Advanced Algorithms

Winter term 2019/20

Lecture 3. 2D Linear Programming via sweep-lines and randomization

Source: CG: A&A §4

Steven Chaplick & Alexander Wolff

Chair for Computer Science I
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:

- $M_A$: $4x_1 + 11x_2 \leq 880$
- $M_B$: $x_1 + x_2 \leq 150$
- $M_C$: $x_2 \leq 60$

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

**linear constraints:**

- **$M_A$:** $4x_1 + 11x_2 \leq 880$
- **$M_B$:** $x_1 + x_2 \leq 150$
- **$M_C$:** $x_2 \leq 60$

**Ax ≤ b**

**x ≥ 0**

**linear objective fct.:**

maximize $c^T x$

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

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

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

= maximum value of objective fct. given constraints

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

"iso-profit line" (orthogonal to $(300, 500)$)
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]

\[ \bigcap H = \emptyset \]
\[ \bigcap H \text{ unbd. in dir. } c \]
set of optima: segment vs. point

\[ \bigcap H \text{ bounded.} \]
First Approach

- compute $\cap H$ iteratively
- walk $\partial (\cap H)$, find vertex $x$ w/ $cx$ maximum, $O(n)$ time

**IntersectHalfplanes($H$)**

Let $H = (h_1, \ldots, h_n)$

$C \leftarrow h_1$

**foreach** $i$ from 2 to $n$ do

$C \leftarrow C \cap h_i$

**return** $C$

Running time: $T_{IH}(n) = n \cdot O(n)$

Total Time: $O(n^2)$ :

Exercise: Compute $C \cap h_i$ faster.
Second Approach

- compute $\bigcap H$ via divide and conquer
- walk $\partial (\bigcap H)$, find vertex $x$ w/ $cx$ maximum, $O(n)$ time

```plaintext
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 $C$
```

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

How complex can the new region be?
Theorem. The intersection of two convex polygonal regions can be computed in linear time.
Sweep-Line Algorithm

Done, since we have finished C!
Data Structures

1) event (-point) queue $Q$

\[ p \prec q \iff_{\text{def.}} y_p > y_q \quad \text{or} \quad (y_p = y_q \text{ and } x_p < x_q) \]

Store event pts in \textit{sorted order} acc. to $\prec$

nextEvent() : either, next point (by $\prec$), or the intersection pt. of two active segments (below the sweep-line)

... runtime? $O(1)$, since num. active segments $\leq 4$ :)

2) (sweep-line) status $T$

Store the segments intersected by $\ell$ in left-to-right order.

Also, maintain the new convex hull.
Second Approach: Halfplane Intersection

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

\[
\text{IntersectHalfplanes}(H)
\]

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

return \(C\)

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

**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

$\bigcap H = \emptyset \quad \bigcap H \text{ unbd. in dir. } c \quad \bigcap H \text{ bounded.}$

- Add two bounding halfplanes $m_1$ and $m_2$

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

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.

⇒ Set of solutions is either empty or a uniquely defined pt.

Idea: $M$ based on obj.fct. $c$. See §4.5 of CG: A&A for more on unbounded LPs.
**Incremental Approach**

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

Randomized

2DBoundedLP($H, c, m_1, m_2$)
compute random permutation of $H$

$H_0 = \{m_1, m_2\}$

$v_0 \leftarrow \text{corner of } m_1 \cap m_2$

for $i \leftarrow 1$ to $n$ do

if $v_{i-1} \in h_i$ then

$v_i \leftarrow v_{i-1}$

else

$v_i \leftarrow \text{1DBoundedLP}(\pi_{\partial h_i}(H_{i-1}), \pi_{\partial h_i}(c))$

if $v_i = \text{nil}$ then

return $\text{nil}$

$H_i = H_{i-1} \cup \{h_i\}$

return $v_n$

**w-c running time:**

$T(n) = \sum_{i=1}^{n} O(i) =$

$= O(n^2)$

:-(
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

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

We fix the $i$ random halfplanes in $H_i$.

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

Proof technique: Backward analysis! 

\[
\Pr[X_i = 1] = \text{probability that the optimal solution changes when } h_i \text{ is removed from } H_i.
\]

$\leq 2/i$. This is independent of the choice of $H_i$.\]
Alt. for Intersecting Convex Regions

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

Running time $T_{MO}(n) = O((n + I) \log n)$, where $I = \#$ intersection points.

Running time $T_{IH}(n) = 2T_{IH}(n/2) + T_{ICR}(n) \leq 2T_{IH}(n/2) + O(n \log n) \in O(n \log^2 n)$

As this is more general, it is unsurprisingly worse ...

~> Better to use specialized algorithm for intersecting convex regions/polygons

* it can happen sometimes that general algorithms give optimal runtimes for special cases