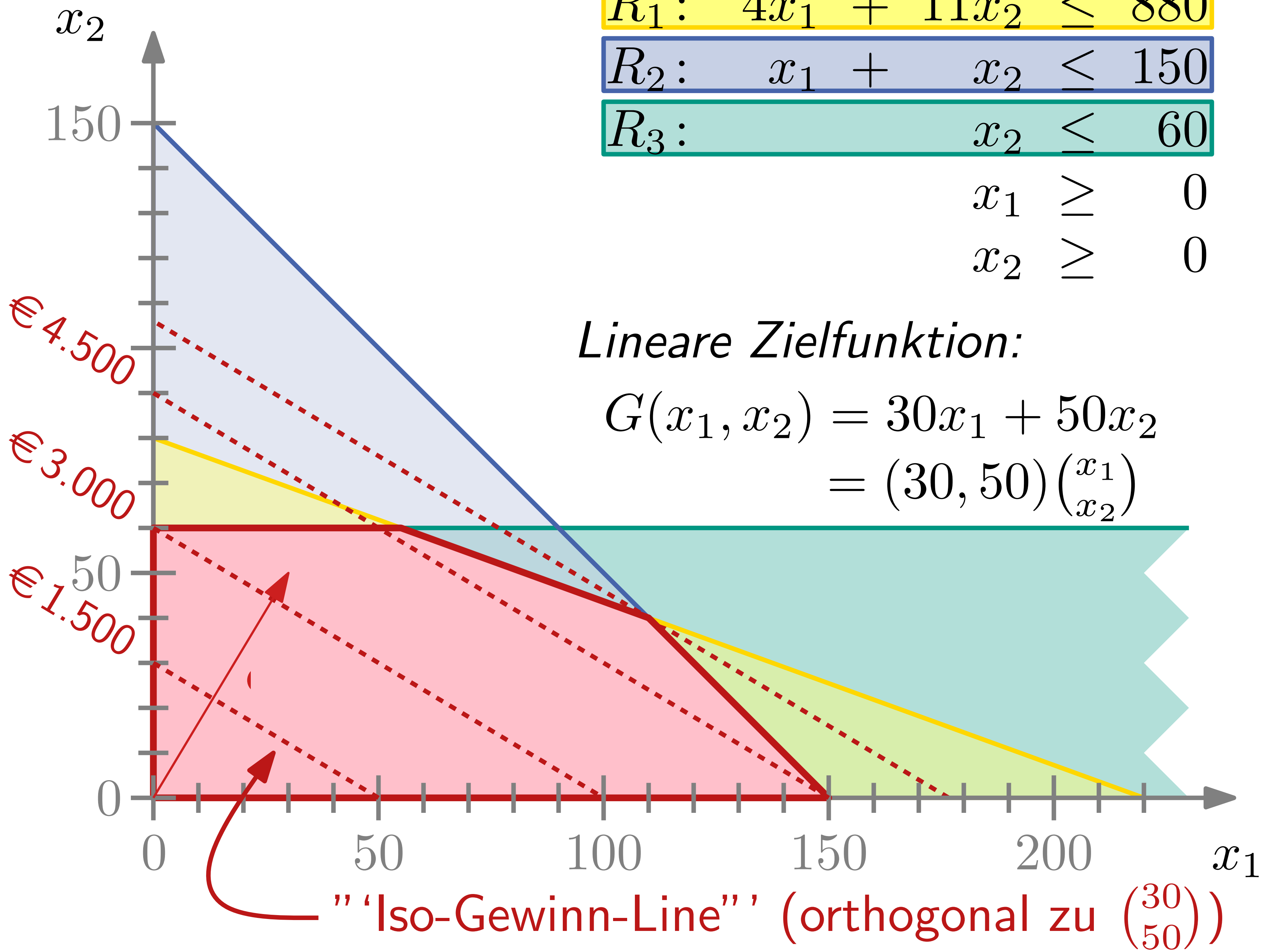
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$G(x_1, x_2) = 30x_1 + 50x_2 = (30, 50)\binom{x_1}{x_2}$$



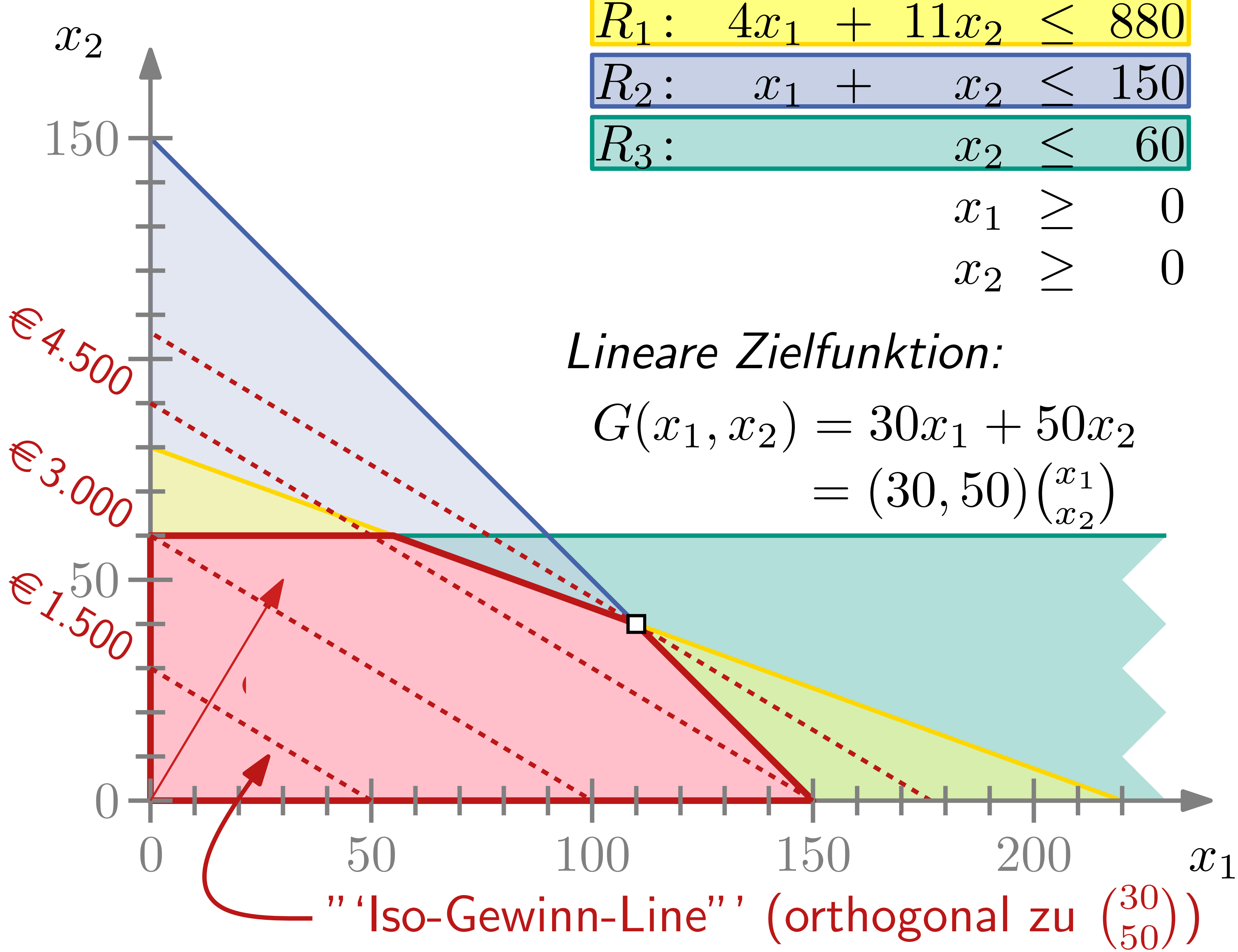
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$
\begin{aligned}
G(x_1, x_2) &= 30x_1 + 50x_2 \\
&= (30, 50)\binom{x_1}{x_2}
\end{aligned}
$$



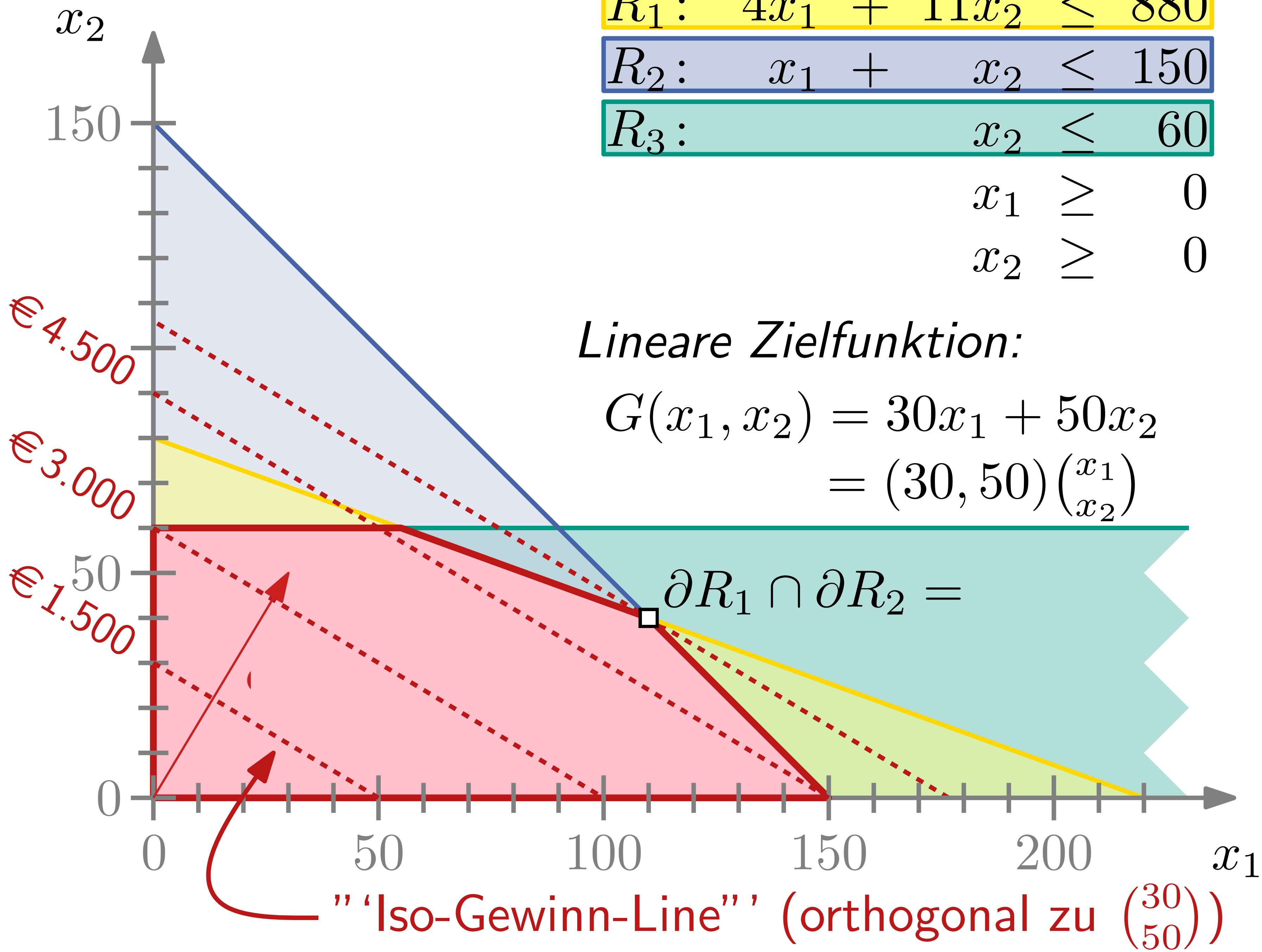
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$G(x_1, x_2) = 30x_1 + 50x_2 = (30, 50)\binom{x_1}{x_2}$$



