Computational Geometry • Lecture

Linear Programming

INSTITUTE FOR THEORETICAL INFORMATICS · FACULTY OF INFORMATICS

Tamara Mchedlidze

09.5.2018

Profit optimization

- You are the boss of a company, that produces two products P_1 und P_2 from three raw materials R_1, R_2 und R_3.
- Let's assume you produce x_1 items of the product P_1 and x_2 items of product P_2.
- Assume that items $P_1,\ P_2$ get profit of $300€$ and $500€$, respectively. Then the total profit is:

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

- Assume that the amout of raw material you need for P_1 and P_2 is:

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

- And in your warehouse there are $880R_1$, $150R_2$ and $60R_3$. So:

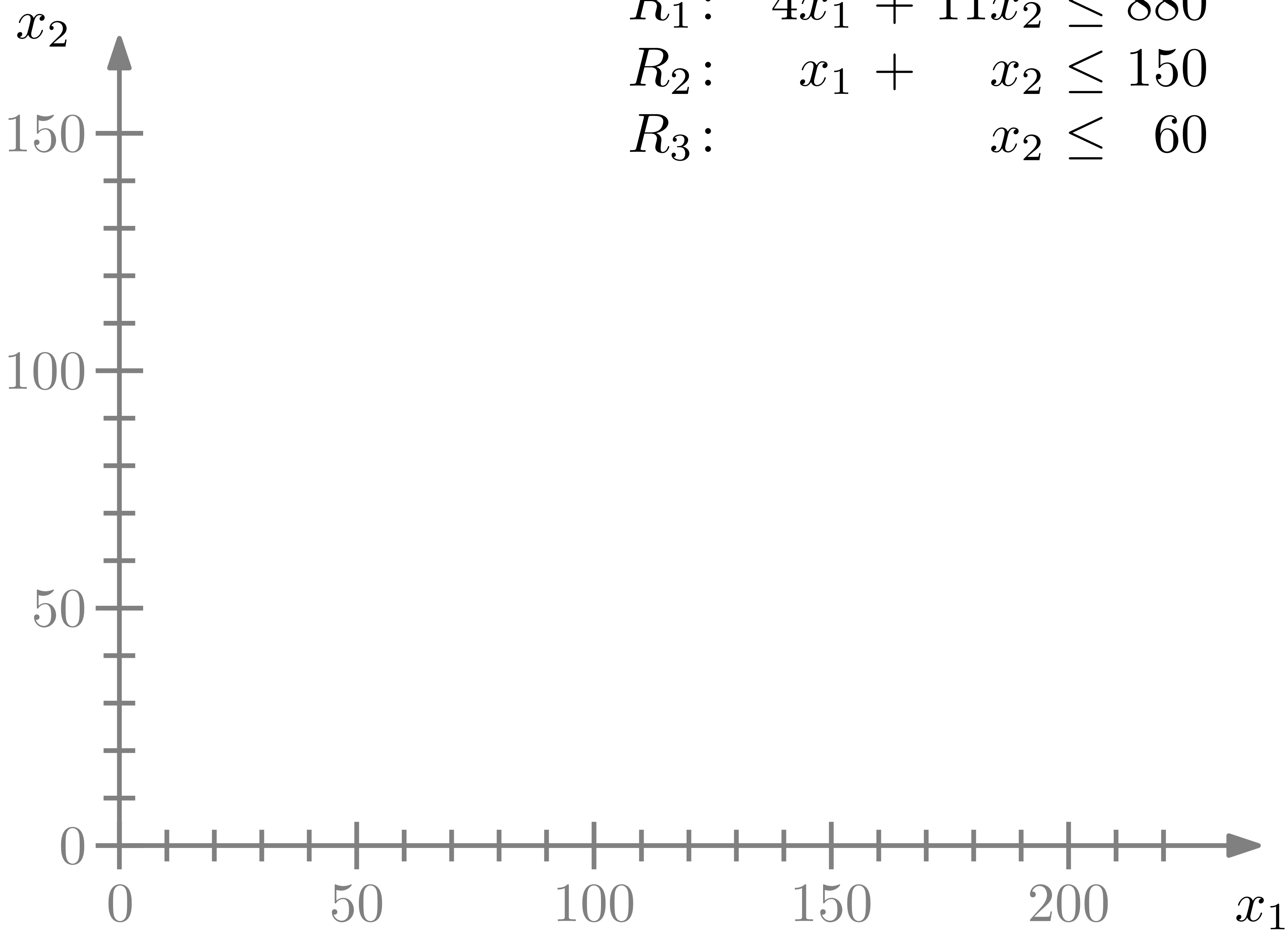
$$\begin{aligned} R_1: &\quad 4x_1 + 11x_2 \leq 880 \\ R_2: &\quad x_1 + x_2 \leq 150 \\ R_3: &\quad x_2 \leq 60 \end{aligned}$$

- Which choice for (x_1, x_2) maximizes your profit?

Solution

Linear constraints:

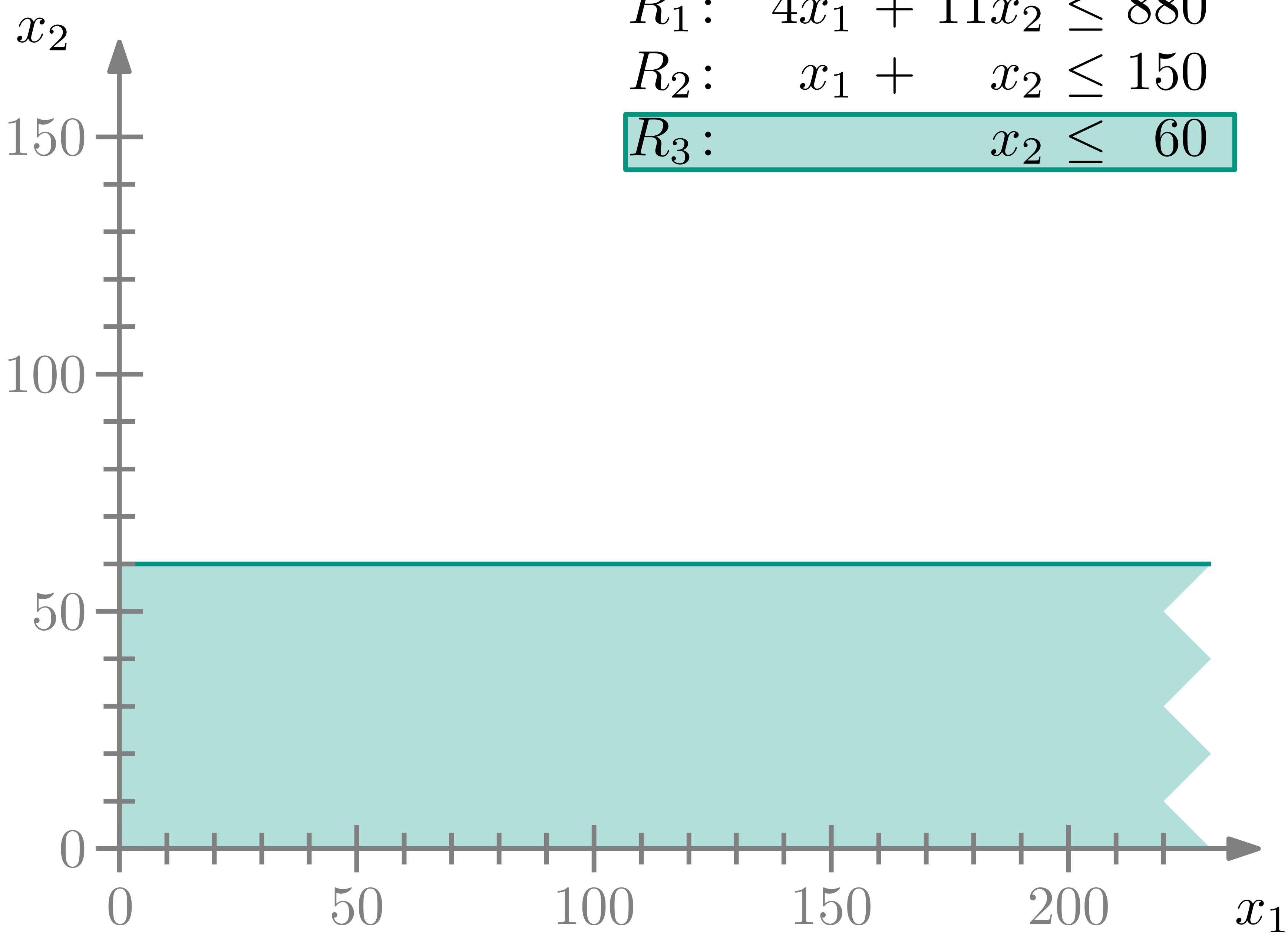
$$R_1: \quad 4x_1 + 11x_2 \leq 880$$
$$R_2: \quad x_1 + x_2 \leq 150$$
$$R_3: \quad x_2 \leq 60$$



Solution

Linear constraints:

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



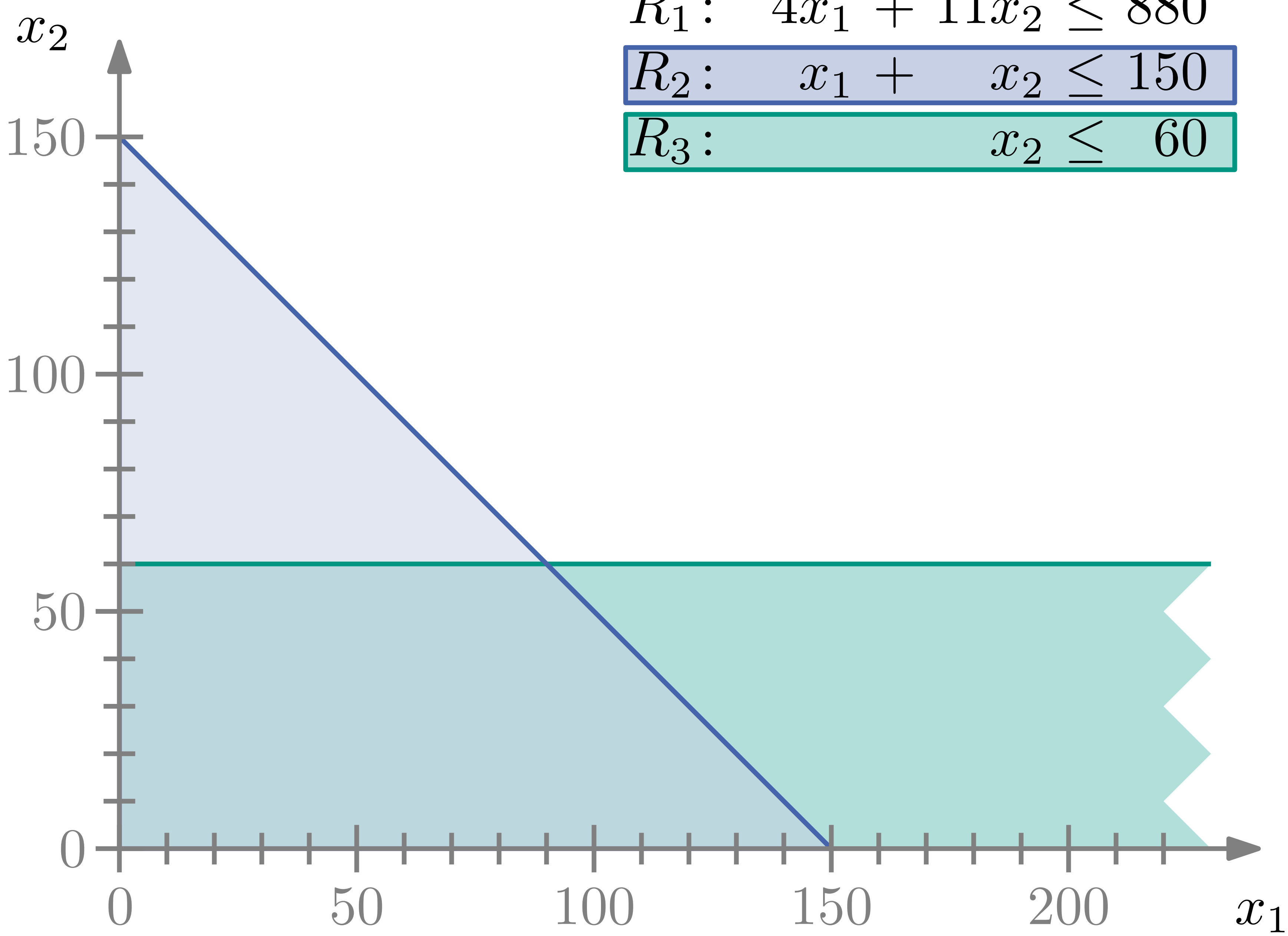
Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$



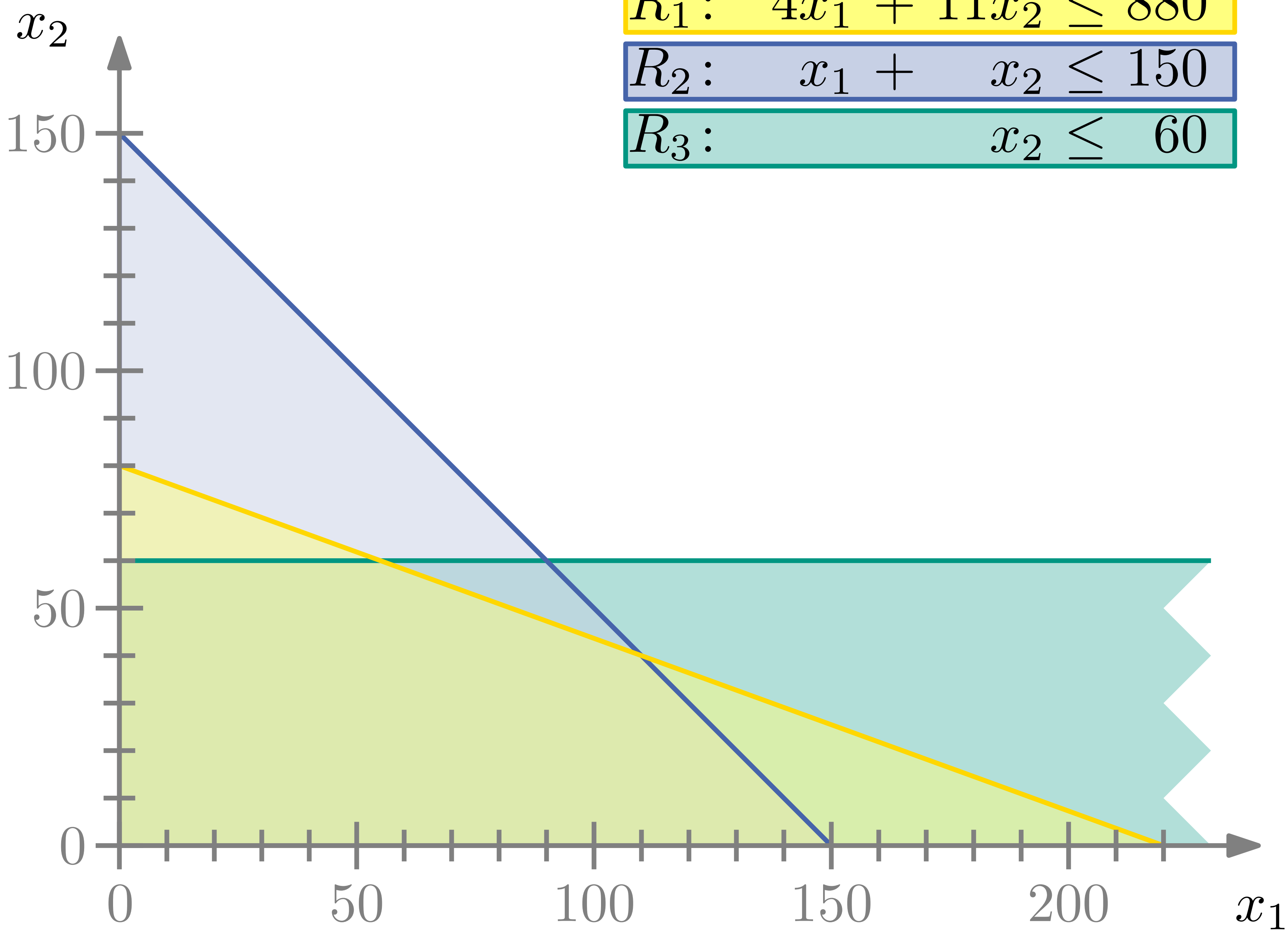
Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$



Solution

Linear constraints:

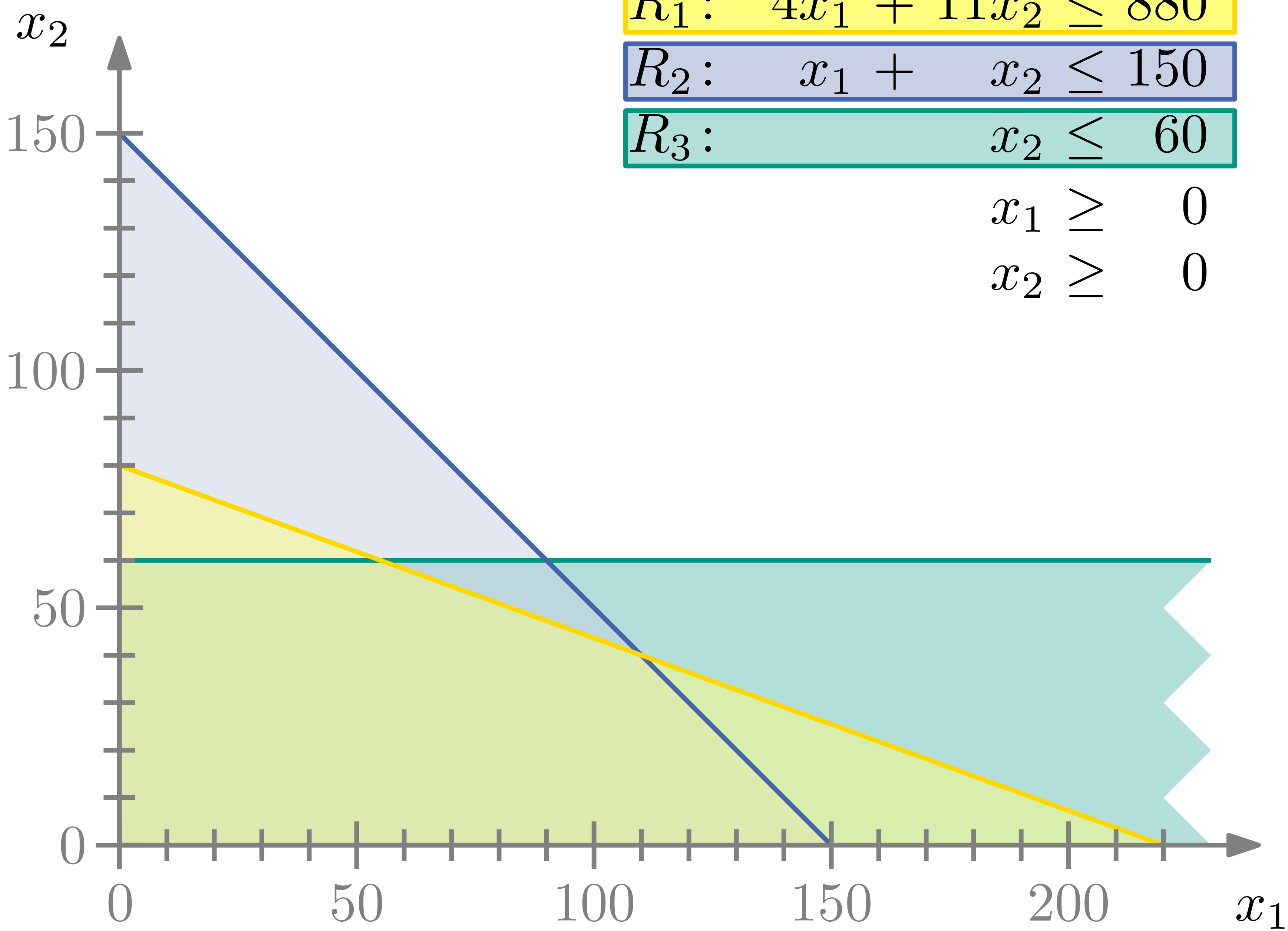
$R_1: \quad 4x_1 + 11x_2 \leq 880$

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

$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

$x_2 \geq 0$



Solution

Linear constraints:

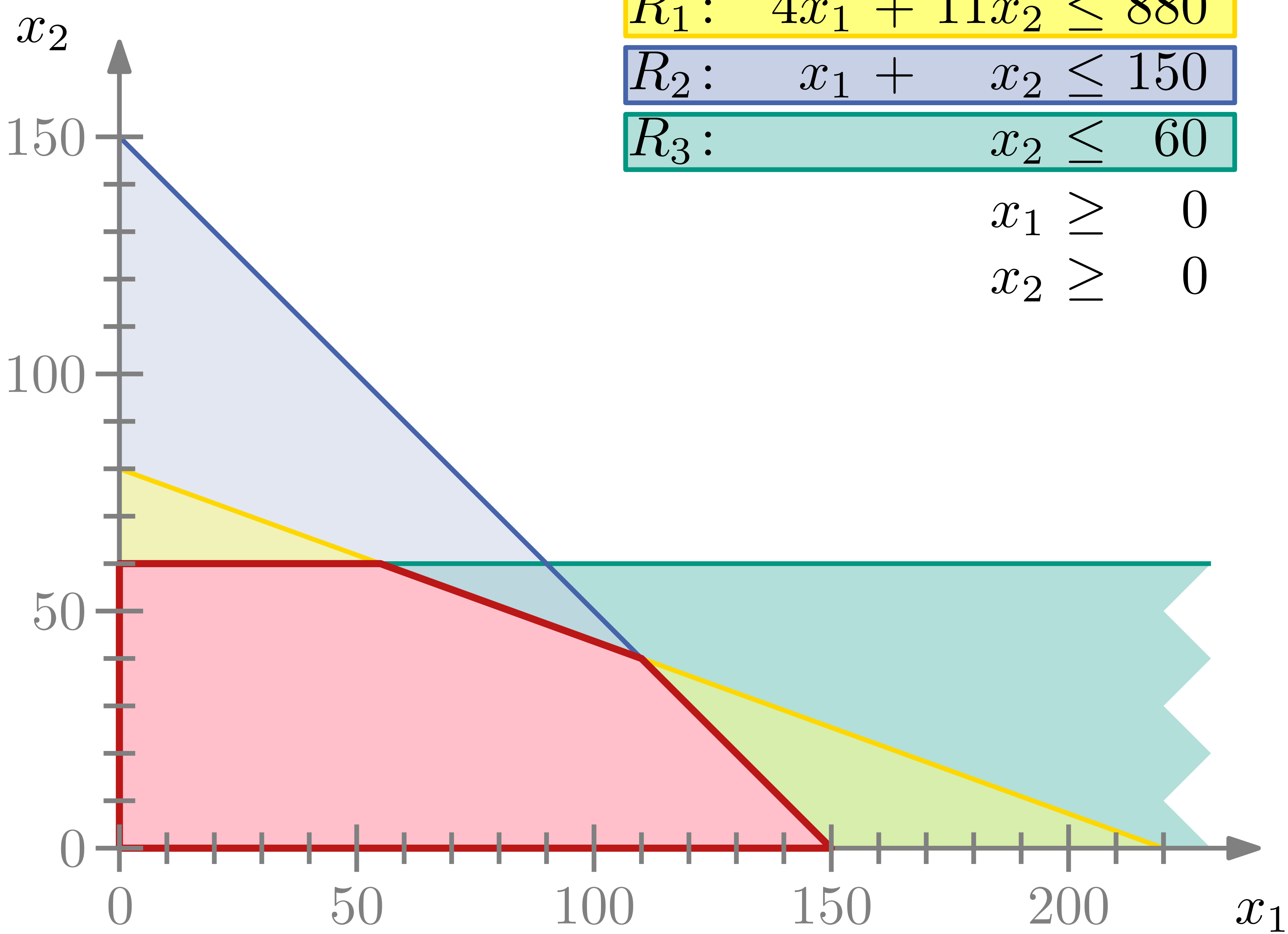
$R_1: \quad 4x_1 + 11x_2 \leq 880$

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

$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

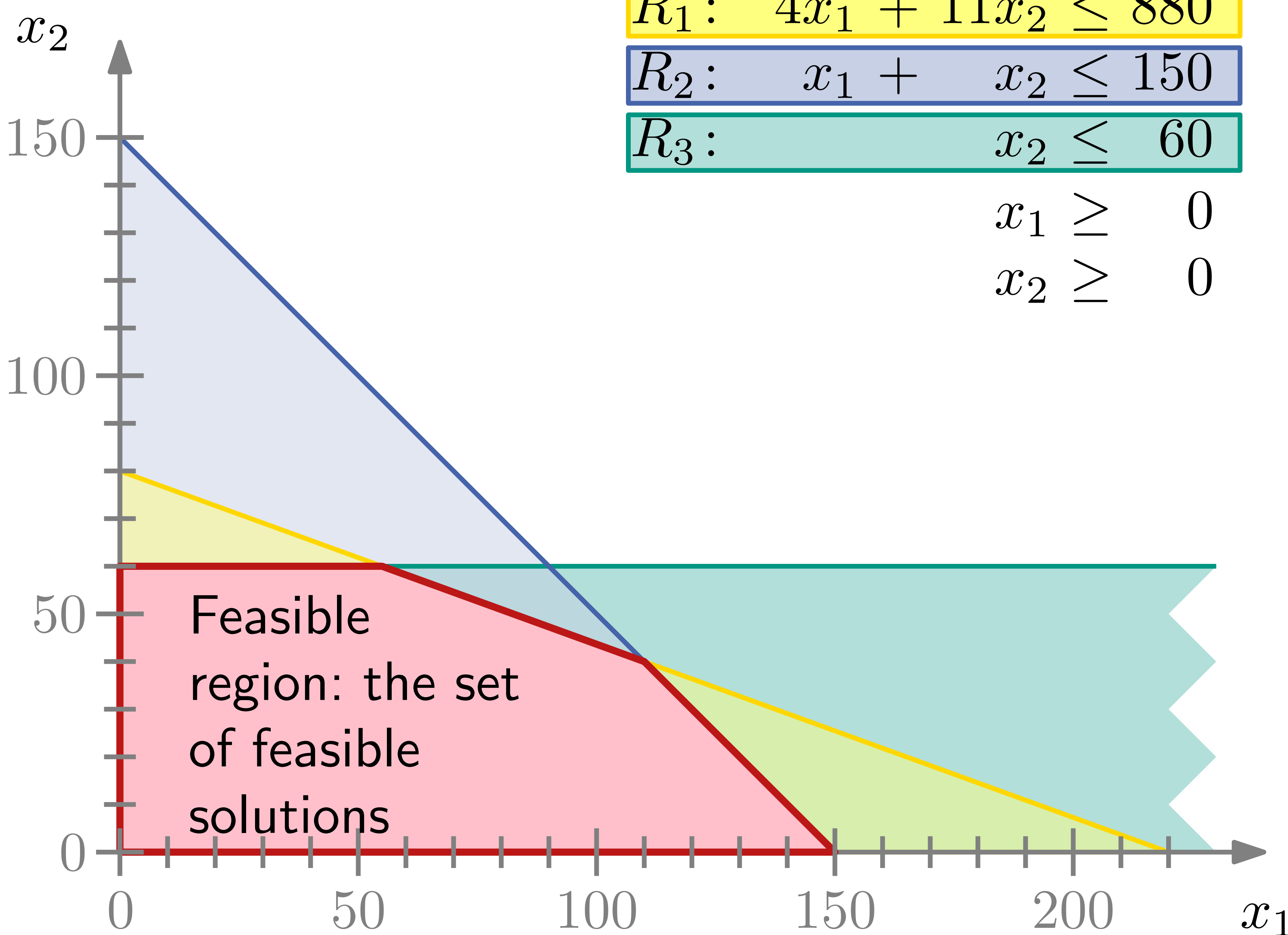
$x_2 \geq 0$



Solution

Linear constraints:

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



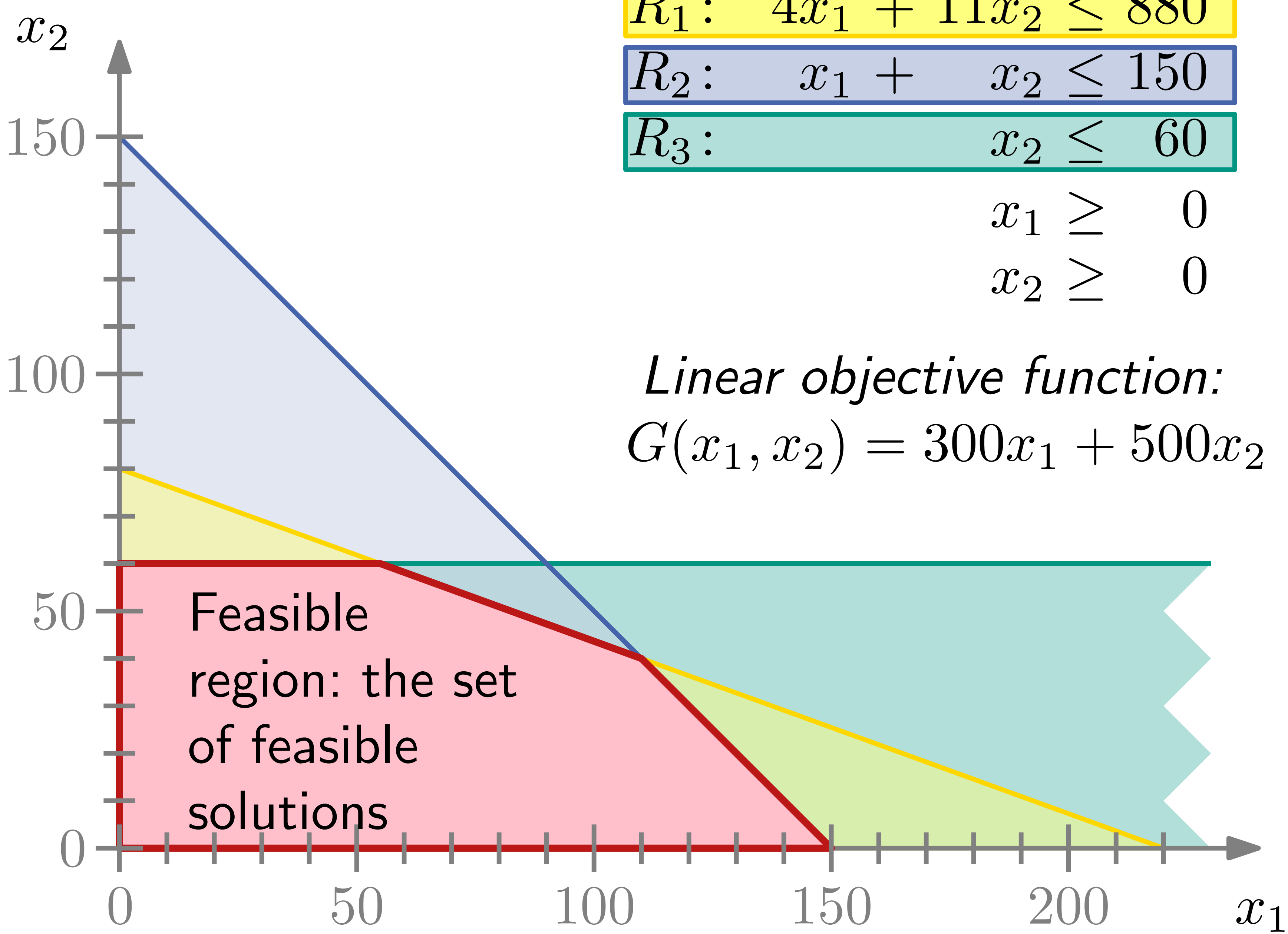
Solution

Linear constraints:

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

Linear objective function:

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



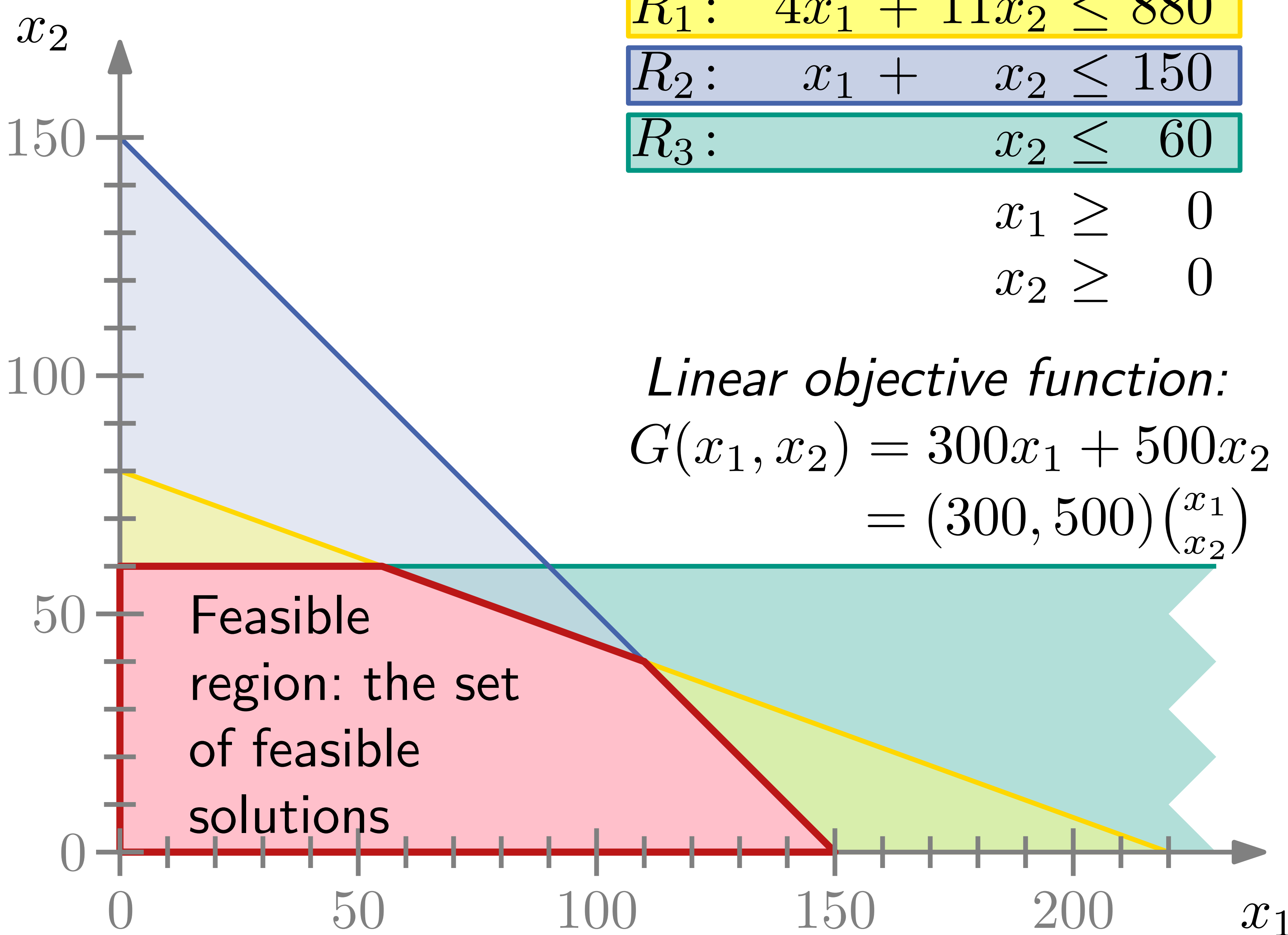
Solution

Linear constraints:

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

Linear objective function:

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



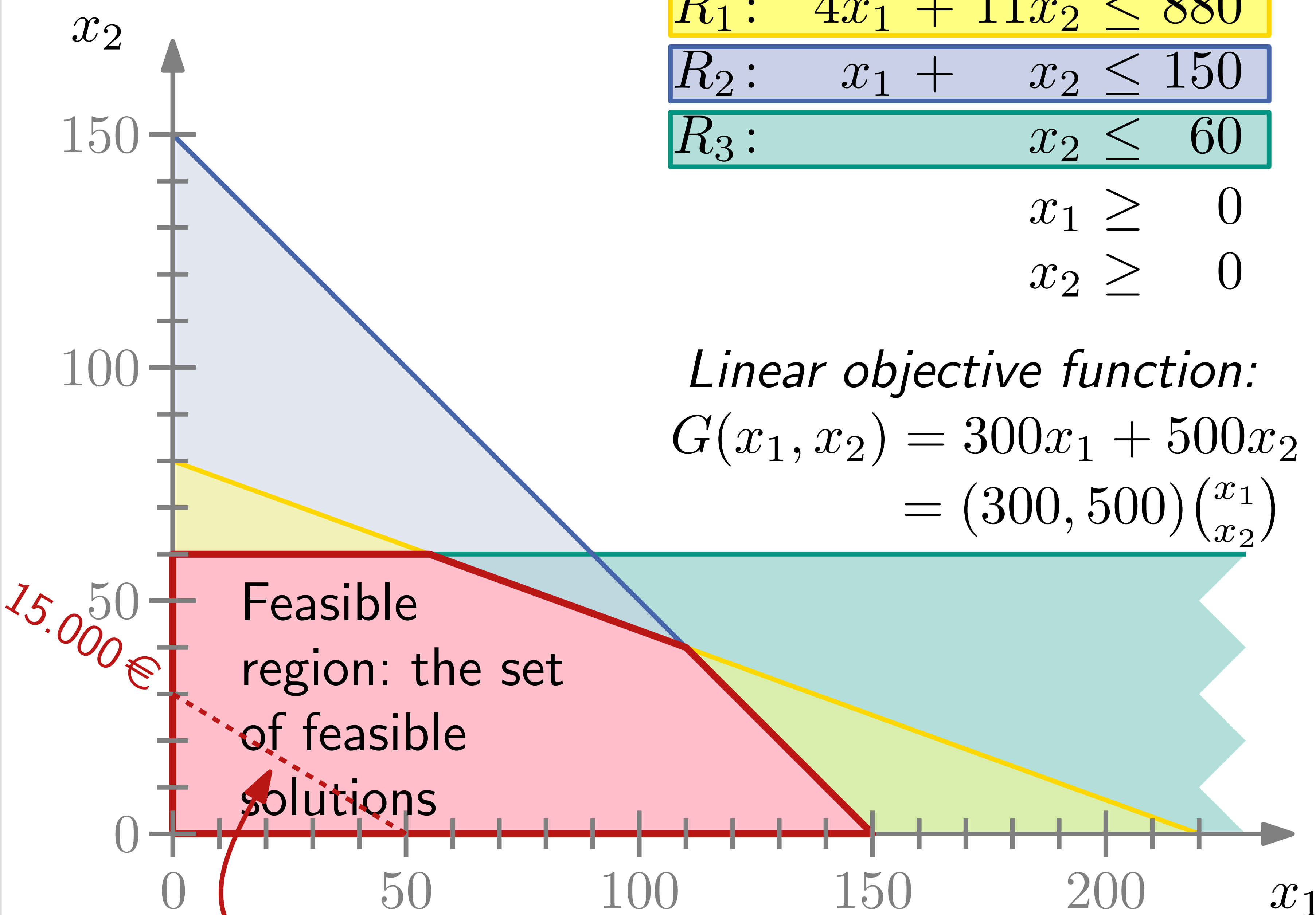
Solution

Linear constraints:

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

Linear objective function:

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



„Isocost line" (orthogonal to $\binom{300}{500}$)

