Lecture 7 - Linear programming

- Consider $f: \mathbb{R}^2 \to \mathbb{R} : (x,y) \mapsto c_1 x + c_2 y$ where $(c_1, c_2) \neq (0,0)$

& a set $H = \{h_1, \dots, h_n\}$ of half-planes.

- Goal: Find a point (x,y) in intersection $\cap H$ at which f attains max value.

- Write

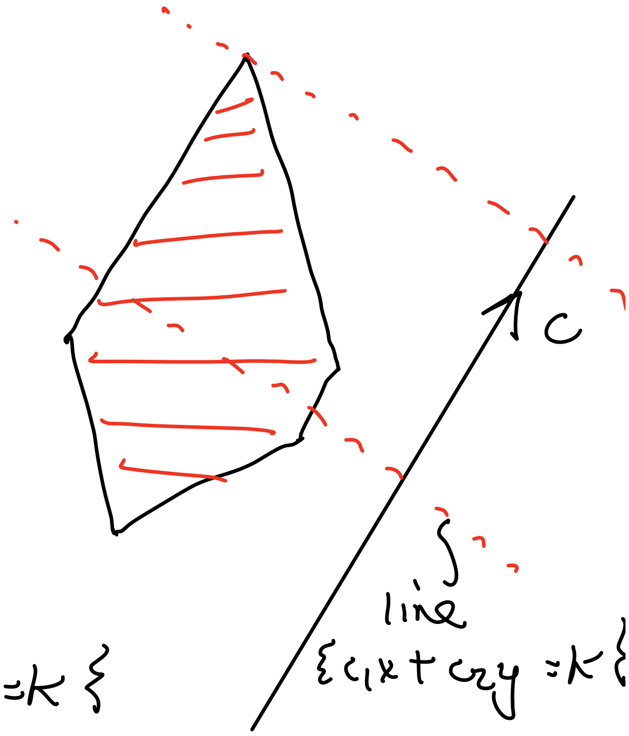
$$h_i : a_{i1} x + a_{i2} y \leq b_i \text{ for } i = \{1, \dots, n\}$$

Example?

Geometric significance

- f determined by a vector $\vec{c} = (c_1, c_2)$.
- As we move in the $\vec{c}$-direction f increases: ie. for $t > 0$,

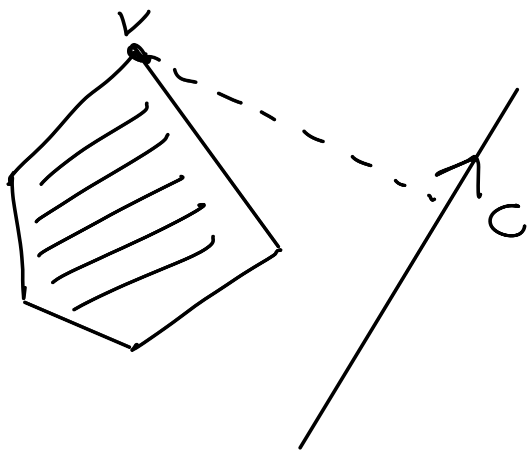
$$f((x,y) + t(c_1, c_2))$$
$$= f(x,y) + t f(c_1, c_2)$$
$$= f(x,y) + t(c_1^2 + c_2^2)$$
$$> f(x,y).$$



- At lines $\{(x,y) : c_1 x + c_2 y = k\}$ f has constant value. These are lines perpendicular to $\vec{c}$.
- Hence f obtains maximal value at any point v in intersection which is extreme in direction of c.

