Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

Computational Geometry

Linear Programming
or
Profit Maximization

Lecture #4

Dr. Philipp Kindermann

Winter Semester 2018/19

Maximizing Profit

You are the boss of a small company that produces two products, P_1 and P_2. If you produce x_1 units of P_1 and x_2 units of P_2, your profit in € is

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

Your production runs on three machines M_A, M_B, and M_C with the following capacities:

$$\begin{aligned} M_A: &\quad 4x_1 + 11x_2 \leq 880 \\ M_B: &\quad x_1 + x_2 \leq 150 \\ M_C: &\quad x_2 \leq 60 \end{aligned}$$

Which choice of (x_1, x_2) maximizes your profit?

The Answer

linear constraints:

$M_A:$	$4x_1 + 11x_2 \leq 880$	
$M_B:$	$x_1 + x_2 \leq 150$	$Ax \leq b$
$M_C:$	$x_2 \leq 60$	
	$x_1 \geq 0$	
	$x_2 \geq 0$	$x \geq 0$

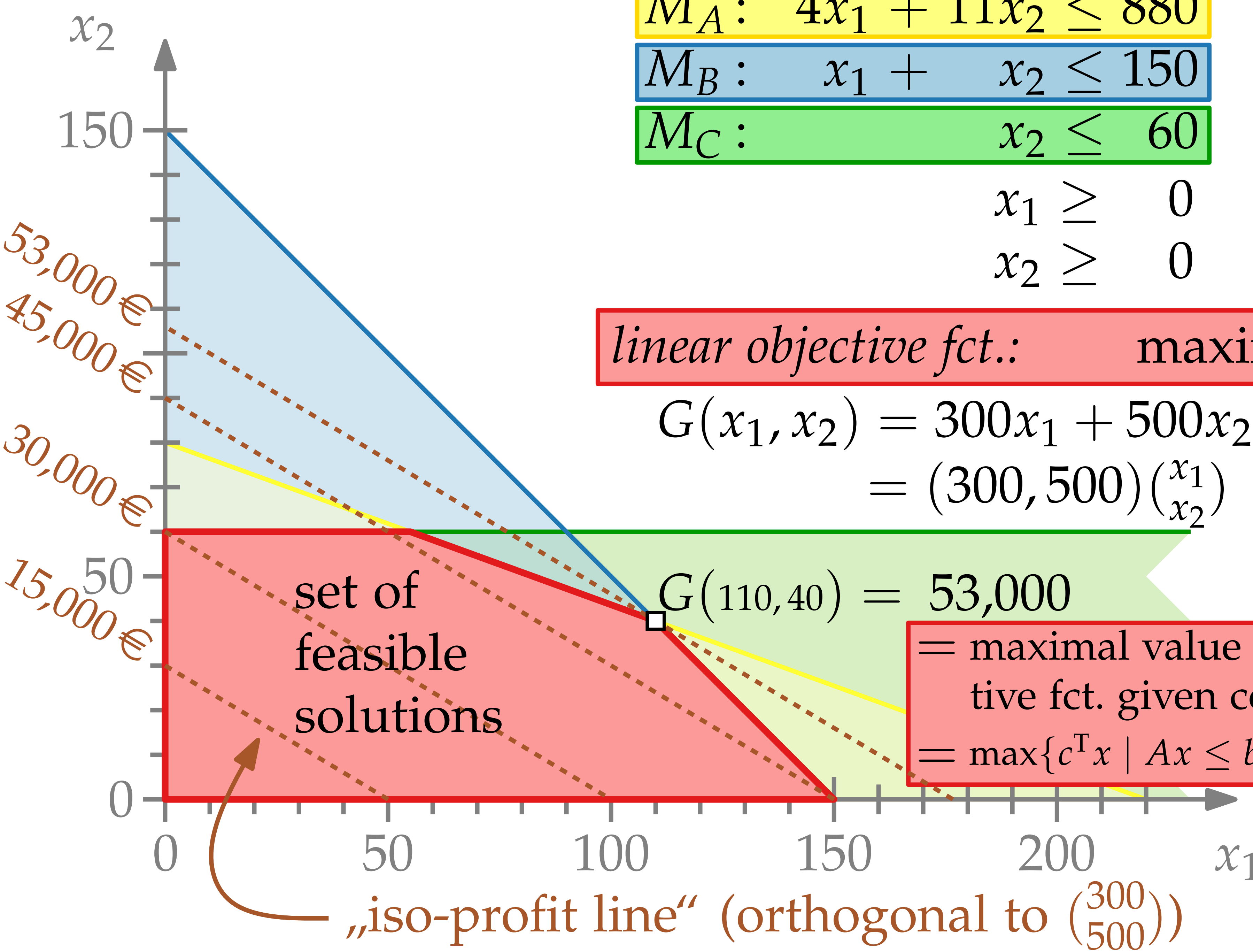
linear objective fct.: maximize $c^T x$

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

$$G(110, 40) = 53{,}000$$

$= $ maximal value of objective fct. given constraints

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



Definition and Known Algorithms