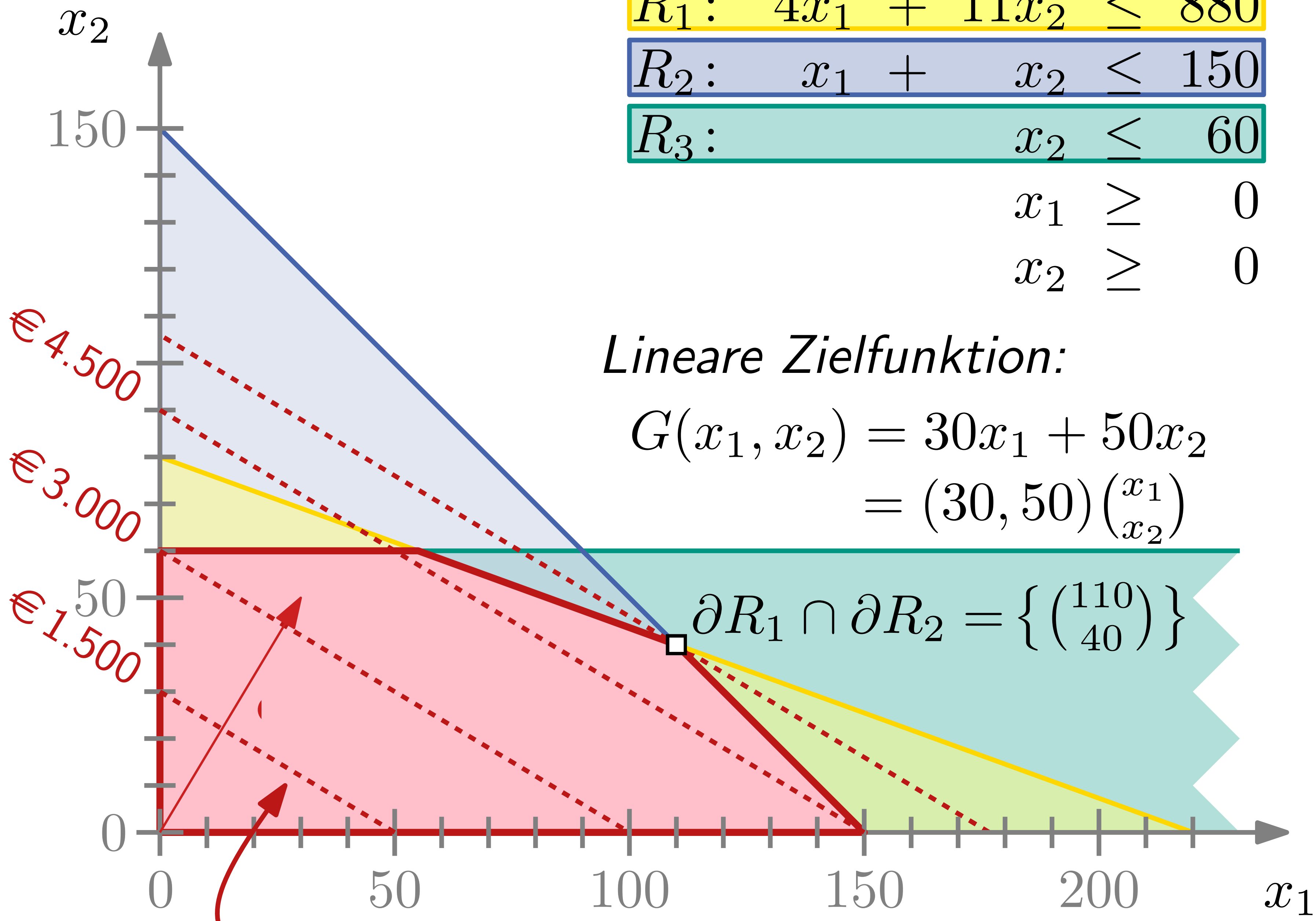
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$
\begin{aligned}
G(x_1, x_2) &= 30x_1 + 50x_2 \\
&= (30, 50)\binom{x_1}{x_2}
\end{aligned}
$$



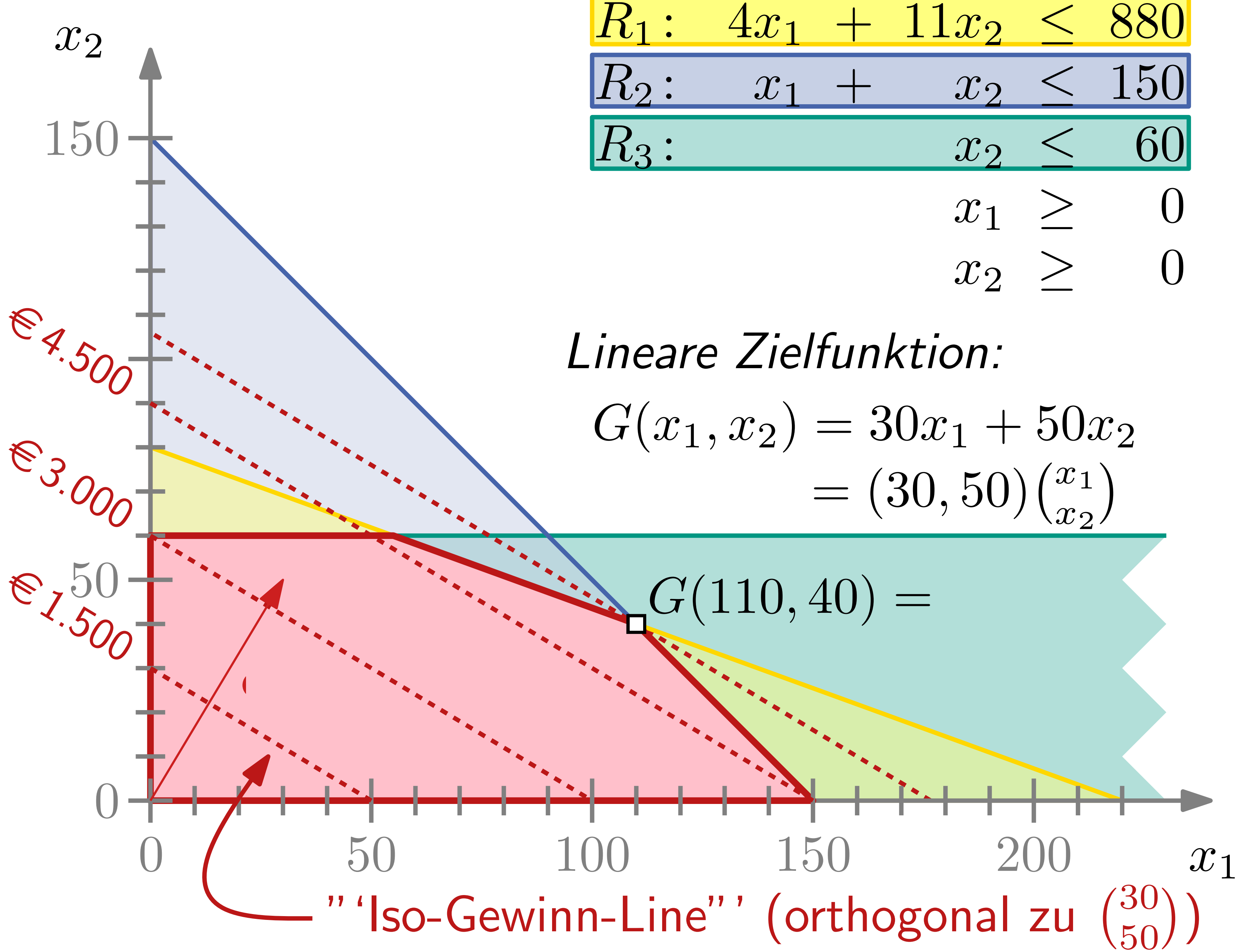
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$\begin{aligned} G(x_1, x_2) &= 30x_1 + 50x_2 \\ &= (30, 50)\binom{x_1}{x_2} \end{aligned}$$



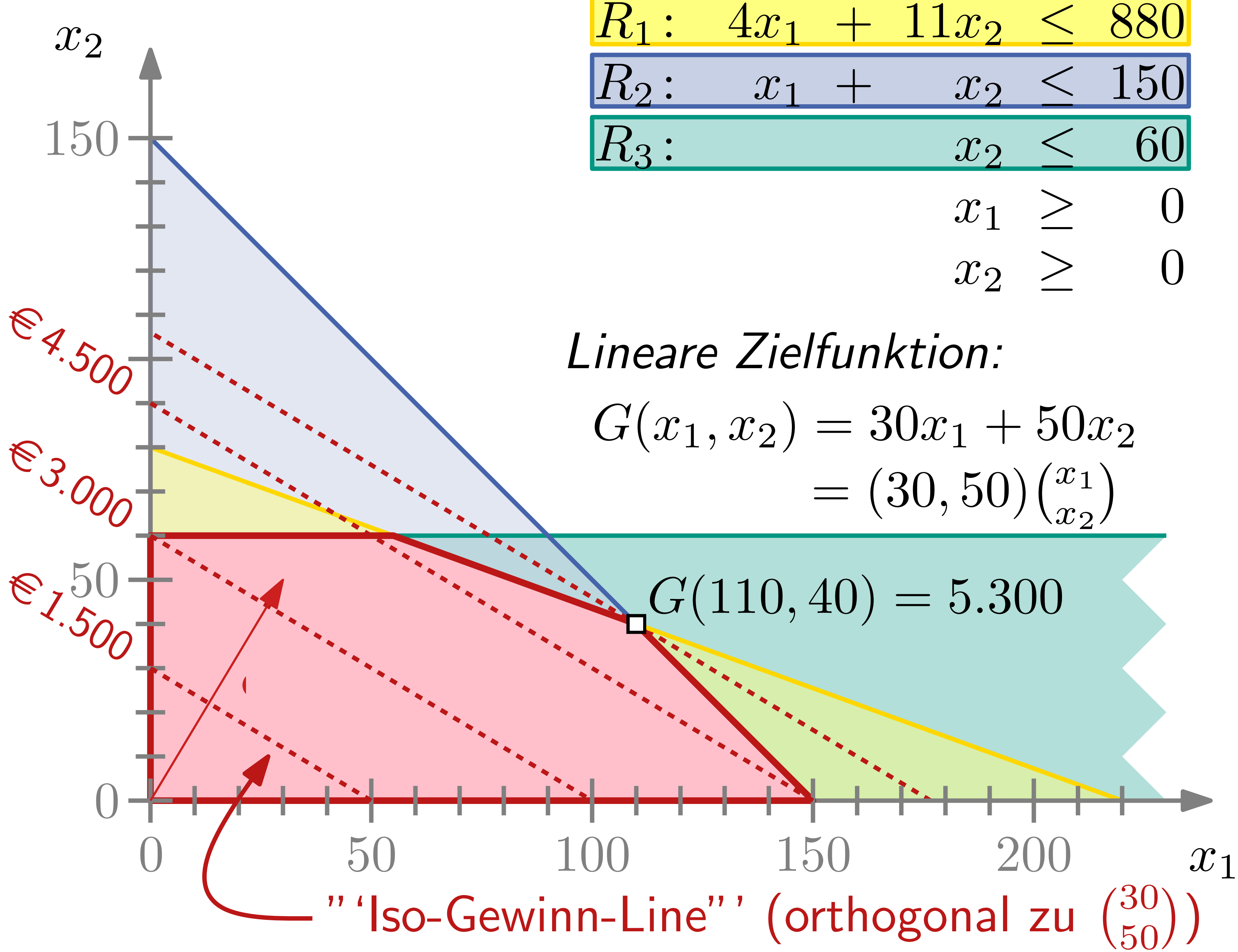
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$\begin{aligned} G(x_1, x_2) &= 30x_1 + 50x_2 \\ &= (30, 50)\binom{x_1}{x_2} \end{aligned}$$