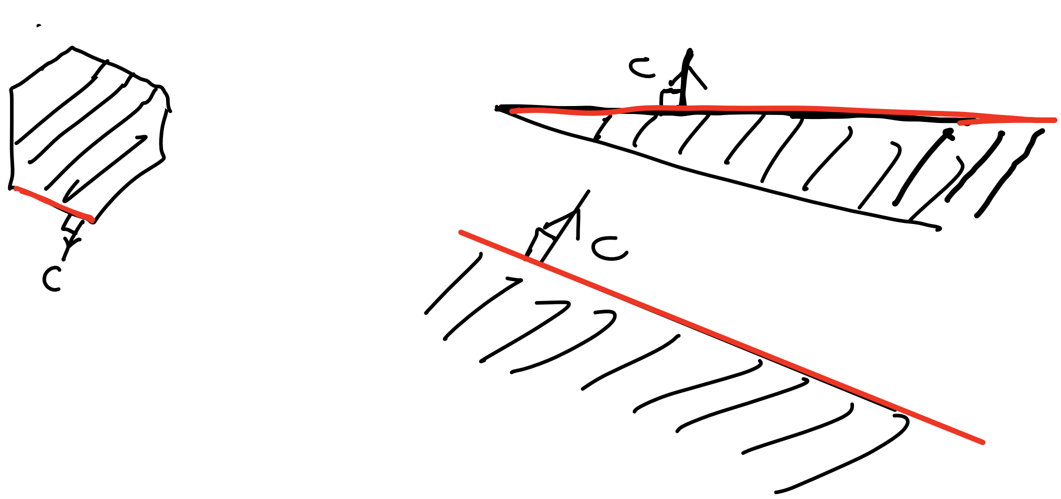
Different possibilities

1) $\cap H$ is empty. No solution - problem is infeasible.

2) 

Just 1 point v at which f obtains a max. value.

3) Infinitely many solutions - These form a segment, a half-line or a line.



4) 

The function f is unbounded on intersection: there exists a half-line in the intersection along which f is increasing (p in picture).

Input to algorithm:

vector $\vec{c}$ & $H = \{h_1, \dots, h_n\}$ set of half-planes.

Output:

- If infeasible, provide 3 half-planes with empty intersection.
- If f achieves a maximum, provide such a point in the intersection – min. wrt lex ordering)

- If f not bounded above on $\cap H$ provide a half-line in $\cap H$ along which f is increasing.

Firstly, 1-d case:

- $fx = cx$ for $c \neq 0$.
- half-planes $a_i x \leq b_i$ for $a_i \neq 0$, $i = 1, \ldots, n$.

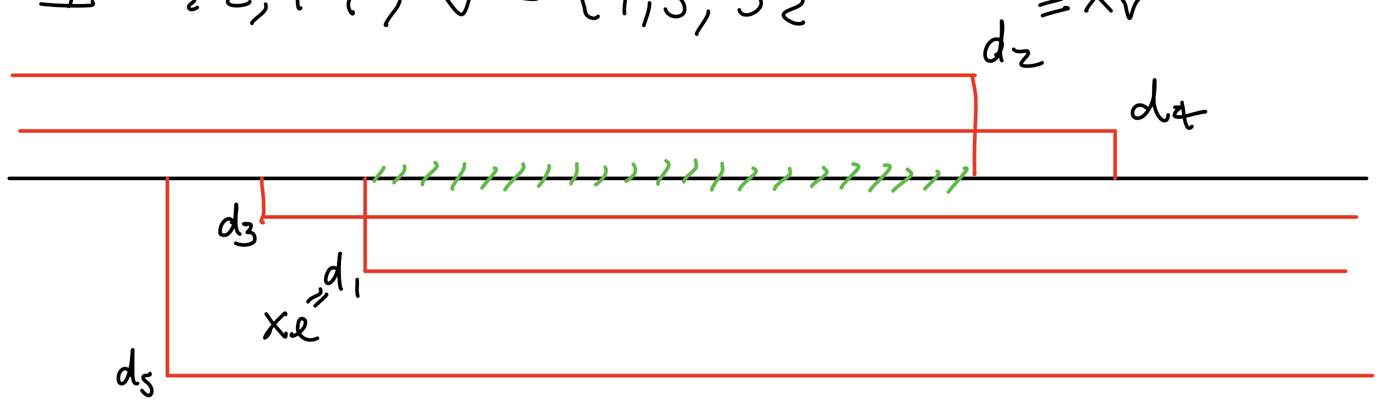
Goal: Find pt in intersection at which f attains max value.