Given a set H of n halfspaces in $\mathbb{R}^d$ and a direction c, find a point $x \in \bigcap H$ such that cx is maximum (or minimum).

Many algorithms known, e.g.:

- Simplex [Dantzig '47]
- Ellipsoid method [Khatchiyan '79]
- Inner-point method [Karmakar' 84]

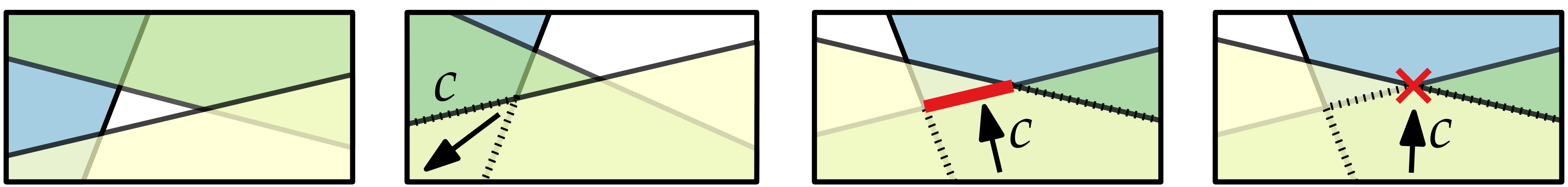
Good for instances where n *and* d are large.

We consider $d = 2$.

VERY important problem, e.g., in Operations Research.

["Book" application: casting]

$\bigcap H$ bounded.



$\bigcap H = \emptyset$ $\bigcap H$ unbnd. in dir. c set of optima: segment vs. point

First Approach

- compute $\bigcap H$ explicitly
- walk along $\partial\left(\bigcap H\right)$ to find a vertex x with cx maximum

IntersectHalfplanes(H)

if $|H| = 1$ **then**
  $C \leftarrow h$, where $\{h\} = H$
else
  split H into sets H_1 and H_2 with $|H_1|, |H_2| \approx |H|/2$
  $C_1 \leftarrow$ IntersectHalfplanes(H_1)
  $C_2 \leftarrow$ IntersectHalfplanes(H_2)
  $C \leftarrow$ IntersectConvexRegions(C_1, C_2)
return C

How??

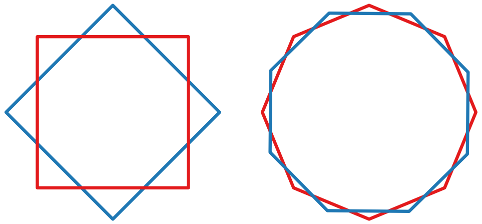
Running time: $T_{\mathrm{IH}}(n) = 2T_{\mathrm{IH}}(n/2) + T_{\mathrm{ICR}}(n)$

Intersecting Convex Regions

Any ideas?

Use sweep-line alg. for map overlay (line-segment intersections)!

