Linear Programming

Outline:

I. Introduction to linear programming (LP)

II. Incremental solution to LP in 2D

The Diet Problem*

How much money to spend in order to get what Polly needs every day?

- energy (2,000 kcal)
- protein (55 g)
- calcium (800 mg)

* V. Chvatal. *Linear Programming.* W. H. Freeman and Company, 1983.

The Diet Problem*

How much money to spend in order to get what Polly needs every day?

- energy (2,000 kcal)
- protein (55 g)
- calcium (800 mg)

Food	Serving size	Energy (kcal)	Protein (g)	Calcium (mg)	Price per serving (cents)
Oat meal	28 g	110	4	2	3
Chicken	100 g	205	32	12	24
Eggs	2 large	160	13	54	13
Whole milk	237 cc	160	8	285	9
Cherry pie	170 g	420	4	22	20
Pork with beans	260 g	260	14	80	19

* V. Chvatal. *Linear Programming.* W. H. Freeman and Company, 1983.

Daily Serving Limits

	Servings at most per day
Oatmeal	4
Chicken	3
Eggs	2
Milk	8
Cherry pie	2
Pork with beans	2

Task: Design the *most economical* menu.

Formulating the Problem

x_1: servings of oatmeal

x_2: servings of chicken

x_3: servings of eggs

x_4: servings of whole milk

x_5: servings of cherry pie

x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Formulating the Problem

x_1: servings of oatmeal

x_2: servings of chicken

x_3: servings of eggs

x_4: servings of whole milk

x_5: servings of cherry pie

x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

subject to

and

Formulating the Problem

x_1: servings of oatmeal

x_2: servings of chicken

x_3: servings of eggs

x_4: servings of whole milk

x_5: servings of cherry pie

x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

subject to

and

Formulating the Problem

x_1: servings of oatmeal

x_2: servings of chicken

x_3: servings of eggs

x_4: servings of whole milk

x_5: servings of cherry pie

x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\begin{aligned} \min \quad & 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6 \\ \text{subject to} \quad & 0 \le x_1 \le 4 \\ & 0 \le x_2 \le 3 \\ & 0 \le x_3 \le 2 \\ & 0 \le x_4 \le 8 \\ & 0 \le x_5 \le 2 \\ & 0 \le x_6 \le 2 \\ \text{and} \quad & \end{aligned}$$

Formulating the Problem

x_1: servings of oatmeal

x_2: servings of chicken

x_3: servings of eggs

x_4: servings of whole milk

x_5: servings of cherry pie

x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

$$\text{subject to} \quad 0 \le x_1 \le 4$$
$$0 \le x_2 \le 3$$
$$0 \le x_3 \le 2$$
$$0 \le x_4 \le 8$$
$$0 \le x_5 \le 2$$
$$0 \le x_6 \le 2$$

Servings-per-day limits

and

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

$$\begin{aligned} \text{subject to} \quad & 0 \le x_1 \le 4 \\ & 0 \le x_2 \le 3 \\ & 0 \le x_3 \le 2 \\ & 0 \le x_4 \le 8 \\ & 0 \le x_5 \le 2 \\ & 0 \le x_6 \le 2 \end{aligned}$$

Servings-per-day limits

and

$$\begin{aligned} \text{energy} \quad & 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000 \\ \text{protein} \quad & 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55 \\ \text{calcium} \quad & 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800 \end{aligned}$$

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

Constraints (linear)

$$\text{subject to} \quad 0 \le x_1 \le 4$$
$$0 \le x_2 \le 3$$
$$0 \le x_3 \le 2$$
$$0 \le x_4 \le 8$$
$$0 \le x_5 \le 2$$
$$0 \le x_6 \le 2$$

Servings-per-day limits

and

$$\text{energy} \quad 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000$$
$$\text{protein} \quad 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55$$
$$\text{calcium} \quad 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800$$

Formulating the Problem

x_1: servings of oatmeal
x_2: servings of chicken
x_3: servings of eggs
x_4: servings of whole milk
x_5: servings of cherry pie
x_6: servings of pork with beans

Food	Price per serving (cents)
Oat meal	3
Chicken	24
Eggs	13
Whole milk	9
Cherry pie	20
Pork with beans	19

Objective function (linear)

$$\min \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$$

Linear Program!

Constraints (linear)

$$\begin{aligned}
\text{subject to} \quad & 0 \le x_1 \le 4 \\
& 0 \le x_2 \le 3 \\
& 0 \le x_3 \le 2 \\
& 0 \le x_4 \le 8 \\
& 0 \le x_5 \le 2 \\
& 0 \le x_6 \le 2
\end{aligned}$$

Servings-per-day limits

and

$$\begin{aligned}
\text{energy} \quad & 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 \ge 2000 \\
\text{protein} \quad & 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 \ge 55 \\
\text{calcium} \quad & 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 \ge 800
\end{aligned}$$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\ x_d$

subject to

$$a_{11} x_1 + a_{12} x_2 + \cdots + a_{1d} x_d \leq b_1$$
$$a_{21} x_1 + a_{22} x_2 + \cdots + a_{2d} x_d \leq b_2$$
$$\vdots$$
$$a_{n1} x_1 + a_{n2} x_2 + \cdots + a_{nd} x_d \leq b_n$$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\ x_d$

subject to
$$a_{11} x_1 + a_{12} x_2 + \cdots + a_{1d} x_d \leq b_1$$
$$a_{21} x_1 + a_{22} x_2 + \cdots + a_{2d} x_d \leq b_2$$
$$\vdots$$
$$a_{n1} x_1 + a_{n2} x_2 + \cdots + a_{nd} x_d \leq b_n$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\ x_d$ $= \boldsymbol{cx^T}$

subject to
$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d \le b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d \le b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d \le b_n$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$

Linear Programming (LP)

Max $c_1 x_1 + c_2 x_2 + \cdots + c_d\, x_d$ $= \boldsymbol{c x^T}$

subject to

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d}x_d &\le b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d}x_d &\le b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd}x_d &\le b_n \end{aligned} \quad \Big\} \; A\boldsymbol{x}^T \le \boldsymbol{b}^T$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$

$\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$

$\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$

Half-Space Constraints

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i$$

Half-Space Constraints

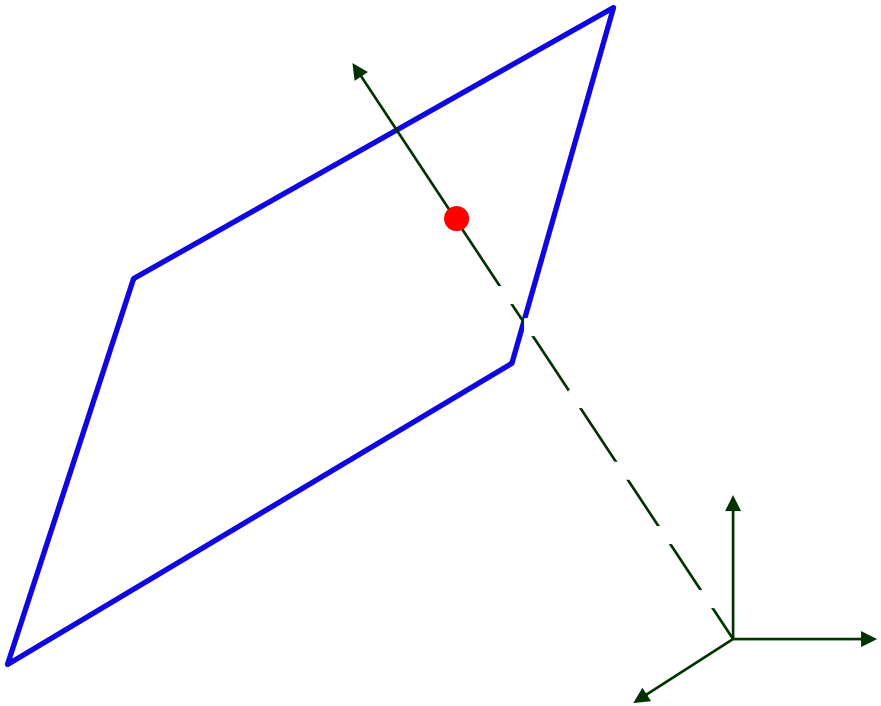
$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i$$

$$\frac{(a_{i1}, a_{i2}, \ldots, a_{id})}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}} \boldsymbol{x}^T \leq \frac{b_i}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$

Half-Space Constraints

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i$$

$$\frac{(a_{i1}, a_{i2}, \ldots, a_{id})}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}} \boldsymbol{x}^T \leq \frac{b_i}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$



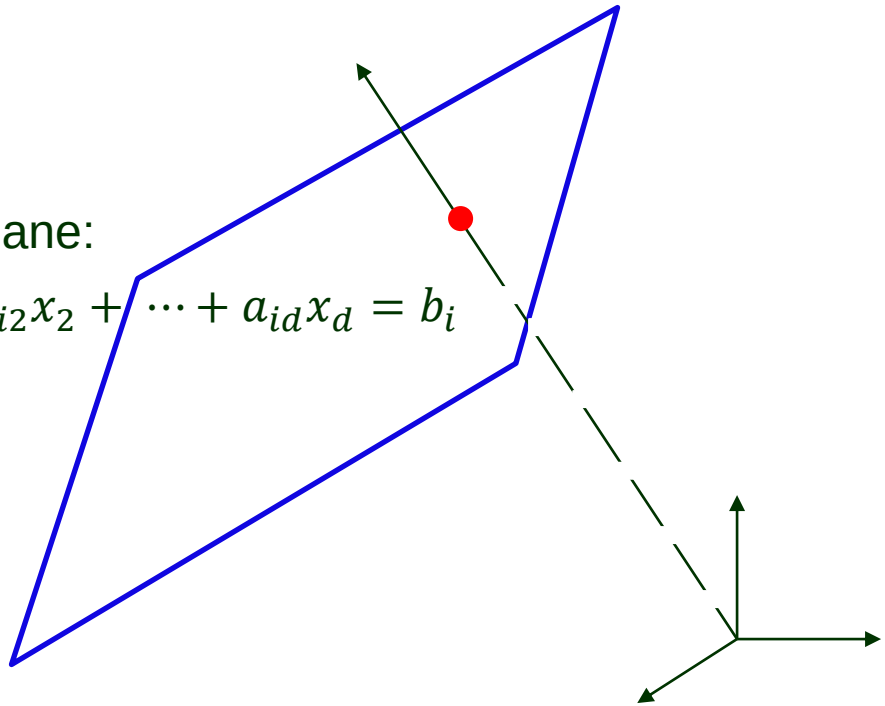
Half-Space Constraints

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i$$

$$\frac{(a_{i1}, a_{i2}, \ldots, a_{id})}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}} \boldsymbol{x}^T \leq \frac{b_i}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$

Hyperplane:

$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d = b_i$



Half-Space Constraints

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i$$

$$\Downarrow$$

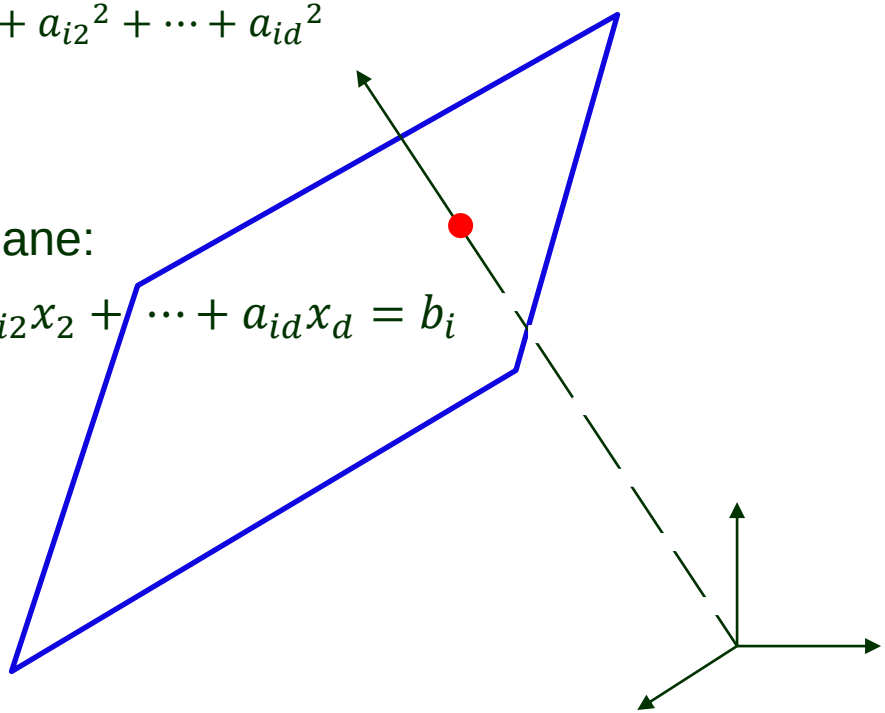
$$\frac{(a_{i1}, a_{i2}, \ldots, a_{id})}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}\, \boldsymbol{x}^T \leq \frac{b_i}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$