- Let $I = \{i : a_i > 0\}$, $J = \{j : a_j < 0\}$
- Half-plane equations become

$$x \leq b_i / a_i = d_i \text{ for } i \in I$$
$$\& \quad x \geq b_j / a_j = d_j \text{ for } j \in J.$$

- let $x_\ell = \max\{-\infty, d_j : j \in J\}$
& $x_r = \min\{d_i, \infty : i \in I\}$.

$I = \{2, 4\}$, $J = \{1, 3, 5\}$



Cases: (1) $x_r < x_\ell$. (Int. empty & problem infeasible)

(2) $x_\ell \leq x_r < \infty$ & $c > 0$.

f has max at x_r.

③ $-\infty < x_\ell \leq x_r$ & $c < 0$

f has max at x_ℓ.

④ $x_r = \infty$ (I is empty) &

$c > 0$: $[x_\ell, \infty]$ is half-line along which f increases.

⑤ $x_\ell = -\infty$ & $c < 0$ –

then $(-\infty, x_r)$ is half-line along which f increases.

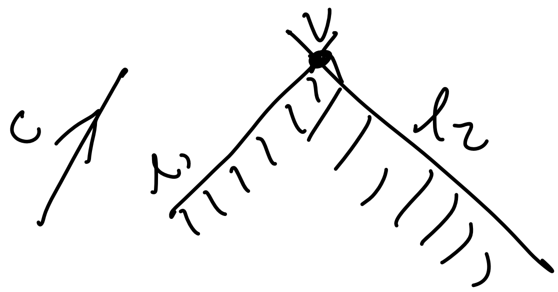
Complexity

Find max & min from sets of n points – complexity

$O(n)$

2-dimensional case (bounded case)

- In bounded case, we are given 2 half-planes h_1, h_2 such that f is <u>bounded</u> <u>from above</u> on $h_1 \cap h_2$.



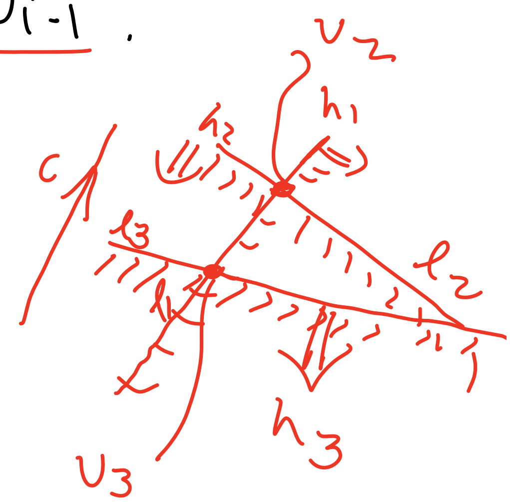
- Then $h_1 \cap h_2$ has maximum at $l_1 \cap l_2 = v$
In case there is more than 1 solution, we are also given <u>lex ordering</u> st. v is minimal solution.

Algorithm is incremental:

given optimal point $v_{i-1} \in C_{i-1} = h_1 \cap \dots \cap h_{i-1}$ we search for search for an optimal point $v_i \in h_i \cap C_{i-1} = C_i$.

Optimal point v_i: F achieves maximum at v_i in C_i, & v_i is least such with respect to the lex ordering.

- If $v_{i-1} \in C_i$, then $v_i = v_{i-1}$.
- Otherwise, C_i is empty or v_i lies on boundary ℓ_i of half-plane h_i.
- How to find v_i in this case?



- We search for max v_i function f restricted to line ℓ_i in $\underbrace{\ell_i \cap C_1}_{\text{1-d half plane}}, \dots, \underbrace{\ell_i \cap C_{i-1}}_{\text{1-d half plane}}$

1-d linear program!