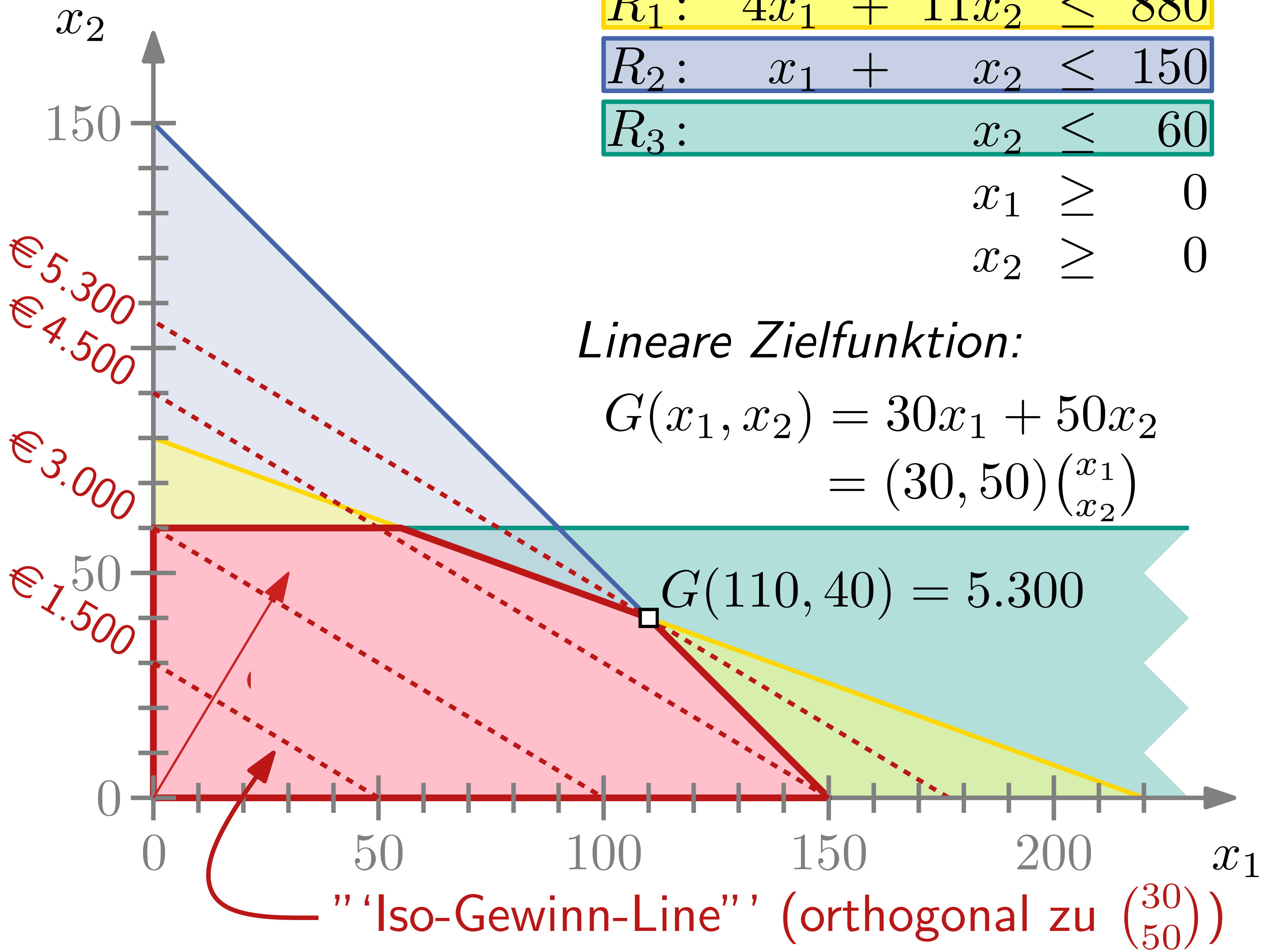
Lösung

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:

$$\begin{aligned}
G(x_1, x_2) &= 30x_1 + 50x_2\\
&= (30, 50)\binom{x_1}{x_2}
\end{aligned}$$



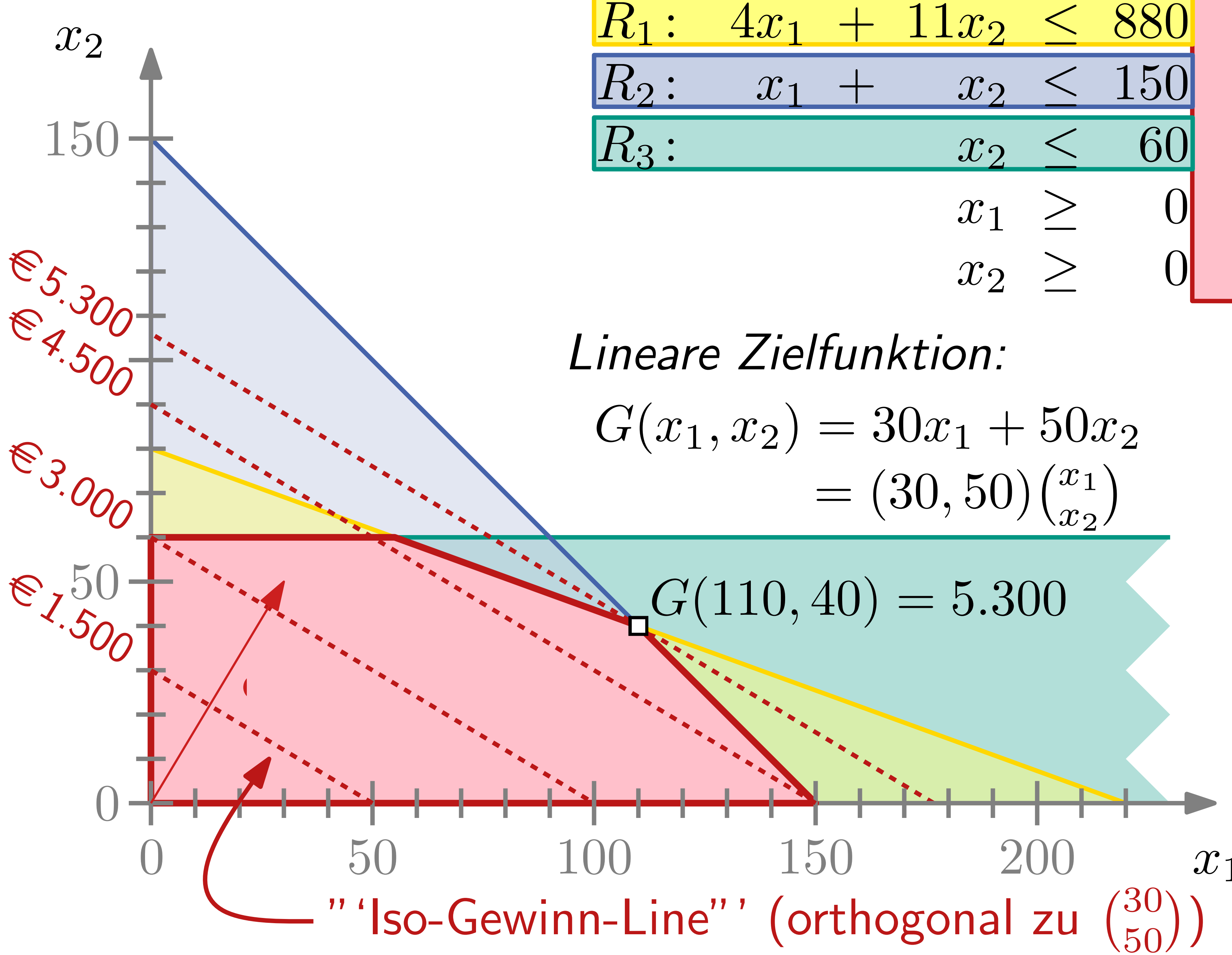
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$G(x_1, x_2) = 30x_1 + 50x_2 = (30, 50)\binom{x_1}{x_2}$$



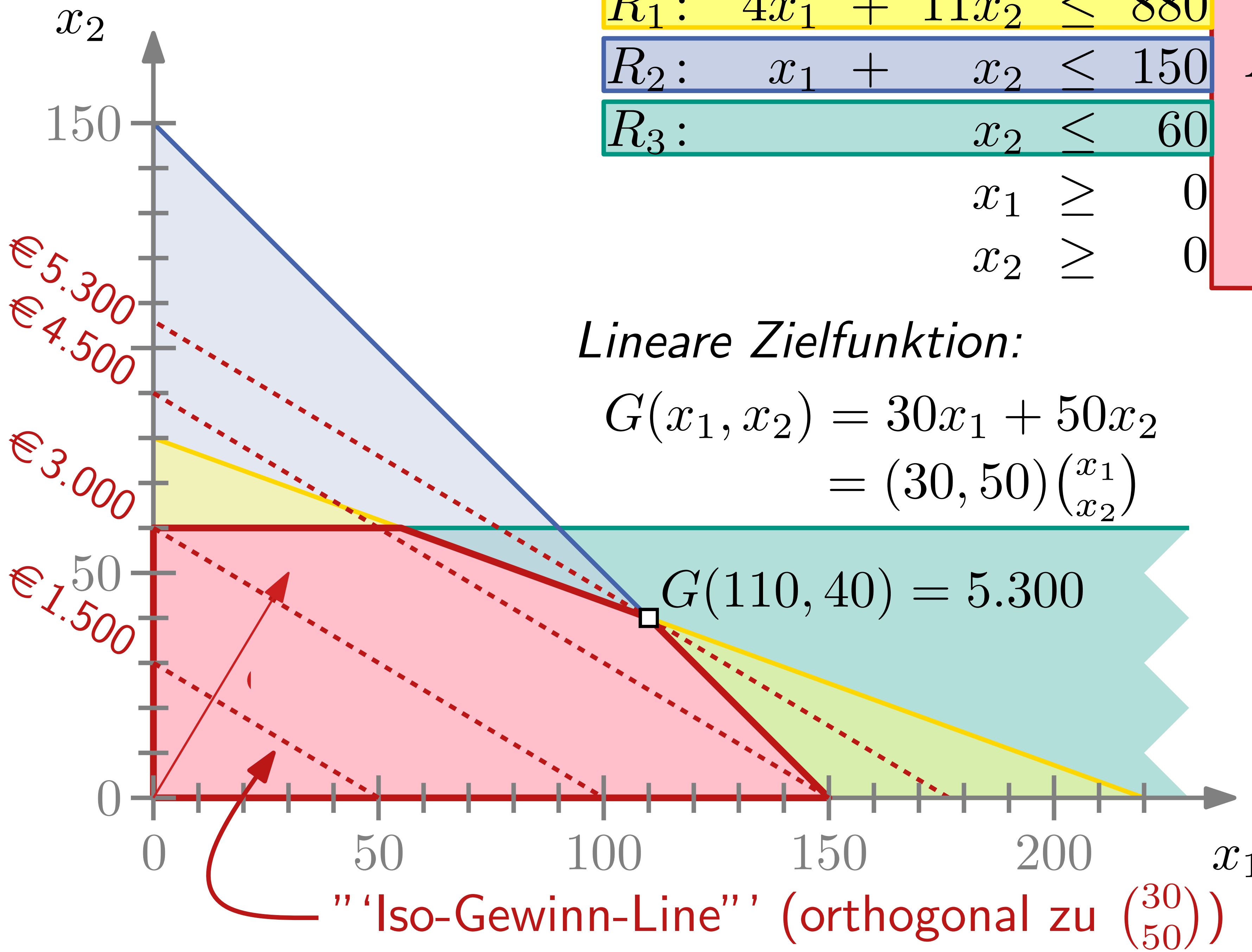
Lösung

Lineare Beschränkungen:

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

Lineare Zielfunktion:

$$
\begin{aligned}
G(x_1, x_2) &= 30x_1 + 50x_2 \\
&= (30, 50)\binom{x_1}{x_2}
\end{aligned}
$$