Unit normal to hyperplane:

$$\frac{(a_{i1}, a_{i2}, \ldots, a_{id})}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$

Hyperplane:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d = b_i$$



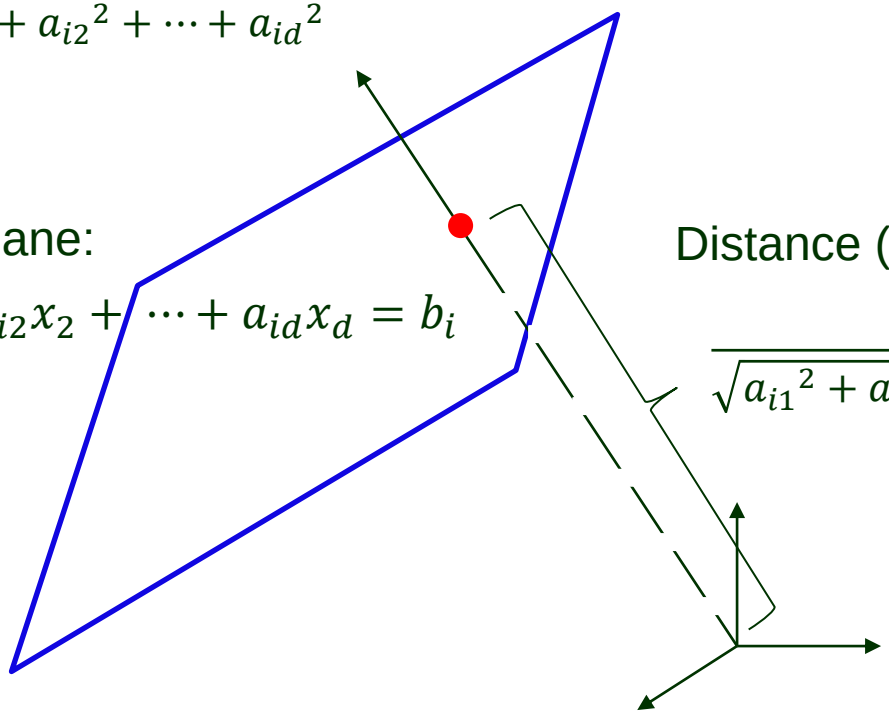
Half-Space Constraints

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d \leq b_i$$

$$\frac{(a_{i1}, a_{i2}, \ldots, a_{id})}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}\, \boldsymbol{x}^T \leq \frac{b_i}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$

Unit normal to hyperplane:

$$\frac{(a_{i1}, a_{i2}, \ldots, a_{id})}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$



Hyperplane:

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{id}x_d = b_i$$

Distance (signed) from the origin:

$$\frac{b_i}{\sqrt{{a_{i1}}^2 + {a_{i2}}^2 + \cdots + {a_{id}}^2}}$$

General LP

High dimensions ($d > 2$): Simplex algorithm (George Dantzig, 1947)

General LP

High dimensions ($d > 2$): Simplex algorithm (George Dantzig, 1947)

One of 10 greatest algorithms in the 20th century

General LP

High dimensions ($d > 2$): Simplex algorithm (George Dantzig, 1947)

One of 10 greatest algorithms in the 20th century

$O(2^d)$

General LP

High dimensions ($d > 2$): Simplex algorithm (George Dantzig, 1947)

One of 10 greatest algorithms in the 20th century

$O(2^d)$

Polynomial time algorithms:

- Ellipsoid algorithm (Leonid Khachivan, 1979)
- Karmarka's algorithm (Narenda Karmarkar, 1984) $O(d^{3.5}L)$

Running time is a high order polynomial in d variables and #bits L in input.

General LP

High dimensions ($d > 2$): Simplex algorithm (George Dantzig, 1947)

One of 10 greatest algorithms in the 20th century

$O(2^d)$

Polynomial time algorithms:

Running time is a high order polynomial in d variables and #bits L in input.

- Ellipsoid algorithm (Leonid Khachivan, 1979)
- Karmarka's algorithm (Narenda Karmarkar, 1984) $O(d^{3.5}L)$

In practice, the simplex algorithm is a better choice because

- its average running time is polynomial under various probability distributions, and
- it is numerically more stable.

2-Dimensional (2D) LP

Traditional LP algorithms are *inefficient* for low dimensions.

2-Dimensional (2D) LP

Traditional LP algorithms are *inefficient* for low dimensions.

We will focus on $d = 2$:

- cost $\boldsymbol{c} = (c_x, c_y)$
- cost $f(p) = c_x p_x + c_y p_y$ at a point (p_x, p_y)
- constraint set H of n half-planes
- *feasible region*: $R = \cap_{h \in H} h$

2-Dimensional (2D) LP

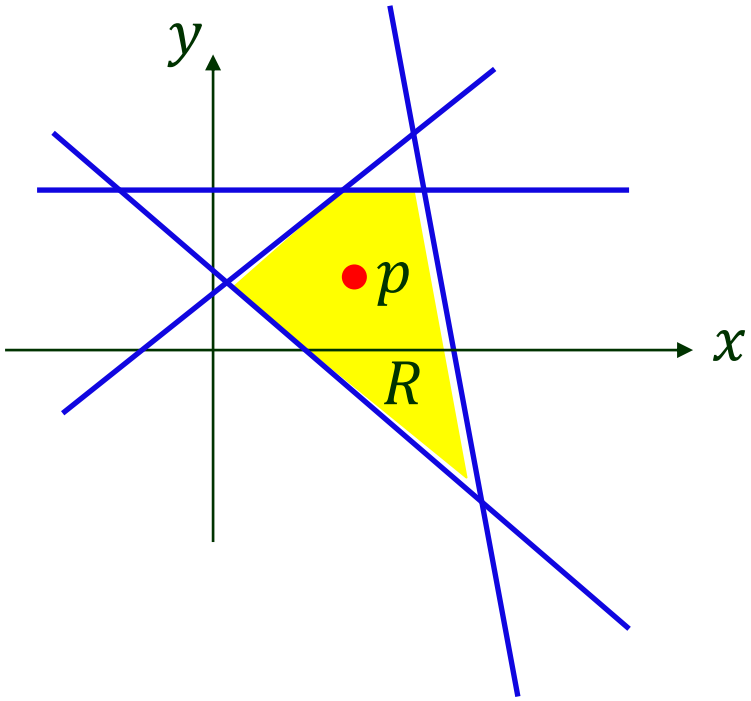
Traditional LP algorithms are *inefficient* for low dimensions.

We will focus on $d = 2$:

- cost $\boldsymbol{c} = (c_x, c_y)$
- cost $f(p) = c_x p_x + c_y p_y$ at a point (p_x, p_y)
- constraint set H of n half-planes
- *feasible region*: $R = \cap_{h \in H} h$

e.g., $H = \{x \geq 0, x + y \leq 1, x - y \geq 2\}$

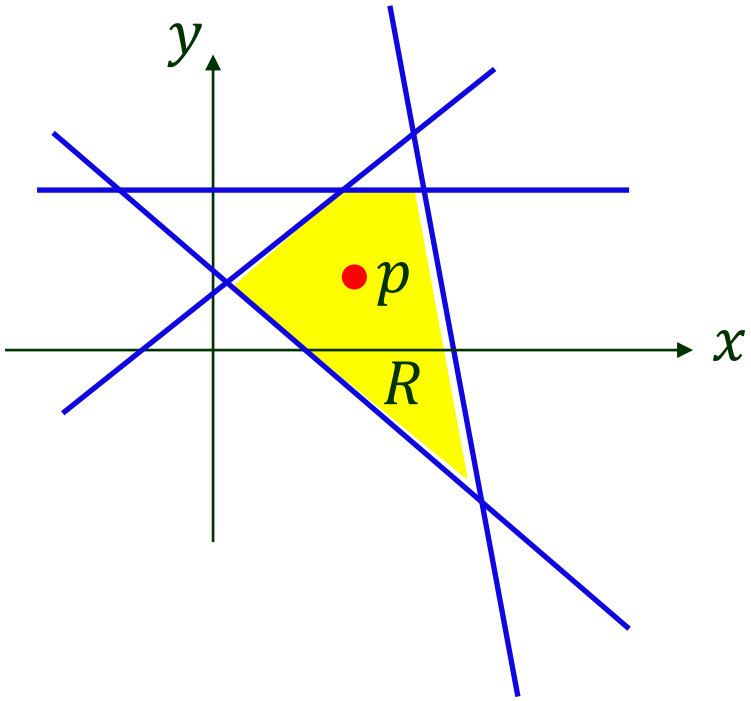
Feasible Region



Intersection of all half-planes.

feasible solution: a point p in the region

Feasible Region

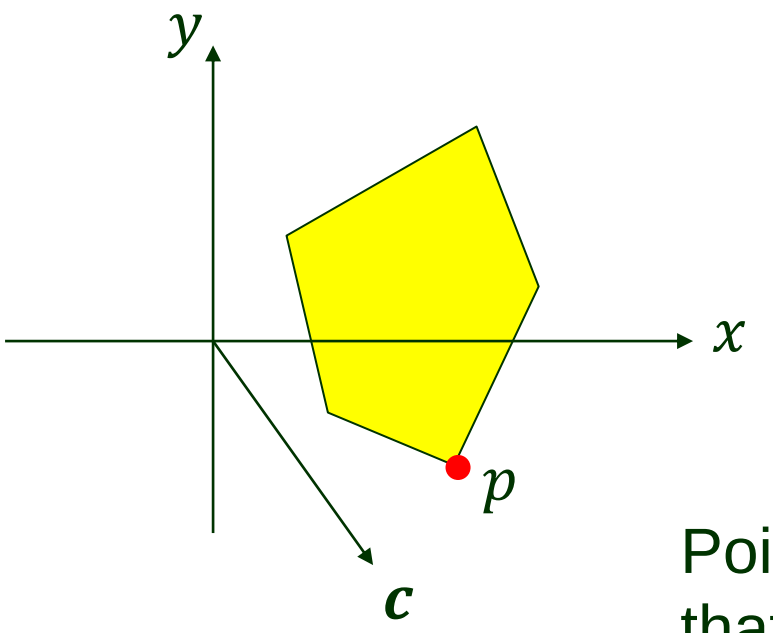


Intersection of all half-planes.

feasible solution: a point p in the region

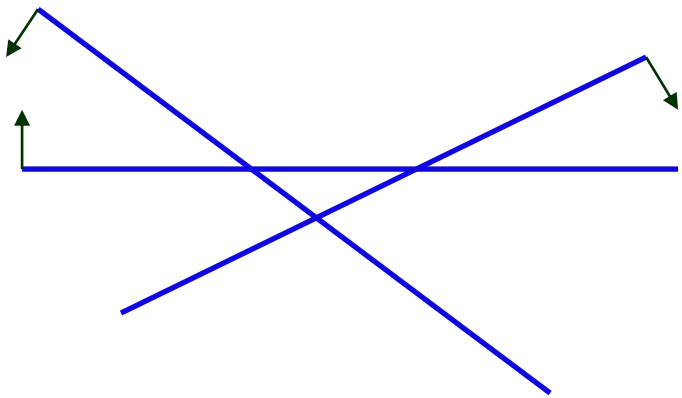
The LP is *infeasible* if $\bigcap_{h \in H} h = \emptyset$.

Optimal Solution for $(H, \boldsymbol{c})$



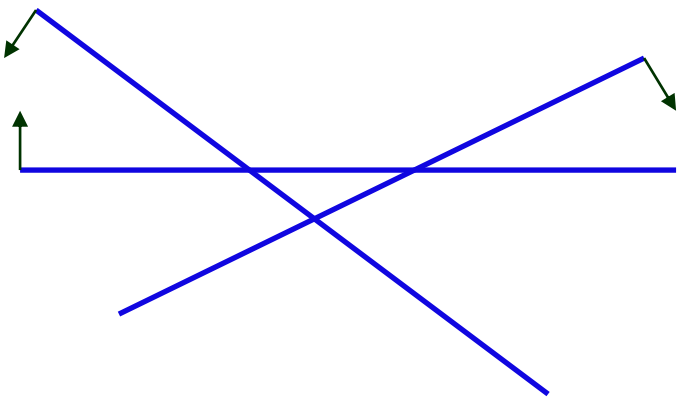
Point in the feasible region
that is *extreme* in the direction $\boldsymbol{c}$.