In more detail,

- let $v_i = (x, y)$
  then $a_{i_1} x + a_{i_2} y = b_i$.
- Assuming $a_{i_2} \neq 0$ (otherwise $a_{i_1} \neq 0$)
  we have $y = \dfrac{b_i - a_{i_1} x}{a_{i_2}}$.
- We search for max. value of f on this line.
- On this line, consider f as a function of one variable

$$g(x) = c_1 x + c_2\big((b_i - a_{i_1} x)/a_{i_2}\big)$$

$$= \left(\frac{c_1 - c_2 a_{i_1}}{a_{i_2}}\right) x + c_2\left(b_i/a_{i_2}\right)$$

- Want max of g - does not depend on constant !

so we must find max value of

$$g^*(x) = \left(\frac{c_1 - c_2 a_{i_1}}{a_{i_2}}\right) x$$

- Searching for max ~~of~~ f on $\mathrm{lin}\, C_{i-1}$
  $\sim$ a max of g^* at

$a_{i_1} x + a_{i_2} \left(b_i - a_{i} x\right) \leq b_i$ f...

$\ldots \left(\frac{\ldots}{a_{i2}} \right) \leq b_j$ for $j = 1, \ldots, i-1$.

- Rewrite as

$$(*) \quad \left(a_{j1} - \frac{a_{j2} a_{i1}}{a_{i2}} \right) x \leq b_j - \frac{a_{j2} b_i}{a_{i2}} .$$

- Now we find v_i (or that C_i is empty) by solving 1-d linear program for g^* at cases $*$.

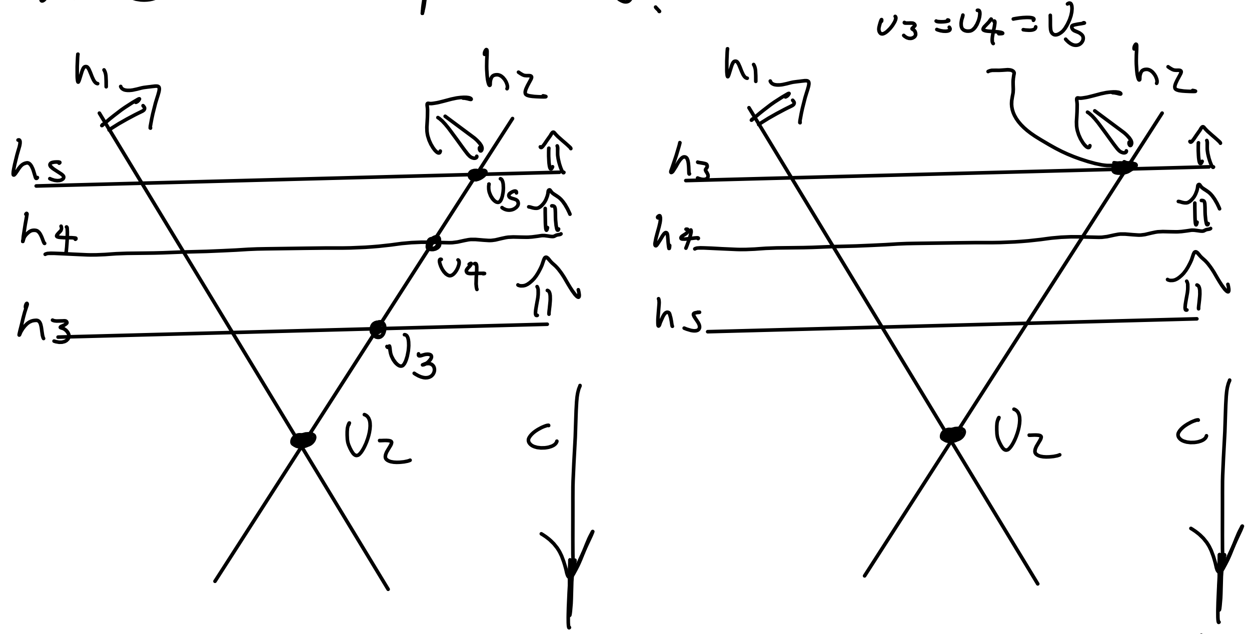
(See code in E-Learning - Lines 7-17, except line 9)

- Possible to extend to higher dim - always calling back to lower dimensional cases.