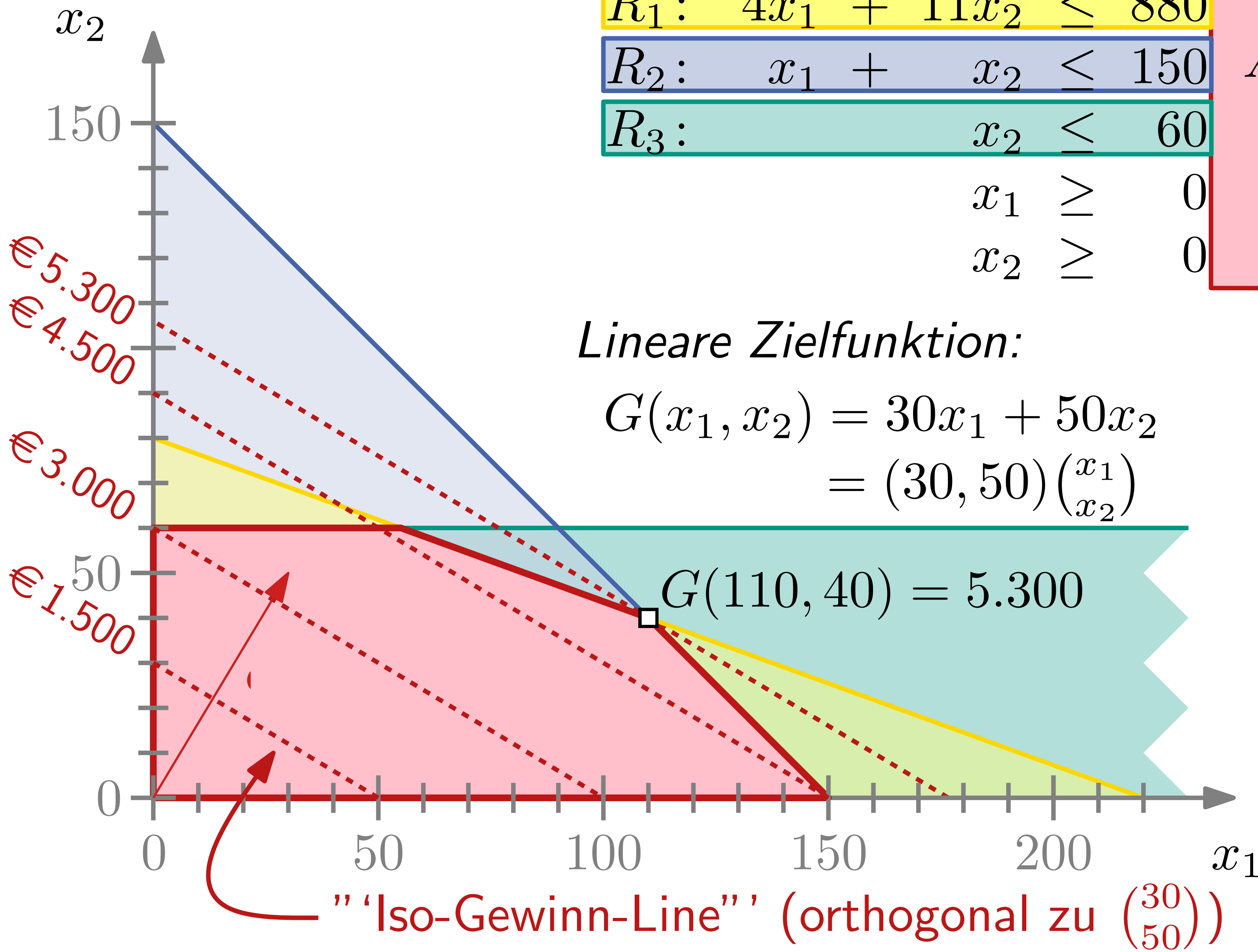
Lösung

Lineare Beschränkungen:

$$\begin{array}{lrcrcl} R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\ R_2: & x_1 & + & x_2 & \leq & 150 \\ R_3: & & & x_2 & \leq & 60 \\ & & & x_1 & \geq & 0 \\ & & & x_2 & \geq & 0 \end{array} \qquad \begin{array}{c} Ax \leq b \\ \\ x \geq 0 \end{array}$$

Lineare Zielfunktion:

$$G(x_1, x_2) = 30x_1 + 50x_2 = (30, 50)\binom{x_1}{x_2}$$



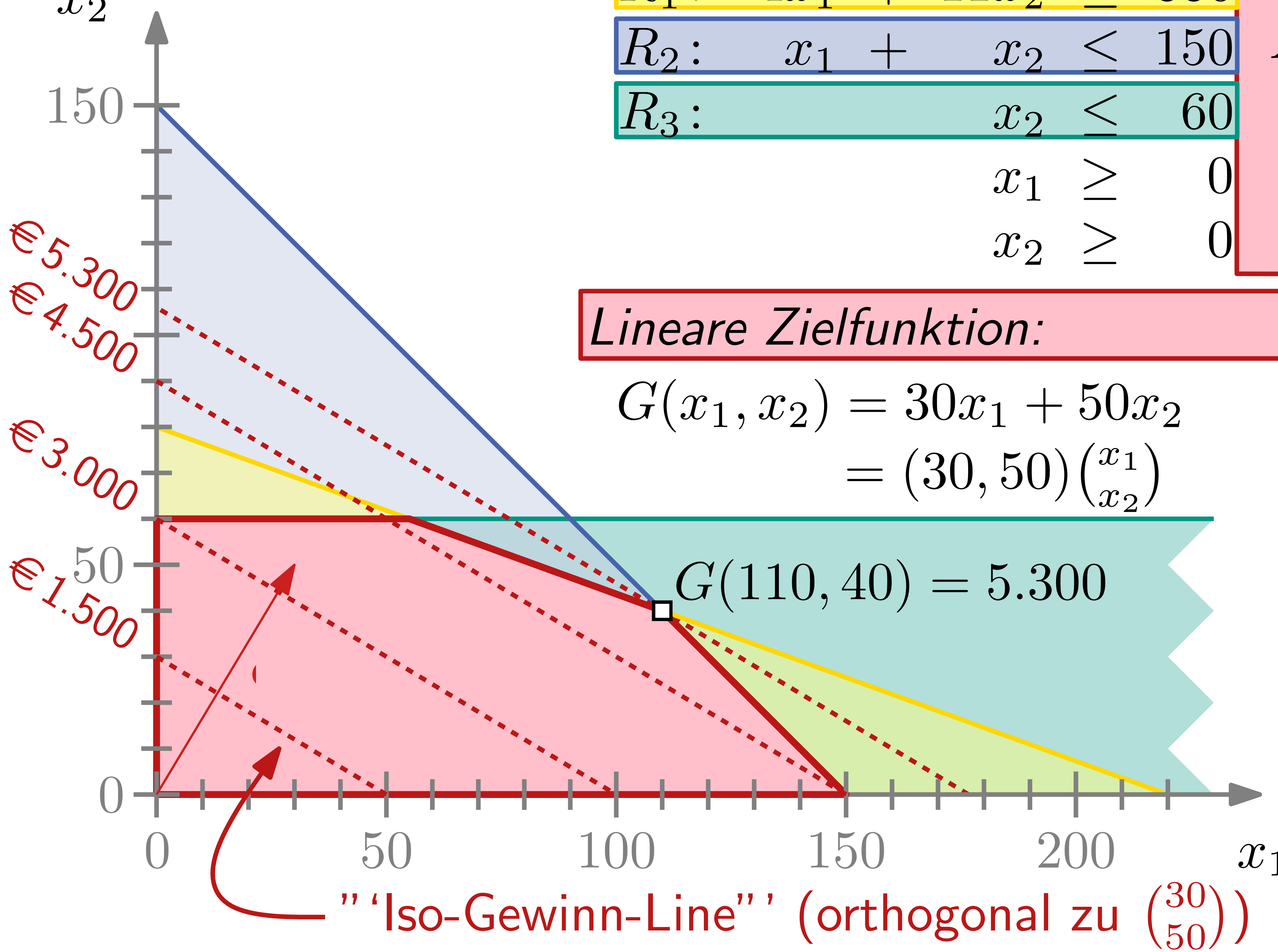
Lösung

Lineare Beschränkungen:

$$
\begin{array}{lrcrcr}
R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\
R_2: & x_1 & + & x_2 & \leq & 150 \\
R_3: & & & x_2 & \leq & 60 \\
 & & & x_1 & \geq & 0 \\
 & & & x_2 & \geq & 0
\end{array}
\qquad
\begin{array}{c}
Ax \leq b \\
\\
x \geq 0
\end{array}
$$

Lineare Zielfunktion:

$$
\begin{aligned}
G(x_1, x_2) &= 30x_1 + 50x_2 \\
&= (30, 50)\binom{x_1}{x_2}
\end{aligned}
$$



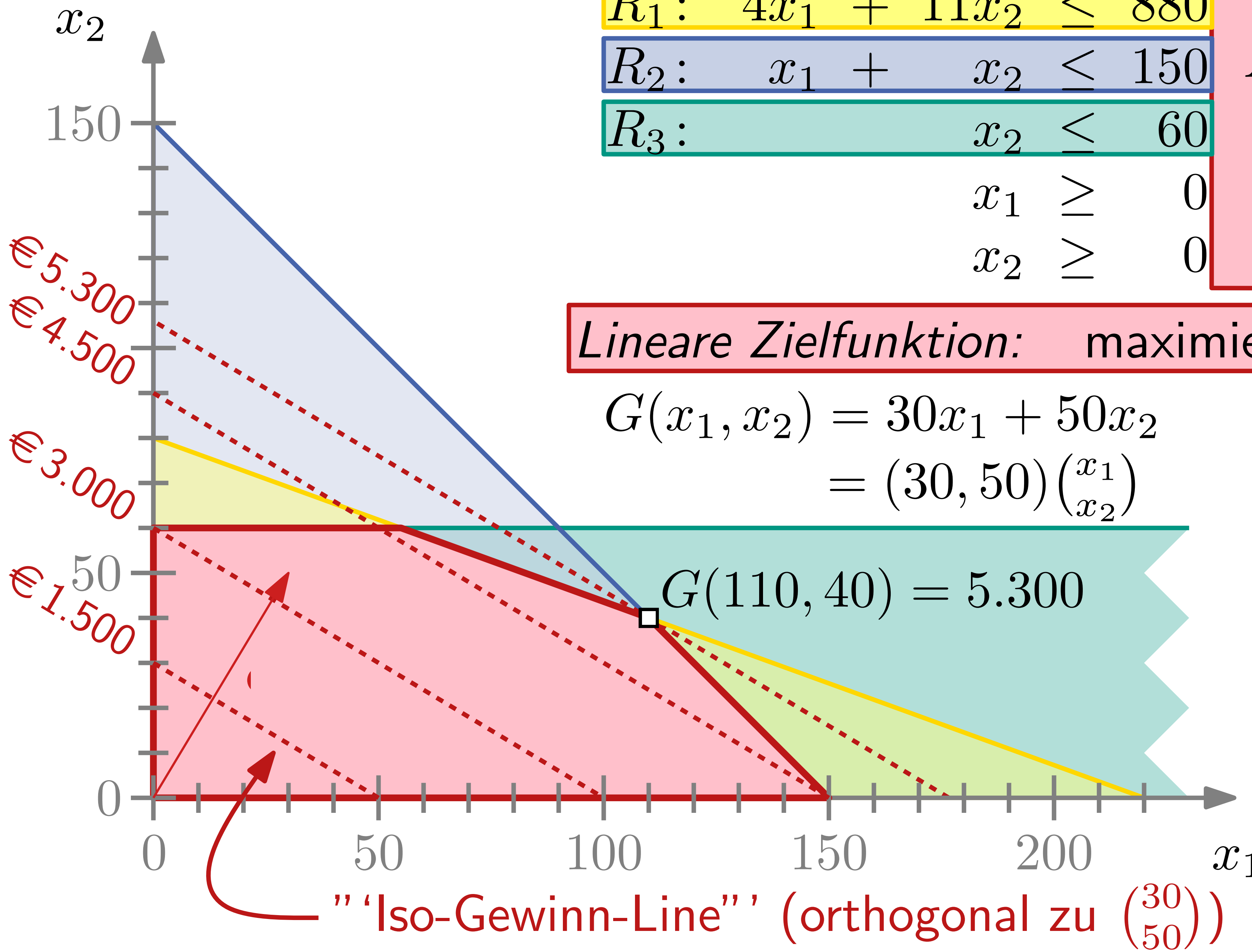
Lösung

Lineare Beschränkungen:

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

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

$$G(x_1, x_2) = 30x_1 + 50x_2 = (30, 50)\binom{x_1}{x_2}$$



Lösung

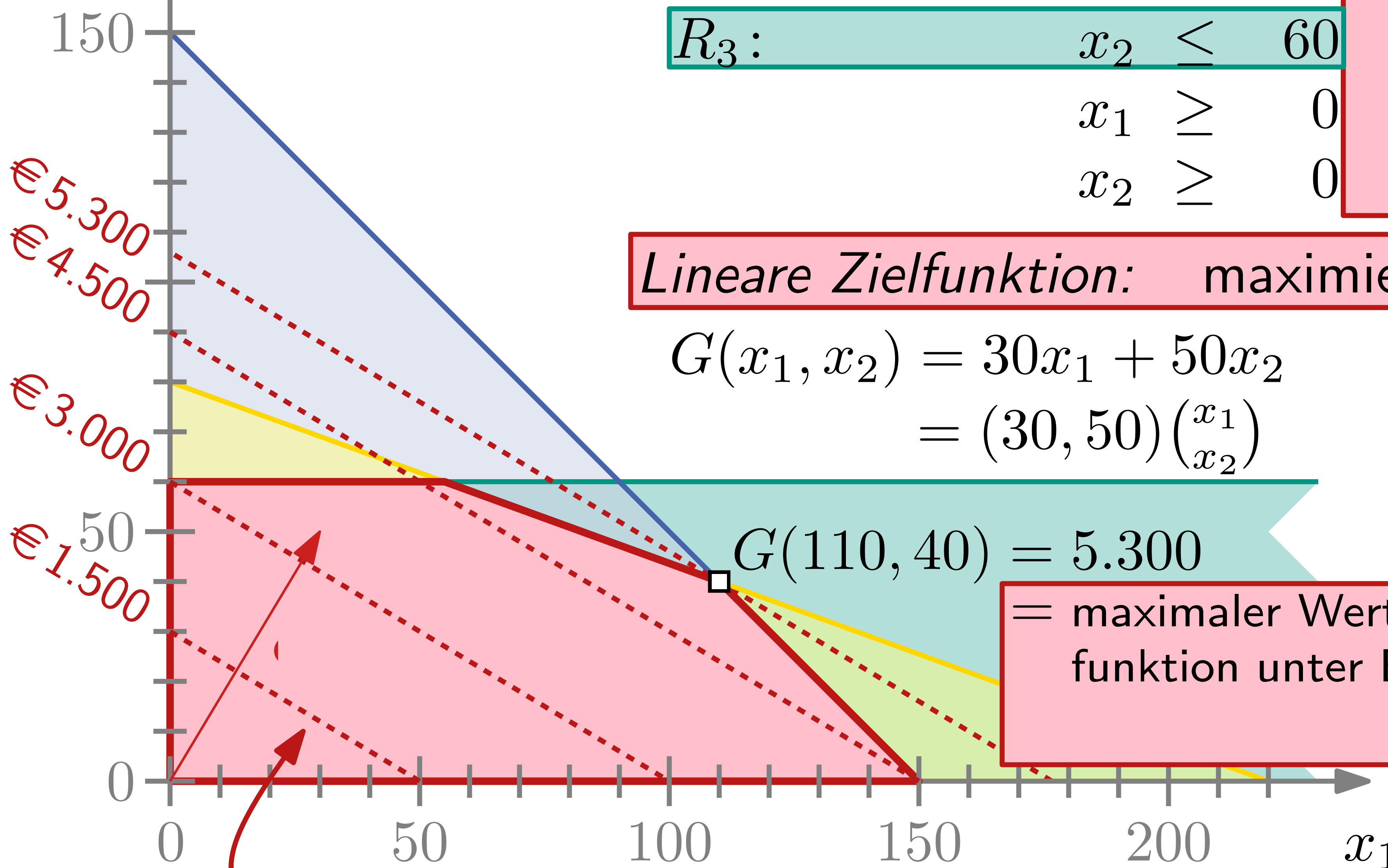
Lineare Beschränkungen:

$$
\left.
\begin{array}{lrcrcr}
R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\
R_2: & x_1 & + & x_2 & \leq & 150 \\
R_3: & & & x_2 & \leq & 60
\end{array}
\right\} \quad Ax \leq b
$$

$$
\left.
\begin{array}{rcr}
x_1 & \geq & 0 \\
x_2 & \geq & 0
\end{array}
\right\} \quad x \geq 0
$$

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

$$G(x_1, x_2) = 30x_1 + 50x_2 = (30, 50)\binom{x_1}{x_2}$$



"'Iso-Gewinn-Line"' (orthogonal zu $\binom{30}{50}$)

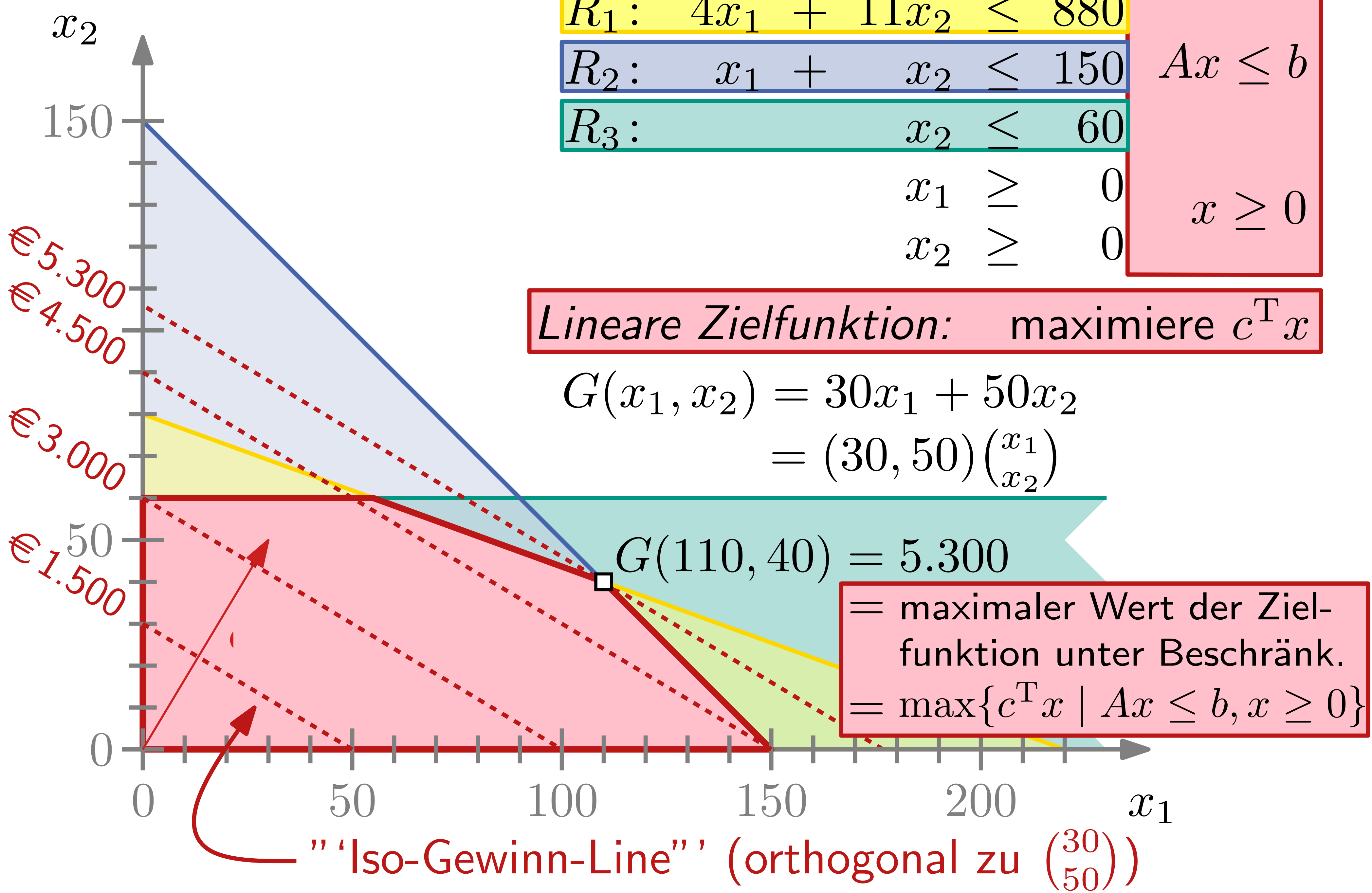
Lösung

Lineare Beschränkungen:

$$\begin{array}{lrcrcr} R_1: & 4x_1 & + & 11x_2 & \leq & 880 \\ R_2: & x_1 & + & x_2 & \leq & 150 \\ R_3: & & & x_2 & \leq & 60 \\ & & & x_1 & \geq & 0 \\ & & & x_2 & \geq & 0 \end{array} \qquad \begin{array}{l} Ax \leq b \\ \\ x \geq 0 \end{array}$$

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

$$G(x_1, x_2) = 30x_1 + 50x_2 = (30, 50)\binom{x_1}{x_2}$$



Lineares Programmieren

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

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

Lineares Programmieren

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

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

- H entspricht einer Menge von Halbebenen in $\mathbb{R}^d$.
- Gesucht ist ein Punkt $x \in \bigcap_{h \in H} h$, der $c^T x$ maximiert, also $\max\{c^{\mathrm{T}}x \mid Ax \leq b, x \geq 0\}$.
- LP ist ein zentrales Optimierungsverfahren im Operations Research (zuerst entwickelt 1939 von Kantorovich).

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 $d = 2$

Frage: Wie kann der Lösungsraum aussehen?

Algorithmen für LPs

Viele Algorithmen zum Lösen von LPs in der Praxis existieren:

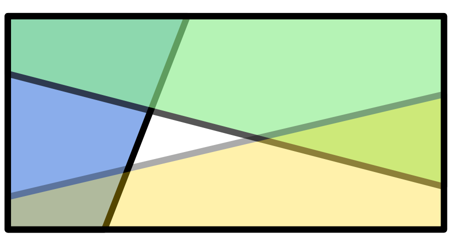
- Simplex-Algorithmus [Dantzig, 1947]
- Ellipsoid-Methode [Khatchiyan, 1979]
- Innere-Punkt-Methode [Karmarkar, 1979]

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

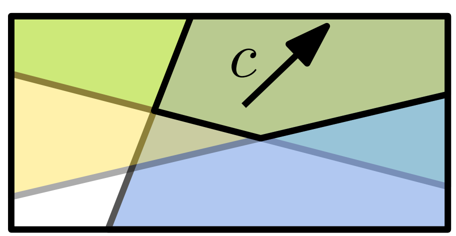
Heute: Spezialfall $d = 2$

Möglichkeiten für den Lösungsraum:

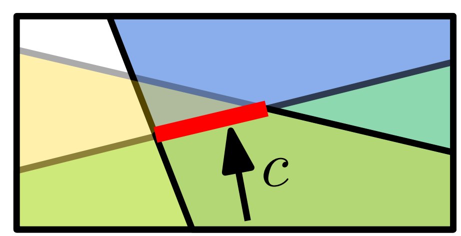
$\bigcap H$ beschränkt



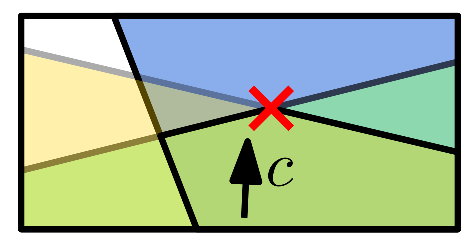
$\bigcap H = \emptyset$
unlösbar



$\bigcap H$ 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

Wie lässt sich effizient der Schnitt zweier konvexer Regionen bestimmen?