Two Possibilities

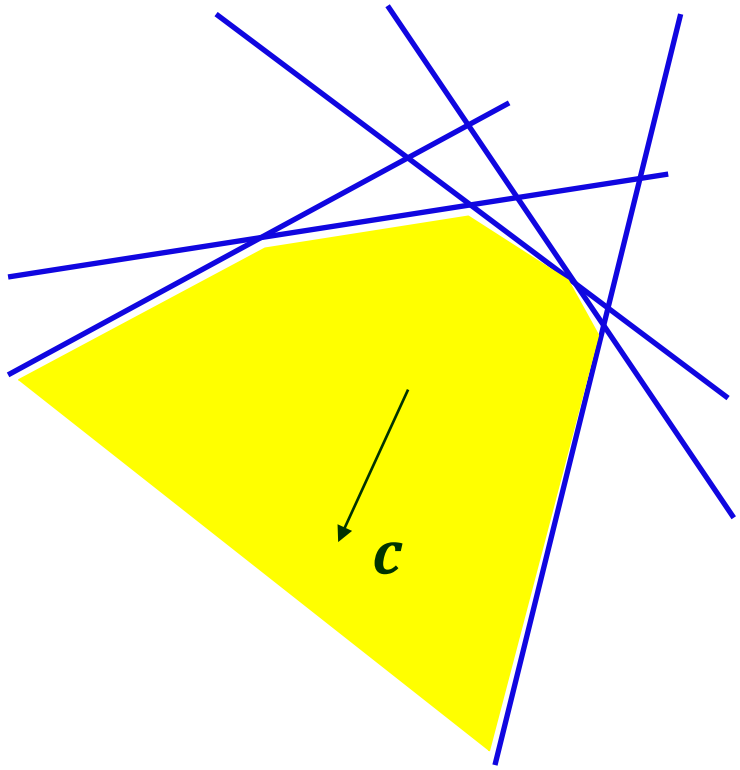


(a) infeasible LP

Two Possibilities



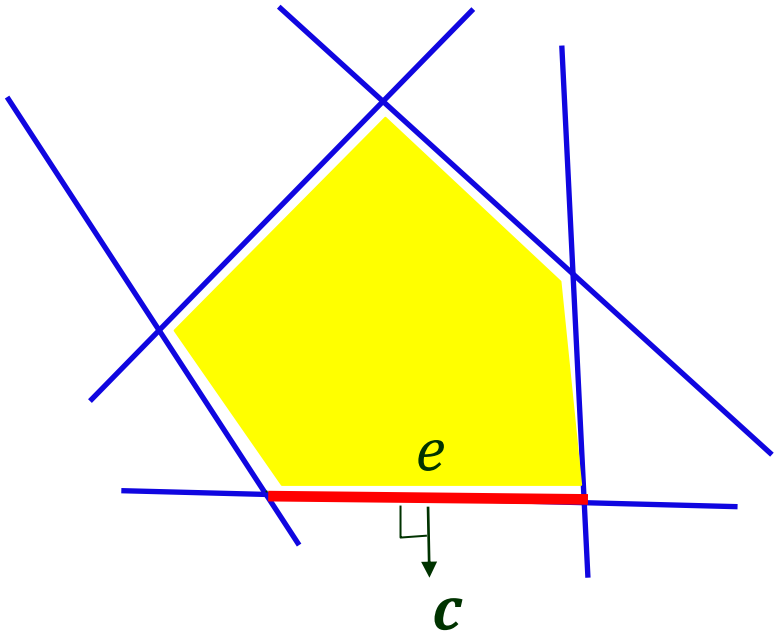
(a) infeasible LP



(b) unbounded LP

(f arbitrarily large)

Two More Possibilities

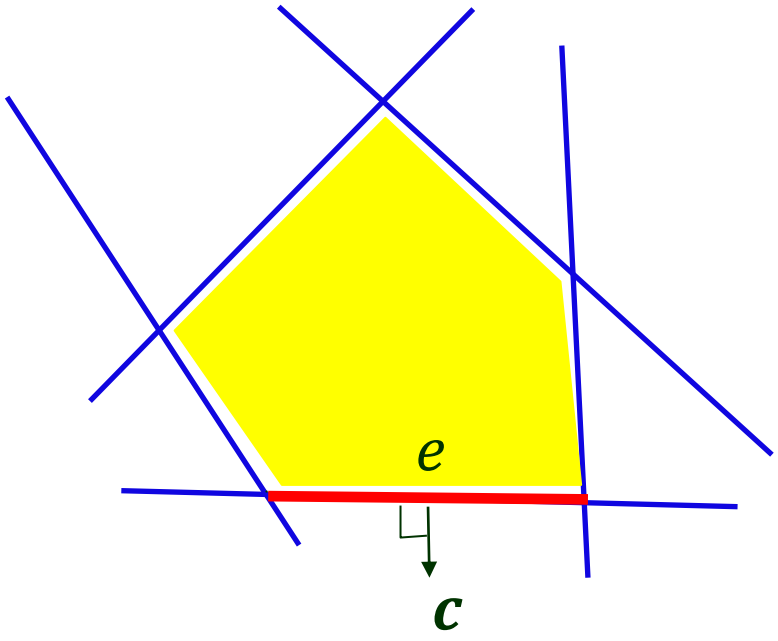


$e \perp \boldsymbol{c}$

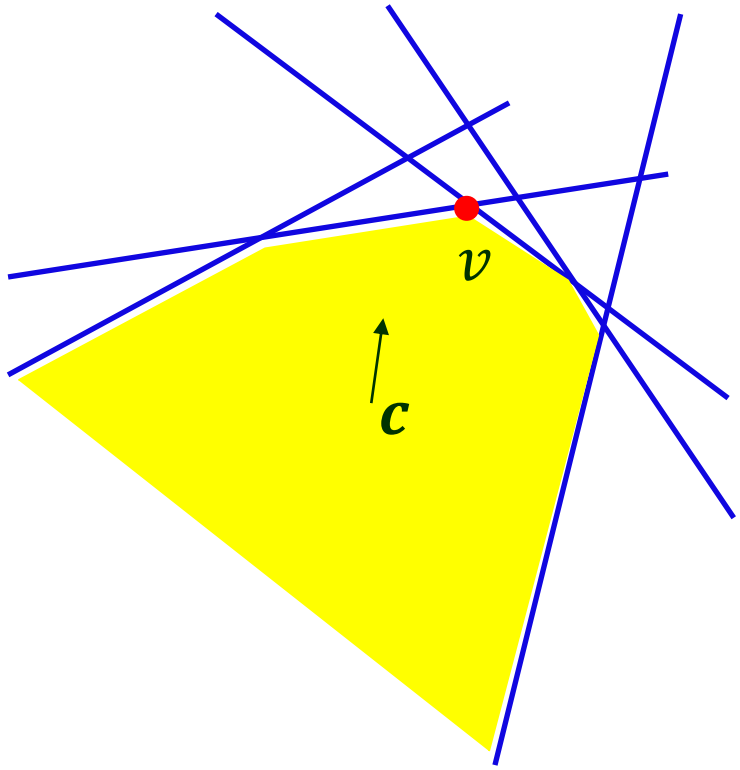
(c) a continuum of solutions
(every point on the edge e is optimal)

For uniqueness, choose lexicographically the smallest vertex.

Two More Possibilities



$e \perp \boldsymbol{c}$



(c) a continuum of solutions
(every point on the edge e is optimal)

For uniqueness, choose lexicographically the smallest vertex.

(d) unique solution

Incremental Construction

Subroutine *UNBOUNDEDLP*$(H, \boldsymbol{c})$ outputs

- a ray completely contained in R if unbounded;
- two half-planes $h_1, h_2 \in H$ such that the LP $(\{h_1, h_2\}, c)$ is bounded otherwise.

Incremental Construction

Subroutine *UNBOUNDEDLP*$(H, \boldsymbol{c})$ outputs

- a ray completely contained in R if unbounded;
- two half-planes $h_1, h_2 \in H$ such that the LP $(\{h_1, h_2\}, c)$ is bounded otherwise.

Remaining half-planes $h_3, h_4, \ldots, h_n$

Incremental Construction

Subroutine *UNBOUNDEDLP*$(H, \boldsymbol{c})$ outputs

- a ray completely contained in R if unbounded;
- two half-planes $h_1, h_2 \in H$ such that the LP $(\{h_1, h_2\}, c)$ is bounded otherwise.

Remaining half-planes $h_3, h_4, \ldots, h_n$

$$R_i = h_1 \cap h_2 \cap \cdots \cap h_i$$

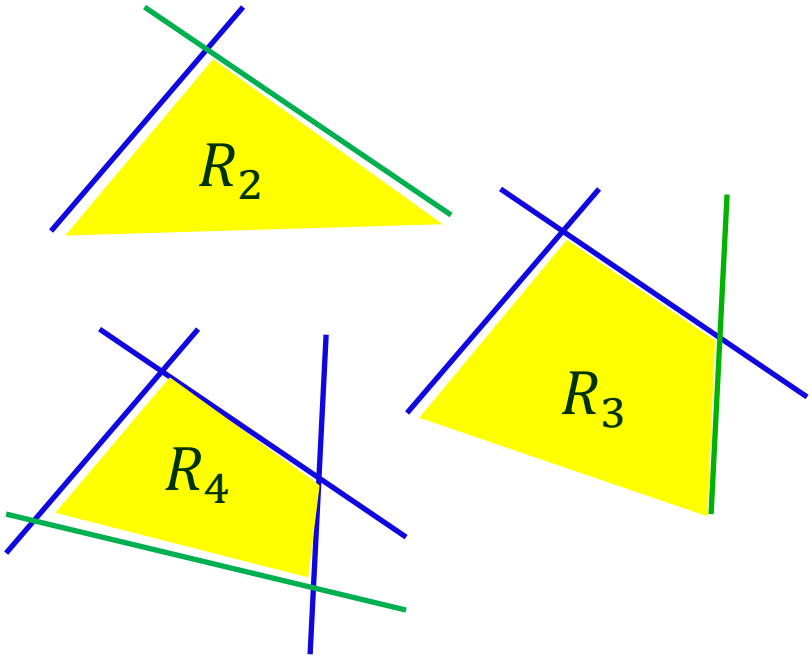
Incremental Construction

Subroutine *UNBOUNDEDLP*$(H, \boldsymbol{c})$ outputs

- a ray completely contained in R if unbounded;
- two half-planes $h_1, h_2 \in H$ such that the LP $(\{h_1, h_2\}, c)$ is bounded otherwise.

Remaining half-planes $h_3, h_4, \ldots, h_n$

$$R_i = h_1 \cap h_2 \cap \cdots \cap h_i$$



Incremental Construction

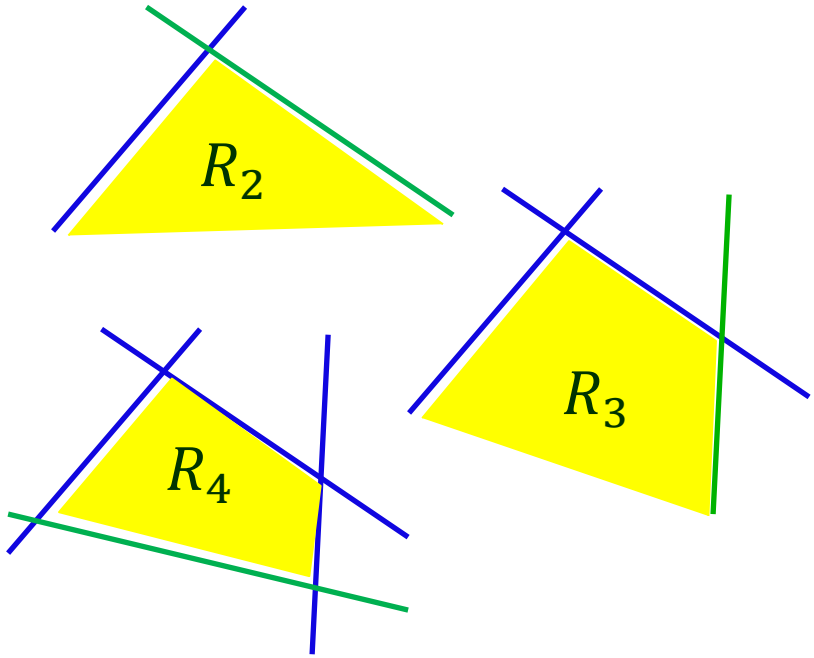
Subroutine *UNBOUNDEDLP*$(H, \boldsymbol{c})$ outputs

- a ray completely contained in R if unbounded;
- two half-planes $h_1, h_2 \in H$ such that the LP $(\{h_1, h_2\}, c)$ is bounded otherwise.