Running time

- If $v_{i-1} \in h_i$, constant time to set $v_i = v_{i-1}$.
- Otherwise, time to calculate v_i is linear in i - so $O(i)$.
- Complexity: $O(3) + O(4) + \ldots + O(n)$
$$= O(3+4+\ldots+n)$$
$$= O(n^2)$$

This is quite high running time & depends heavily on order of the half-planes.



- Introduce randomization into algorithm - considering a random ordering of the half-planes (see L9 of code in

E-Learning)

- Randomized expected time of alg. is much lower: average time of calculation taking into account all of possible orders
- Calculation of randomised exp. time:

X_i a random variable def. by $X_i = \begin{cases} 1 \text{ if } v_{i-1} \notin h_i \\ 0 \text{ if } v_{i-1} \in h_i \end{cases}$

- Time of alg. is estimated by

$$\sum_{i=3}^{n} O(i)\, X_i$$

- Rand. Expected time $E(X) = \sum_{i=3}^{n} O(i)\, E(X_i)$

where $E(X_i) = \text{prob}(X_i = 1) = \text{prob}(v_{i-1} \notin h_i)$.

Can show

$$E(X) = O(n)$$

expected time

- As we will show, prob$(v_{i-1} \notin h_i) = 2/i$.

Therefore $E(X) = \sum_{3}^{n} O(i) \cdot 2/i = \sum_{3}^{n} O(1)$
$= O(n)$.

Expected time is linear.

- We must show prob$(v_{i-1} \notin h_i) = 2/i$.
- Now $v_i = l_j \cap l_k$ for $j, k \leq i$

& j, k minimal with these properties.

Then
$p(v_{i-1} \notin h_i) = p(v_i \neq v_{i-1})$
$= p(i = j \text{ or } i = k)$.

- There are $i(i-1)$ choices of pairs $j, k \leq i$.
- There are $i-1$ choices in which $j = i$.
- - - - $i-1$ — — — — — $k = i$,

so $2(i-1)$ choices in which either j or k equals i.

So $\text{prob}(i=j \text{ or } i=k) = \dfrac{2(i-1)}{i(i-1)}$

$= 2/i$.

$\Rightarrow$ Expected randomised complexity is $O(n)$ for the bounded case.

(Note: in exam, won't be proof of complexity of algorithms.)

Examples for class

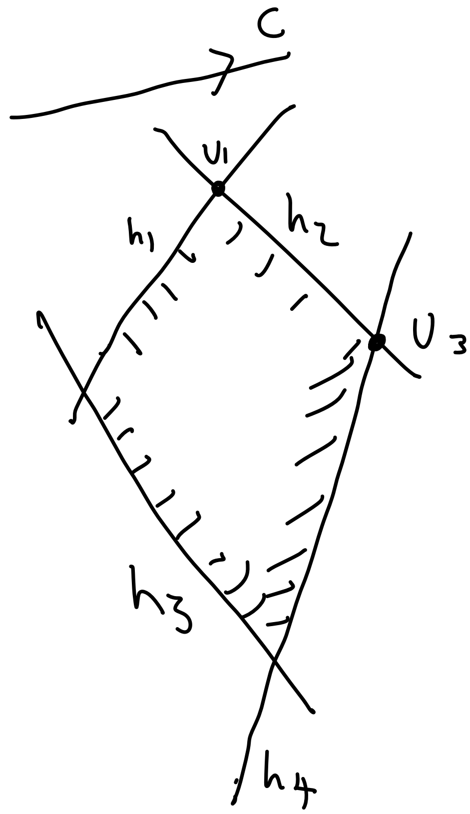
Lex ordering

$\hookrightarrow$

v_1

$v_2 = v_1$

v_3



See E-Learning for unbounded case (also reduces to 1-d linear program)