Randomized Linear Programming

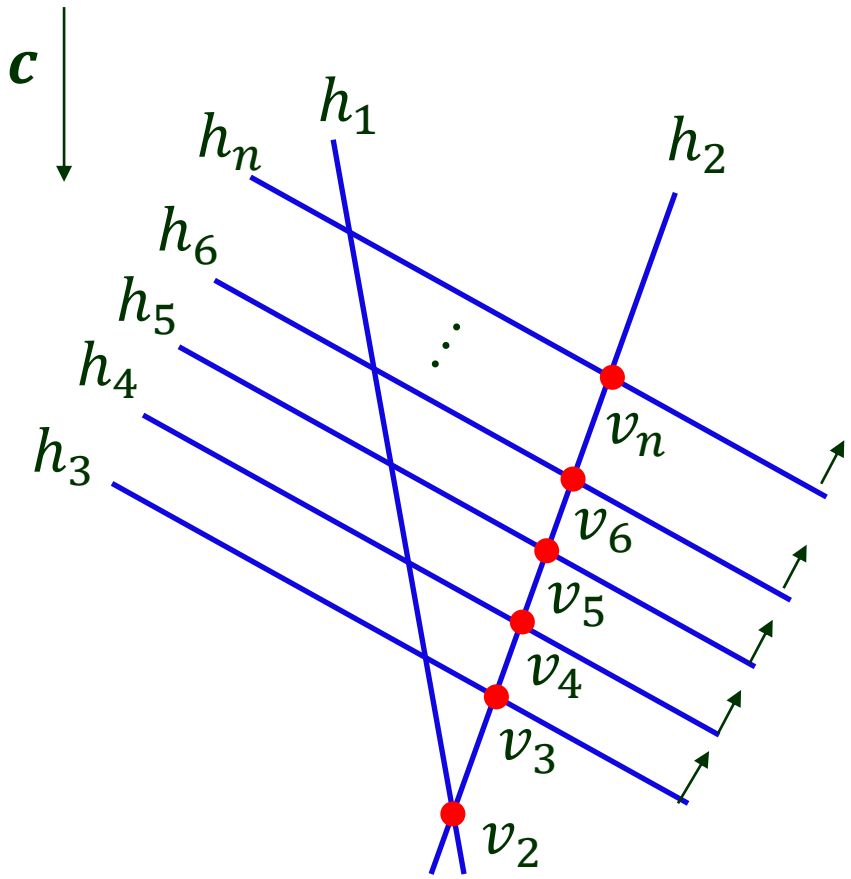
Outline:

I. Generation of a random permutation

II. *Backward analysis* of running time

III. Unbounded linear program

How to Avoid Worst Case?



Pick a *random* ordering of the set H of half-planes.

Bad order still leads to $O(n^2)$.

Most orders lead to a faster algorithm.

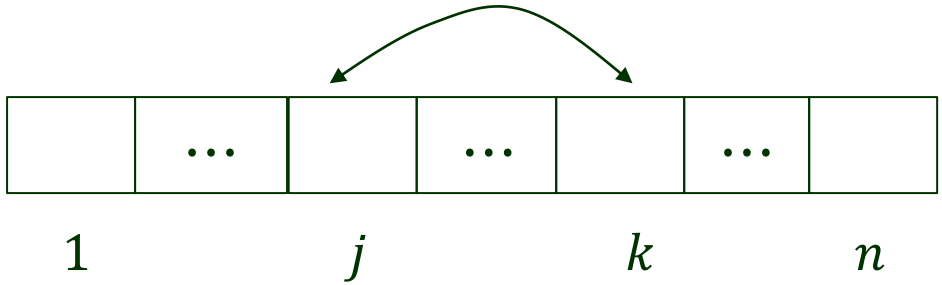
Algorithm Modification

⋮

$h_1, h_2 \leftarrow$ certificate half-planes
$v_2 \leftarrow l_1 \cap l_2$
generate a random permutation of $h_3, h_4, \ldots, h_n$
for $i \leftarrow 3$ to n
    do if $v_{i-1} \in h_i$
        then $v_i \leftarrow v_{i-1}$
        else $v_i \leftarrow$ point p on l_i that maximizes f
                subject to $h_1, h_2, \ldots, h_{i-1}$
            if p does not exist
                then LP is infeasible
return v_n

I. Random Permutation

Array $A[1..n]$



for $k \leftarrow n$ down to 2

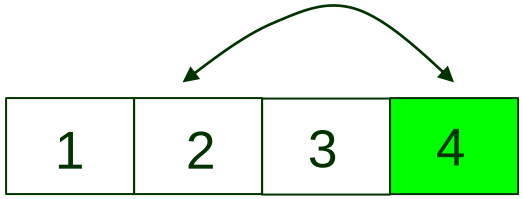
do $j \leftarrow$RANDOM(k)

$A[k] \leftrightarrow A[j]$

// returns a random integer between
//1 and k under uniform distribution.