Remaining half-planes $h_3, h_4, \ldots, h_n$

$$R_i = h_1 \cap h_2 \cap \cdots \cap h_i$$

$$R_2 \supseteq R_3 \supseteq \cdots \supseteq R_n$$



Incremental Construction

Subroutine *UNBOUNDEDLP*$(H, \boldsymbol{c})$ outputs

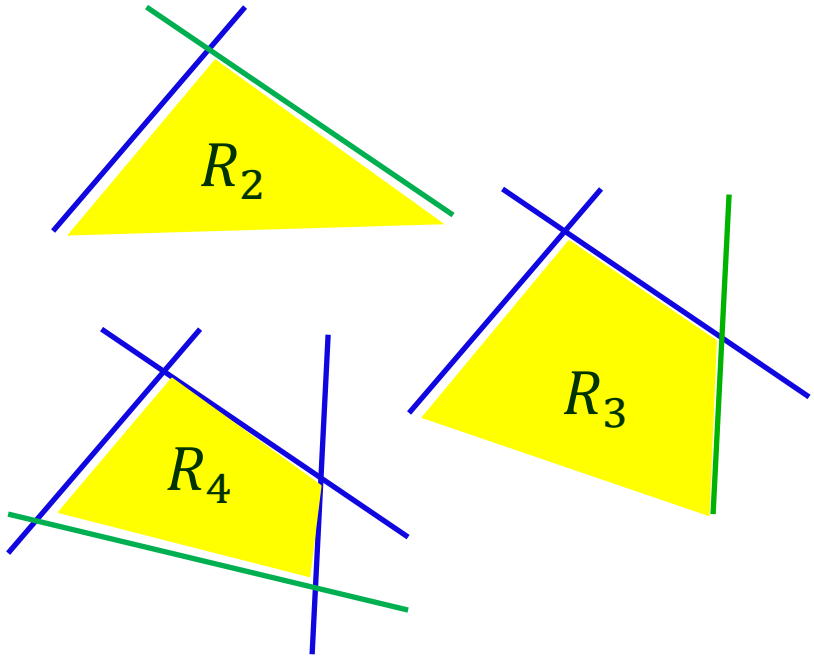
- a ray completely contained in R if unbounded;
- two half-planes $h_1, h_2 \in H$ such that the LP $(\{h_1, h_2\}, c)$ is bounded otherwise.

Remaining half-planes $h_3, h_4, \ldots, h_n$

$$R_i = h_1 \cap h_2 \cap \cdots \cap h_i$$

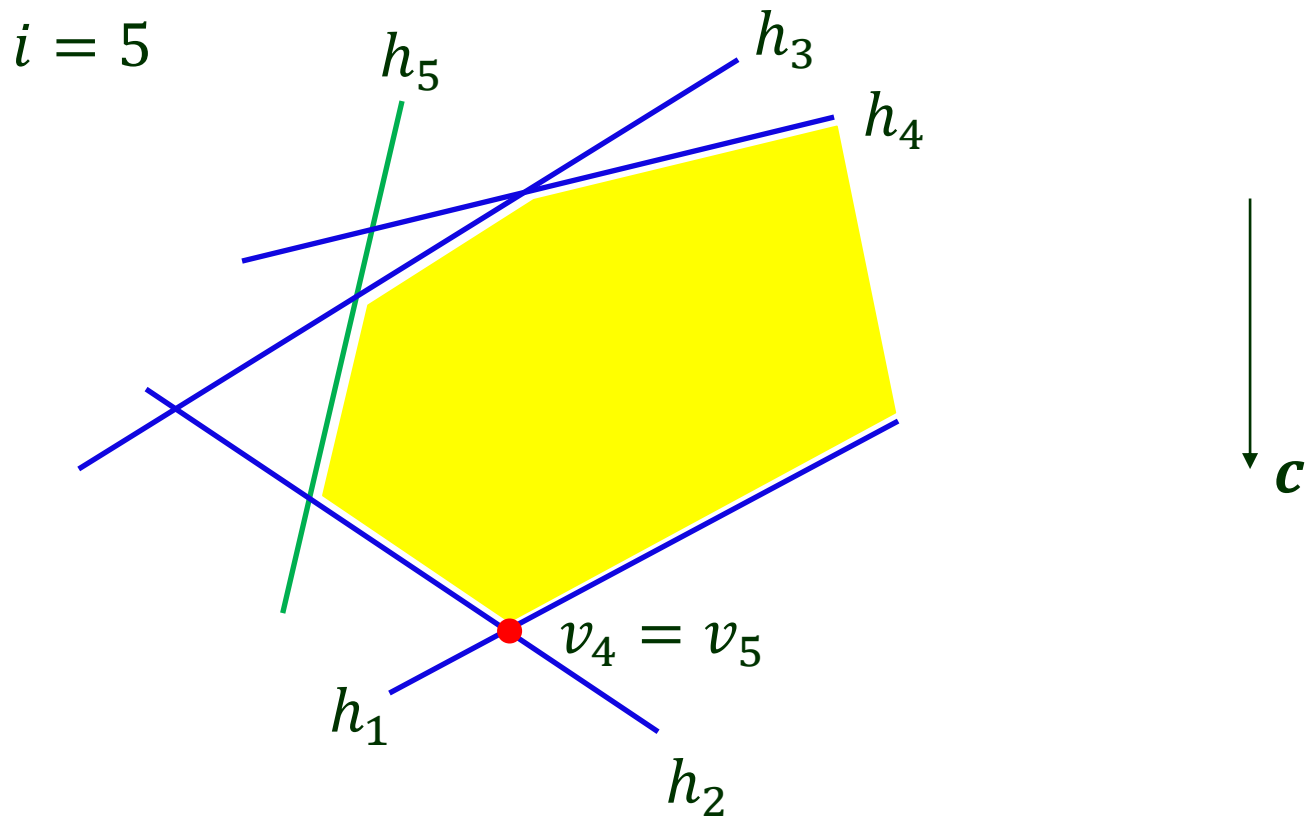
$$R_2 \supseteq R_3 \supseteq \cdots \supseteq R_n$$

If $R_i = \emptyset$ then $R_j = \emptyset$ for all $j > i$, infeasible!



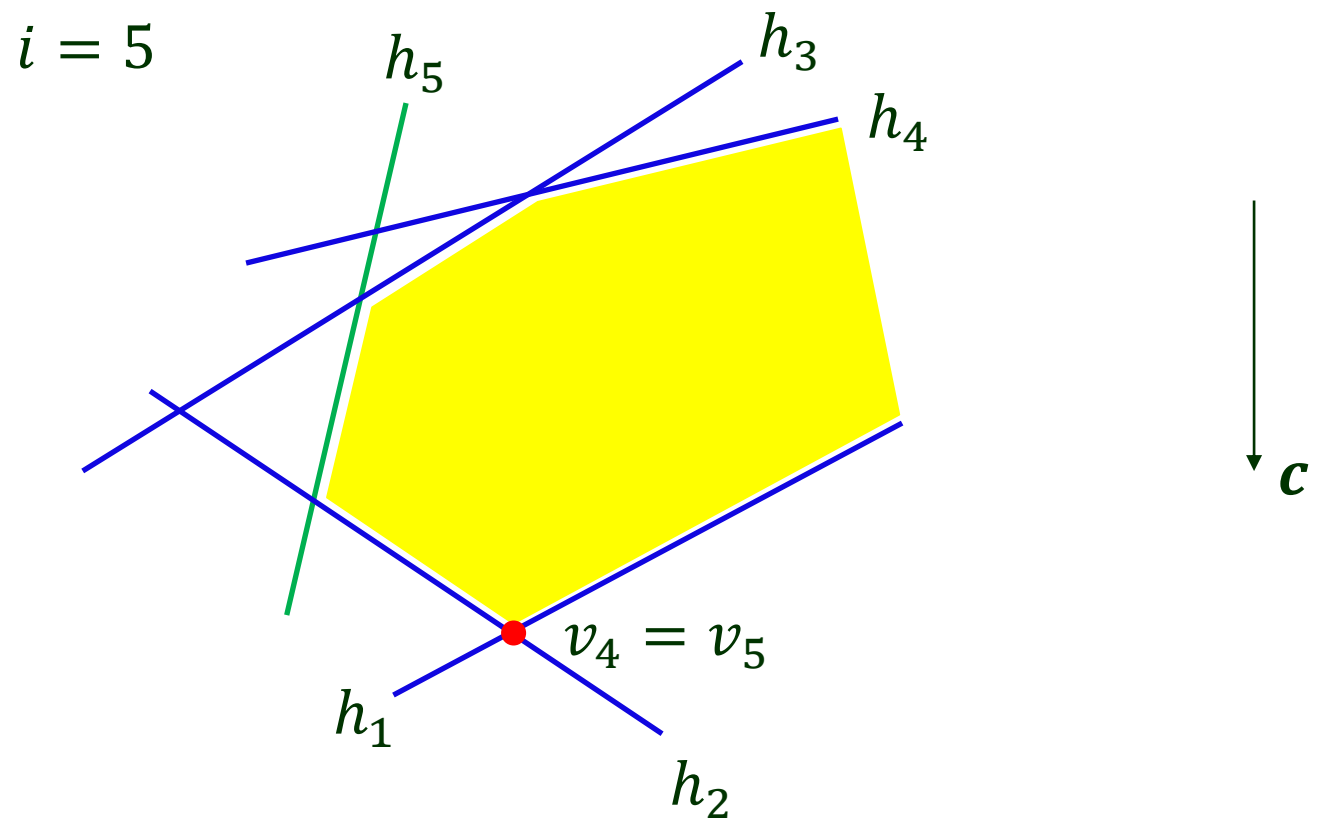
i) No Change of the Optimal Vertex

v_i: optimal vertex of R_i .



i) No Change of the Optimal Vertex

v_i: optimal vertex of R_i .

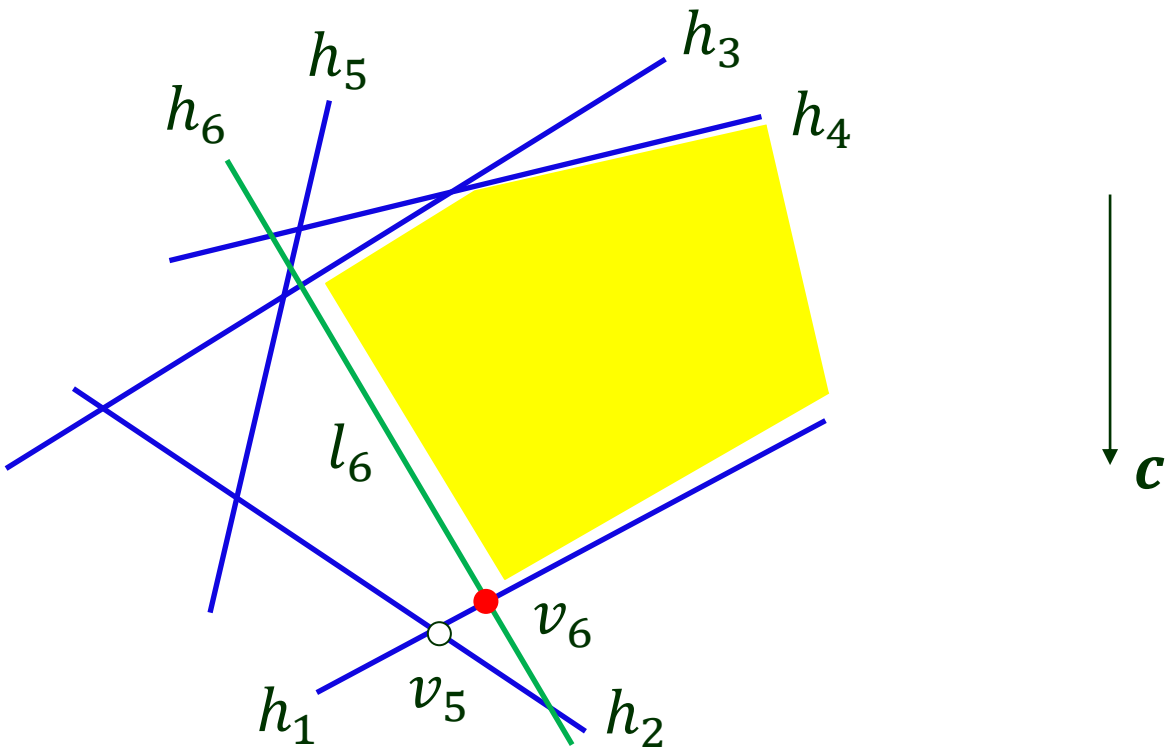


$v_4 \in h_5 \Rightarrow$ No change

ii) Change of the Optimal Vertex

v_i: optimal vertex of $R_i = h_1 \cap h_2 \cap \cdots \cap h_i$.