Running time $T_{\mathrm{ICR}}(n) = O((n + I) \log n)$,



where $I = \#$ intersection points.

here: $I \leq n$

Running time $T_{\mathrm{IH}}(n) = 2T_{\mathrm{IH}}(n/2) + T_{\mathrm{ICR}}(n)$

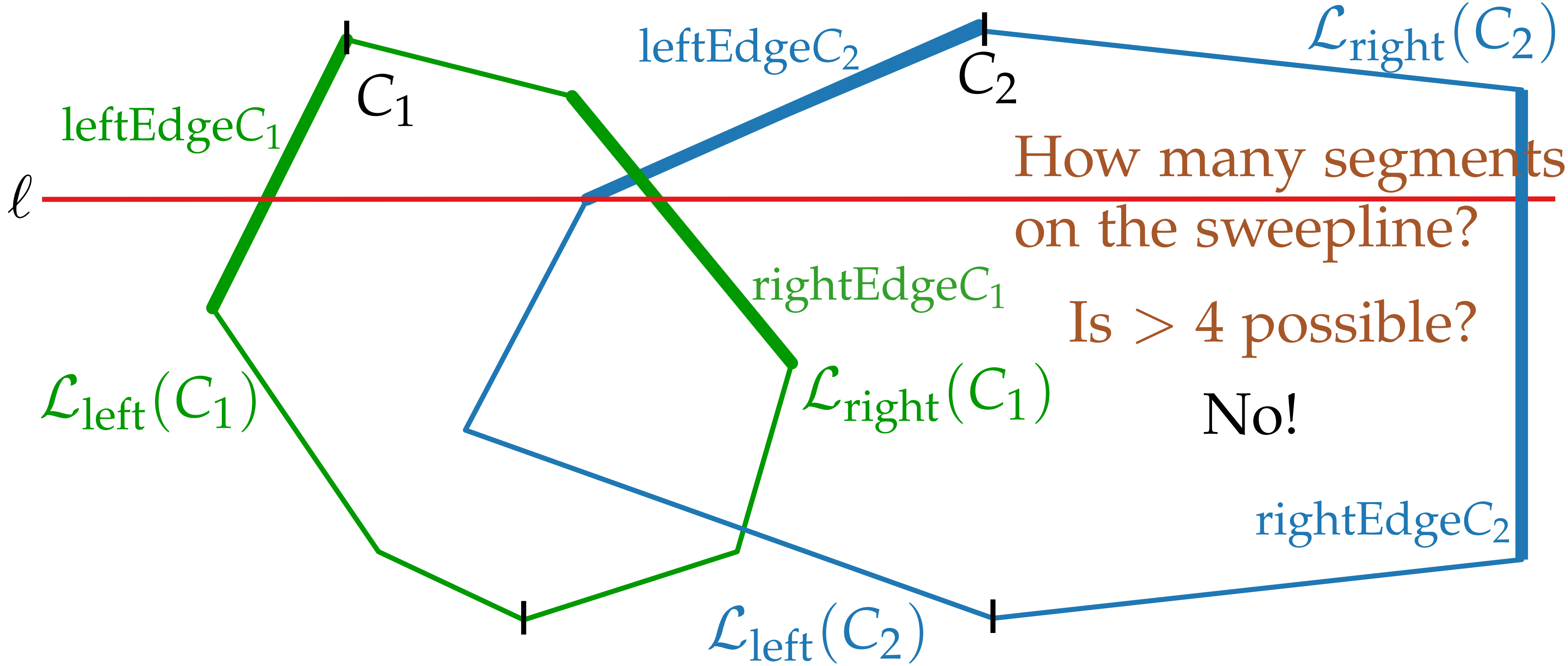
$$\leq 2T_{\mathrm{IH}}(n/2) + O(n \log n)$$

$$\in O(n \log^2 n)$$

Better ideas?

Use specialized algorithm for intersecting *convex* regions/polygons

Intersecting Convex Regions Faster



How many segments on the sweepline?

Is > 4 possible?