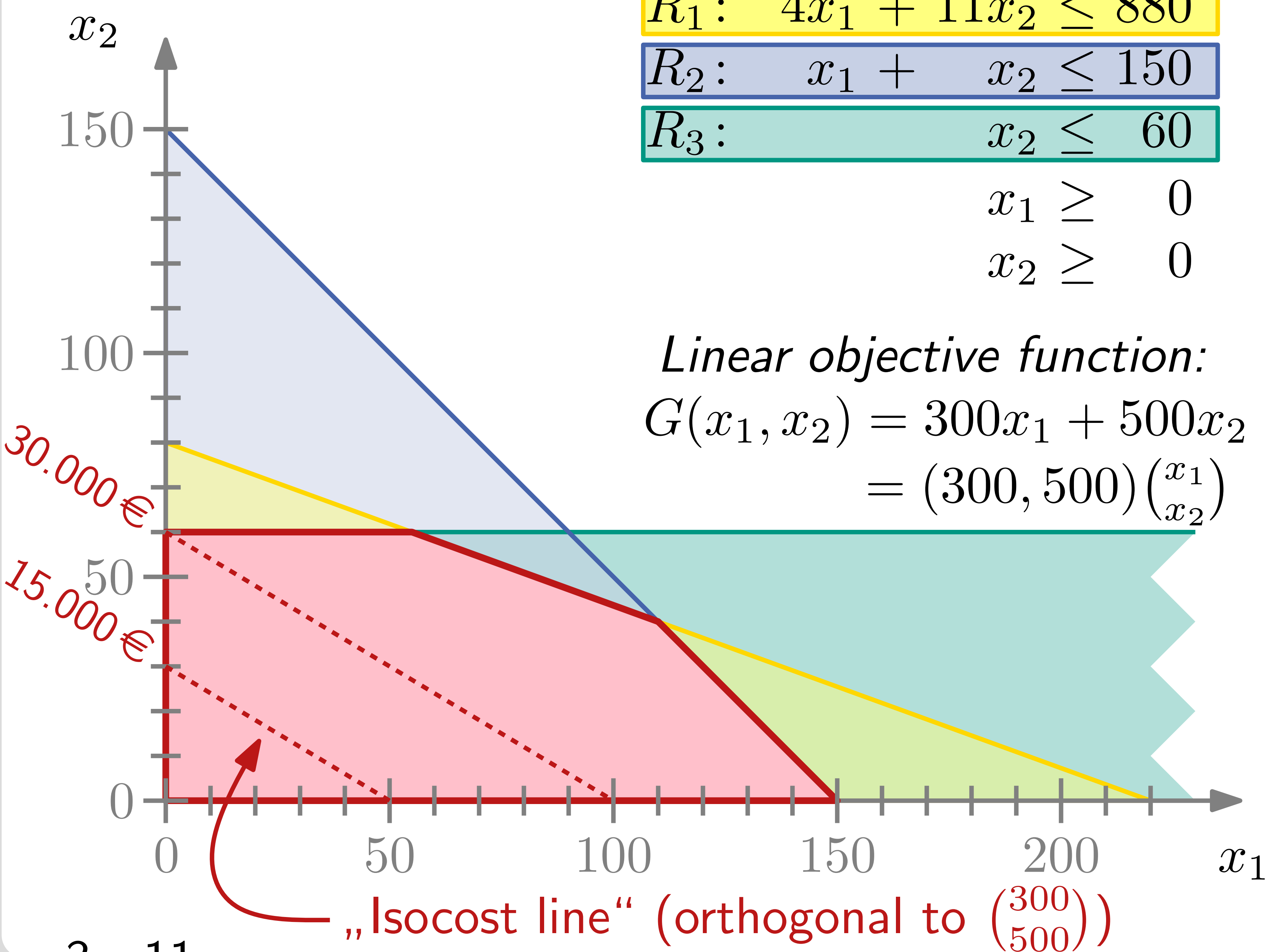
Solution

Linear constraints:

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

Linear objective function:

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



Solution

Linear constraints:

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

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

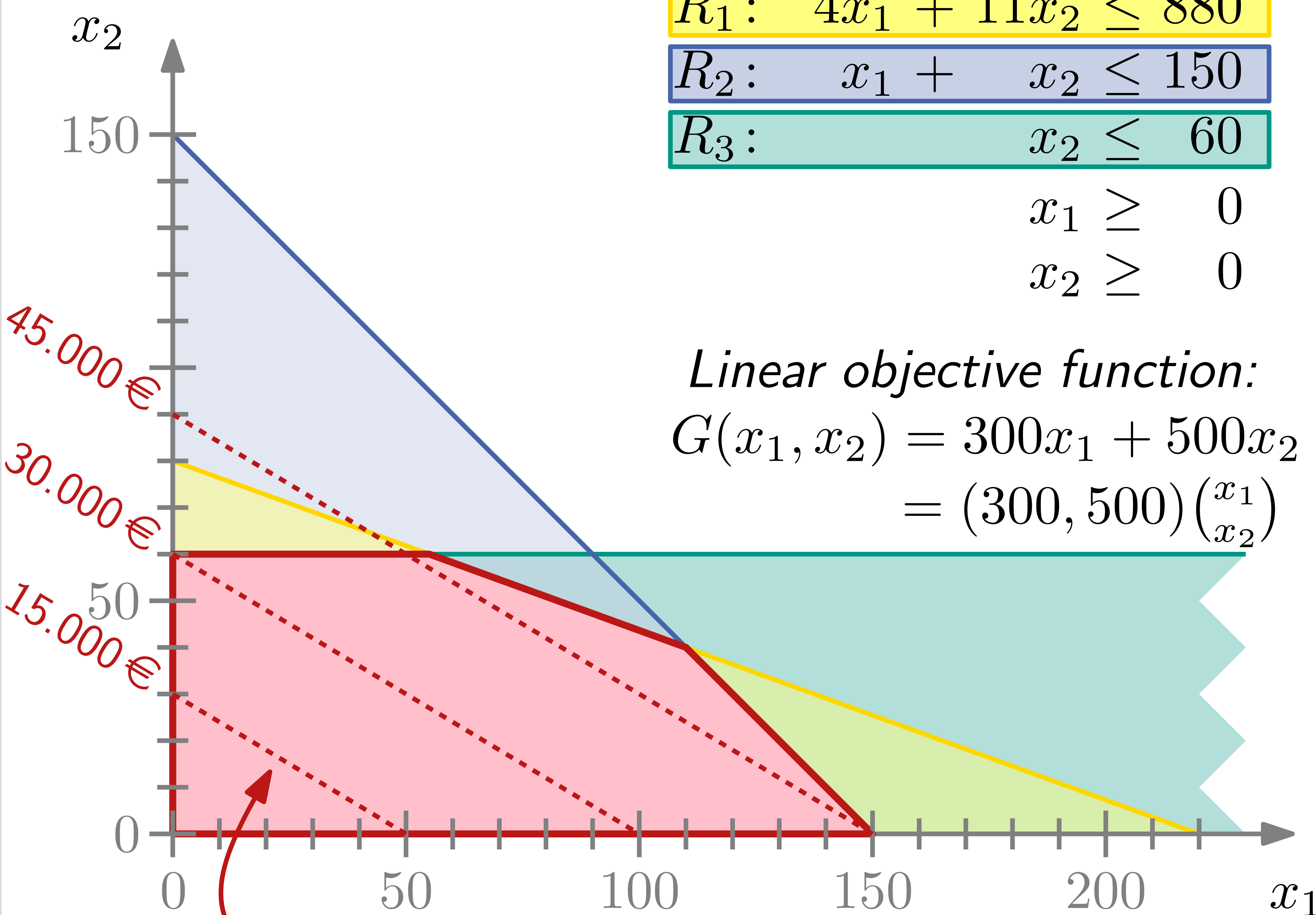
$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

$x_2 \geq 0$

Linear objective function:

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



„Isocost line“ (orthogonal to $\binom{300}{500}$)

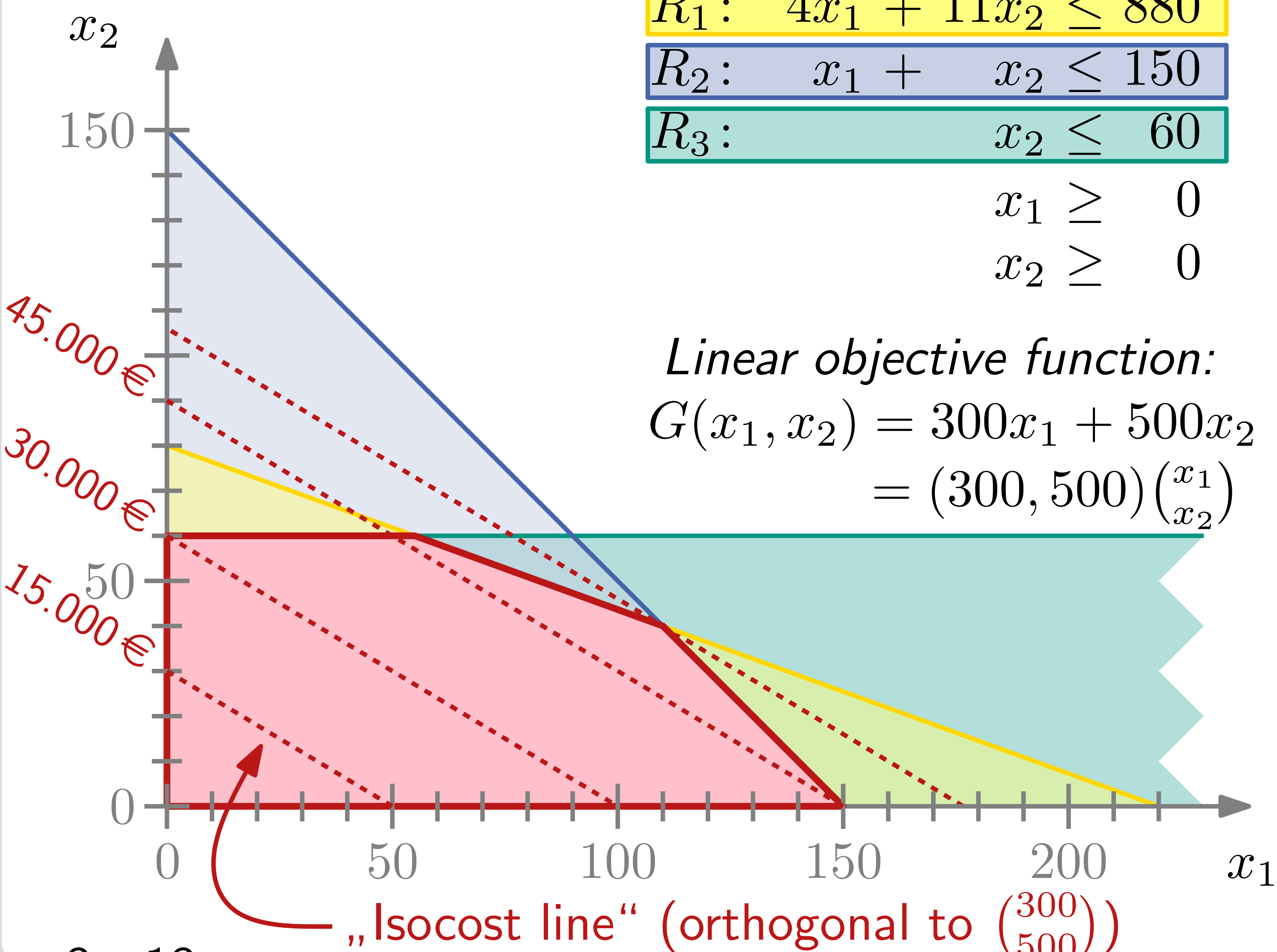
Solution

Linear constraints:

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

Linear objective function:

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



Solution

Linear constraints:

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

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

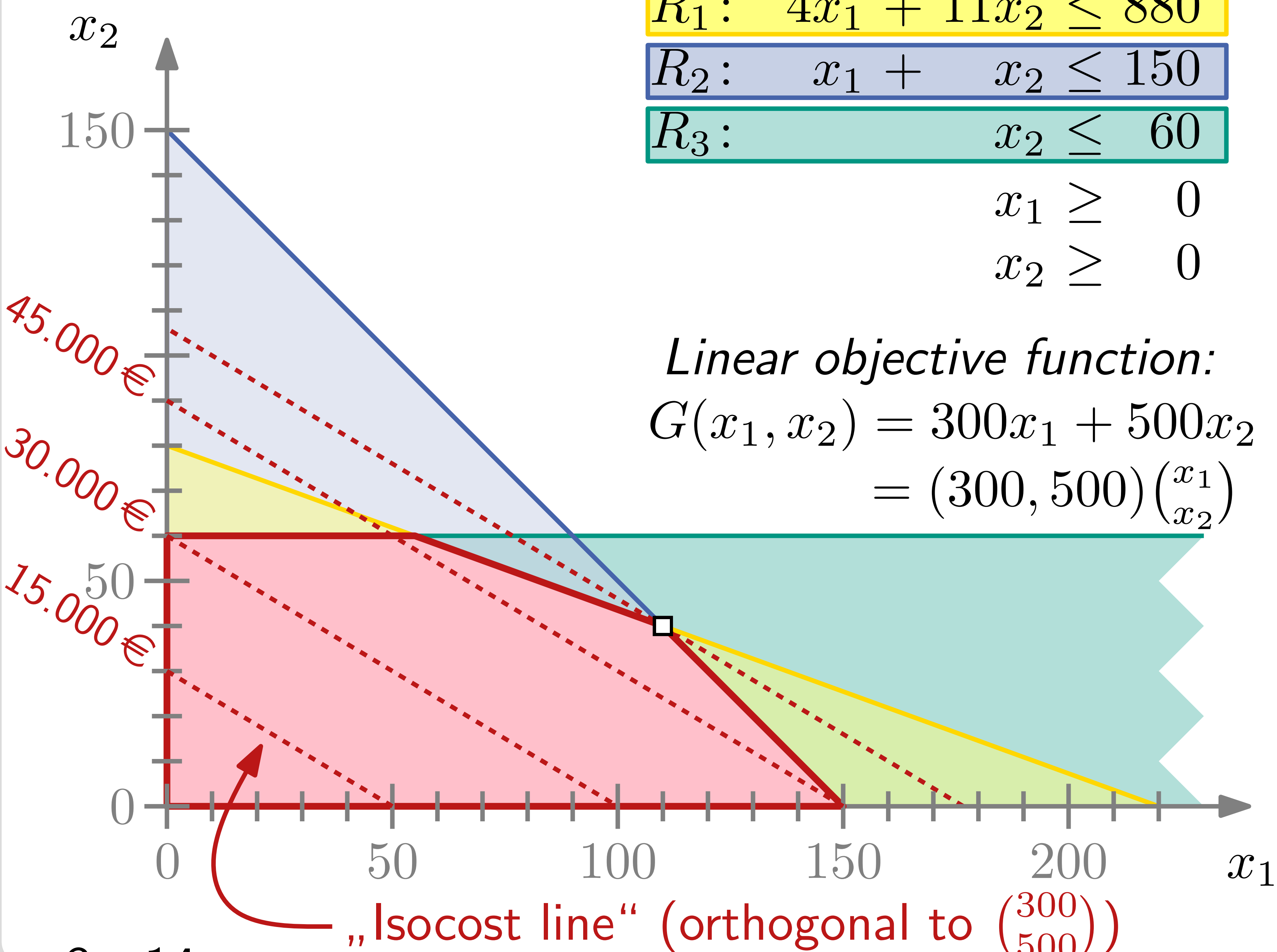
$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

$x_2 \geq 0$

Linear objective function:

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



Solution

Linear constraints:

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

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

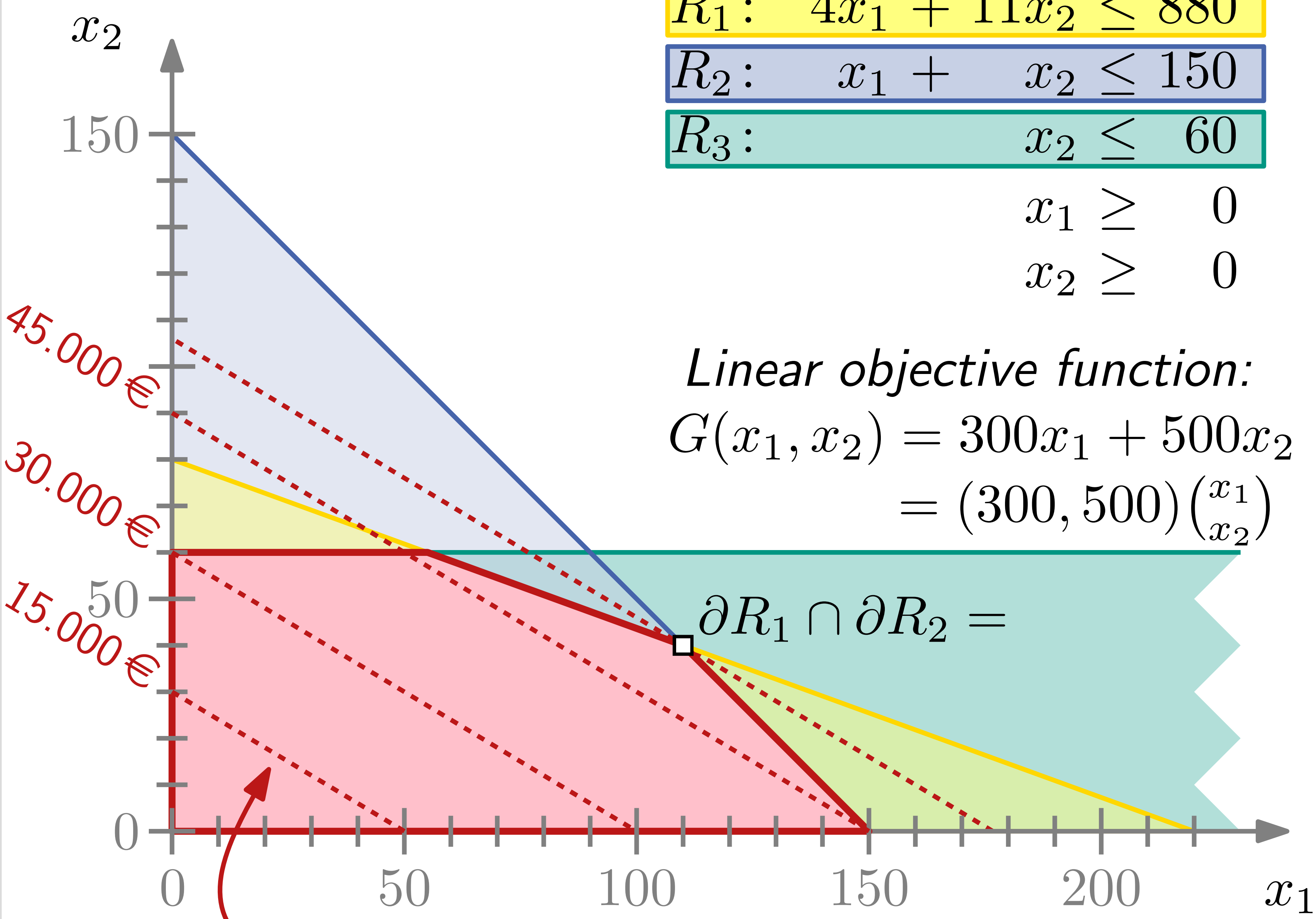
$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

$x_2 \geq 0$

Linear objective function:

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



„Isocost line“ (orthogonal to $\binom{300}{500}$)