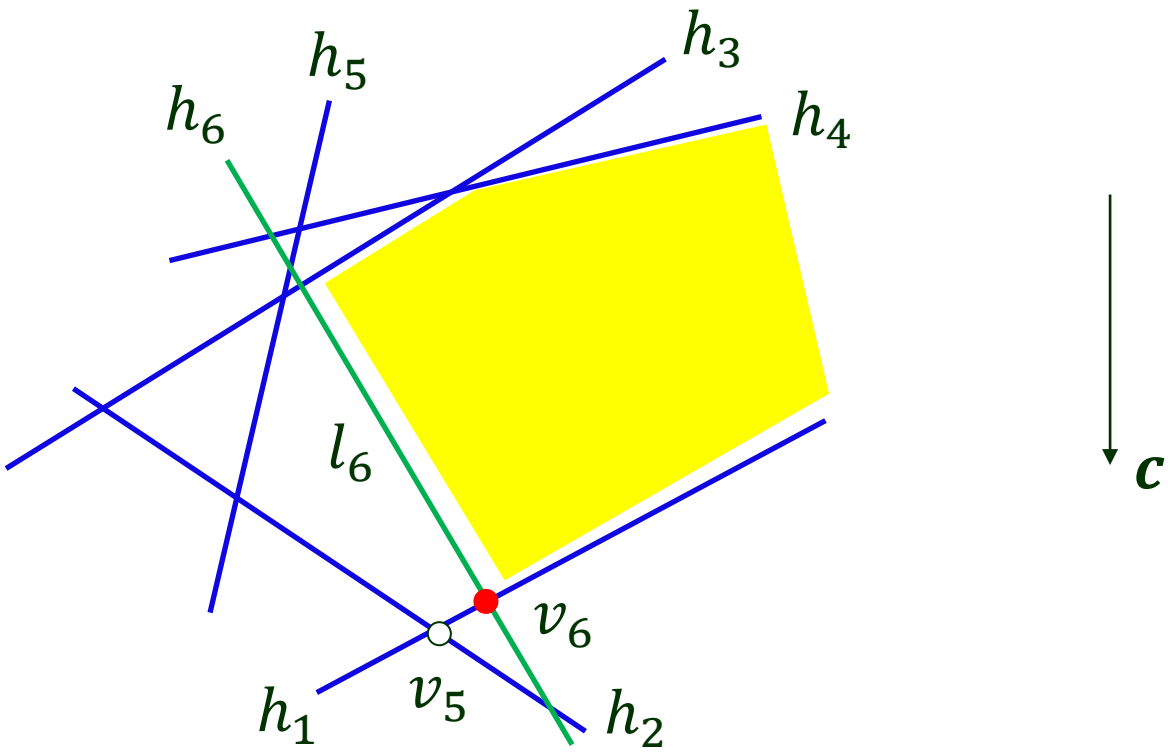
$i = 6$



ii) Change of the Optimal Vertex

v_i: optimal vertex of $R_i = h_1 \cap h_2 \cap \cdots \cap h_i$.

$i = 6$



$v_5 \notin h_6 \Rightarrow$ new optimal vertex $v_6 \in l_6$ (bounding line of h_6)

Evolution of the Optimal Vertex

Lemma i) $v_{i-1} \in h_i \Rightarrow v_i = v_{i-1}$

ii) $v_{i-1} \notin h_i \Rightarrow$ either $R_i = \varnothing$ or $v_i \in l_i$

Proof i) Suppose $v_{i-1} \in h_i$.

Evolution of the Optimal Vertex

Lemma i) $v_{i-1} \in h_i \Rightarrow v_i = v_{i-1}$

ii) $v_{i-1} \notin h_i \Rightarrow$ either $R_i = \varnothing$ or $v_i \in l_i$

Proof i) Suppose $v_{i-1} \in h_i$.

$R_i = R_{i-1} \cap h_i$

$v_{i-1} \in h_i$

Evolution of the Optimal Vertex

Lemma i) $v_{i-1} \in h_i \implies v_i = v_{i-1}$

ii) $v_{i-1} \notin h_i \implies$ either $R_i = \emptyset$ or $v_i \in l_i$

Proof i) Suppose $v_{i-1} \in h_i$.

$$\left.\begin{array}{l} R_i = R_{i-1} \cap h_i \\ v_{i-1} \in h_i \end{array}\right\} \implies v_{i-1} \in R_i \subseteq R_{i-1}$$

Evolution of the Optimal Vertex

Lemma i) $v_{i-1} \in h_i \implies v_i = v_{i-1}$

ii) $v_{i-1} \notin h_i \implies$ either $R_i = \emptyset$ or $v_i \in l_i$

Proof i) Suppose $v_{i-1} \in h_i$.

$$\left.\begin{array}{l} R_i = R_{i-1} \cap h_i \\ v_{i-1} \in h_i \end{array}\right\} \implies v_{i-1} \in R_i \subseteq R_{i-1}$$

$\implies v_{i-1}$ is the optimal vertex in R_i as well.

Evolution of the Optimal Vertex

Lemma i) $v_{i-1} \in h_i \Rightarrow v_i = v_{i-1}$

ii) $v_{i-1} \notin h_i \Rightarrow$ either $R_i = \emptyset$ or $v_i \in l_i$

Proof i) Suppose $v_{i-1} \in h_i$.

$$\left.\begin{array}{l} R_i = R_{i-1} \cap h_i \\ v_{i-1} \in h_i \end{array}\right\} \Rightarrow v_{i-1} \in R_i \subseteq R_{i-1}$$

$\Rightarrow v_{i-1}$ is the optimal vertex in R_i as well.

$\Rightarrow v_i = v_{i-1}$

Proof (cont'd)

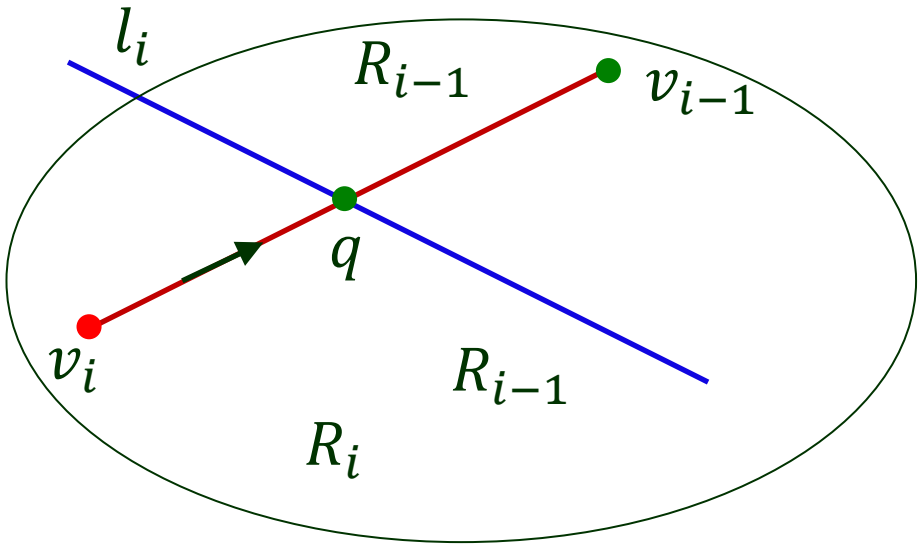
ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.

Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

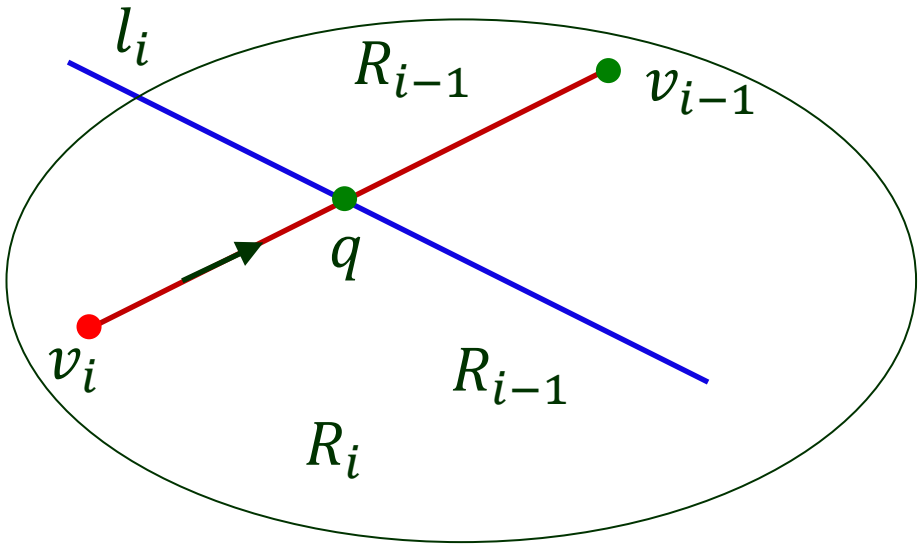
By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