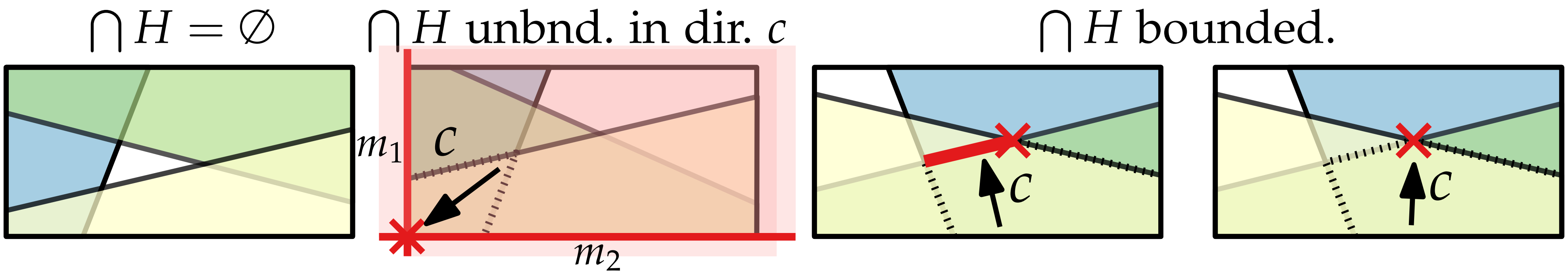
No!

Theorem. The intersection of two convex polygonal regions can be computed in linear time.

Corollary. The intersection of n half planes can be computed in $O(n \log n)$ time.

Can we do better?

A Small Trick: Make Solution Unique



- Add two bounding halfplanes m_1 and m_2

$$m_1 = \begin{cases} x \leq M & \text{if } c_x > 0, \\ x \geq M & \text{otherwise,} \end{cases} \quad \text{for some sufficiently large } M$$

$$m_2 = \begin{cases} y \leq M & \text{if } c_y > 0, \\ y \geq M & \text{otherwise.} \end{cases}$$

- Take the lexicographically largest solution.

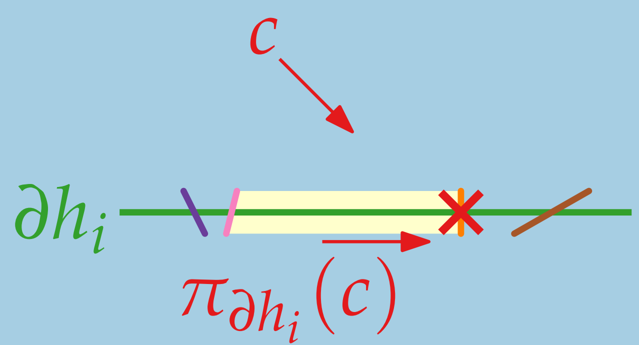
$\Rightarrow$ Set of solutions is either empty or a uniquely defined pt.

Incremental Approach

Idea: Don't compute $\bigcap H$, but just *one* (optimal) point!

Randomized

2DBoundedLP(H, c, m_1, m_2)
  compute random permutation of H
  $H_0 = \{m_1, m_2\}$
  $v_0 \leftarrow$ corner of $m_1 \cap m_2$
  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$(\pi_{\partial h_i}(H_{i-1}), \pi_{\partial h_i}(c))$ $O(i)$
      if $v_i =$ nil **then**
        return nil
    $H_i = H_{i-1} \cup \{h_i\}$ $O(1)$
  return v_n



w-c running time:

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

Result

Theorem. The 2D bounded LP problem can be solved in $O(n)$ expected time.

Proof. Let $X_i = \begin{cases} 1 & \text{if } v_{i-1} \notin h_i, \\ 0 & \text{else.} \end{cases}$ (indicator random variable).

Then the expected running time is

$$\mathbf{E}[T_{2d}(n)] = \mathbf{E}[\sum_{i=1}^{n}(1 - X_i) \cdot O(1) + X_i \cdot O(i)]$$
$$= O(n) + \sum \mathbf{E}[X_i] \cdot O(i)$$
$$= O(n) + \sum \mathbf{Pr}[X_i = 1] \cdot O(i) \quad = O(n).$$

We fix the i random halfplanes in H_i.

$\mathbf{Pr}[X_i = 1]$ = probability that the optimal solution changes when h_i is added to H_{i-1}.

Proof technique: *Backward analysis!*

= probability that the optimal solution changes when h_i is removed from H_i.

$\leq 2/i$. This is independent of the choice of H_i. □