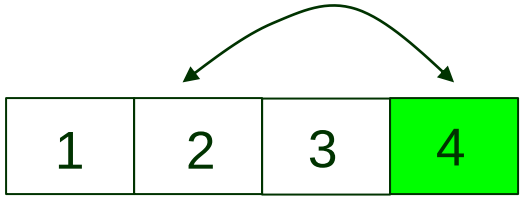
$O(n)$

How Does It Work?

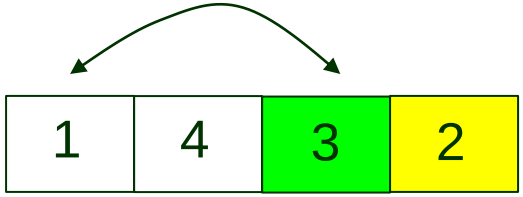


$\text{RANDOM}(4) = 2$

How Does It Work?

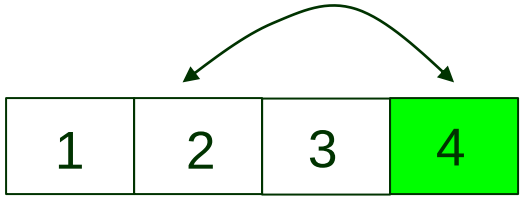


RANDOM(4) = 2

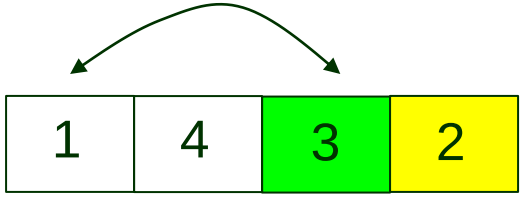


RANDOM(3) =1

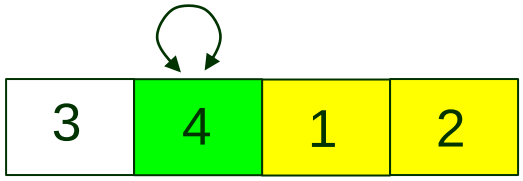
How Does It Work?



RANDOM(4) = 2

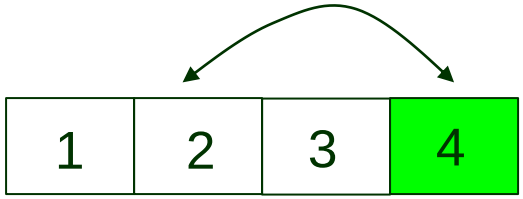


RANDOM(3) =1

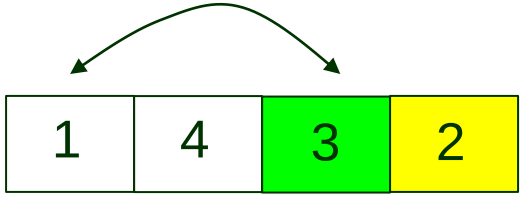


RANDOM(2) = 2

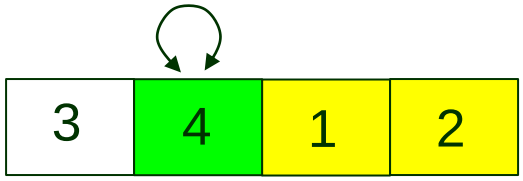
How Does It Work?



RANDOM(4) = 2

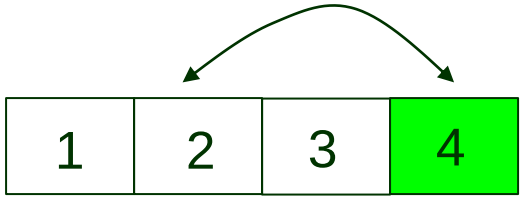


RANDOM(3) =1

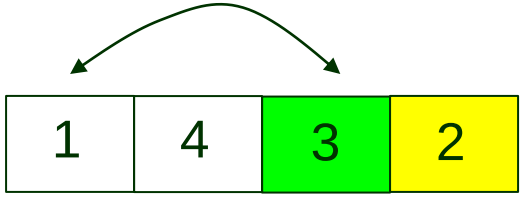


RANDOM(2) = 2

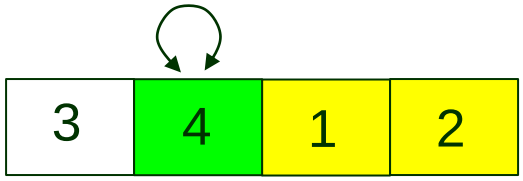
How Does It Work?