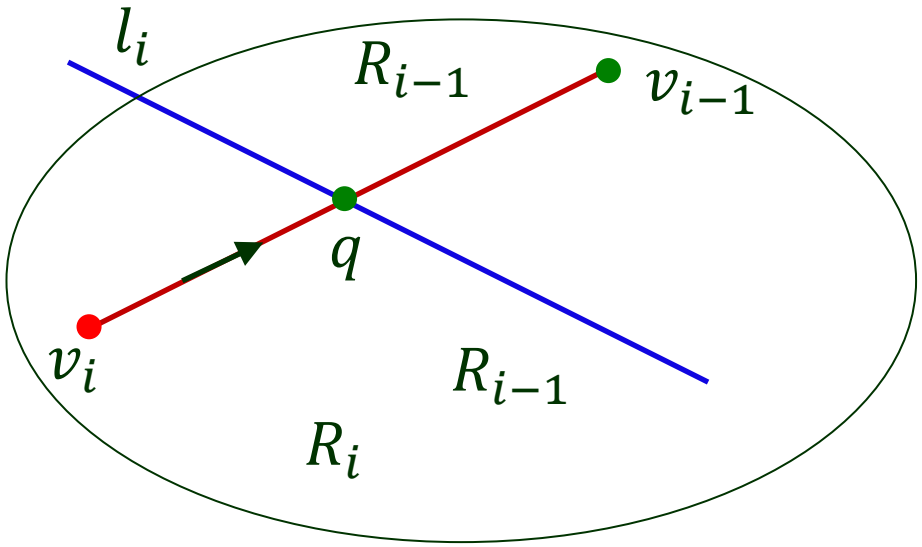
$v_{i-1} \in R_{i-1}$

$v_i \in R_i \subseteq R_{i-1}$

Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



$v_{i-1} \in R_{i-1}$

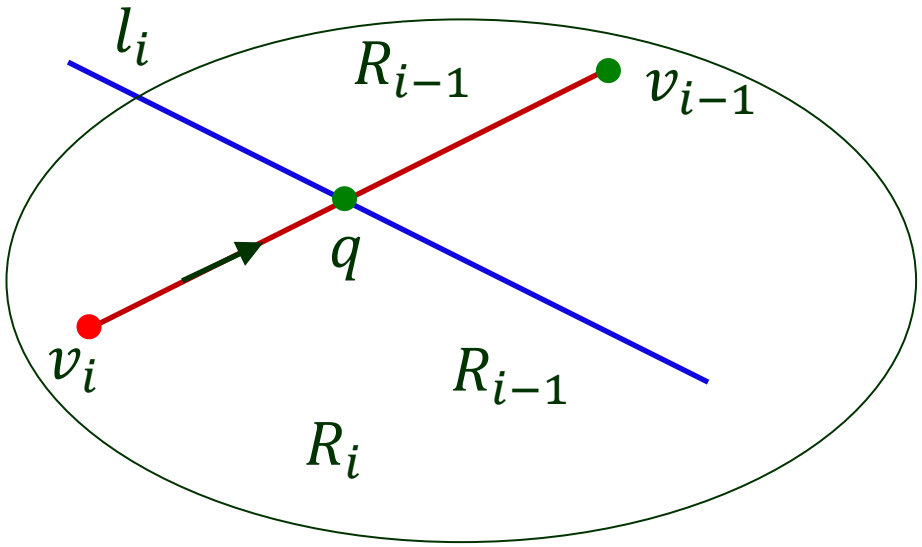
$v_i \in R_i \subseteq R_{i-1}$

Convexity of R_{i-1}

Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



$v_{i-1} \in R_{i-1}$

$v_i \in R_i \subseteq R_{i-1}$

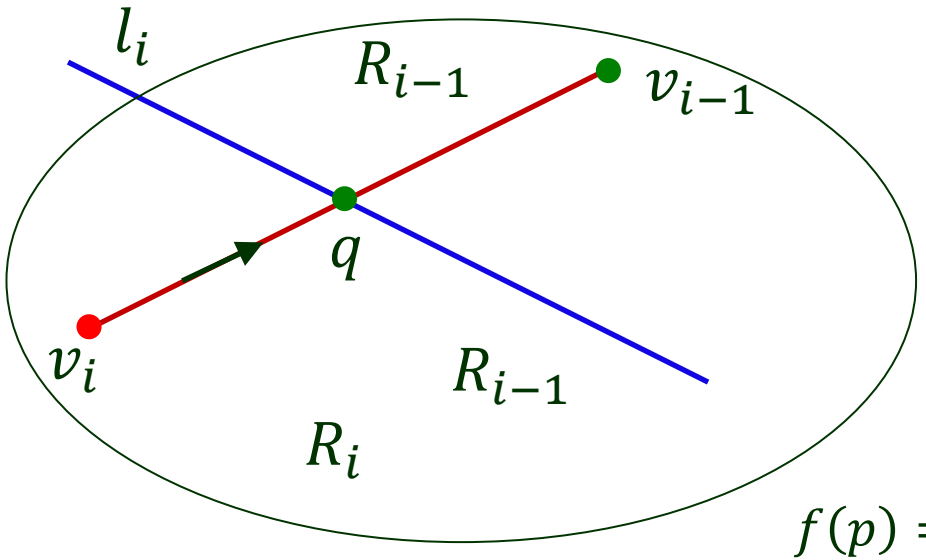
⇩ Convexity of R_{i-1}

line segment $\overline{v_{i-1}v_i} \subseteq R_{i-1}$

Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



$v_{i-1} \in R_{i-1}$

$v_i \in R_i \subseteq R_{i-1}$

⇩ Convexity of R_{i-1}

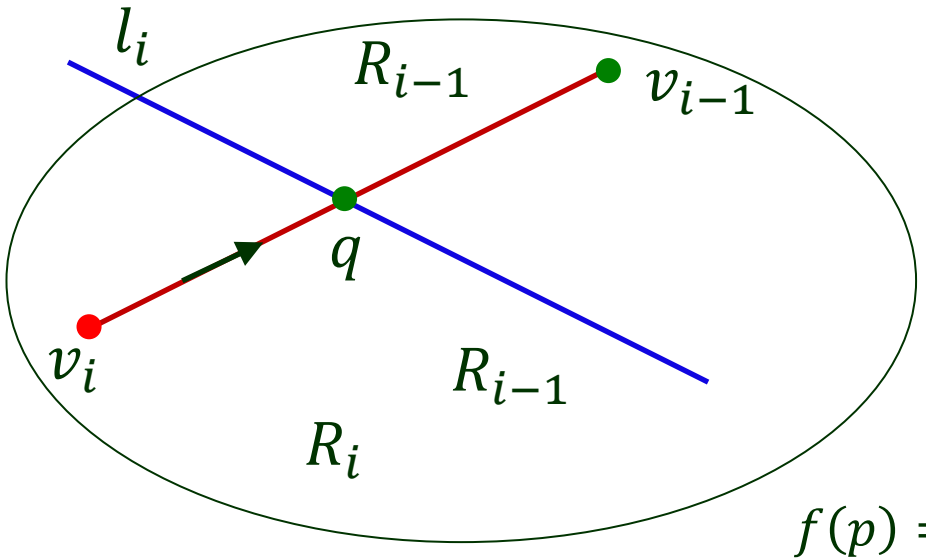
line segment $\overline{v_{i-1}v_i} \subseteq R_{i-1}$

$f(p) = \boldsymbol{c}\, p^T$ is a linear function.

Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



$v_{i-1} \in R_{i-1}$

$v_i \in R_i \subseteq R_{i-1}$

⇩ Convexity of R_{i-1}

line segment $\overline{v_{i-1}v_i} \subseteq R_{i-1}$

$f(p) = \boldsymbol{c}\, p^T$ is a linear function.

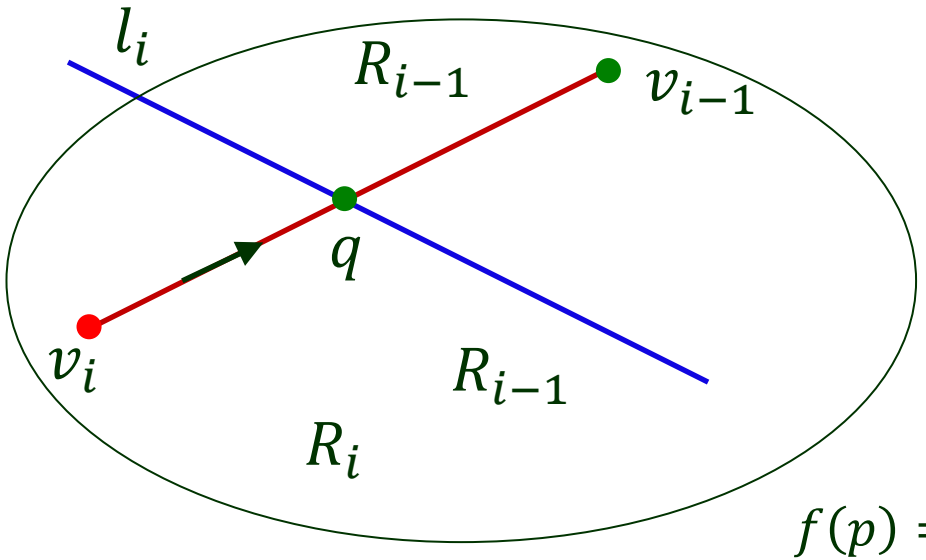
⇩

As p moves on $\overline{v_{i-1}v_i}$ from v_i to v_{i-1}, $f(p)$ increases monotonically.

Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



$v_{i-1} \in R_{i-1}$

$v_i \in R_i \subseteq R_{i-1}$

⇩ Convexity of R_{i-1}

line segment $\overline{v_{i-1}v_i} \subseteq R_{i-1}$

$f(p) = \boldsymbol{c}\, p^T$ is a linear function.

⇩

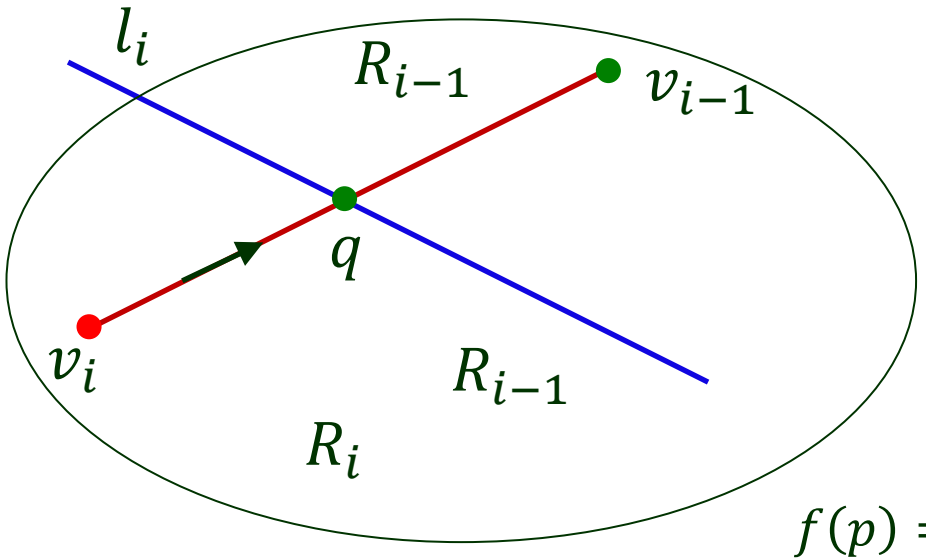
As p moves on $\overline{v_{i-1}v_i}$ from v_i to v_{i-1}, $f(p)$ increases monotonically.

At the intersection $q = l_i \cap \overline{v_{i-1}v_i} \in R_i$, we have $f(q) > f(v_i)$.

Proof (cont'd)

ii) Suppose $v_{i-1} \notin h_i$.

By contradiction. Suppose $R_i \neq \emptyset$ and $v_i \notin l_i$.



$v_{i-1} \in R_{i-1}$

$v_i \in R_i \subseteq R_{i-1}$

⇩ Convexity of R_{i-1}

line segment $\overline{v_{i-1}v_i} \subseteq R_{i-1}$

$f(p) = \boldsymbol{c}\, p^T$ is a linear function.

⇩

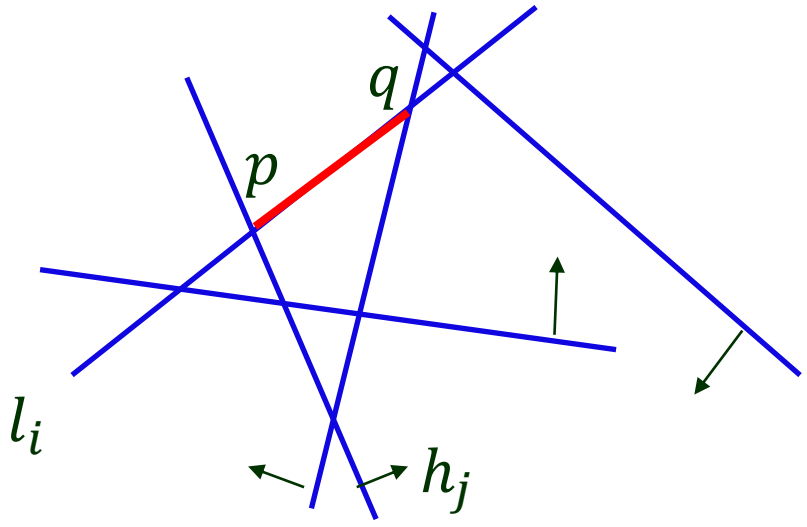
As p moves on $\overline{v_{i-1}v_i}$ from v_i to v_{i-1}, $f(p)$ increases monotonically.

At the intersection $q = l_i \cap \overline{v_{i-1}v_i} \in R_i$, we have $f(q) > f(v_i)$. Contradiction!

□

Updating the Optimal Vertex

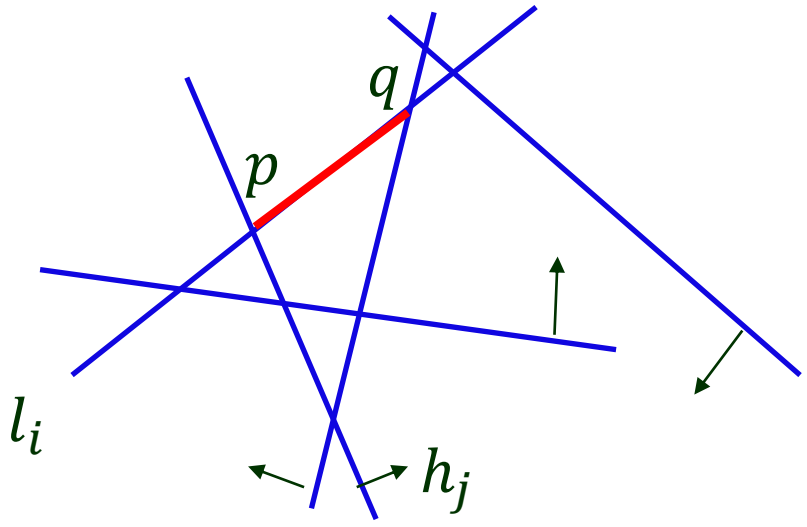
How to find v_i if $v_{i-1} \notin h_i$?



Updating the Optimal Vertex

How to find v_i if $v_{i-1} \notin h_i$?

Easy, it lies on l_i.

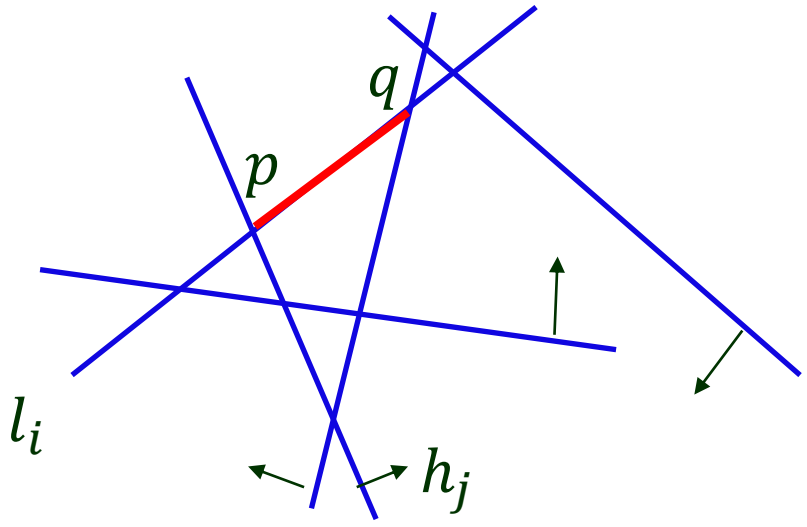


$h_j \cap l_i$, $1 \leq j < i$, is a ray.

Updating the Optimal Vertex

How to find v_i if $v_{i-1} \notin h_i$?

Easy, it lies on l_i.



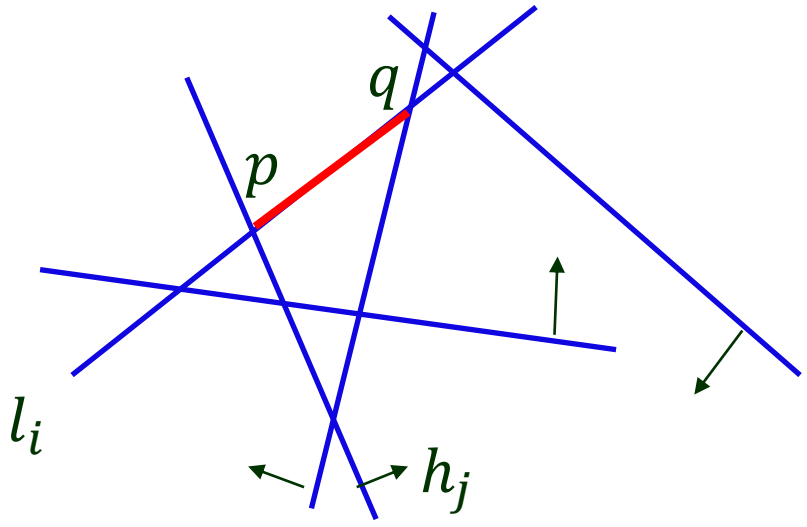
$h_j \cap l_i$, $1 \leq j < i$, is a ray.