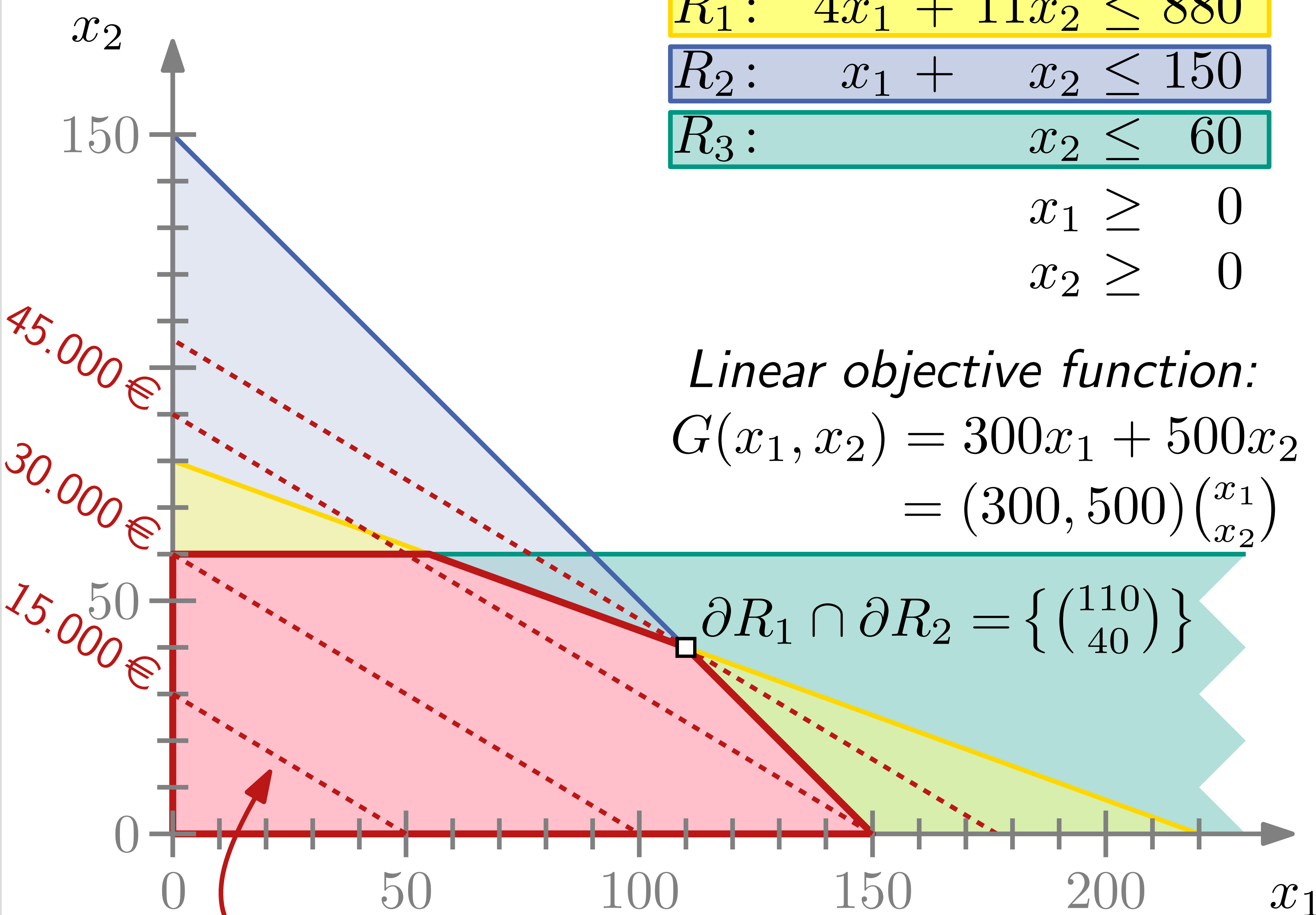
Solution

Linear constraints:

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

Linear objective function:

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



„Isocost line“ (orthogonal to $\binom{300}{500}$)

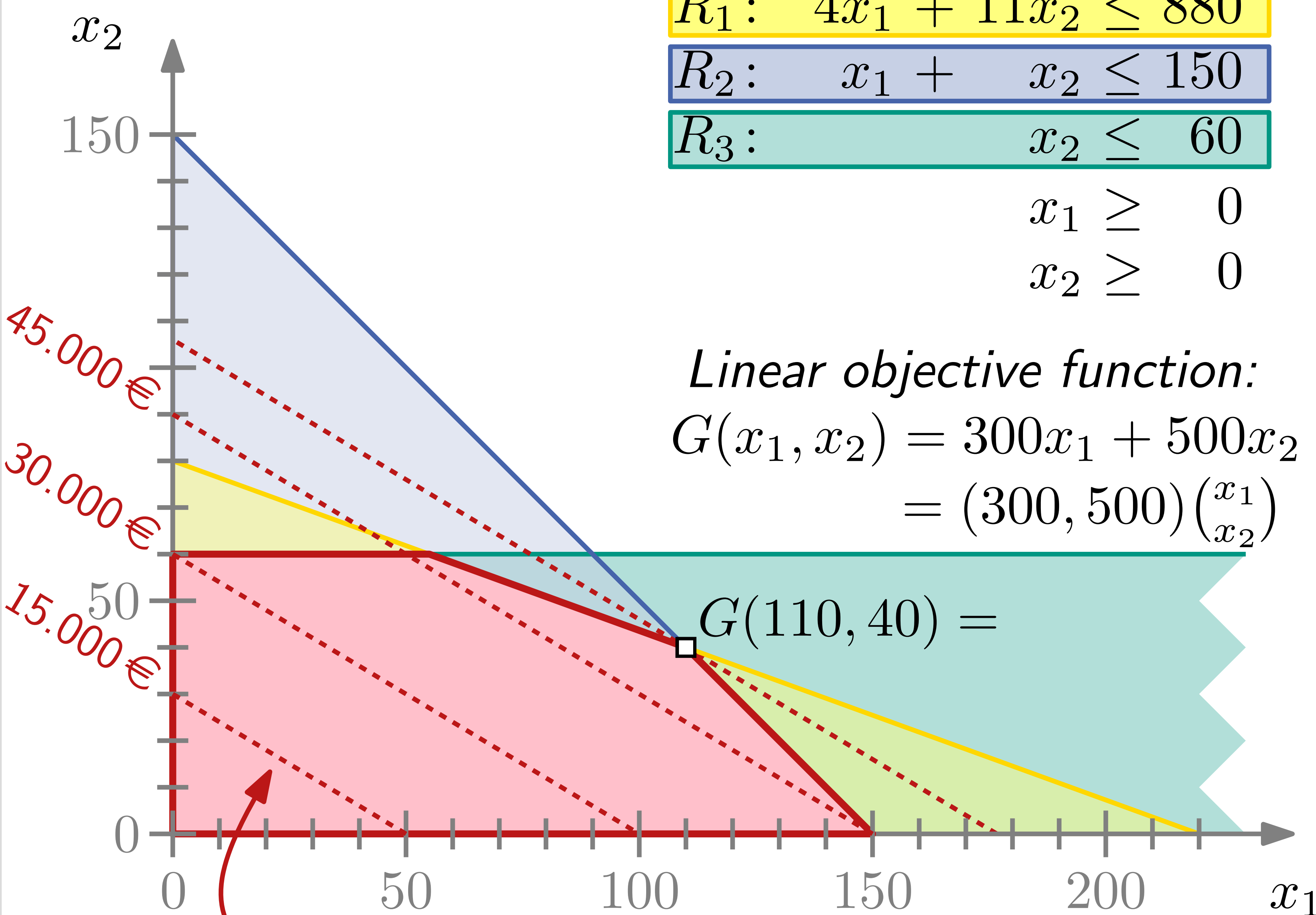
Solution

Linear constraints:

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

Linear objective function:

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



„Isocost line“ (orthogonal to $\binom{300}{500}$)

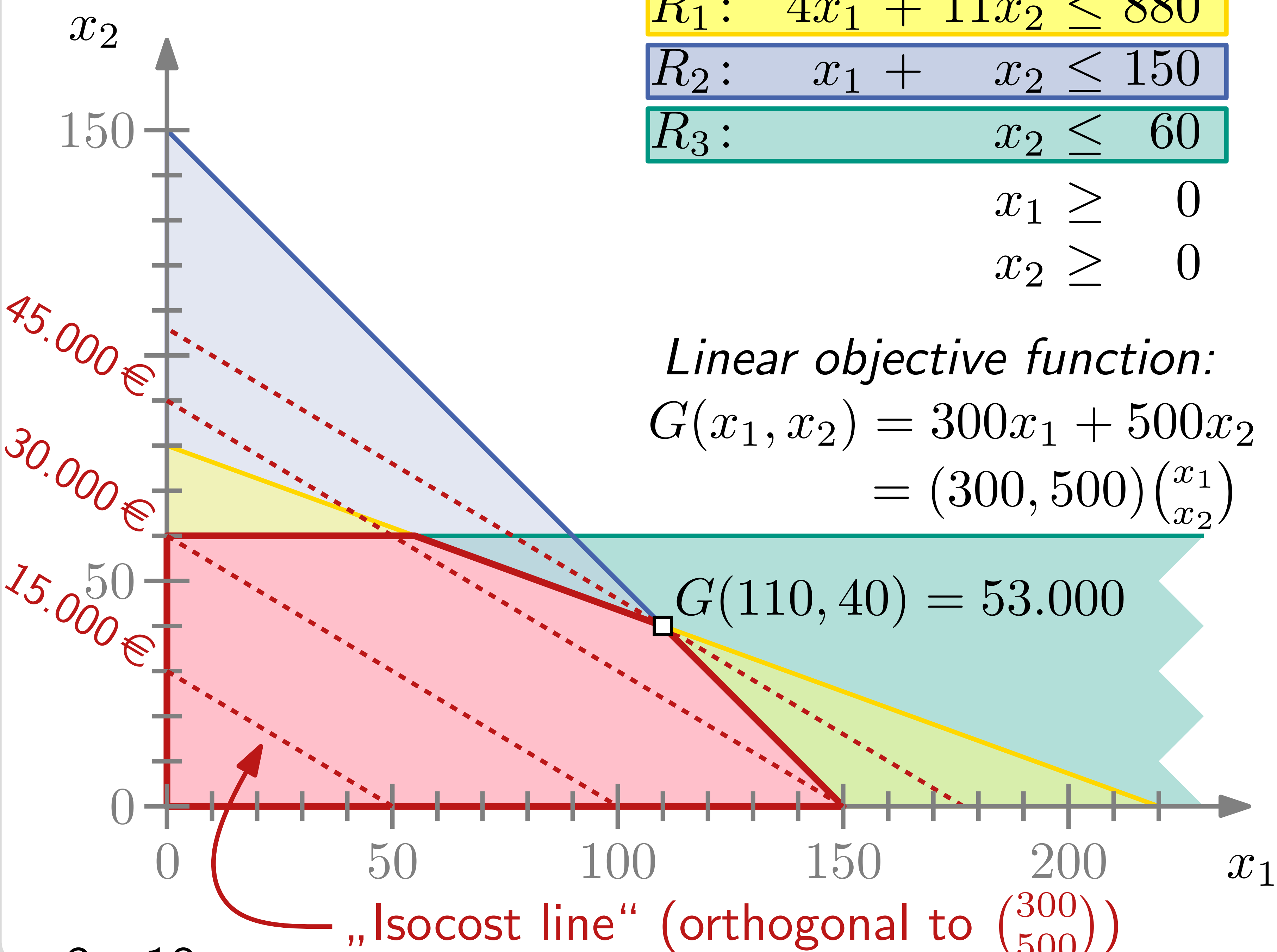
Solution

Linear constraints:

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

Linear objective function:

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



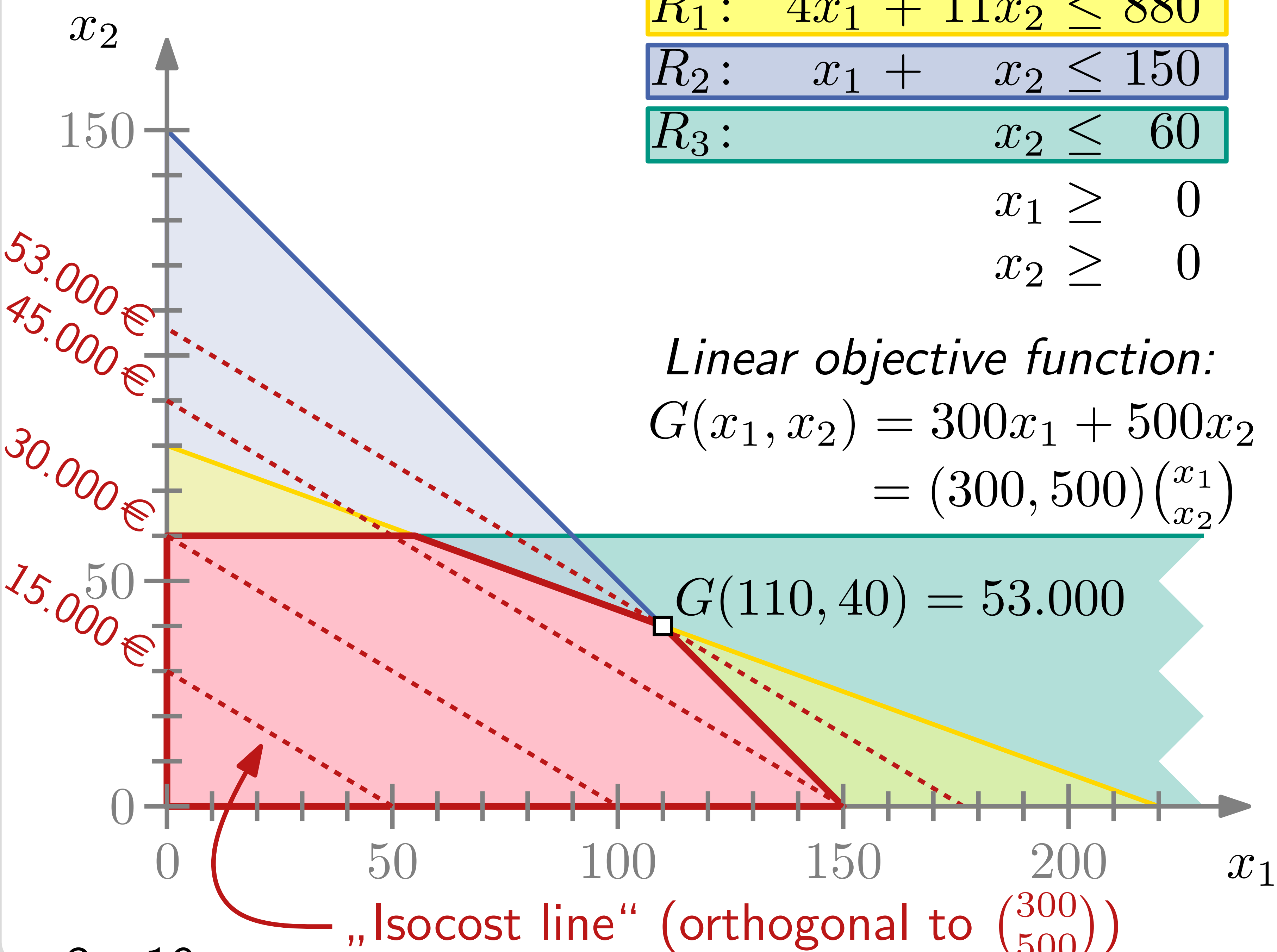
Solution

Linear constraints:

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

Linear objective function:

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



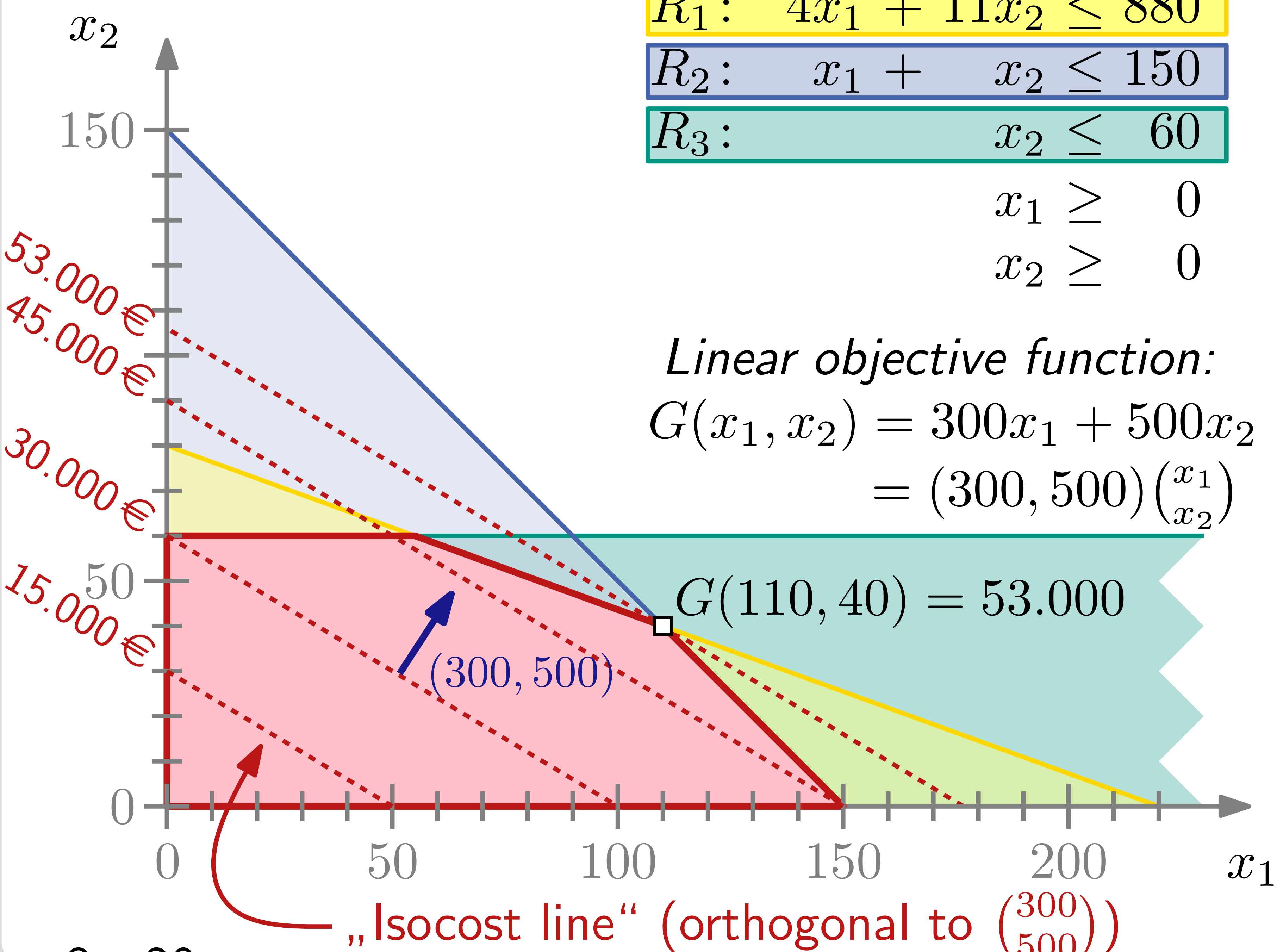
Solution

Linear constraints:

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

Linear objective function:

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



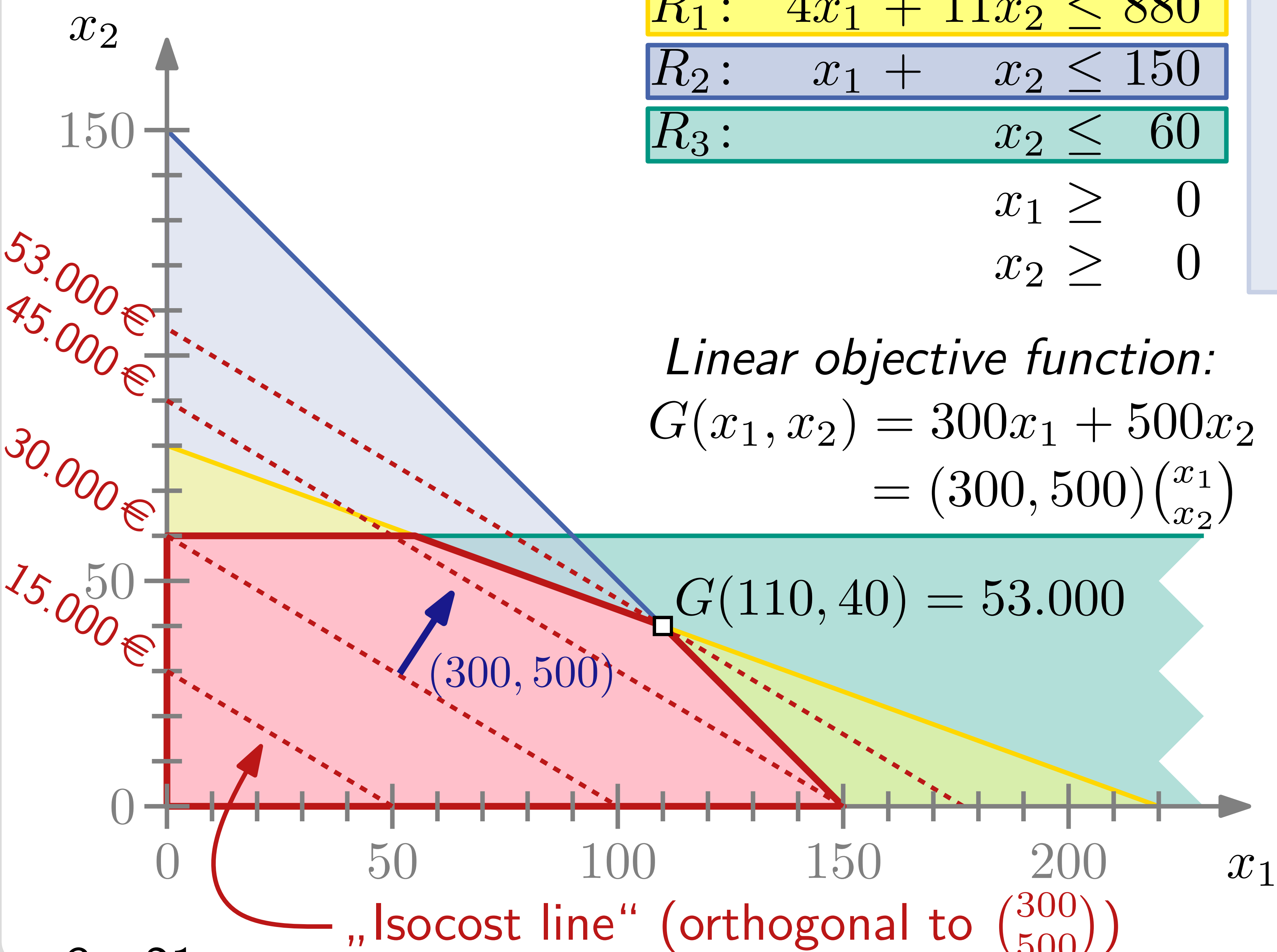
Solution

Linear constraints:

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

Linear objective function:

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



Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$

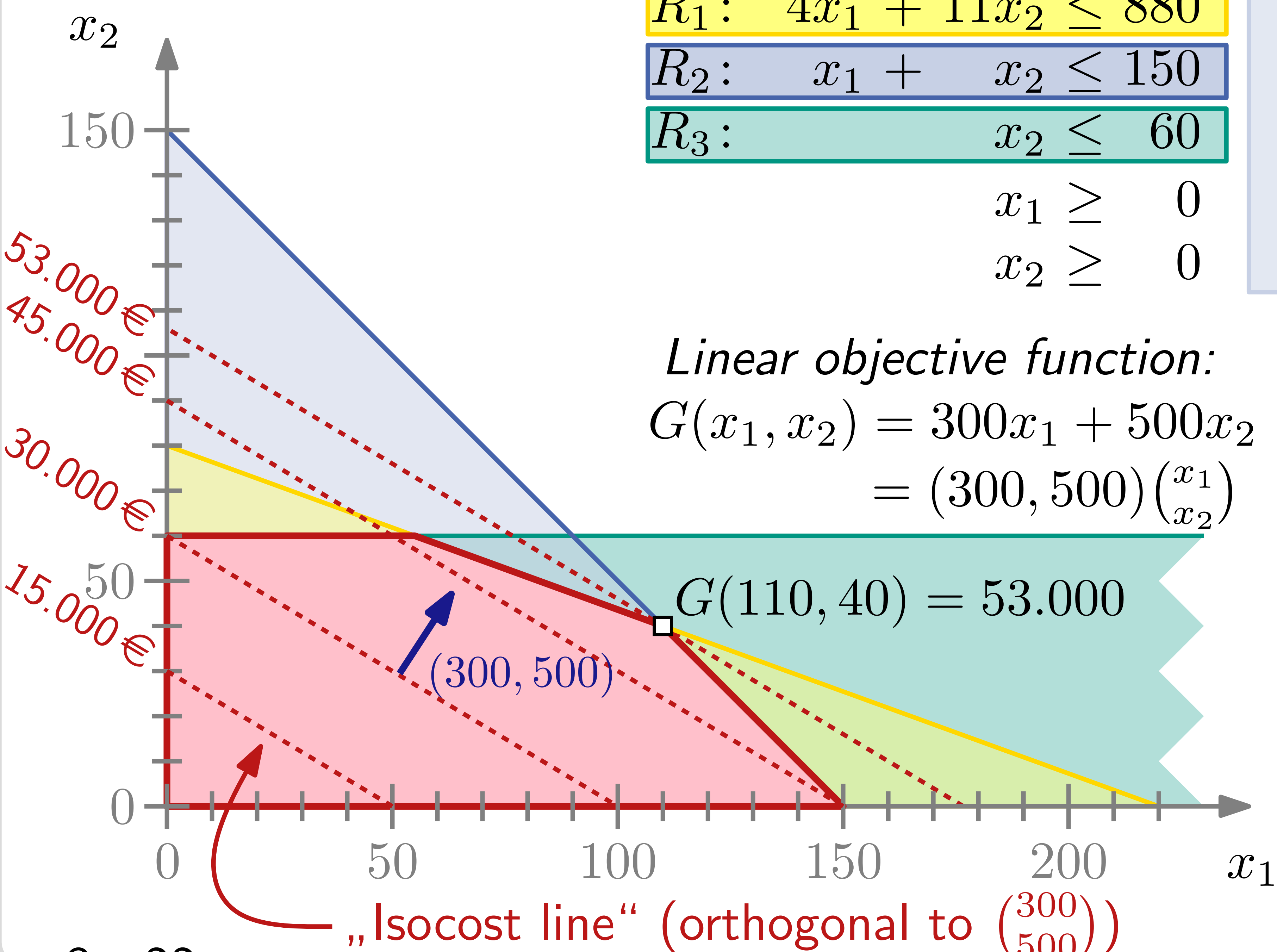
$x_1 \geq 0$

$x_2 \geq 0$

$Ax \leq b$

Linear objective function:

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



Solution

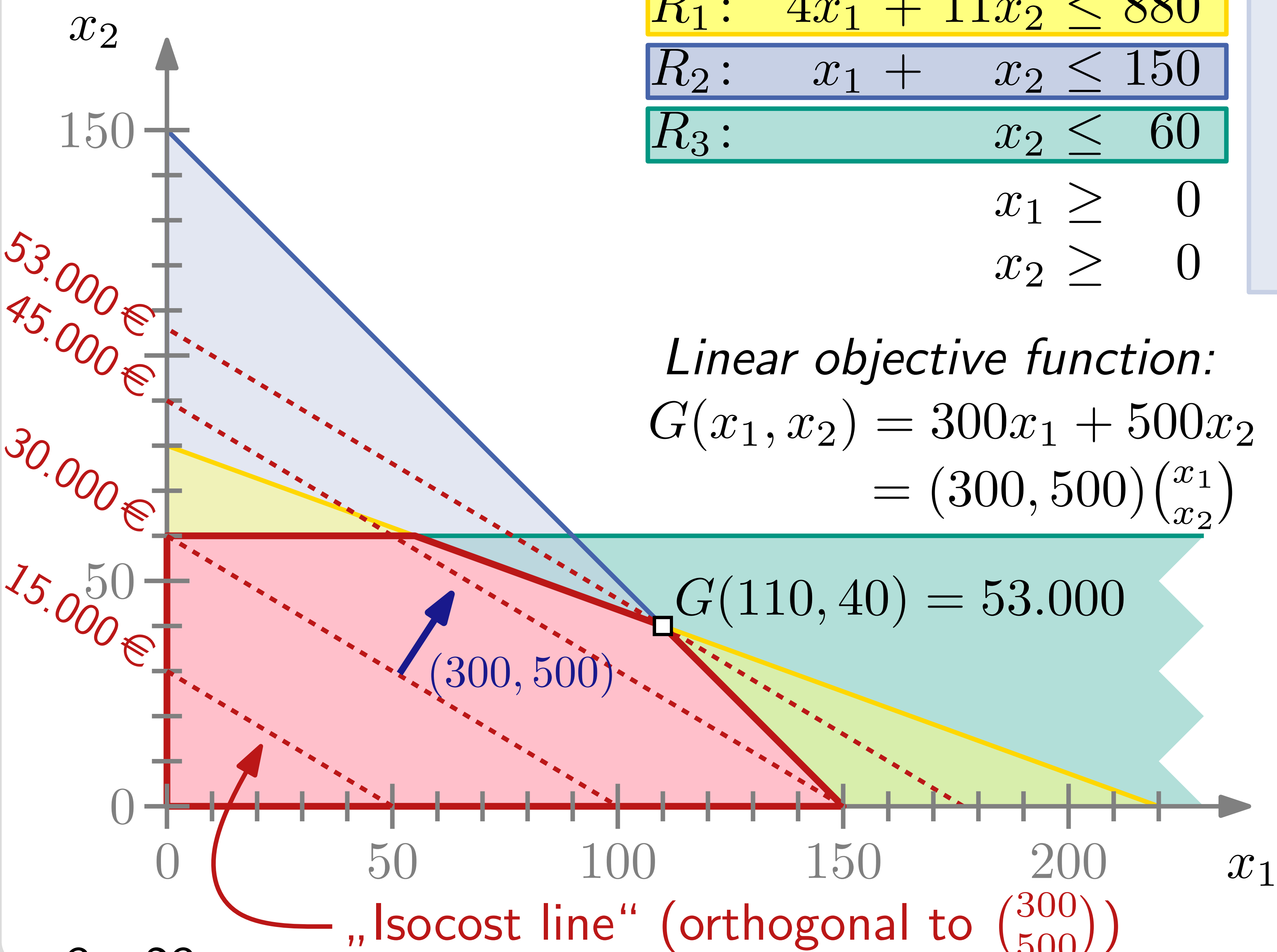
Linear constraints:

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

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

Linear objective function:

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



Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

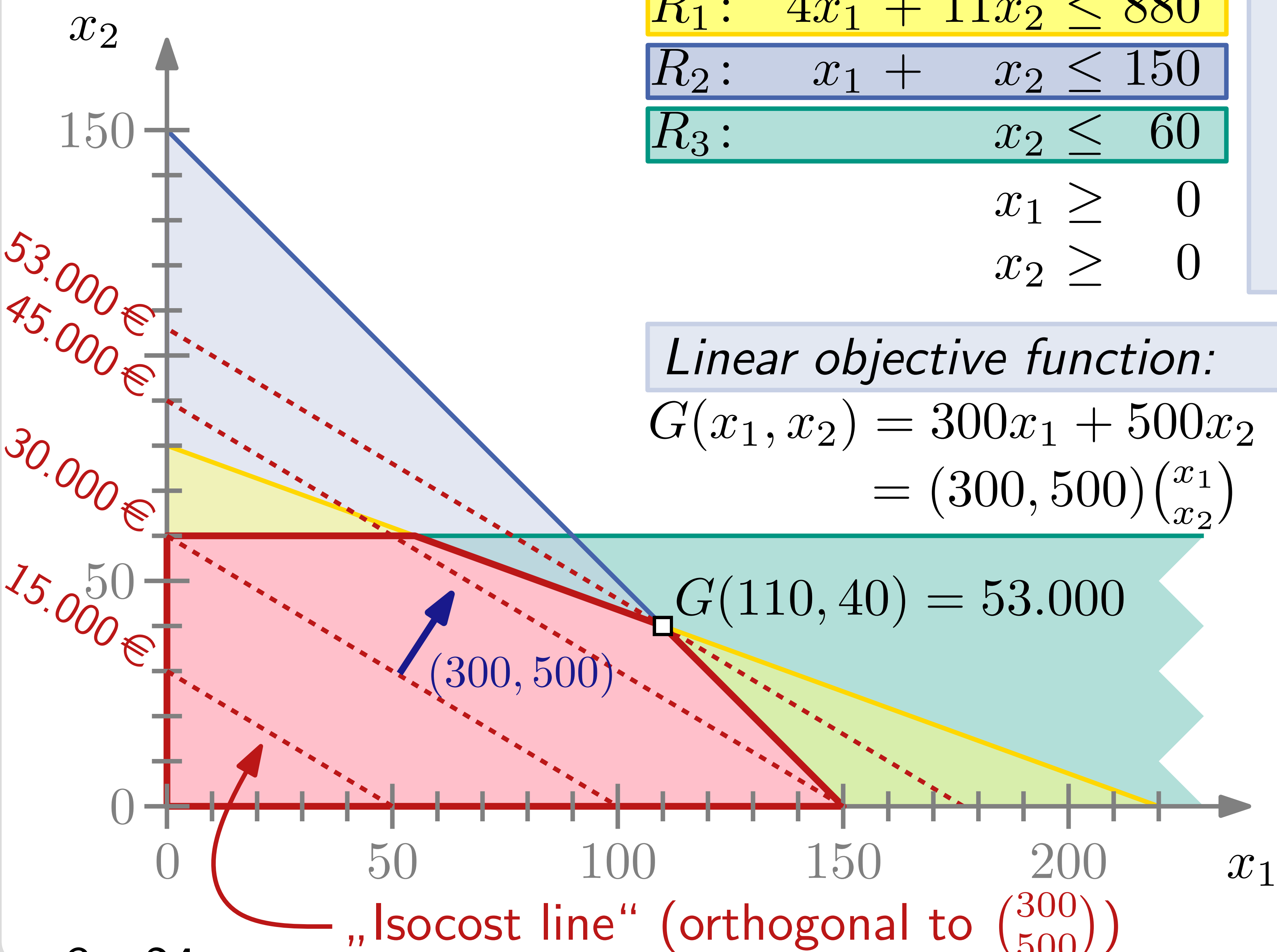
$x_2 \geq 0$

$Ax \leq b$

$x \geq 0$

Linear objective function:

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



Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

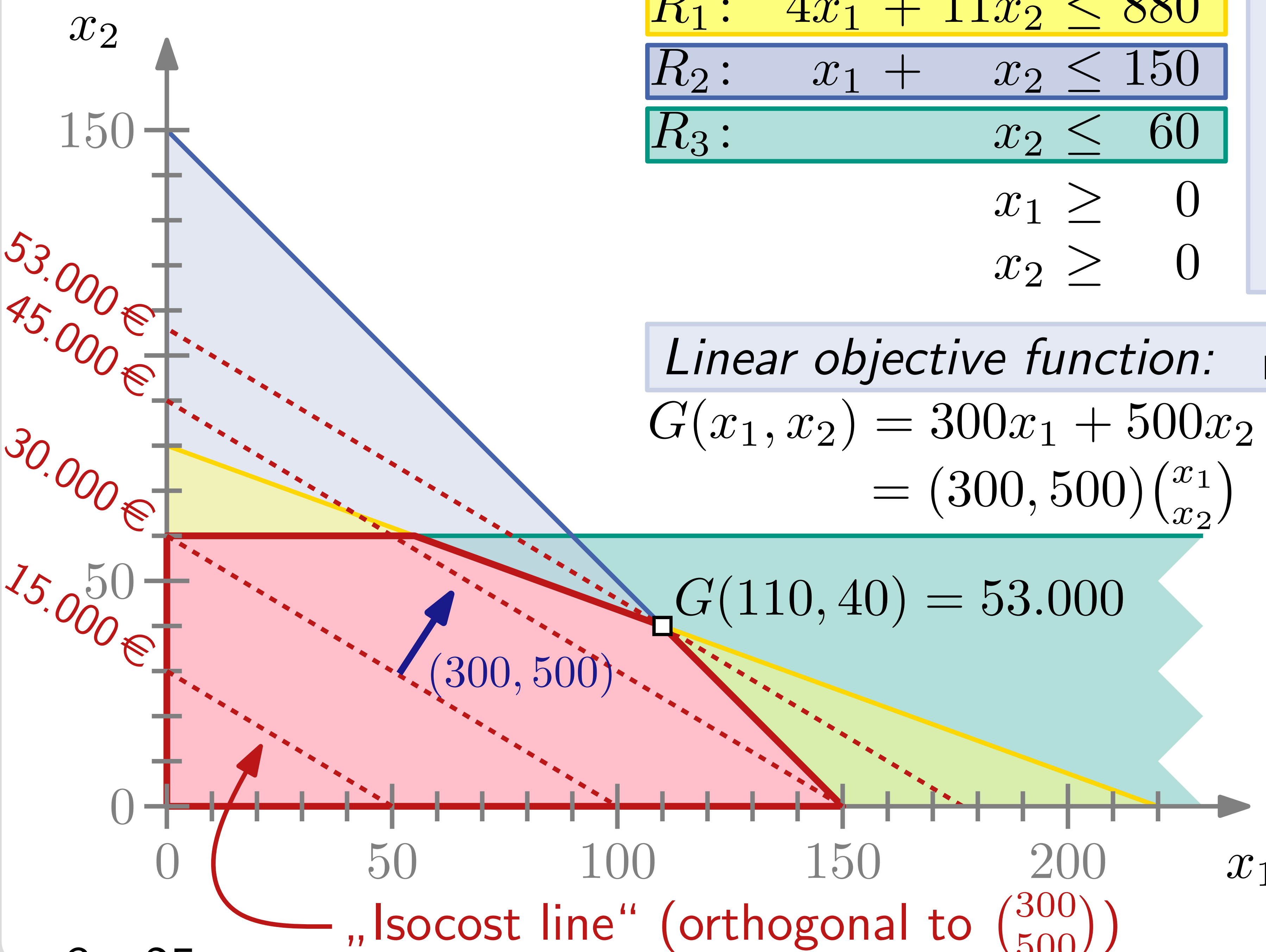
$x_2 \geq 0$

$Ax \leq b$

$x \geq 0$

Linear objective function: max $c^{\mathrm{T}}x$

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



Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

$x_2 \geq 0$

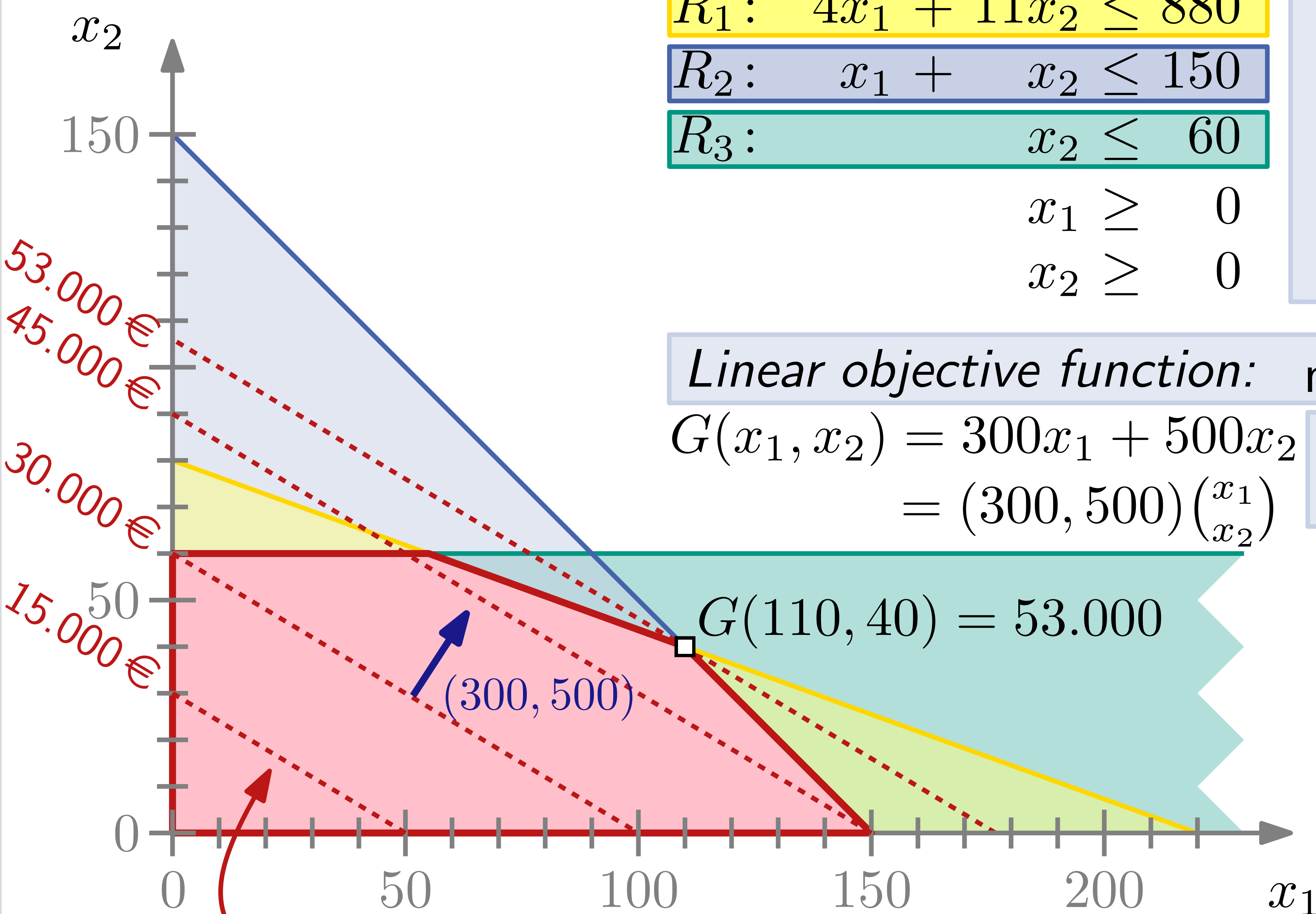
$Ax \leq b$

$x \geq 0$

Linear objective function: max $c^{\mathrm{T}} x$

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

c - *normal vector*



„Isocost line" (orthogonal to $\binom{300}{500}$)

Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

$x_2 \geq 0$

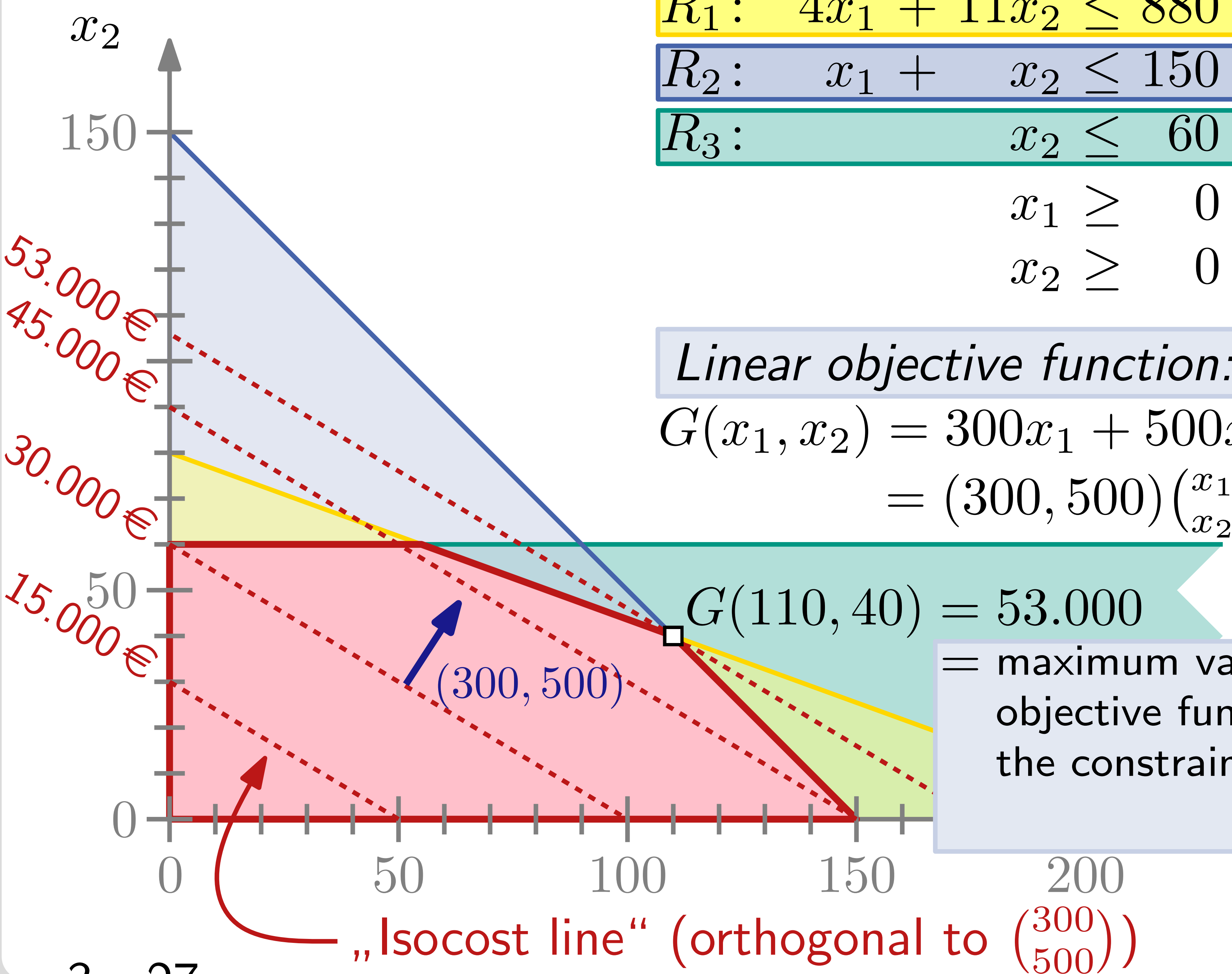
$Ax \leq b$

$x \geq 0$

Linear objective function: max $c^{\mathrm{T}}x$

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

c - *normal vector*



= maximum value of the objective function under the constraints.

Solution

Linear constraints:

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

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

$R_3: \quad x_2 \leq 60$

$x_1 \geq 0$

$x_2 \geq 0$

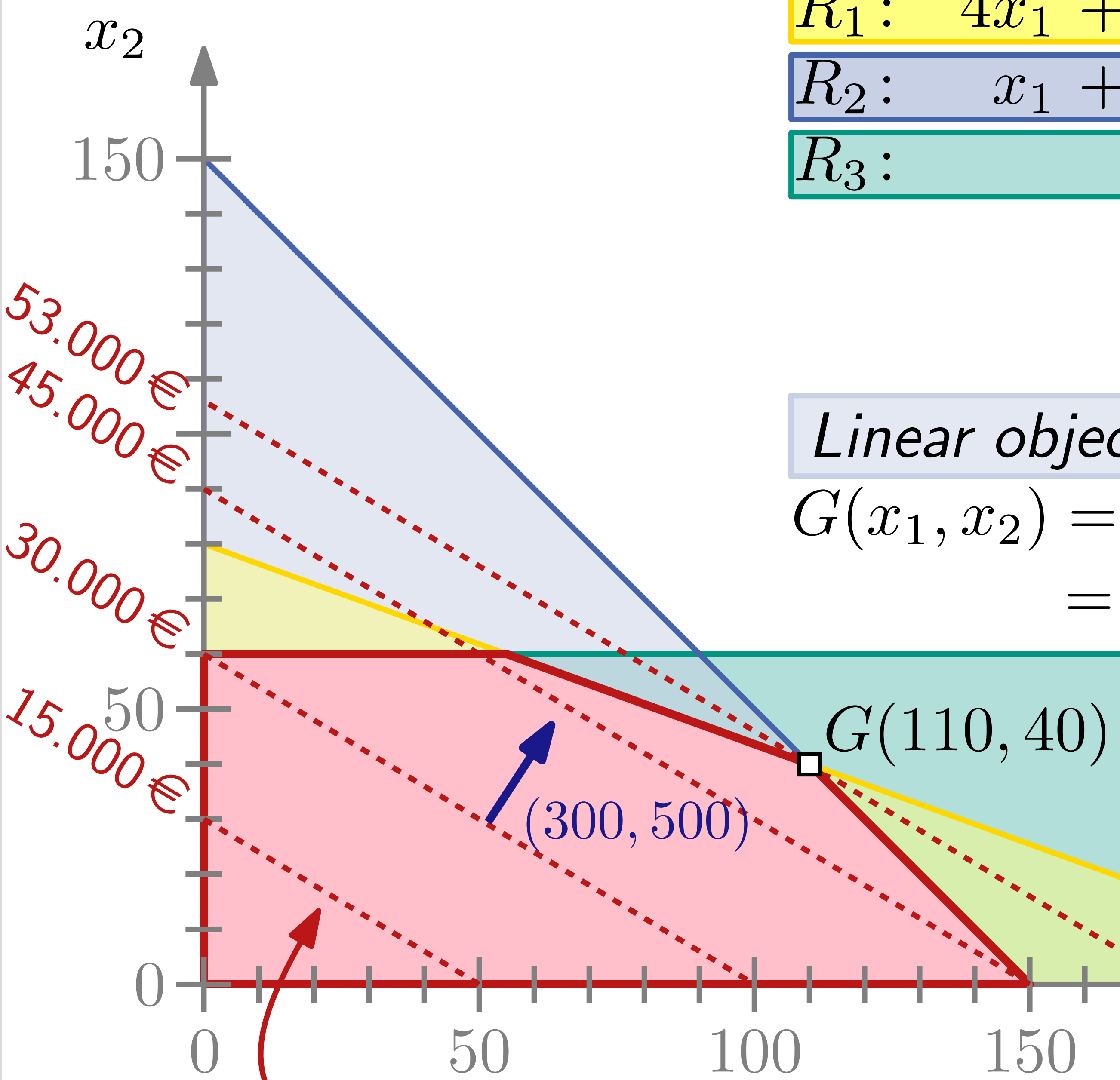
$Ax \leq b$

$x \geq 0$

Linear objective function: max $c^{\mathrm{T}}x$

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

c - *normal vector*



$G(110, 40) = 53.000$

= maximum value of the objective function under the constraints.

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

„Isocost line“ (orthogonal to $\binom{300}{500}$)

Linear programming

Definition: Given a set of linear constraints H and a linear objective function c in $\mathbb{R}^d$, a **linear program** (LP) is formulated as follows:

$$\begin{array}{lrcl} \text{maximize} & c_1x_1 + c_2x_2 + \cdots + c_dx_d & & \\ \text{under constr.} & a_{1,1}x_1 + \cdots + a_{1,d}x_d & \leq & b_1 \\ & a_{2,1}x_1 + \cdots + a_{2,d}x_d & \leq & b_2 \\ & & \vdots & \\ & a_{n,1}x_1 + \cdots + a_{n,d}x_d & \leq & b_n \end{array} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\end{array}}\right\} H$$

Linear programming

Definition: Given a set of linear constraints H and a linear objective function c in $\mathbb{R}^d$, a **linear program** (LP) is formulated as follows:

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

- H is a set of half-spaces in $\mathbb{R}^d$.
- We are searching for a point $x \in \bigcap_{h \in H} h$, that maximizes $c^T x$, i.e. $\max\{c^{\mathrm{T}}x \mid Ax \leq b, x \geq 0\}$.
- Linear programming is a central method in operations research.

Algorithms for LPs

There are many algorithms to solve LPs:

- Simplex-Algorithm [Dantzig, 1947]
- Ellipsoid-Method [Khatchiyan, 1979]
- Interior-Point-Method [Karmarkar, 1979]

They work well in practice, especially for large values of n (number of constraints) and d (number of variables).

Algorithms for LPs

There are many algorithms to solve LPs:

- Simplex-Algorithm [Dantzig, 1947]
- Ellipsoid-Method [Khatchiyan, 1979]
- Interior-Point-Method [Karmarkar, 1979]

They work well in practice, especially for large values of n (number of constraints) and d (number of variables).

Today: Special case $d = 2$

Algorithms for LPs

There are many algorithms to solve LPs:

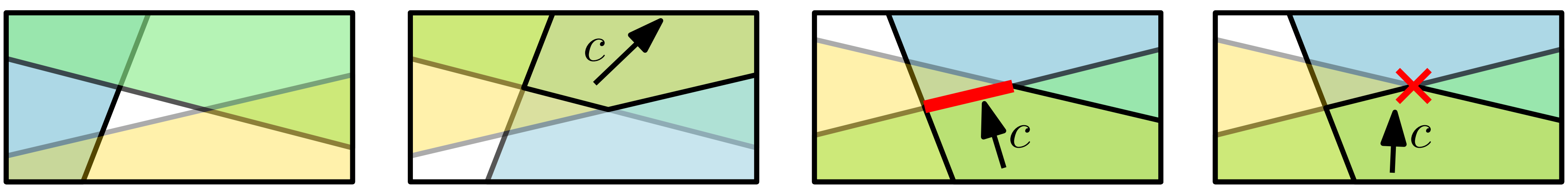
- Simplex-Algorithm [Dantzig, 1947]
- Ellipsoid-Method [Khatchiyan, 1979]
- Interior-Point-Method [Karmarkar, 1979]

They work well in practice, especially for large values of n (number of constraints) and d (number of variables).

Today: Special case $d = 2$

Possibilities for the solution space

feasible region $\bigcap H$ is bounded



$\bigcap H = \emptyset$ infeasible | $\bigcap H$ is unbounded in the direction c | solution is not unique | unique solution

First approach

Idea: Compute the feasible region $\bigcap H$ and search for the angle p, that maximizes $c^T p$.

First approach

Idea: Compute the feasible region $\bigcap H$ and search for the angle p, that maximizes $c^T p$.

How can we proceed?

First approach

Idea: Compute the feasible region $\bigcap H$ and search for the angle p, that maximizes $c^T p$.

- The half-planes are convex
- Let's try a simple Divide-and-Conquer Algorithm

First approach

Idea: Compute the feasible region $\bigcap H$ and search for the angle p, that maximizes $c^T p$.

- The half-planes are convex
- Let's try a simple Divide-and-Conquer Algorithm

```
IntersectHalfplanes(H)
  if |H| = 1 then
  |  C ← H
  else
  |  (H_1, H_2) ← SplitInHalves(H)
  |  C_1 ← IntersectHalfplanes(H_1)
  |  C_2 ← IntersectHalfplanes(H_2)
  |_ C ← IntersectConvexRegions(C_1, C_2)
  return C
```

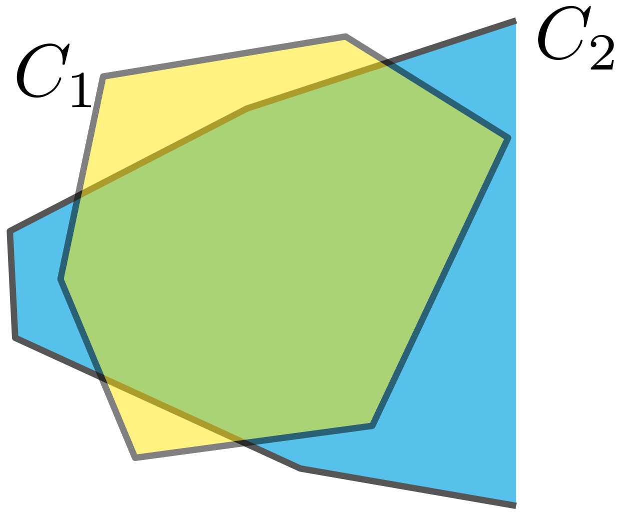
Intersect convex regions

IntersectConvexRegions(C_1, C_2) can be implemented using a sweep line method:

Intersect convex regions

IntersectConvexRegions(C_1, C_2) can be implemented using a sweep line method:

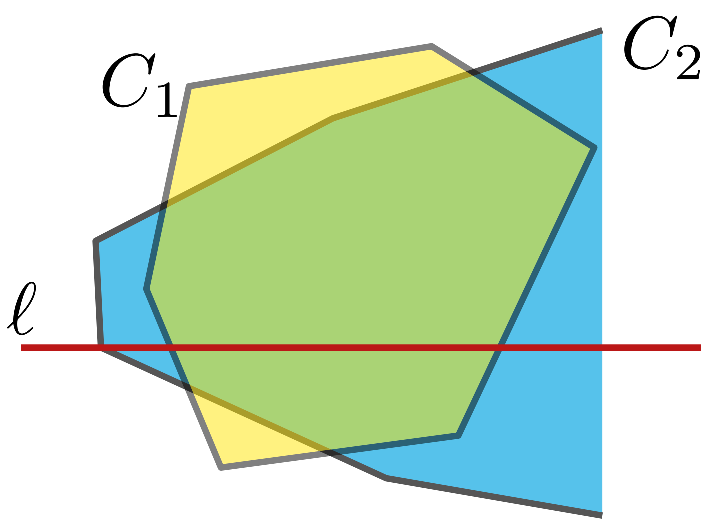
- consider the left and the right boundaries of C_1 and C_2



Intersect convex regions

IntersectConvexRegions(C_1, C_2) can be implemented using a sweep line method:

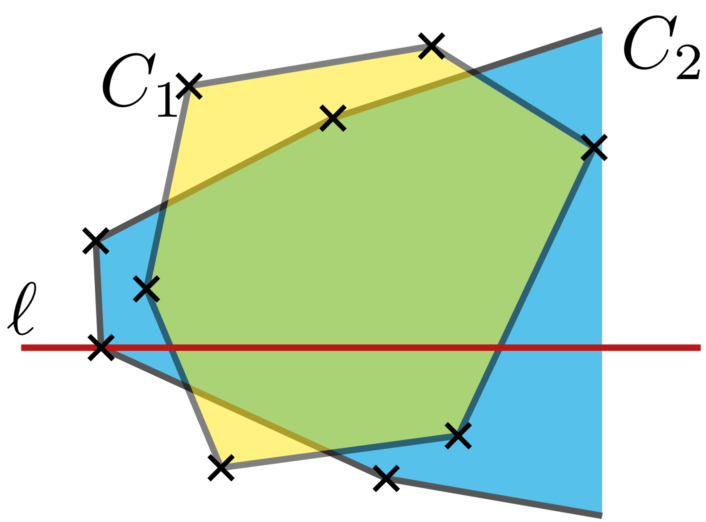
- consider the left and the right boundaries of C_1 and C_2
- move the sweep line ℓ from top to bottom and save the crossed edges (≤ 4)



Intersect convex regions

IntersectConvexRegions(C_1, C_2) can be implemented using a sweep line method:

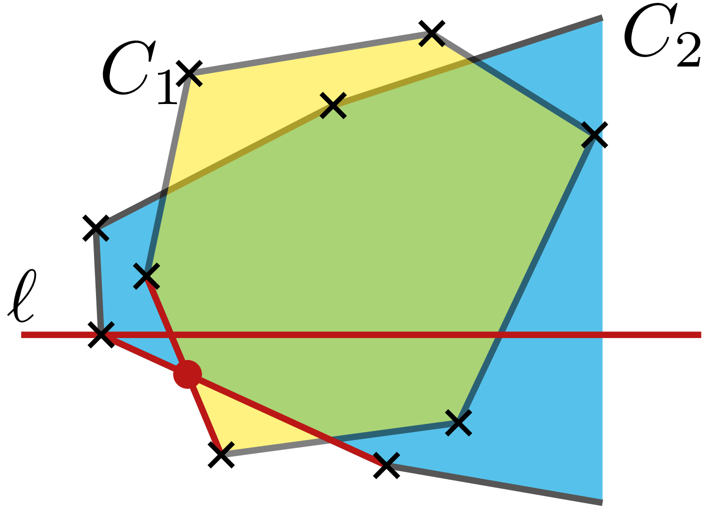
- consider the left and the right boundaries of C_1 and C_2
- move the sweep line ℓ from top to bottom and save the crossed edges (≤ 4)
- The nodes of $C_1 \cup C_2$ define events. We process every event in $O(1)$ time, dependent on the type of the edges incident to the event vertex.