RANDOM$(4) = 2$



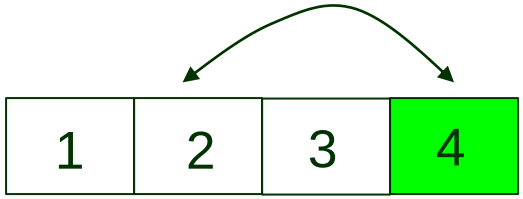
RANDOM$(3) = 1$



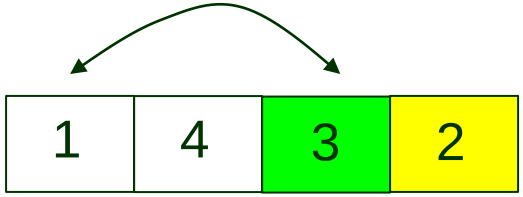
RANDOM$(2) = 2$

Every permutation is equally likely to be generated.

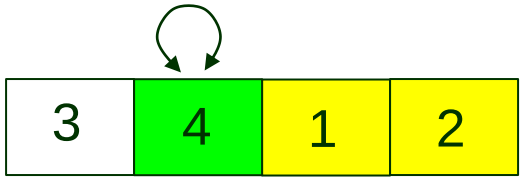
How Does It Work?



RANDOM$(4) = 2$



RANDOM$(3) = 1$



RANDOM$(2) = 2$

Every permutation is equally likely to be generated.

Why?

II. Run Time Analysis

The LP algorithm is randomized.

Its running time depends on random choices.

II. Run Time Analysis

The LP algorithm is randomized.

Its running time depends on random choices.

How to analyze the running time?

II. Run Time Analysis

The LP algorithm is randomized.

Its running time depends on random choices.

How to analyze the running time?

- ♣ $(n-2)!$ permutations of $h_3, h_4, \ldots, h_n$

II. Run Time Analysis

The LP algorithm is randomized.

Its running time depends on random choices.

How to analyze the running time?

- ♣ $(n-2)!$ permutations of $h_3, h_4, \ldots, h_n$
- ♣ Average over all of them (expected time).

Start of Analysis

- UNBOUNDEDLP takes time $O(n)$ – to be shown.
- Random permutation takes time $O(n)$.

```
if UNBOUNDEDLP reports that LP is infeasible
  then return // does not detect all infeasible scenarios
  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
        generate a random permutation of h_3, h_4, …, h_n
        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}
                       if p does not exist
                           then LP is infeasible
        return v_n
```

Start of Analysis

- UNBOUNDEDLP takes time $O(n)$ – to be shown.
- Random permutation takes time $O(n)$.
- Add a half-plane.

```
if UNBOUNDEDLP reports that LP is infeasible
  then return // does not detect all infeasible scenarios
  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
        generate a random permutation of h_3, h_4, …, h_n
        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}
                       if p does not exist
                           then LP is infeasible
        return v_n
```

Start of Analysis

- UNBOUNDEDLP takes time $O(n)$ – to be shown.
- Random permutation takes time $O(n)$.
- Add a half-plane.
  - $O(1)$ if the optimal vertex does not change.

```
if UNBOUNDEDLP reports that LP is infeasible
  then return // does not detect all infeasible scenarios
  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
        generate a random permutation of h_3, h_4, …, h_n
        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}
                       if p does not exist
                           then LP is infeasible
        return v_n
```

Start of Analysis

- UNBOUNDEDLP takes time $O(n)$ – to be shown.
- Random permutation takes time $O(n)$.
- Add a half-plane.
  - $O(1)$ if the optimal vertex does not change.
  - $O(i)$ for solving a 1D LP otherwise.

```
if UNBOUNDEDLP reports that LP is infeasible
  then return // does not detect all infeasible scenarios
  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
        generate a random permutation of h_3, h_4, …, h_n
        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}
                       if p does not exist
                           then LP is infeasible
        return v_n
```

Start of Analysis

- UNBOUNDEDLP takes time $O(n)$ – to be shown.
- Random permutation takes time $O(n)$.
- Add a half-plane.
  - $O(1)$ if the optimal vertex does not change.
  - $O(i)$ for solving a 1D LP otherwise.

