Linear Programming and Halfplane Intersection
Carola Wenk
Word Problem

A company produces tables and chairs. The profit for a chair is $2, and for a table $4. Machine group $A$ needs 4 hours to produce a chair, and 6 hours to produce a table. Machine group $B$ needs 2 hours to produce a chair, and 6 hours to produce a table. Per day there are at most 120 working hours for group $A$ and at most 72 hours for group $B$.

How can the company maximize profit?

Variables:
- $c_A = \#$ chairs produced on machine group $A$
- $c_B = \#$ chairs produced on machine group $B$
- $t_A = \#$ tables produced on machine group $A$
- $t_B = \#$ tables produced on machine group $B$

Constraints:
- $4c_A + 6t_A \leq 120$
- $2c_B + 6t_B \leq 72$

Objective function (profit):
Maximize $2(c_A + c_B) + 4(t_A + t_B)$
Linear Programming

Variables: $x_1, \ldots, x_d$

Objective function: Maximize $f_{\overline{c}}(\overline{x}) = c_1 x_1 + \ldots + c_d x_d$

Constraints:

$h_1$: $a_{11} x_1 + \ldots + a_{1d} x_d \leq b_1$

$h_2$: $a_{21} x_1 + \ldots + a_{2d} x_d \leq b_2$

\ldots

$h_n$: $a_{n1} x_1 + \ldots + a_{nd} x_d \leq b_n$

- Each constraint $h_i$ is a half-space in $\mathbb{R}^d$
- $\bigcap_{i=1}^n h_i$ is the feasible region of the linear program
- Maximizing $f_{\overline{c}}(\overline{x})$ corresponds to finding a point $\overline{x}$ that is extreme in direction $\overline{c}$.
Sub-Problem: Halfspace Intersection (in R²: Halfplane Intersection)

Given: A set $H = \{h_1, h_2, \ldots, h_n\}$ of halfplanes

$h_i: a_i x + b_i y \leq c_i$

with constants $a_i, b_i, c_i$; for $i=1,\ldots,n$.

Find: $\bigcap_{i=1}^{n} h_i$, i.e., the feasible region of all points $(x, y) \in \mathbb{R}^2$ satisfying all $n$ constraints at the same time. This is a convex polygonal region bounded by at most $n$ edges.
D&C Halfplane Intersection

Algorithm Intersect_Halfplanes($H$):
Input: A set $H$ of $n$ halfplanes in $\mathbb{R}^2$
Output: The convex polygonal region $C = \bigcap_{h \in H} h$
if $|H|=1$ then
  $C = h$, where $H = \{h\}$
else
  split $H$ into two sets $H_1$ and $H_2$ of size $n/2$ each
  $C_1 = \text{Intersect}_\text{Halfplanes}(H_1)$
  $C_2 = \text{Intersect}_\text{Halfplanes}(H_2)$
  $C = \text{Intersect}_\text{Convex}_\text{Regions}(C_1, C_2)$
return $C$

- Use a plane-sweep to develop an $O(n)$-time algorithm for Intersect_Convex_Regions
- $T(n) = 2T(n/2)+n \Rightarrow T(n)\in O(n \log n)$
Incremental Linear Programming

- 2D linear program (LP)
- Assume the LP is bounded (otherwise add constraints)
- Assume there is one unique solution (if any); take the lexicographically smallest solution

- **Incremental approach:** Add one halfplane after the other.

\[ H_i = \{h_1, \ldots, h_i\} \quad C_i = h_1 \cap \cdots \cap h_i \quad C = C_n = \bigcap_{h \in H} h \]

Let \( v_i = \) unique optimal vertex for feasible region \( C_i \), for \( i \geq 2 \).

Then \( C_1 \supseteq C_2 \supseteq \ldots \supseteq C_n = C \), and hence

if \( C_i = \emptyset \) for some \( i \) then \( C_j = \emptyset \) for all \( j \geq i \).
**Incremental Linear Programming**

**Lemma:** Let $2 \leq i \leq n$.

(i) If $v_{i-1} \in h_i$ then $v_i = v_{i-1}$

(ii) If $v_{i-1} \notin h_i$ then

$$C_i = \emptyset$$

or $v_i \in l_i = \text{the line bounding } h_i$

Handling case (ii) involves solving a 1-dimensional LP on $l_i$:

- The feasible region is just an interval, that can be computed in linear time
  [rightmost left-bounded halfplane, leftmost right-bounded halfplane]
- $\Rightarrow$ We can compute a new $v_i$, or decide that the LP is infeasible, in $O(i)$ time
Algorithm 2D_Bounded_LP($H$, $\vec{c}$):

**Input:** A two-dimensional LP ($H$, $\vec{c}$)

**Output:** Report if ($H$, $\vec{c}$) is infeasible. Otherwise report the lexicographically smallest point that maximizes $f_\vec{c}$.

Let $h_1, ..., h_n$ be the halfplanes of $H$

Let $v_2$ be the corner of $C_2$, which exists because LP is bounded

for i=3 to n do
  if $v_{i-1} \in h_i$ then $v_i = v_{i-1}$
  else // Case (ii)
    $v_i =$ point on $l_i$ that maximizes $f_\vec{c}$ subject to constraints in $H_{i-1}$
    if such a point does not exist then
      Report that the LP is infeasible
      break;
  
return $v_n$

- **Runtime:** $\sum_{i=1}^{n} O(i) = O(n^2)$
- **Storage:** $O(n)$
Randomized Incremental LP

Depending on the insertion order of the halfplanes the runtime varies between $O(n)$ and $O(n^2)$.
⇒ Randomize the input order of the halfplanes.

**Theorem:** 2D_Randomized_Bounded_LP runs in $O(n)$ expected time and $O(n)$ deterministic space.

**Proof:** Define a random variable $X_i = \begin{cases} 1, & v_{i-1} \notin h_i \\ 0, & \text{else} \end{cases}$

The total time spent to resolve case (ii), over all $h_1, ..., h_n$ is

$$\sum_{i=1}^{n} O(i)X_i$$
Randomized Incremental LP

We now need to bound the expected value
$$E(\sum_{i=1}^{n} O(i)X_i) = \sum_{i=1}^{n} O(i)E(X_i)$$
and we know that $E(X_i) = P(X_i) = P(v_{i-1} \notin h_i)$.

Apply backwards analysis to bound $E(X_i)$:
- Fix $H_i = \{h_1, ..., h_i\}$ which determines $C_i$.
- Analyze what happened in last step when $h_i$ was added.
- $P(\text{had to compute new optimal vertex when adding } h_i)$
  $\leq \frac{2}{i}$

$$\Rightarrow E(X_i) \leq \frac{2}{i}$$

$$\Rightarrow \text{Total expected runtime is } \sum_{i=1}^{n} O(i) \frac{2}{i} = O(n)$$