Intersect convex regions

IntersectConvexRegions(C_1, C_2) can be implemented using a sweep line method:

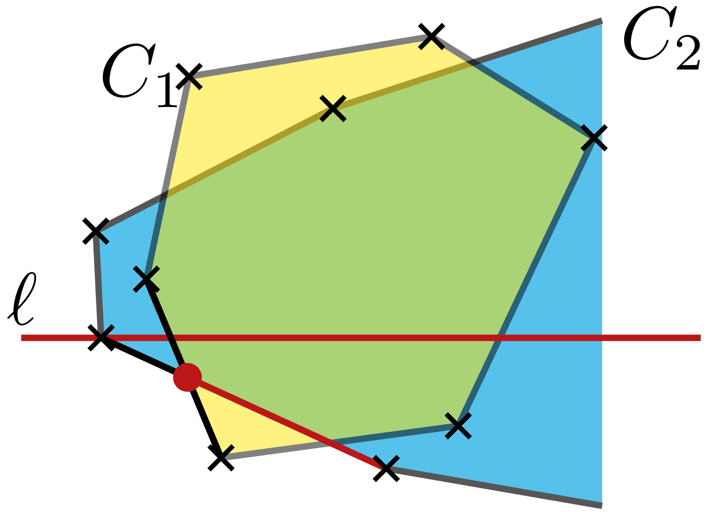
- consider the left and the right boundaries of C_1 and C_2
- move the sweep line ℓ from top to bottom and save the crossed edges (≤ 4)
- The nodes of $C_1 \cup C_2$ define events. We process every event in $O(1)$ time, dependent on the type of the edges incident to the event vertex.



Intersect convex regions

IntersectConvexRegions(C_1, C_2) can be implemented using a sweep line method:

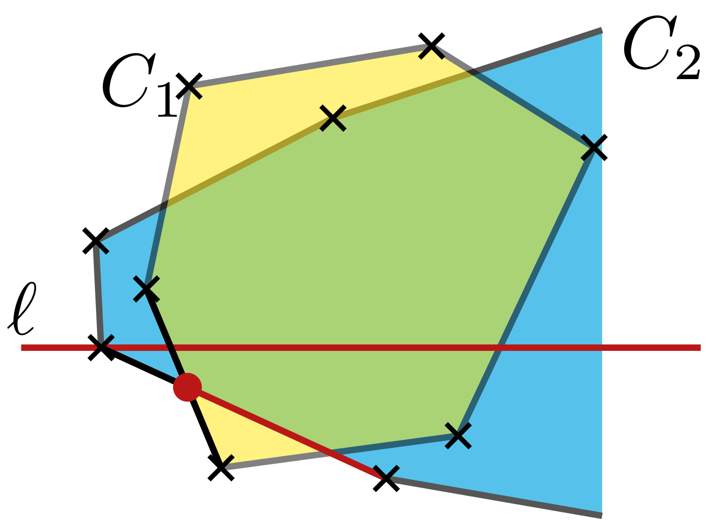
- consider the left and the right boundaries of C_1 and C_2
- move the sweep line ℓ from top to bottom and save the crossed edges (≤ 4)
- The nodes of $C_1 \cup C_2$ define events. We process every event in $O(1)$ time, dependent on the type of the edges incident to the event vertex.



Intersect convex regions

IntersectConvexRegions(C_1, C_2) can be implemented using a sweep line method:

- consider the left and the right boundaries of C_1 and C_2
- move the sweep line ℓ from top to bottom and save the crossed edges (≤ 4)
- The nodes of $C_1 \cup C_2$ define events. We process every event in $O(1)$ time, dependent on the type of the edges incident to the event vertex.



Theorem 1:
The intersection of two convex polygonal regions in the plane with $n_1 + n_2 = n$ nodes can be computed in $O(n)$ time.

Running time of IntersectHalfplanes(H)

IntersectHalfplanes(H)

```
if |H| = 1 then
    C ← H
else
    (H_1, H_2) ← SplitInHalves(H)
    C_1 ← IntersectHalfplanes(H_1)
    C_2 ← IntersectHalfplanes(H_2)
    C ← IntersectConvexRegions(C_1, C_2)
return C
```

Task: What is the running time of IntersectHalfplanes(H)?

Running time of 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

Task: What is the running time of IntersectHalfplanes(H)?

Recursive formula

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

Running time of IntersectHalfplanes(H)

KIT Karlsruhe Institute of Technology

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

Task: What is the running time of IntersectHalfplanes(H)?

Recursive formula

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

Master Theorem $\Rightarrow$
$T(n) \in O(n \log n)$

Running time of IntersectHalfplanes(H)

IntersectHalfplanes(H)

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

- feasible region $\bigcap H$ can be found in $O(n \log n)$ time
- $\bigcap H$ has $O(n)$ nodes
- the node p that maximizes $c^T p$ can therefore be found in $O(n \log n)$ time

Task: What is the running time of IntersectHalfplanes(H)?

Recursive formula

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

Master Theorem $\Rightarrow$ $T(n) \in O(n \log n)$

Running time of IntersectHalfplanes(H)

IntersectHalfplanes(H)

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

- feasible region $\bigcap H$ can be found in $O(n \log n)$ time
- $\bigcap H$ has $O(n)$ nodes
- the node p that maximizes $c^T p$ can therefore be found in $O(n \log n)$ time

Can we do better?

Task: What is the running time of IntersectHalfplanes(H)?

Recursive formula

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

Master Theorem $\Rightarrow$
$T(n) \in O(n \log n)$

Bounded LPs

Idea: Instead of computing the feasible region and then searching for the optimal angle, do this incrementally.

Bounded LPs

Idea: Instead of computing the feasible region and then searching for the optimal angle, do this incrementally.

Invariant: Current best solution is a unique corner of the current feasible polygon

Bounded LPs

Idea: Instead of computing the feasible region and then searching for the optimal angle, do this incrementally.

Invariant: Current best solution is a unique corner of the current feasible polygon

How to deal with the unbounded feasible regions?

Bounded LPs

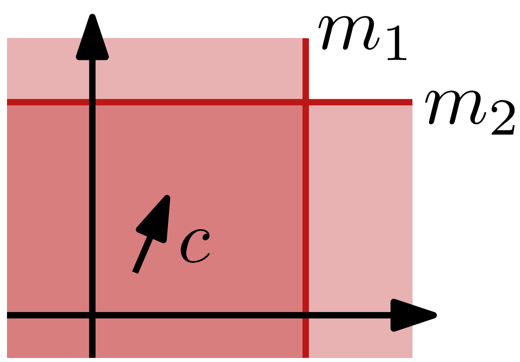
Idea: Instead of computing the feasible region and then searching for the optimal angle, do this incrementally.

Invariant: Current best solution is a unique corner of the current feasible polygon

How to deal with the unbounded feasible regions?

Define two half-planes for a big enough value M

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



Bounded LPs

Idea: Instead of computing the feasible region and then searching for the optimal angle, do this incrementally.

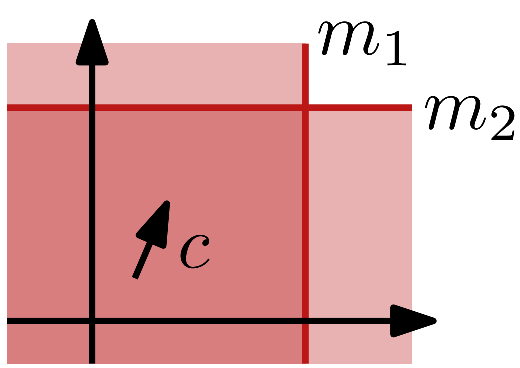
Invariant: Current best solution is a unique corner of the current feasible polygon

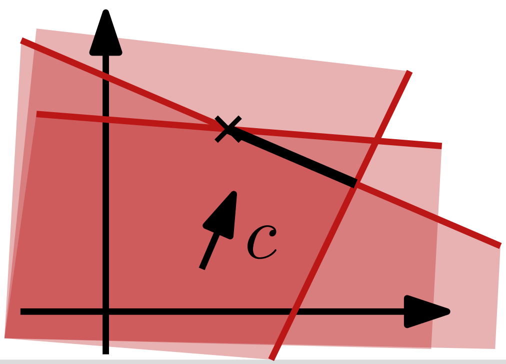
How to deal with the unbounded feasible regions?

When the optimal point is not unique, select lexicographically smallest one!

Define two half-planes for a big enough value M

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





Bounded LPs

Idea: Instead of computing the feasible region and then searching for the optimal angle, do this incrementally.

Invariant: Current best solution is a unique corner of the current feasible polygon

How to deal with the unbounded feasible regions?

When the optimal point is not unique, select lexicographically smallest one!

Define two half-planes for a big enough value M

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

Consider a LP (H, c) with $H = \{h_1, \ldots, h_n\}$, $c = (c_x, c_y)$. We denote the first i constraints by $H_i = \{m_1, m_2, h_1, \ldots, h_i\}$, and the feasible polygon defineed by them by $C_i = m_1 \cap m_2 \cap h_1 \cap \cdots \cap h_i$

Properties

- each region C_i has a single optimal angle v_i

Properties

- each region C_i has a single optimal angle v_i
- it holds that: $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

Properties

- each region C_i has a single optimal angle v_i
- it holds that: $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

How the optimal angle v_{i-1} changes when the half plane h_i is added?

Properties

- each region C_i has a single optimal angle v_i
- it holds that: $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

How the optimal angle v_{i-1} changes when the half plane h_i is added?

Lemma 1: For $1 \leq i \leq n$ and bounding line ℓ_i of h_i holds that:

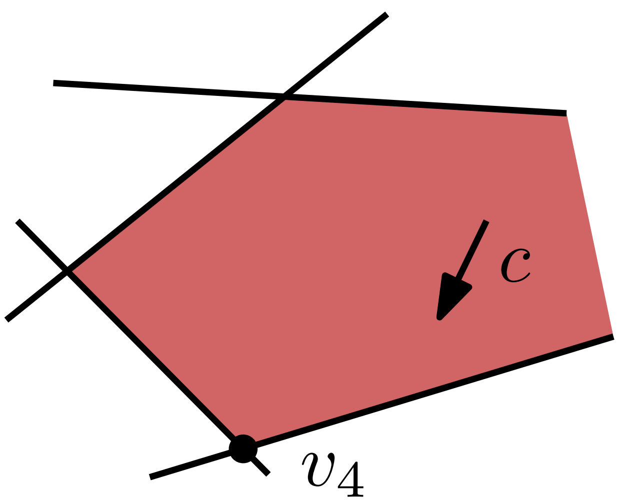
(i) If $v_{i-1} \in h_i$ then $v_i = v_{i-1}$,
(ii) otherwise, either $C_i = \emptyset$ or $v_i \in \ell_i$.

Properties

- each region C_i has a single optimal angle v_i
- it holds that: $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

How the optimal angle v_{i-1} changes when the half plane h_i is added?

Lemma 1: For $1 \leq i \leq n$ and bounding line ℓ_i of h_i holds that:
(i) If $v_{i-1} \in h_i$ then $v_i = v_{i-1}$,
(ii) otherwise, either $C_i = \emptyset$ or $v_i \in \ell_i$.

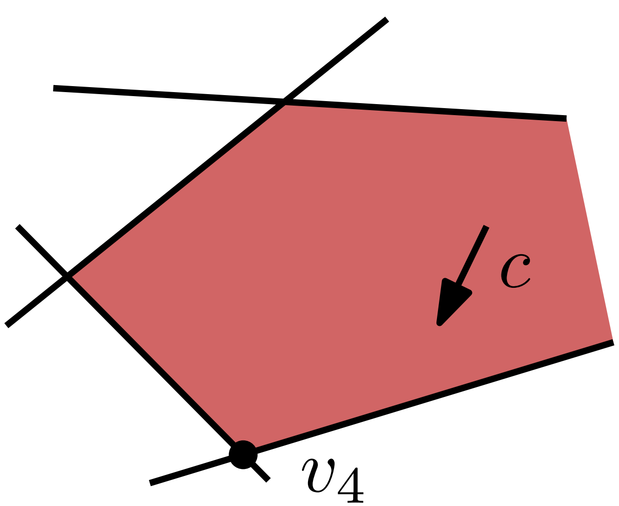


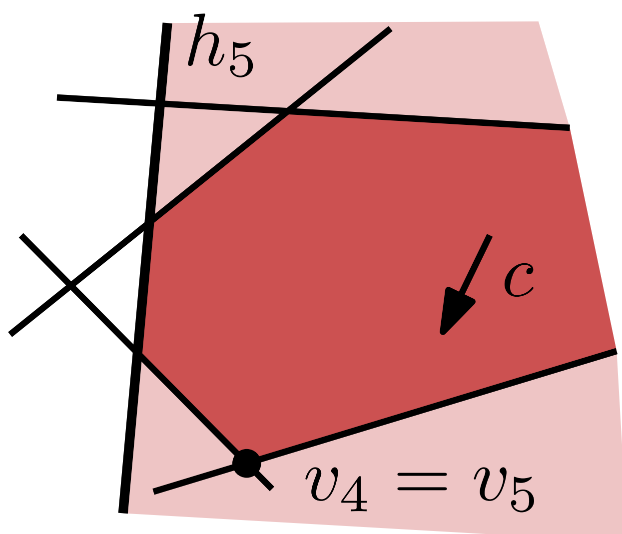
Properties

- each region C_i has a single optimal angle v_i
- it holds that: $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

How the optimal angle v_{i-1} changes when the half plane h_i is added?

Lemma 1: For $1 \leq i \leq n$ and bounding line ℓ_i of h_i holds that:
(i) If $v_{i-1} \in h_i$ then $v_i = v_{i-1}$,
(ii) otherwise, either $C_i = \emptyset$ or $v_i \in \ell_i$.



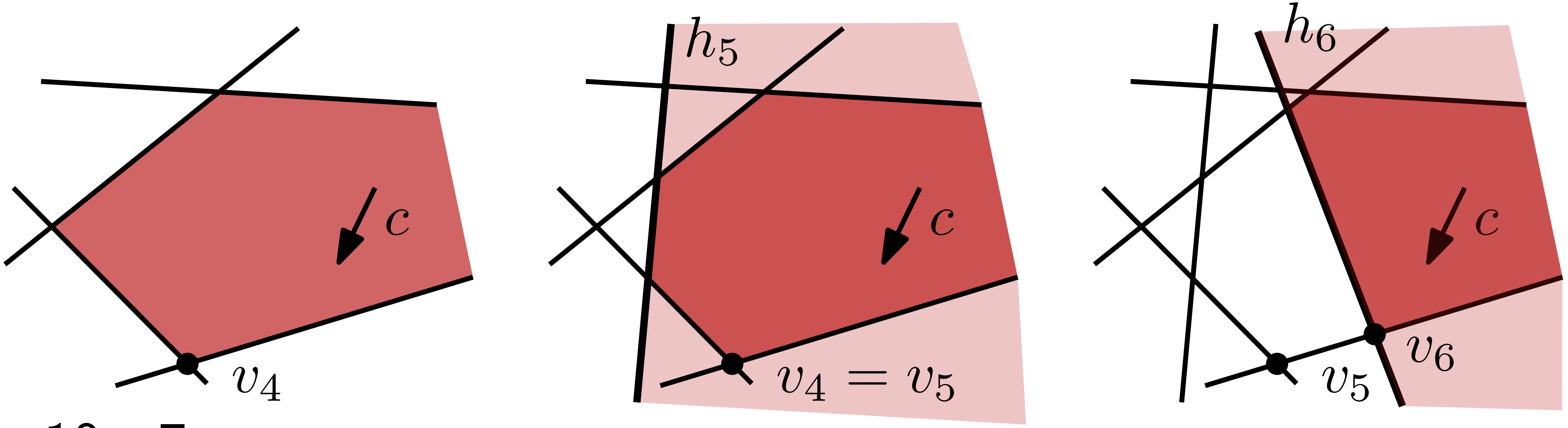


Properties

- each region C_i has a single optimal angle v_i
- it holds that: $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

How the optimal angle v_{i-1} changes when the half plane h_i is added?

Lemma 1: For $1 \leq i \leq n$ and bounding line ℓ_i of h_i holds that:
(i) If $v_{i-1} \in h_i$ then $v_i = v_{i-1}$,
(ii) otherwise, either $C_i = \emptyset$ or $v_i \in \ell_i$.



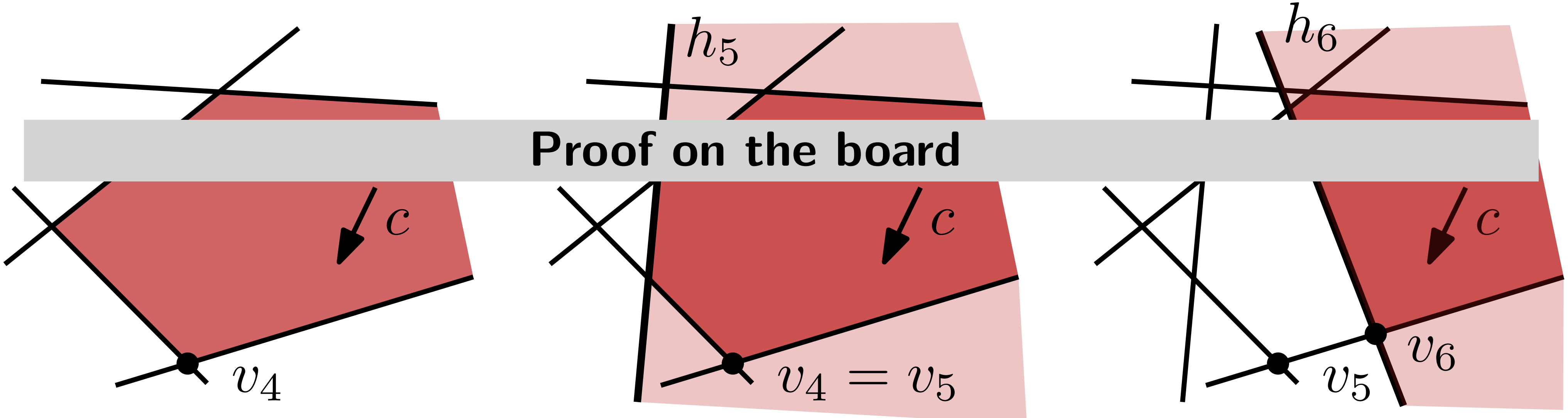
Properties

- each region C_i has a single optimal angle v_i
- it holds that: $C_0 \supseteq C_1 \supseteq \cdots \supseteq C_n = C$

How the optimal angle v_{i-1} changes when the half plane h_i is added?

Lemma 1: For $1 \leq i \leq n$ and bounding line ℓ_i of h_i holds that:
(i) If $v_{i-1} \in h_i$ then $v_i = v_{i-1}$,
(ii) otherwise, either $C_i = \emptyset$ or $v_i \in \ell_i$.

Proof on the board



One-dimentional LP

In case(ii) of Lemma 1, we search for the best point on the segment $\ell_i \cap C_{i-1}$:

- we parametrize $\ell_i : y = ax + b$

One-dimentional LP

In case(ii) of Lemma 1, we search for the best point on the segment $\ell_i \cap C_{i-1}$:

- we parametrize $\ell_i : y = ax + b$
- define new objective function $f_c^i(x) = c^T \binom{x}{ax+b}$

One-dimentional LP

In case(ii) of Lemma 1, we search for the best point on the segment $\ell_i \cap C_{i-1}$:

- we parametrize $\ell_i : y = ax + b$
- define new objective function $f_c^i(x) = c^T \binom{x}{ax+b}$
- for $j \leq i-1$ let $\sigma_x(\ell_j, \ell_i)$ denote the x-coordinate of $\ell_j \cap \ell_i$

One-dimentional LP

In case(ii) of Lemma 1, we search for the best point on the segment $\ell_i \cap C_{i-1}$:

- we parametrize $\ell_i : y = ax + b$
- define new objective function $f_c^i(x) = c^T \binom{x}{ax+b}$
- for $j \leq i-1$ let $\sigma_x(\ell_j, \ell_i)$ denote the x-coordinate of $\ell_j \cap \ell_i$

This gives us the following one-dimentional LP:

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

with constr. $x \leq \sigma_x(\ell_j, \ell_i)$ if $\ell_i \cap h_j$ is limited to the righ
$x \geq \sigma_x(\ell_j, \ell_i)$ if $\ell_i \cap h_j$ is limited to the lef

One-dimentional LP

In case(ii) of Lemma 1, we search for the best point on the segment $\ell_i \cap C_{i-1}$:

- we parametrize $\ell_i : y = ax + b$
- define new objective function $f_c^i(x) = c^T \binom{x}{ax+b}$
- for $j \leq i-1$ let $\sigma_x(\ell_j, \ell_i)$ denote the x-coordinate of $\ell_j \cap \ell_i$

This gives us the following one-dimentional LP:

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

$$\begin{array}{llll} \text{with constr.} & x \leq \sigma_x(\ell_j, \ell_i) & \text{if } \ell_i \cap h_j \text{ is limited to the righ} \\ & x \geq \sigma_x(\ell_j, \ell_i) & \text{if } \ell_i \cap h_j \text{ is limited to the lef} \end{array}$$

How to solve this LP? Running time?