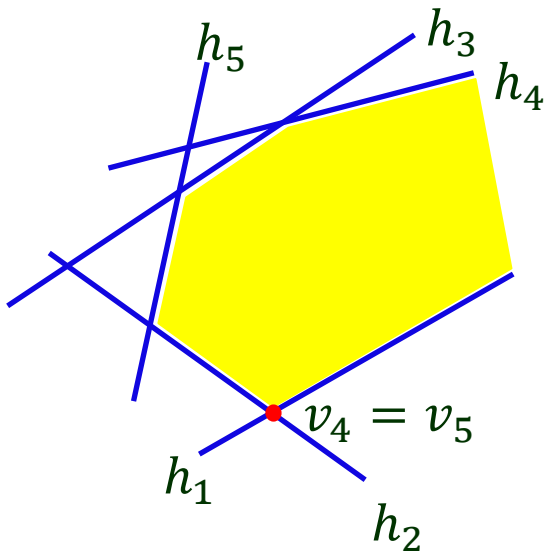
```
if UNBOUNDEDLP reports that LP is infeasible
  then return // does not detect all infeasible scenarios
  else if LP is unbounded  // to be discussed
      then return a ray along which it is unbounded
      else
        h_1, h_2 ← certificate half-planes
           v_2 ← l_1 ∩ l_2
        generate a random permutation of h_3, h_4, …, h_n
        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}
                       if p does not exist
                           then LP is infeasible
        return v_n
```

Bounding the average time for solving all 1D LPs.

Random Variable

For stage i

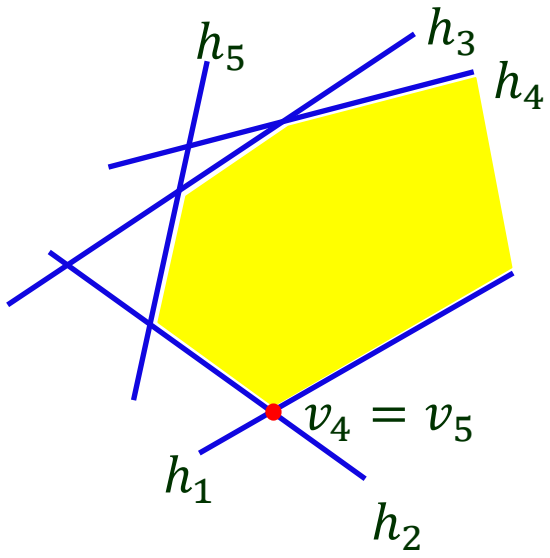
$$X_i = \begin{cases} 0 \text{ if } v_{i-1} \in h_i \\ \\ \\ \\ \end{cases}$$

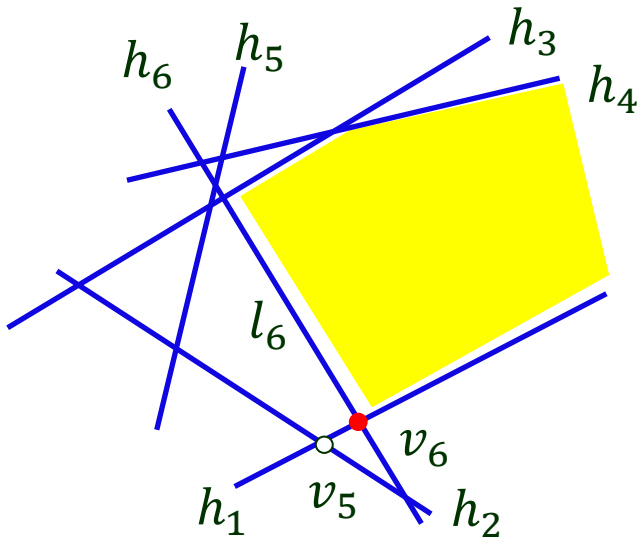


Random Variable

For stage i

$$X_i = \begin{cases} 0 \text{ if } v_{i-1} \in h_i \\ 1 \text{ otherwise} \end{cases}$$





Expected Time

Total time over all half-planes $h_3, h_4, \ldots, h_n$.

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

Expected Time

Total time over all half-planes $h_3, h_4, \ldots, h_n$.