Three outcomes of intersecting these rays:

Updating the Optimal Vertex

How to find v_i if $v_{i-1} \notin h_i$?

Easy, it lies on l_i.



$h_j \cap l_i$, $1 \leq j < i$, is a ray.

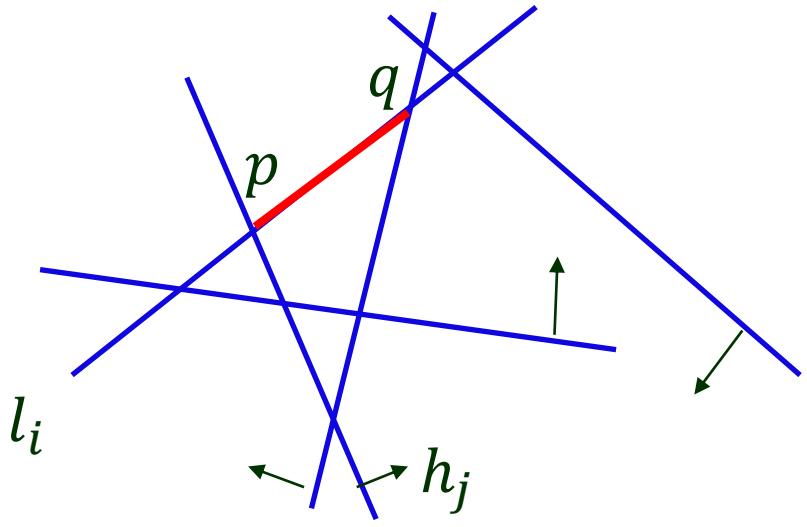
Three outcomes of intersecting these rays:

i) a line segment $\overline{pq}$, either $v_i = p$ or $v_i = q$.

Updating the Optimal Vertex

How to find v_i if $v_{i-1} \notin h_i$?

Easy, it lies on l_i.



$h_j \cap l_i$, $1 \leq j < i$, is a ray.

Three outcomes of intersecting these rays:

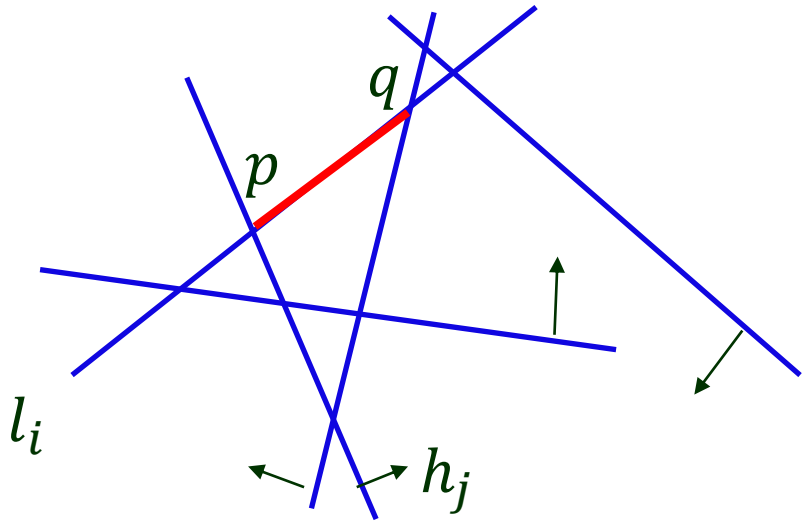
i) a line segment $\overline{pq}$, either $v_i = p$ or $v_i = q$.

ii) $\varnothing$ ⇨ infeasible

Updating the Optimal Vertex

How to find v_i if $v_{i-1} \notin h_i$?

Easy, it lies on l_i.



$h_j \cap l_i$, $1 \leq j < i$, is a ray.

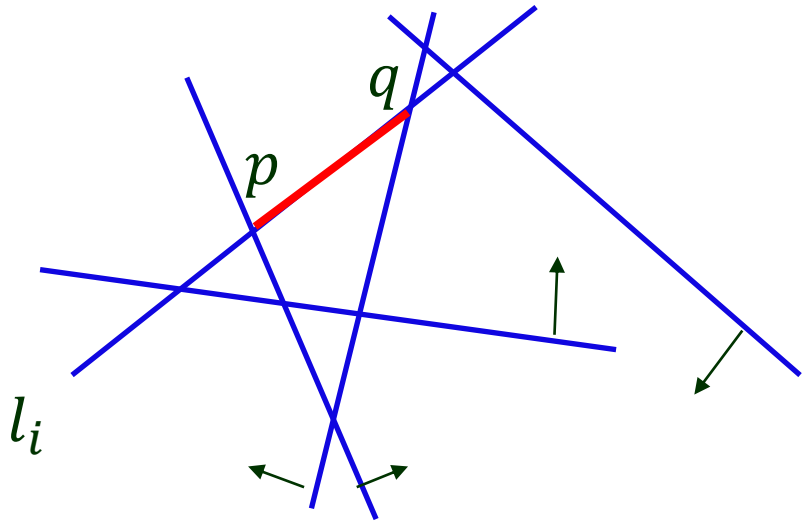
Three outcomes of intersecting these rays:

i) a line segment $\overline{pq}$, either $v_i = p$ or $v_i = q$.

ii) $\varnothing$ ⇨ infeasible

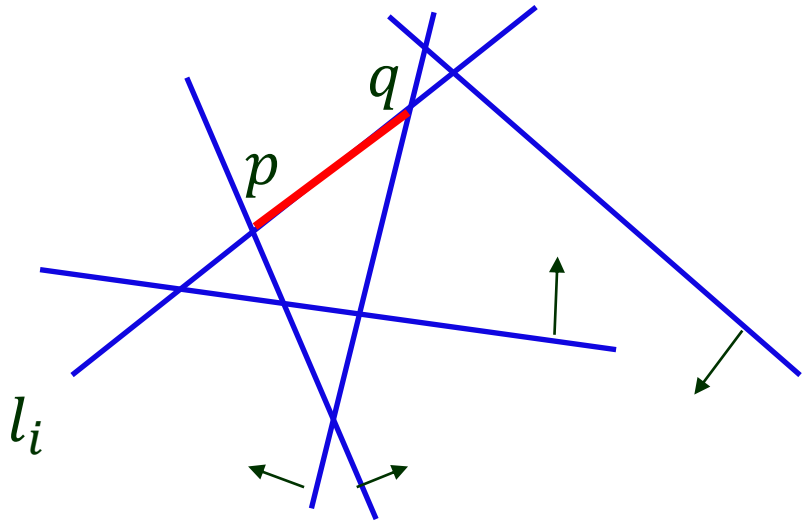
iii) a line ray, either the vertex is v_i or lies at infinity along the ray.

Time of Update



Iterate over half-planes $h_1, h_2, \ldots, h_{i-1}$ to intersect with l_i.

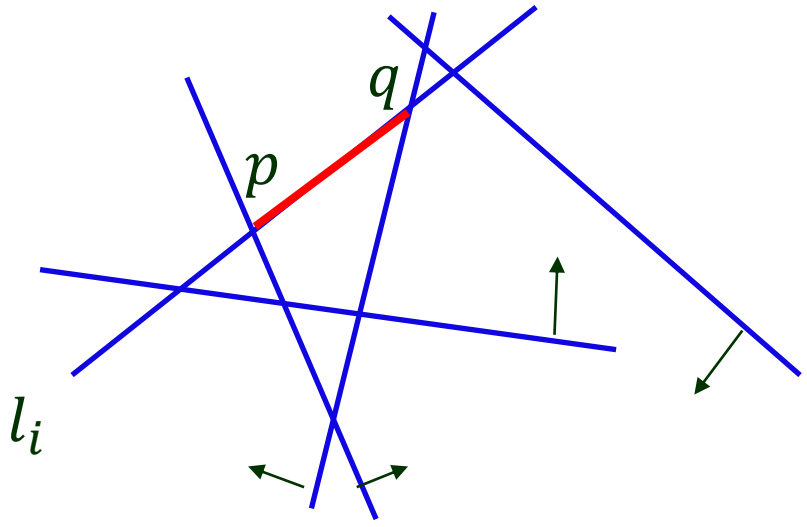
Time of Update



Iterate over half-planes $h_1, h_2, \ldots, h_{i-1}$ to intersect with l_i.

$O(i)$

Time of Update



Iterate over half-planes $h_1, h_2, \ldots, h_{i-1}$ to intersect with l_i.

$O(i)$

It may end earlier once the intersection becomes empty.

The Incremental Algorithm

```
if UNBOUNDEDLP reports that LP is infeasible
   then return
   else if LP is unbounded  // to be discussed
           then return a ray along which it is unbounded
           else h_1, h_2 ← certificate half-planes
                v_2 ← l_1 ∩ l_2
                for i ← 3 to n
                   do if v_{i-1} ∈ h_i
                         then v_i ← v_{i-1}
                         else v_i ← point p on l_i that maximizes f
                                    subject to h_1, h_2, …, h_{i-1}  // O(i)
                              if p does not exist
                                 then LP is infeasible
                return v_n // optimal solution
```

Correctness, Run Time & Storage

Theorem The algorithm solves an LP with n constraints and 2 variables in $O(n^2)$ time and $O(n)$ storage.

Correctness, Run Time & Storage

Theorem The algorithm solves an LP with n constraints and 2 variables in $O(n^2)$ time and $O(n)$ storage.

Correctness by induction – after every stage,

Invariant: The point v_i is the optimal point for R_i.

Correctness, Run Time & Storage

Theorem The algorithm solves an LP with n constraints and 2 variables in $O(n^2)$ time and $O(n)$ storage.

Correctness by induction – after every stage,

Invariant: The point v_i is the optimal point for R_i.

Stage i of adding v_i takes $O(i)$ time.

- dominated by finding the line segment $l_i \cap (\cap_{1 \le j < i} h_j)$

Correctness, Run Time & Storage

Theorem The algorithm solves an LP with n constraints and 2 variables in $O(n^2)$ time and $O(n)$ storage.

Correctness by induction – after every stage,

Invariant: The point v_i is the optimal point for R_i.

Stage i of adding v_i takes $O(i)$ time.

- dominated by finding the line segment $l_i \cap (\cap_{1 \le j < i} h_j)$

Total running time $\sum_{i=3}^{n} O(i) = O(n^2)$

Worst Case

Stage i takes $\Theta(i)$ time only when $v_{i-1} \notin h_i$.

$O(1)$ time when $v_{i-1} \in h_i$.

Worst Case

Stage i takes $\Theta(i)$ time only when $v_{i-1} \notin h_i$.

$O(1)$ time when $v_{i-1} \in h_i$.

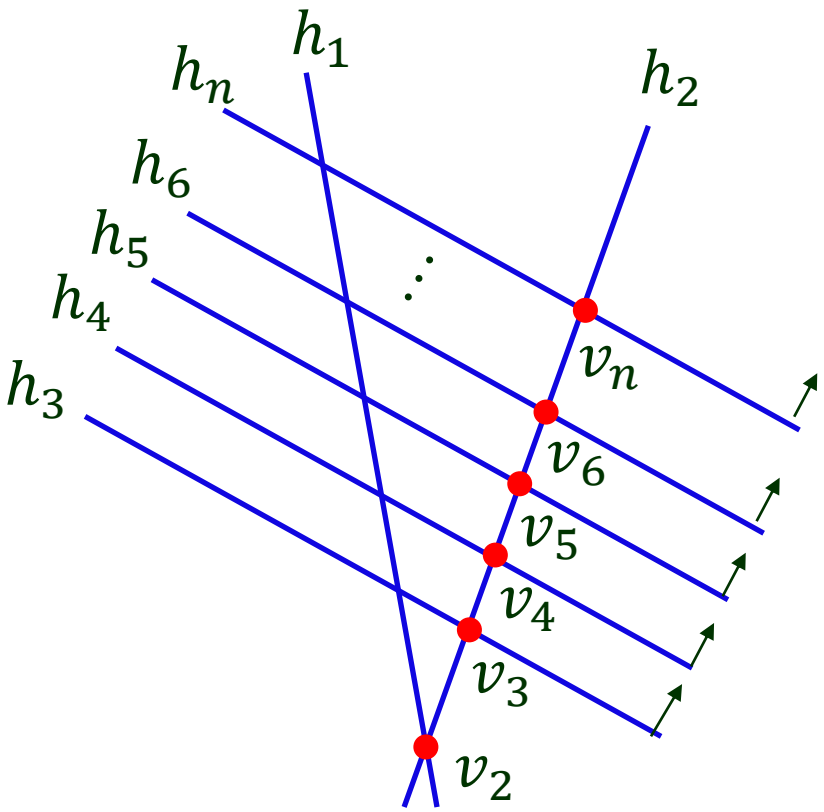
Worst case can happen that the optimal vertex changes $n - 2$ times.