One-dimentional LP

In case(ii) of Lemma 1, we search for the best point on the segment $\ell_i \cap C_{i-1}$:

- we parametrize $\ell_i : y = ax + b$
- define new objective function $f_c^i(x) = c^T \binom{x}{ax+b}$
- for $j \leq i-1$ let $\sigma_x(\ell_j, \ell_i)$ denote the x-coordinate of $\ell_j \cap \ell_i$

This gives us the following one-dimentional LP:

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

$$\begin{array}{llll} \text{with constr.} & x \leq \sigma_x(\ell_j, \ell_i) & \text{if } \ell_i \cap h_j \text{ is limited to the righ} \\ & x \geq \sigma_x(\ell_j, \ell_i) & \text{if } \ell_i \cap h_j \text{ is limited to the left} \end{array}$$

Lemma 2: A one-dimentional LP can be solved in linear time. In particular, in case (ii), one can compute the new angle v_i or decide whether $C_i = \emptyset$ in $O(i)$ time.

Incremental Algorithm

2dBoundedLP(H, c, m_1, m_2)

```
C_0 ← m_1 ∩ m_2
v_0 ← unique angle of C_0
for i ← 1 to n do
    if v_{i-1} ∈ h_i then
        v_i ← v_{i-1}
    else
        v_i ← 1dBoundedLP(σ(H_{i-1}), f_c^i)
        if v_i = nil then
            return infeasible
return v_n
```

Incremental Algorithm

2dBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ unique angle of 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 infeasible $O(i)$
return v_n

Incremental Algorithm

2dBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ unique angle of 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 infeasible — $O(i)$
return v_n

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

Incremental Algorithm

2dBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ unique angle of 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 infeasible
return v_n

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

Question: Can in reality the case(ii) happen n times?

Incremental Algorithm

2dBoundedLP(H, c, m_1, m_2)

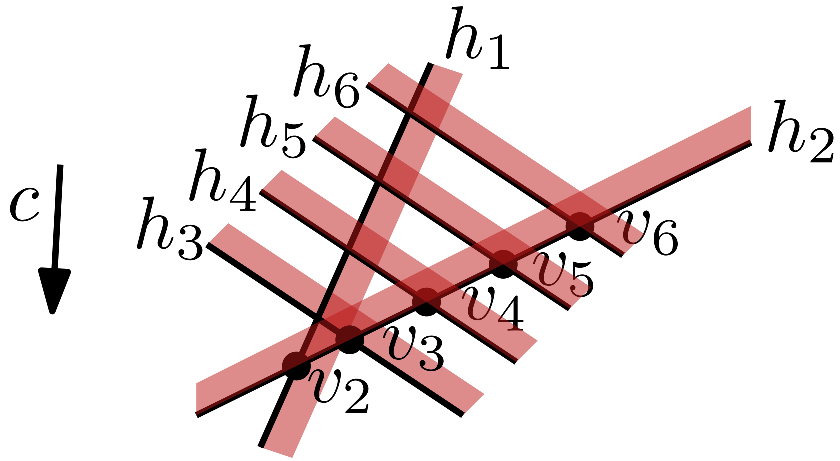
$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ unique angle of C_0
for $i \leftarrow 1$ **to** n **do**
  if $v_{i-1} \in h_i$ **then**
    $v_i \leftarrow v_{i-1}$ $O(1)$
  else
    $v_i \leftarrow$ 1dBoundedLP$(\sigma(H_{i-1}), f_c^i)$
    if $v_i =$ nil **then**
      return infeasible $O(i)$
return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

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

Lemma 3: Algorithmus 2dBoundedLP needs $\Theta(n^2)$ time to solve an LP with n contraints and 2 variables.

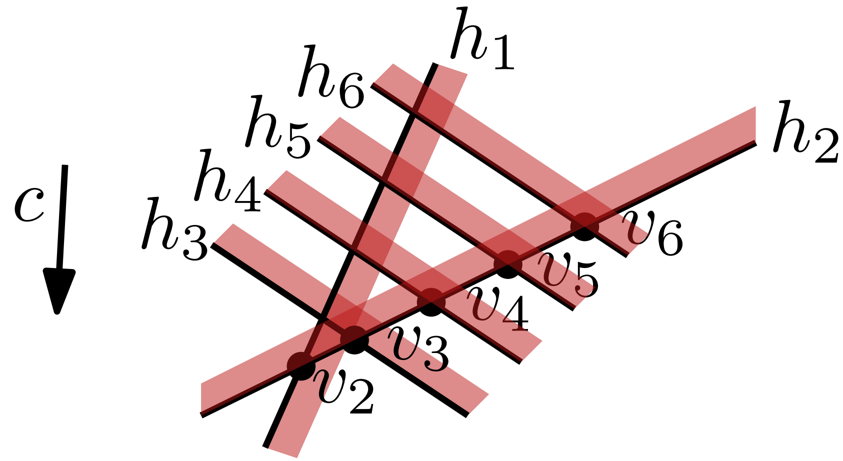
What else can we do?

Obs.: It is not the half-planes H that force the high running time, but the order in which we consider them.



What else can we do?

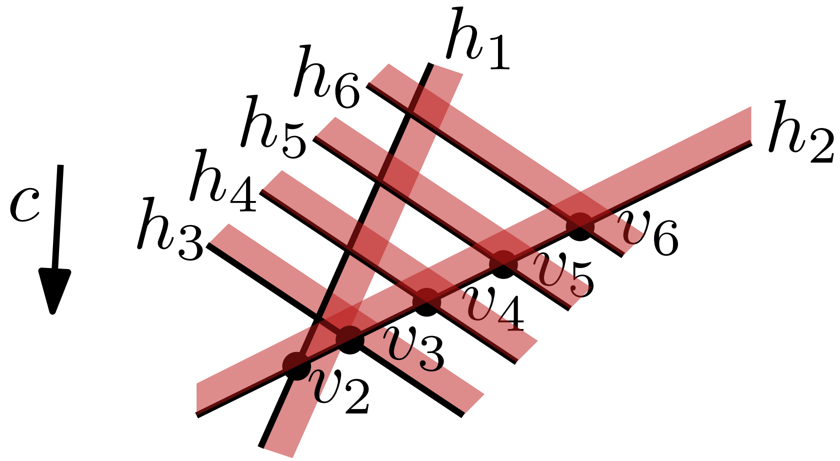
Obs.: It is not the half-planes H that force the high running time, but the order in which we consider them.



How to find (quickly) a good ordering?

What else can we do?

Obs.: It is not the half-planes H that force the high running time, but the order in which we consider them.



How to find (quickly) a good ordering?

Randomized incremental algorithm

2dRandomizedBoundedLP(H, c, m_1, m_2)

$C_0 \leftarrow m_1 \cap m_2$
$v_0 \leftarrow$ unique angle of 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 infeasible
return v_n

Random permutation

RandomPermutation(A)

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

Output: Array A, rearranged into a random permutation

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

  $r \leftarrow$ Random(k)

  exchange $A[r]$ and $A[k]$

Random permutation

RandomPermutation(A)

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

Output: Array A, rearranged into a random permutation

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

  $r \leftarrow$ Random(k) — Random num. between 1 and k

  exchange $A[r]$ and $A[k]$

Random permutation

RandomPermutation(A)

Input: Array $A[1 \ldots n]$
Output: Array A, rearranged into a random permutation
for $k \leftarrow n$ **to** 2 **do**
  $r \leftarrow$ Random(k)   Random num. between 1 and k
  exchange $A[r]$ and $A[k]$

Obs.: The running time of 2dRandomizedBoundedLP depends on the random permutation computed by the procedure RandomPermutation. In the following we compute the **expected running time**.

Random permutation

RandomPermutation(A)

Input: Array $A[1 \ldots n]$
Output: Array A, rearranged into a random permutation
for $k \leftarrow n$ **to** 2 **do**
  $r \leftarrow$ Random(k)   — Random num. between 1 and k
  exchange $A[r]$ and $A[k]$

Obs.: The running time of 2dRandomizedBoundedLP depends on the random permutation computed by the procedure RandomPermutation. In the following we compute the **expected running time**.

Theorem 2: A two-dimentional LP with n constraints can be solved in randomized expected time.

Random permutation

RandomPermutation(A)

Input: Array $A[1 \ldots n]$
Output: Array A, rearranged into a random permutation
for $k \leftarrow n$ **to** 2 **do**
  $r \leftarrow$ Random(k)
  exchange $A[r]$ and $A[k]$

Random num. between 1 and k

Obs.: The running time of 2dRandomizedBoundedLP depends on the random permutation computed by the procedure RandomPermutation. In the following we compute the **expected running time**.

Theorem 2: A two-dimentional LP with n constraints can be solved in $O(n)$ randomized expected time.

Random permutation

RandomPermutation(A)

Input: Array $A[1 \ldots n]$
Output: Array A, rearranged into a random permutation
for $k \leftarrow n$ **to** 2 **do**
  $r \leftarrow$ Random(k)
  exchange $A[r]$ and $A[k]$

Random num. between 1 and k

Obs.: The running time of 2dRandomizedBoundedLP depends on the random permutation computed by the procedure RandomPermutation. In the following we compute the **expected running time**.

Theorem 2: A two-dimentional LP with n constraints can be solved in $O(n)$ randomized expected time.

Proof on the board

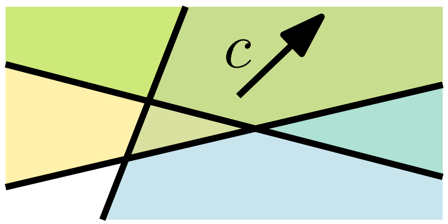
Unbounded LPs

Till now: Artificial contraints to bound C by m_1 and m_2

Unbounded LPs

Till now: Artificial contraints to bound C by m_1 and m_2

Next: recongnize and deal with an unbonded LP

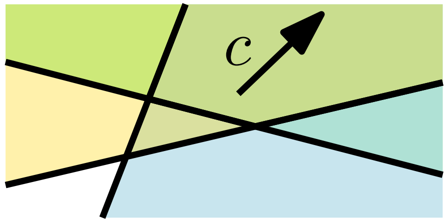


$\bigcap H$ unbounded in the direction c

Unbounded LPs

Till now: Artificial contraints to bound C by m_1 and m_2

Next: recongnize and deal with an unbonded LP



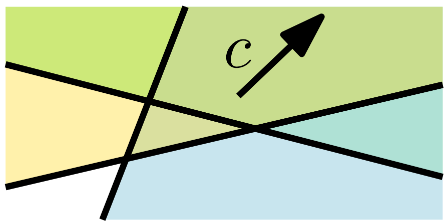
$\bigcap H$ unbounded in the direction c

Def.: A LP (H, c) is called **unbounded**, if there exists a ray $\rho = \{p + \lambda d \mid \lambda > 0\}$ in $C = \bigcap H$, such that the value of the objective function f_c becomes arbitrarily large along ρ.

Unbounded LPs

Till now: Artificial contraints to bound C by m_1 and m_2

Next: recongnize and deal with an unbonded LP



$\bigcap H$ unbounded in the direction c

Def.: A LP (H, c) is called **unbounded**, if there exists a ray $\rho = \{p + \lambda d \mid \lambda > 0\}$ in $C = \bigcap H$, such that the value of the objective function f_c becomes arbitrarily large along ρ.

It must be that:

- $\langle d, c \rangle > 0$
- $\langle d, \eta(h) \rangle \geq 0$ for all $h \in H$ where $\eta(h)$ is the **normal** vector of h oriented towards the feasible side of h

Characterization

Lemma 4: A LP (H, c) is unbounded iff there is a vector $d \in \mathbb{R}^2$ such that

- $\langle d, c \rangle > 0$
- $\langle d, \eta(h) \rangle \geq 0$ for all $h \in H$
- LP (H', c) with $H' = \{h \in H \mid \langle d, \eta(h) \rangle = 0\}$ is feasible.

Characterization

Lemma 4: A LP (H, c) is unbounded iff there is a vector $d \in \mathbb{R}^2$ such that

- $\langle d, c \rangle > 0$
- $\langle d, \eta(h) \rangle \geq 0$ for all $h \in H$
- LP (H', c) with $H' = \{h \in H \mid \langle d, \eta(h) \rangle = 0\}$ is feasible.

Test whether (H, c) is unbounded with a one-dimentional LP:

Step 1:

- rotate coordinate system till $c = (0, 1)$
- normalize vector d with $\langle d, c \rangle > 0$ as $d = (d_x, 1)$
- For normal vector $\eta(h) = (\eta_x, \eta_y)$ it should hold that $\langle d, \eta(h) \rangle = d_x \eta_x + \eta_y \geq 0$
- Let $\bar{H} = \{d_x \eta_x + \eta_y \geq 0 | h \in H\}$
- Check whether this one-dim. LP $\bar{H}$ is feasible

Test auf Unbeschränktheit

Step 2: If Step 1 returns a feasible solution $d_x^\star$

- compute $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- Normals to H' are orthogonal to $d = (d_x, 1) \Rightarrow$ lines bounding half-planes of H' are parallel to d
- intersect the bounding lines of H' with x-axis $\rightarrow$ 1d-LP

Test auf Unbeschränktheit

Step 2: If Step 1 returns a feasible solution $d_x^\star$

- compute $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- Normals to H' are orthogonal to $d = (d_x, 1) \Rightarrow$ lines bounding half-planes of H' are parallel to d
- intersect the bounding lines of H' with x-axis $\rightarrow$ 1d-LP

If the two steps result in a feasible solution, the LP (H, c) is unbounded and we can construct the ray ρ.

Test auf Unbeschränktheit

Step 2: If Step 1 returns a feasible solution $d_x^\star$

- compute $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- Normals to H' are orthogonal to $d = (d_x, 1) \Rightarrow$ lines bounding half-planes of H' are parallel to d
- intersect the bounding lines of H' with x-axis $\to$ 1d-LP

If the two steps result in a feasible solution, the LP (H, c) is unbounded and we can construct the ray ρ.

If the LP H' in Step 2 is infeasible, then then so is the initial LP (H, c).

Test auf Unbeschränktheit

Step 2: If Step 1 returns a feasible solution $d_x^\star$

- compute $H' = \{h \in H \mid d_x^\star \eta_x(h) + \eta_y(h) = 0\}$
- Normals to H' are orthogonal to $d = (d_x, 1) \Rightarrow$ lines bounding half-planes of H' are parallel to d
- intersect the bounding lines of H' with x-axis $\rightarrow$ 1d-LP

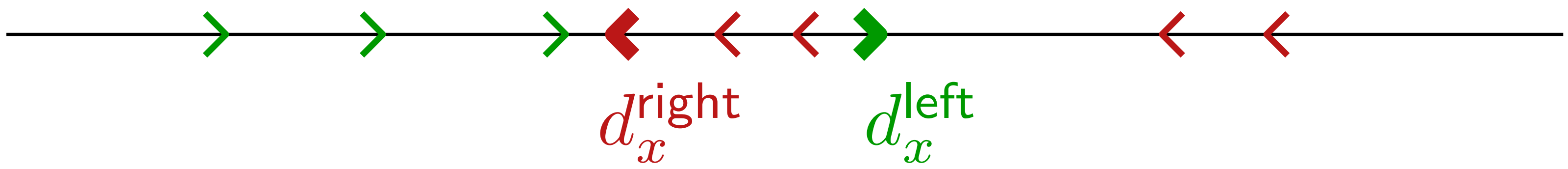
If the two steps result in a feasible solution, the LP (H, c) is unbounded and we can construct the ray ρ.

If the LP H' in Step 2 is infeasible, then then so is the initial LP (H, c).

If the LP $\bar{H}$ of the Step 1 is infeasible, then by Lemma 4, (H, c) is bounded.

Certificates of boundness

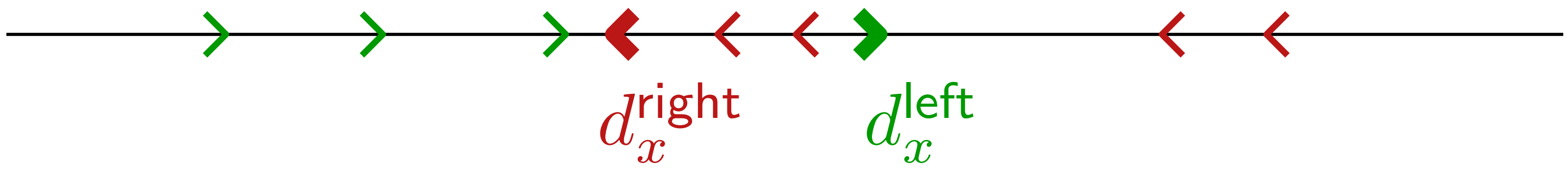
Obs.: When the LP $\bar{H} = \{d_x \eta_x + \eta_y \geq 0 | h \in H\}$ of the Step 1 is infeasible, we can use this information further!



1d-LP $\bar{H}$ is infeasible $\Leftrightarrow$ the interval $[d_x^{\text{left}}, d_x^{\text{right}}] = \emptyset$

Certificates of boundness

Obs.: When the LP $\bar{H} = \{d_x \eta_x + \eta_y \geq 0 | h \in H\}$ of the Step 1 is infeasible, we can use this information further!



1d-LP $\bar{H}$ is infeasible $\Leftrightarrow$ the interval $[d_x^{\text{left}}, d_x^{\text{right}}] = \emptyset$

- let h_1 and h_2 be the half planes corresponding to d_x^{left} and d_x^{right}
- There is no vector d that would "satisfy" h_1 and h_2, thus
- the LP $(\{h_1, h_2\}, c)$ is already bounded
- h_1 and h_2 are **certificates** of the boundness
- use h_1 and h_2 in 2dRandomizedBoundedLP as m_1 and m_2

Algorithms

2dRandomizedLP(H, c)

$\exists$? Vector d with $\langle d, c\rangle > 0$ and $\langle d, \eta(h)\rangle \geq 0$ for all $h \in H$

if d exists **then**

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

  if H' feasible **then**

    return (ray ρ, unbounded)

  else

    return infeasible

else

  $(h_1, h_2) \leftarrow$ Certificates for the boundness of (H, c)

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

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

2dRandomizedLP(H, c)

$\exists$? Vector d with $\langle d, c\rangle > 0$ and $\langle d, \eta(h)\rangle \geq 0$ for all $h \in H$
if d exists **then**
  $H' \leftarrow \{h \in H \mid \langle d, \eta(h)\rangle = 0\}$
  if H' feasible **then**
    return (ray ρ, unbounded)
  else
    return infeasible
else
  $(h_1, h_2) \leftarrow$ Certificates for the boundness of (H, c)
  $\overline{H} \leftarrow H \setminus \{h_1, h_2\}$
  return 2dRandomizedBoundedLP$(\overline{H}, c, h_1, h_2)$

Theorem 3: A two-dimentional LP with n constraints can be solved in $O(n)$ randomized expected time.

Discussion

Can the two-dimentional algorithms be generalized to more dimentions?

Discussion

Can the two-dimentional algorithms be generalized to more dimentions?

Yes! The same way as the two-dimentional LP is solved incrementally with reduction to a one-dimentional LP, a d-dimentional LP can be solved by a randomized incremental algorithm with a reduction to $(d-1)$-dimentional LP. The expected running time is then $O(c^d d!\, n)$ for a constant c. The algorithm is therefore usefull only for small values on d.

Discussion

Can the two-dimentional algorithms be generalized to more dimentions?

Yes! The same way as the two-dimentional LP is solved incrementally with reduction to a one-dimentional LP, a d-dimentional LP can be solved by a randomized incremental algorithm with a reduction to $(d-1)$-dimentional LP. The expected running time is then $O(c^d d!\, n)$ for a constant c. The algorithm is therefore usefull only for small values on d.

The simple randomized incremental algorithm for two and more dimentions given in this lecture is due to Seidel (1991).