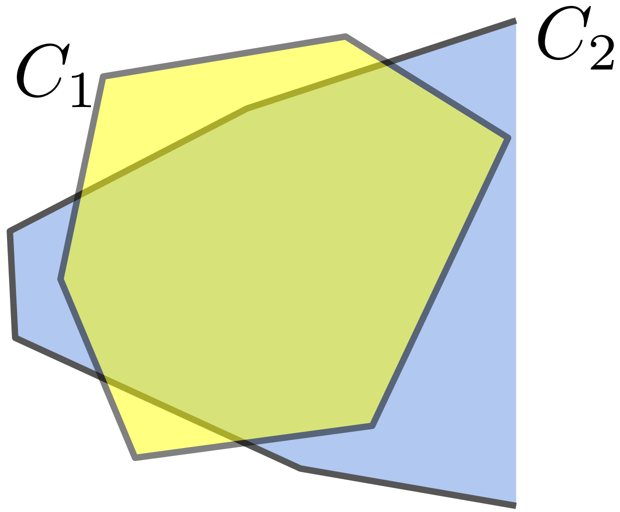
Schnitt konvexer Regionen

Methode IntersectConvexRegions(C_1, C_2) kann mit Sweep-Line Verfahren implementiert werden:

Schnitt konvexer Regionen

Methode IntersectConvexRegions(C_1, C_2) kann mit Sweep-Line Verfahren implementiert werden:

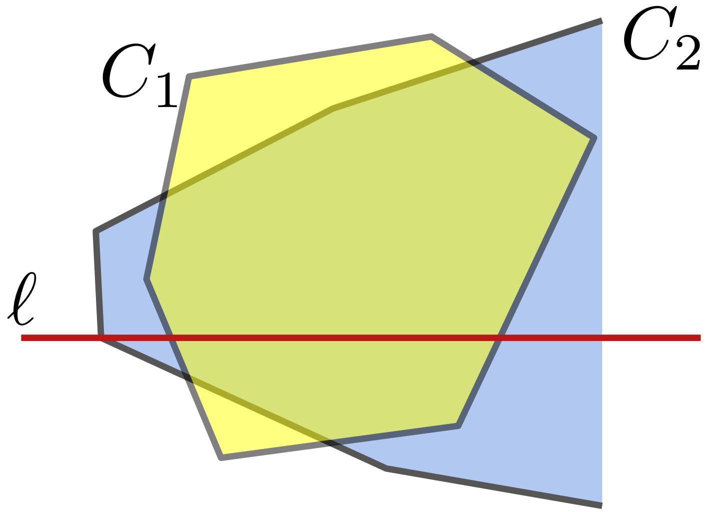
- betrachte jeweils linke und rechte Grenze von C_1 und C_2



Schnitt konvexer Regionen

Methode IntersectConvexRegions(C_1, C_2) kann mit Sweep-Line Verfahren implementiert werden:

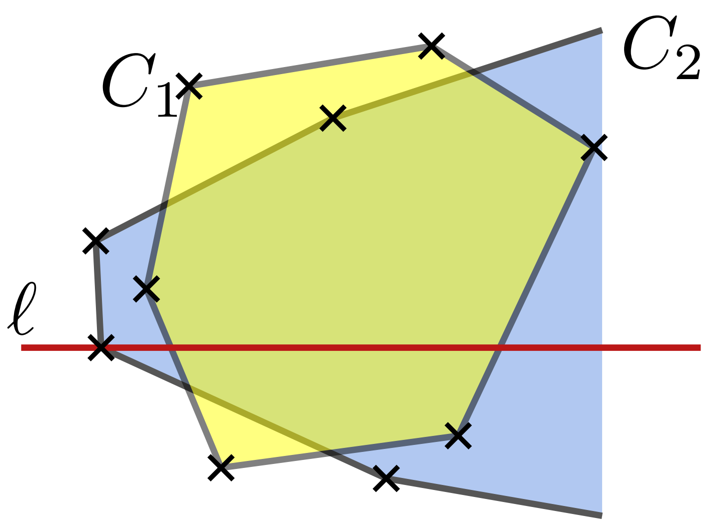
- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten



Schnitt konvexer Regionen

Methode IntersectConvexRegions(C_1, C_2) kann mit Sweep-Line Verfahren implementiert werden:

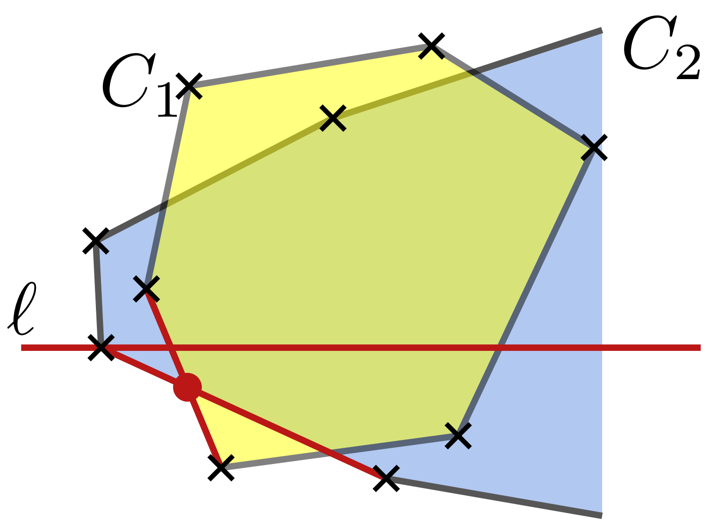
- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten
- Knoten in $C_1 \cup C_2$ definieren Events,



Schnitt konvexer Regionen

Methode IntersectConvexRegions(C_1, C_2) kann mit Sweep-Line Verfahren implementiert werden:

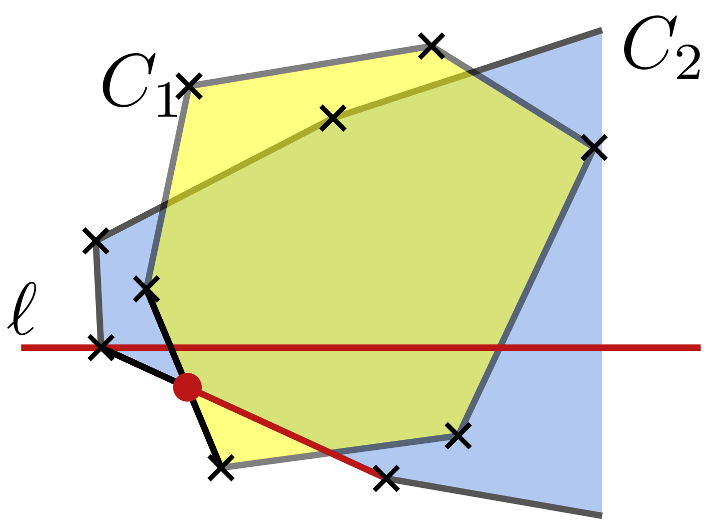
- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten
- Knoten in $C_1 \cup C_2$ definieren Events,



Schnitt konvexer Regionen

Methode IntersectConvexRegions(C_1, C_2) kann mit Sweep-Line Verfahren implementiert werden:

- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line ℓ von oben nach unten und speichere die ≤ 4 schneidenden Kanten
- Knoten in $C_1 \cup C_2$ definieren Events,

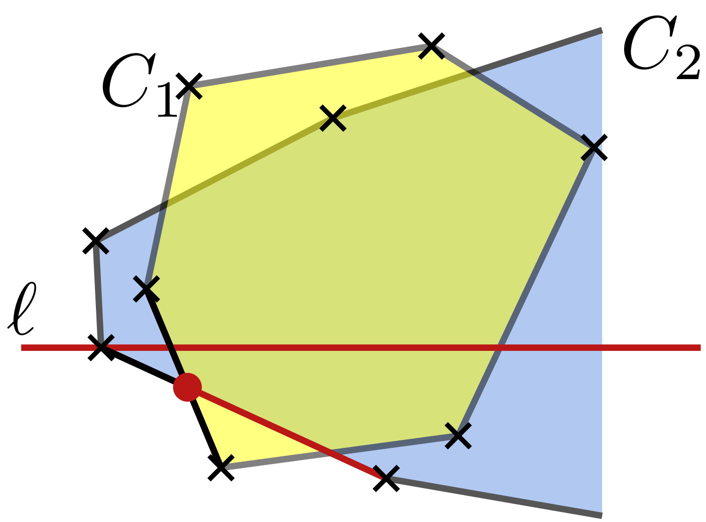


Frage: Welche Laufzeit hat der Sweep-Line Algorithmus?

Schnitt konvexer Regionen

Methode IntersectConvexRegions(C_1, C_2) kann mit Sweep-Line Verfahren implementiert werden:

- betrachte jeweils linke und rechte Grenze von C_1 und C_2
- bewege sweep line 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 1:
Der Schnitt zweier konvexer Polygone mit $n_1 + n_2 = n$ Knoten kann in $O(n)$ Zeit berechnet werden.

Laufzeit von IntersectHalfplanes(H)

IntersectHalfplanes(H)

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

Frage: 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

Frage: 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

Frage: 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?

Frage: 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

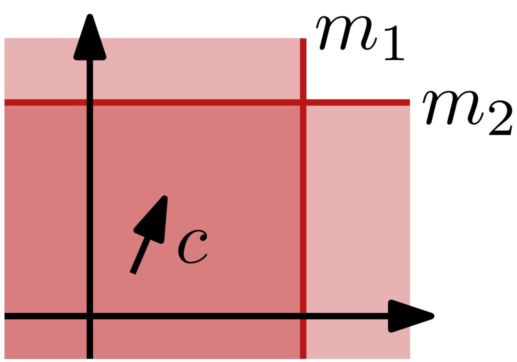
Idee: Statt gesamtes zulässiges Polygon zu berechnen suche inkrementell nach optimaler Ecke.

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.

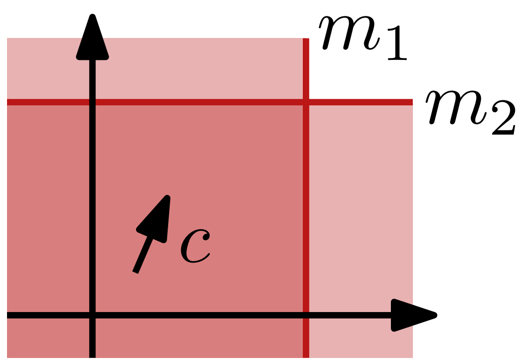
Invariante: aktuell beste Lösung ist eindeutige Ecke des zulässigen aktuellen Polygons

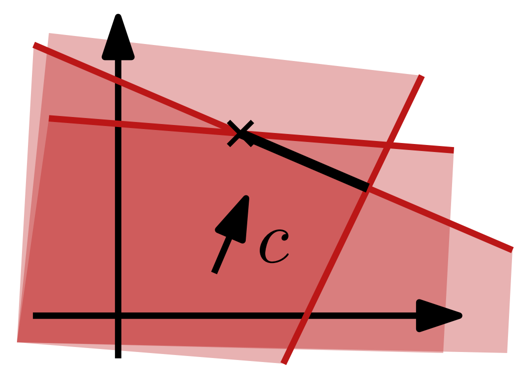
Wie kann man unbeschränkte zulässige Gebiete umgehen?

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 $H = \{h_1, \ldots, h_n\}$, $c = (c_x, c_y)$, zulässigem Polygon C und $1 \leq i \leq n$ definiere

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

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

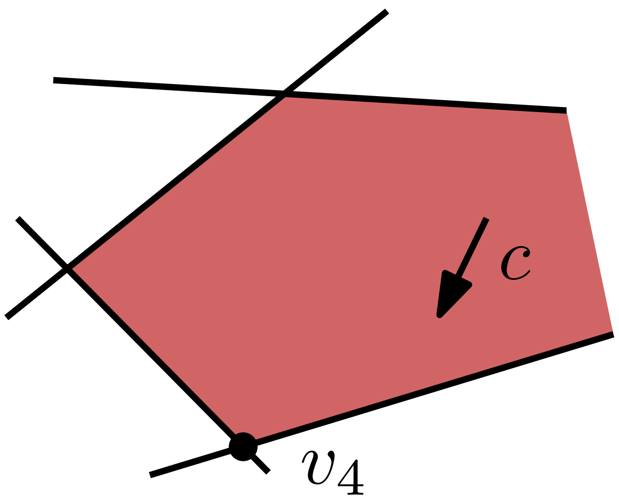
- jede Region C_i hat eine eindeutige optimale Ecke v_i
- es gilt die Inklusionsbeziehung $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

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?