Worst Case

Stage i takes $\Theta(i)$ time only when $v_{i-1} \notin h_i$.

$O(1)$ time when $v_{i-1} \in h_i$.

Worst case can happen that the optimal vertex changes $n - 2$ times.

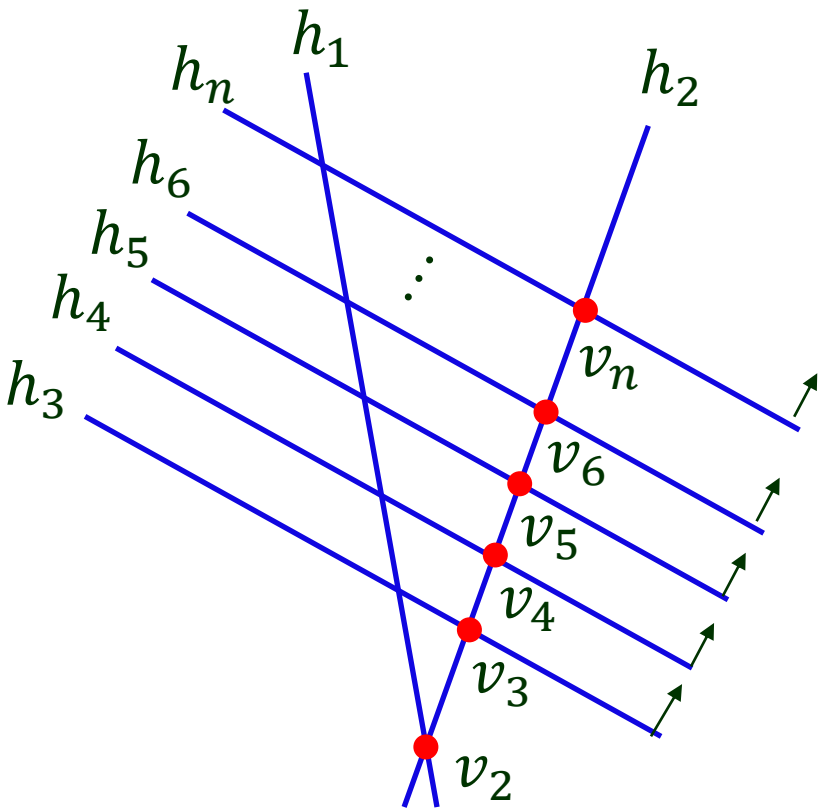


Worst Case

Stage i takes $\Theta(i)$ time only when $v_{i-1} \notin h_i$.

$O(1)$ time when $v_{i-1} \in h_i$.

Worst case can happen that the optimal vertex changes $n-2$ times.



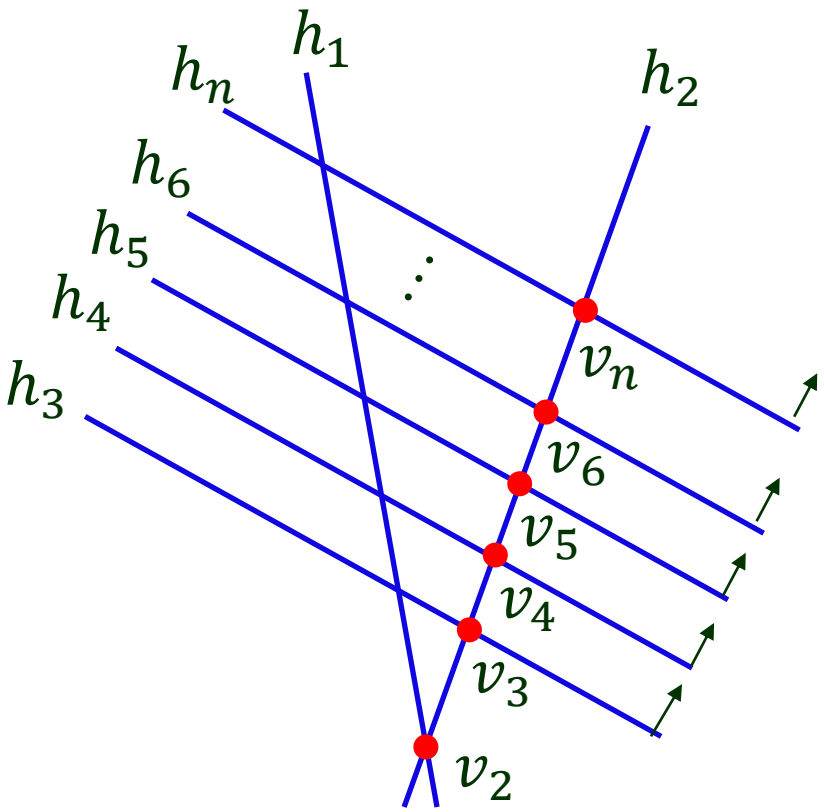
Optimal vertex: $v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_n$

Worst Case

Stage i takes $\Theta(i)$ time only when $v_{i-1} \notin h_i$.

$O(1)$ time when $v_{i-1} \in h_i$.

Worst case can happen that the optimal vertex changes $n-2$ times.



Optimal vertex: $v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_n$

$\Theta(n^2)$ time!

Best Case

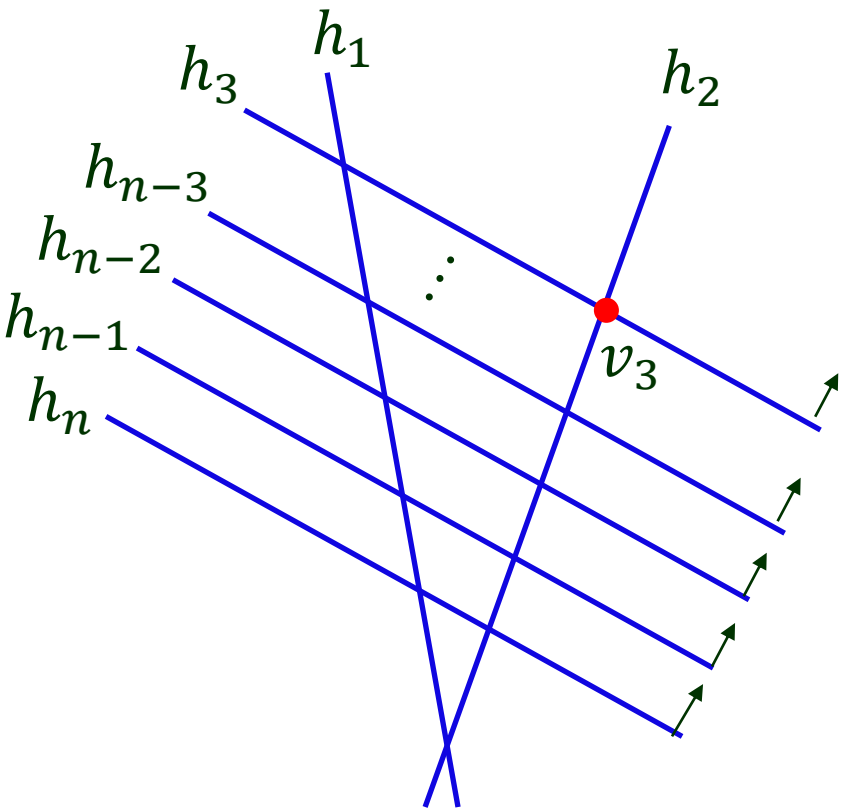
How about adding them in the order $h_n, h_{n-1}, \dots, h_3$?

Namely, change their indices: $n \to 3, n-1 \to 4, \dots, 3 \to n$.

No change of the optimal vertex!

$$R_3 = R_4 = \cdots = R_n$$

$$v_3 = v_4 = \cdots = v_n$$



Best Case

How about adding them in the order $h_n, h_{n-1}, \dots, h_3$?

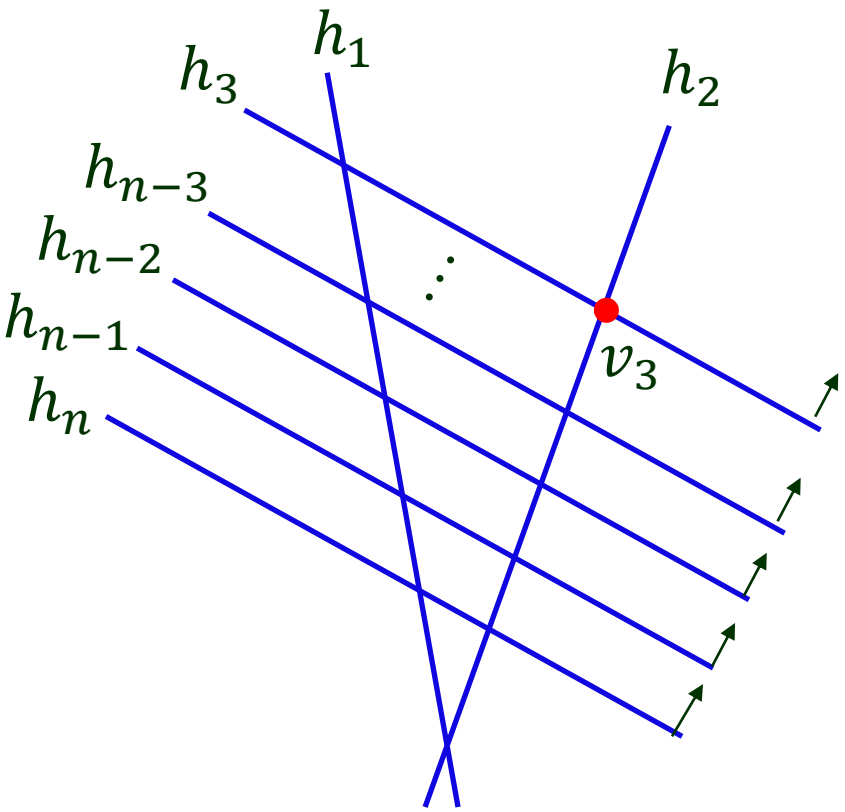
Namely, change their indices: $n \to 3, n-1 \to 4, \dots, 3 \to n$.

No change of the optimal vertex!

$$R_3 = R_4 = \cdots = R_n$$

$$v_3 = v_4 = \cdots = v_n$$

$O(n)$

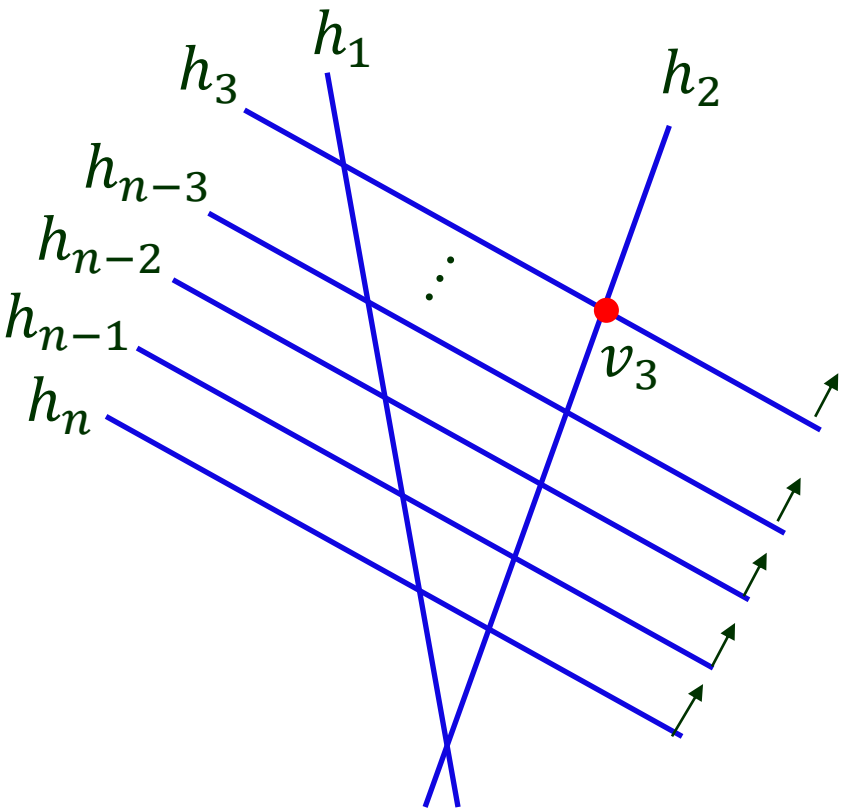


Best Case

How about adding them in the order $h_n, h_{n-1}, \ldots, h_3$?

Namely, change their indices: $n \to 3, n-1 \to 4, \ldots, 3 \to n$.

No change of the optimal vertex!



$$R_3 = R_4 = \cdots = R_n$$

$$v_3 = v_4 = \cdots = v_n$$

$O(n)$

Any set of half-planes has such a good order, but there is no easy way to find it.

Our approach: randomization.