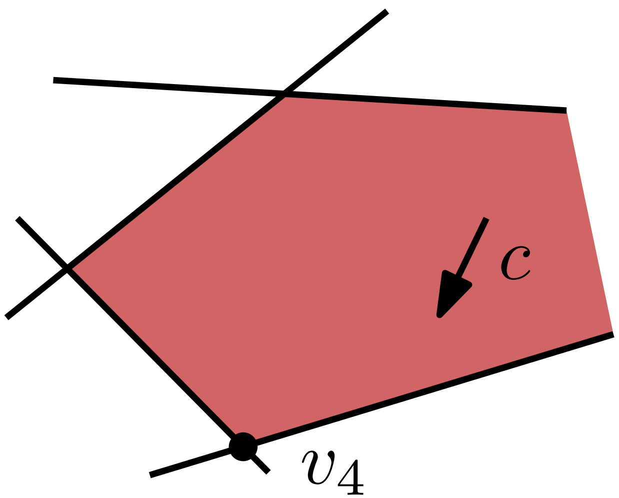


vor dem Einfügen

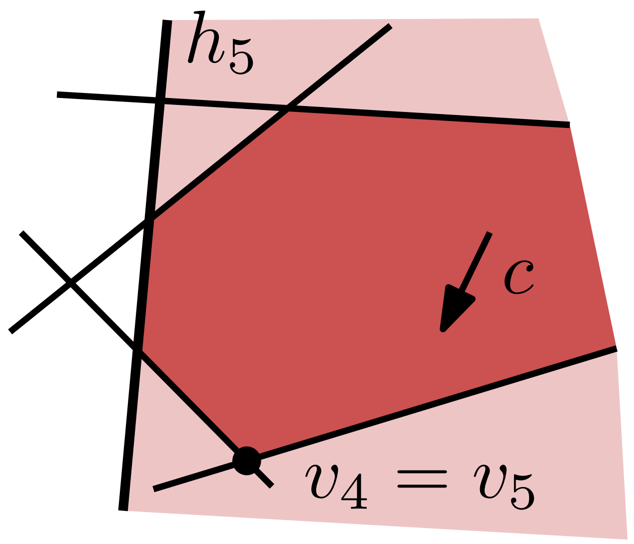
Eigenschaften

- 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?



vor dem Einfügen

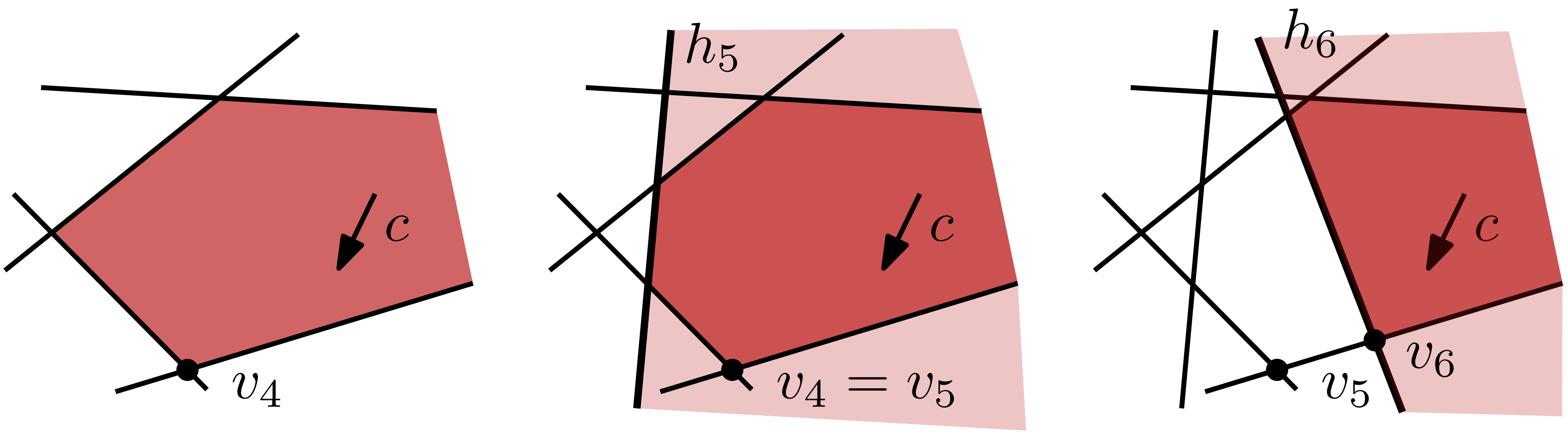


keine Änderung

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?



vor dem Einfügen — keine Änderung — Änderung

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 1: 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 1: 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 $v_i \in \ell_i$.

Beweis: (i) $v_{i-1} \in h_i$ und $v_{i-1} \in C_{i-1}$
$\Rightarrow v_{i-1} \in C_i = C_{i-1} \cap h_i \subseteq C_{i-1}$
weiterhin optimal

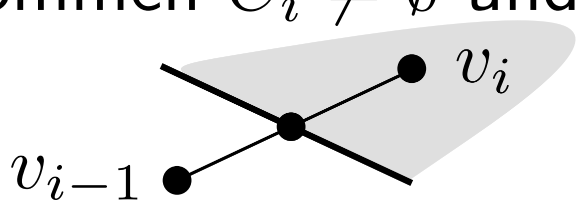
Eigenschaften

- 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 1: 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 $v_i \in \ell_i$.

Beweis: (i) $v_{i-1} \in h_i$ und $v_{i-1} \in C_{i-1}$
$\Rightarrow v_{i-1} \in C_i = C_{i-1} \cap h_i \subseteq C_{i-1}$
weiterhin optimal
(ii) Angenommen $C_i \neq \emptyset$ und $v_i \notin \ell_i$
dann



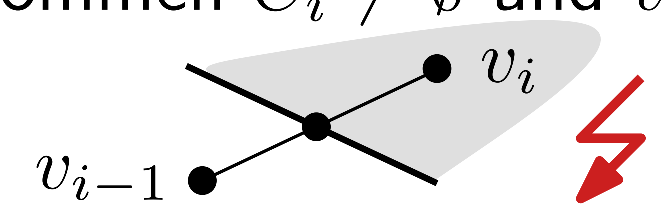
Eigenschaften

- 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 1: 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 $v_i \in \ell_i$.

Beweis:
(i) $v_{i-1} \in h_i$ und $v_{i-1} \in C_{i-1}$
$\Rightarrow v_{i-1} \in C_i = C_{i-1} \cap h_i \subseteq C_{i-1}$
weiterhin optimal
(ii) Angenommen $C_i \neq \emptyset$ und $v_i \notin \ell_i$
dann



Eindimensionales LP

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

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

Eindimensionales LP

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

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

Eindimensionales LP

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

- parametrisiere $\ell_i : y = ax + b$
- definiere neue Zielfunktion $f_c^i(x) = c^T \binom{x}{ax+b}$
- 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) von Lemma 1 suchen wir den besten Punkt auf der Strecke $\ell_i \cap C_{i-1}$:

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

$\qquad\qquad x \geq \sigma_x(\ell_j, \ell_i)$ falls $\ell_i \cap h_j$ nach links beschr.

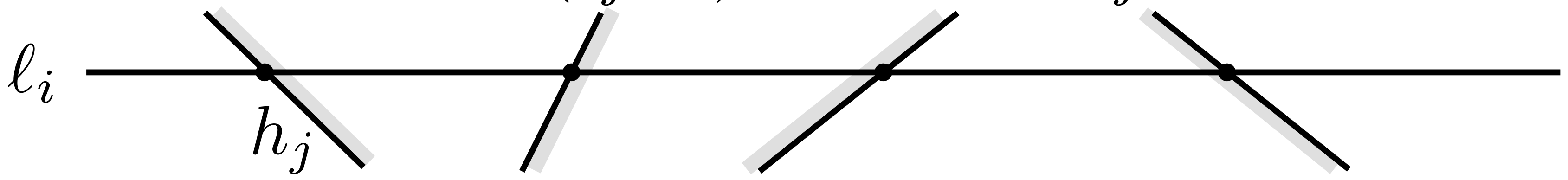
Eindimensionales LP

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

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

$$\begin{array}{lrcll} \text{maximiere} & \multicolumn{4}{l}{f_c^i(x) = c_x x + c_y(ax+b)} \\ \text{mit NB} & x & \leq & \sigma_x(\ell_j, \ell_i) & \text{falls } \ell_i \cap h_j \text{ nach rechts beschr} \\ & x & \geq & \sigma_x(\ell_j, \ell_i) & \text{falls } \ell_i \cap h_j \text{ nach links beschr.} \end{array}$$



Eindimensionales LP

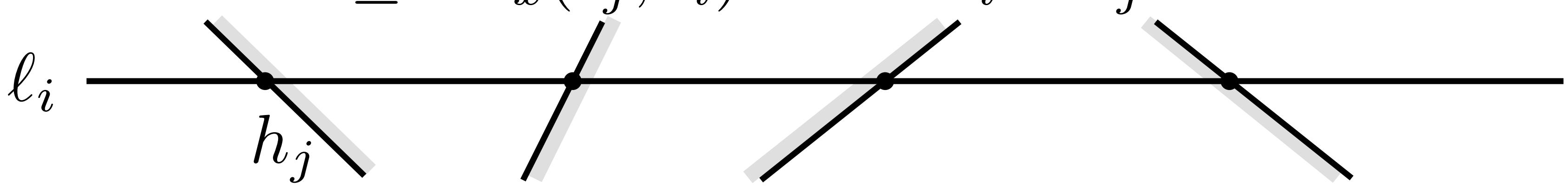
Im Fall (ii) von Lemma 1 suchen wir den besten Punkt auf der Strecke $\ell_i \cap C_{i-1}$:

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

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

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



Wie löst man dieses LP? Welche Laufzeit?

Eindimensionales LP

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

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

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

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

Inkrementeller Algorithmus

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

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**
      return unlösbar
    $O(i)$
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**
      return unlösbar $O(i)$
return v_n

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

Frage: Kann wirklich n-mal der else-Fall auftreten?

Inkrementeller Algorithmus

2dBoundedLP(H, c, m_1, m_2)

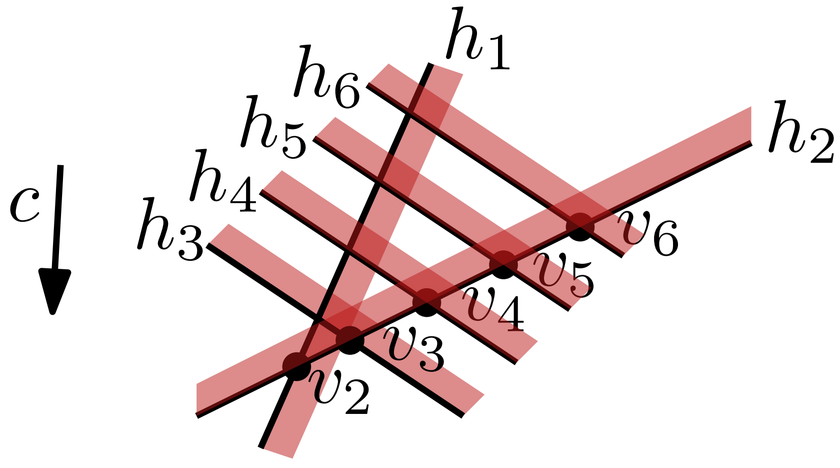
$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ eindeutige Ecke von C_0
for $i \leftarrow 1$ **to** n **do**
  if $v_{i-1} \in h_i$ **then**
    $v_i \leftarrow v_{i-1}$ $\quad O(1)$
  else
    $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
    if $v_i =$ nil **then**
      return unlösbar $\quad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

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

Lemma 3: Algorithmus 2dBoundedLP benötigt $\Theta(n^2)$ Laufzeit um LP mit n Nebenbed. und 2 Variablen zu lösen.

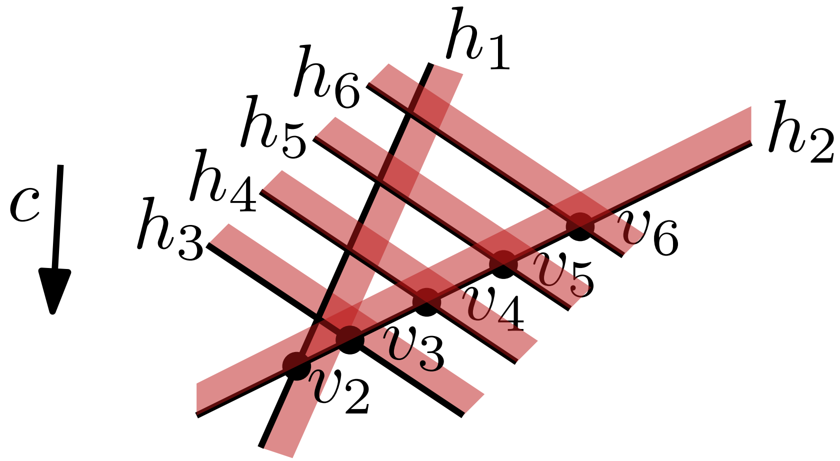
Gibt es einen Ausweg?

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



Gibt es einen Ausweg?

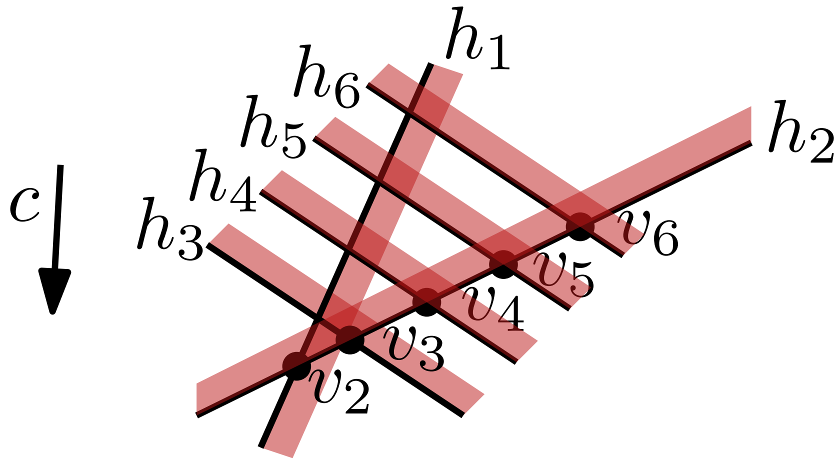
Beob.: Nicht die Halbebenen H sind problematisch für die Laufzeit, sondern die Reihenfolge der Abarbeitung.



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?

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 $A[1 \ldots n]$

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 zufälligen Permutation ab. Betrachte daher die **erwartete Laufzeit**.

Laufzeitanalyse

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

Laufzeitanalyse

2dRandomizedBoundedLP(H, c, m_1, m_2)

```
C_0 ← m_1 ∩ m_2
v_0 ← eindeutige Ecke von C_0                          O(1)
H ← RandomPermutation(H)                               O(n)
for i ← 1 to n do
    if v_{i-1} ∈ h_i then
        v_i ← v_{i-1}                                  O(1)
    else
        v_i ← 1dBoundedLP(σ(H_{i-1}), f_c^i)           O(i)
        if v_i = nil then
            return unlösbar                            O(1)
return v_n                                             O(1)
```

Laufzeitanalyse

2dRandomizedBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ eindeutige Ecke von C_0 — $O(1)$
$H \leftarrow$ RandomPermutation(H) — $O(n)$
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$(\sigma(H_{i-1}), f_c^i)$ — $O(i)$
    if $v_i =$ nil **then**
      return unlösbar — $O(1)$
return v_n — $O(1)$

$$O(n) + \sum_{i=1}^{n} \{P[v_{i-1} \in h_i] * O(1) + P[v_{i-1} \notin h_i] * O(i)\}$$

Laufzeitanalyse

2dRandomizedBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ eindeutige Ecke von C_0 $\quad O(1)$
$H \leftarrow$ RandomPermutation(H) $\quad O(n)$
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$(\sigma(H_{i-1}), f_c^i)$ $\quad O(i)$
    if $v_i =$ nil **then**
      return unlösbar $\quad O(1)$
return v_n $\quad O(1)$

$$O(n) + \sum_{i=1}^{n} \{ \underbrace{P[v_{i-1} \in h_i]}_{\leq 1} * O(1) + \underbrace{P[v_{i-1} \notin h_i]}_{\leq 2/i} * O(i) \}$$

Rückwärtsanalyse: nach Einfügen von h_i ist die Wahrscheinlichkeit, dass v_i durch h_i definiert ist $\leq 2/i$

Laufzeitanalyse

2dRandomizedBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ eindeutige Ecke von C_0 $O(1)$
$H \leftarrow$ RandomPermutation(H) $O(n)$
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$(\sigma(H_{i-1}), f_c^i)$ $O(i)$
    if $v_i =$ nil **then**
      return unlösbar $O(1)$
return v_n $O(1)$

$$O(n) + \sum_{i=1}^{n} \{ \underbrace{P[v_{i-1} \in h_i]}_{\leq 1} * O(1) + \underbrace{P[v_{i-1} \notin h_i]}_{\leq 2/i} * O(i) \}$$
$$= O(n)$$

Rückwärtsanalyse: nach Einfügen von h_i ist die Wahrscheinlichkeit, dass v_i durch h_i definiert ist $\leq 2/i$

Laufzeitanalyse

```
2dRandomizedBoundedLP(H, c, m_1, m_2)
    C_0 ← m_1 ∩ m_2
    v_0 ← eindeutige Ecke von C_0                          O(1)
    H ← RandomPermutation(H)                               O(n)
    for i ← 1 to n do
        if v_{i-1} ∈ h_i then
            v_i ← v_{i-1}                                  O(1)
        else
            v_i ← 1dBoundedLP(σ(H_{i-1}), f_c^i)           O(i)
            if v_i = nil then
                return unlösbar                            O(1)
    return v_n                                             O(1)
```

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

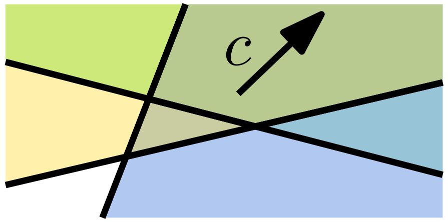
Unbeschränkte LPs

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

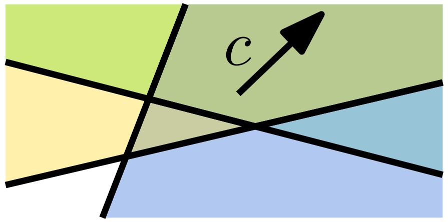


$\bigcap H$ unbeschränkt in Richtung c

Unbeschränkte LPs

Bisher: künstliche Beschränkung von C durch m_1 und m_2

Jetzt: erkenne und behandle unbeschränkte LPs



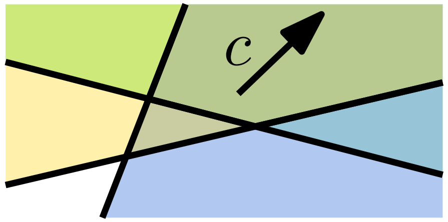
$\bigcap H$ unbeschränkt in Richtung c

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

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

Charakterisierung

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

Schritt 1
- $\langle d, c \rangle > 0$
- $\langle d, \eta(h) \rangle \geq 0$ für alle $h \in H$

Schritt 2 –
- LP (H', c) mit $H' = \{h \in H \mid \langle d, \eta(h) \rangle = 0\}$ ist lösbar.

Teste Unbeschränktheit durch 1dimensionales LP in 2 Schritten:

Charakterisierung

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

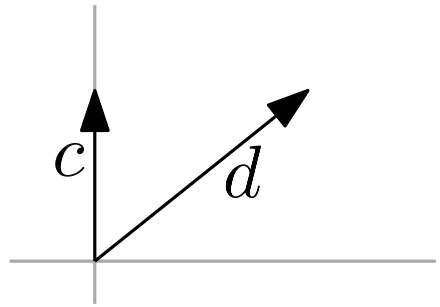
Schritt 1 $\left\{ \begin{array}{l} \bullet\ \langle d, c \rangle > 0 \\ \bullet\ \langle d, \eta(h) \rangle \geq 0 \text{ für alle } h \in H \end{array} \right.$

Schritt 2 – • LP (H', c) mit $H' = \{h \in H \mid \langle d, \eta(h) \rangle = 0\}$ ist lösbar.

Teste Unbeschränktheit durch 1dimensionales LP in 2 Schritten:

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 $\rightarrow$ 1d-LP

Schritt 1: *Existiert d mit $\langle d, c\rangle > 0$ und $\langle d, \eta(h)\rangle \geq 0$ für alle h?*

ja $\downarrow$ **Schritt 2:** *Ist LP (H', c) lösbar?*

nein $\downarrow$ (H, c) ist beschränkt

Schritt 2: *Ist LP (H', c) lösbar?*

ja $\downarrow$ (H, c) ist unbeschränkt

nein $\downarrow$ (H, c) ist 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.

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 auch (H, c) unlösbar und wir können zwei Halbebenen mit leerem Schnitt zurückgeben.

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

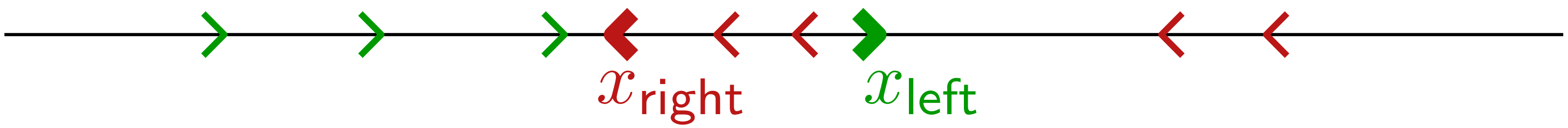
Wenn beide Schritte eine Lösung liefern ist (H, c) unbeschränkt und wir können einen Strahl ρ als Zeugen angeben.

Wenn LP in Schritt 2 unlösbar ist, dann ist auch (H, c) unlösbar und wir können zwei Halbebenen mit leerem Schnitt zurückgeben.

Wenn LP in Schritt 1 unlösbar ist, ist (H, c) nach Lemma 4 beschränkt und wir können zwei beschränkende Halbebenen zurückgeben.

Zeugen für Beschränktheit

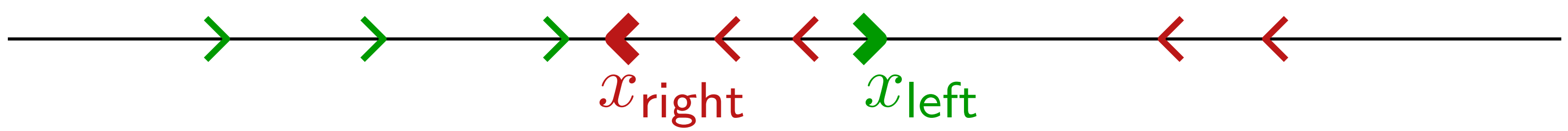
Beob.: Auch wenn LP aus Schritt 1 unlösbar ist, lassen sich die Informationen weiterverwenden!



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 mit 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 als m_1 und m_2

Algorithmus

2dRandomizedLP(H, c)

1d LP: $\exists$? 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)

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

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

Algorithmus

2dRandomizedLP(H, c)

1d LP: ∃? 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)
  $\overline{H} \leftarrow H \setminus \{h_1, h_2\}$
  return 2dRandomizedBoundedLP$(\overline{H}, c, h_1, h_2)$

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

Zusammenfassung

Satz 1: Der Schnitt zweier konvexer Polygone mit $n_1 + n_2 = n$ Knoten kann in $O(n)$ Zeit berechnet werden.

Folgerung: Ein zweidimensionales LP mit n Halbebenen kann in $O(n \log n)$ Zeit gelöst werden.

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

Satz 3: 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! Auch d-dimensionale LPs können randomisiert inkrementell und durch Lösen von $(d-1)$-dimensionalen LPs gelöst werden.

Diskussion

Lässt sich der zweidimensionale Algorithmus auch auf mehr Dimensionen verallgemeinern?

Ja! Auch d-dimensionale LPs können randomisiert inkrementell und durch Lösen von $(d-1)$-dimensionalen LPs gelöst werden.

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! Auch d-dimensionale LPs können randomisiert inkrementell und durch Lösen von $(d-1)$-dimensionalen LPs gelöst werden.

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! Auch d-dimensionale LPs können randomisiert inkrementell und durch Lösen von $(d-1)$-dimensionalen LPs gelöst werden.

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.