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

Expected total time:

$$E\left(\sum_{i=3}^{n} O(i) \cdot X_i\right) = \sum_{i=3}^{n} O(i) \cdot E(X_i)$$

Expected Time

Total time over all half-planes $h_3, h_4, \ldots, h_n$.

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

Expected total time:

$$E\left(\sum_{i=3}^{n} O(i) \cdot X_i\right) = \sum_{i=3}^{n} O(i) \cdot E(X_i)$$

Probability that $v_{i-1} \notin h_i$

Expected Time

Total time over all half-planes $h_3, h_4, \ldots, h_n$.

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

Expected total time:

How to determine?

$$E\left(\sum_{i=3}^{n} O(i) \cdot X_i\right) = \sum_{i=3}^{n} O(i) \cdot E(X_i)$$

Probability that $v_{i-1} \notin h_i$

Expected Time

Total time over all half-planes $h_3, h_4, \ldots, h_n$.

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

Expected total time:

How to determine?

$$E\left(\sum_{i=3}^{n} O(i) \cdot X_i\right) = \sum_{i=3}^{n} O(i) \cdot E(X_i)$$

Probability that $v_{i-1} \notin h_i$

Are we going to exhaust all (topologically) different scenarios of half-plane additions and then assign each a possibility?

Expected Time

Total time over all half-planes $h_3, h_4, \ldots, h_n$.

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

Expected total time:

How to determine?

$$E\left(\sum_{i=3}^{n} O(i) \cdot X_i\right) = \sum_{i=3}^{n} O(i) \cdot E(X_i)$$

Probability that $v_{i-1} \notin h_i$

Are we going to exhaust all (topologically) different scenarios of half-plane additions and then assign each a possibility?

Too complicated even if possible!

Backward Analysis

Look at the algorithm *backwards*.

Backward Analysis

Look at the algorithm *backwards*.

Assume it has finished with the optimal vertex v_n found.

- v_n is a vertex of $R_n = \bigcap_{i=1}^{n} h_i$
- R_{n-1} is from R_n after removing h_n.

Backward Analysis

Look at the algorithm *backwards*.

Assume it has finished with the optimal vertex v_n found.

- v_n is a vertex of $R_n = \bigcap_{i=1}^{n} h_i$
- R_{n-1} is from R_n after removing h_n.
- Suppose that the optimal point changes at stage n.

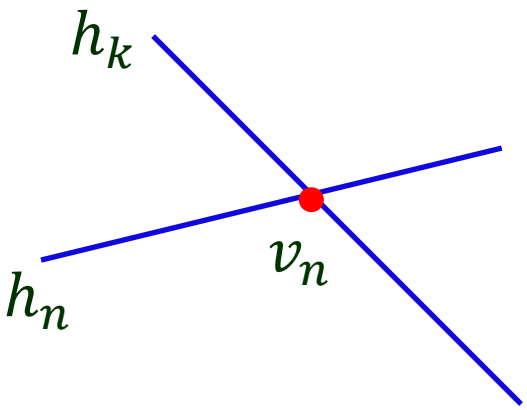
Backward Analysis

Look at the algorithm *backwards*.

Assume it has finished with the optimal vertex v_n found.

- v_n is a vertex of $R_n = \bigcap_{i=1}^{n} h_i$
- R_{n-1} is from R_n after removing h_n.
- Suppose that the optimal point changes at stage n.

$\Longrightarrow$ h_n defines v_n

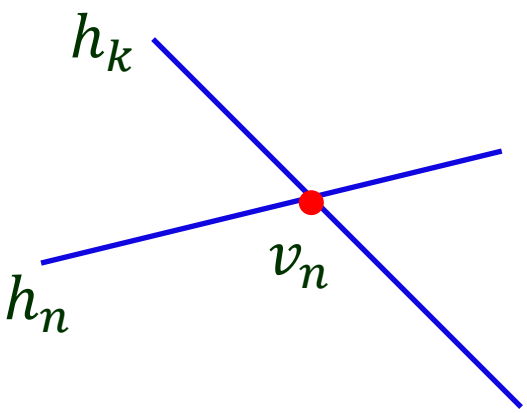


Backward Analysis

Look at the algorithm *backwards*.

Assume it has finished with the optimal vertex v_n found.

- v_n is a vertex of $R_n = \bigcap_{i=1}^{n} h_i$
- R_{n-1} is from R_n after removing h_n.
- Suppose that the optimal point changes at stage n.



$\Longrightarrow$ h_n defines v_n

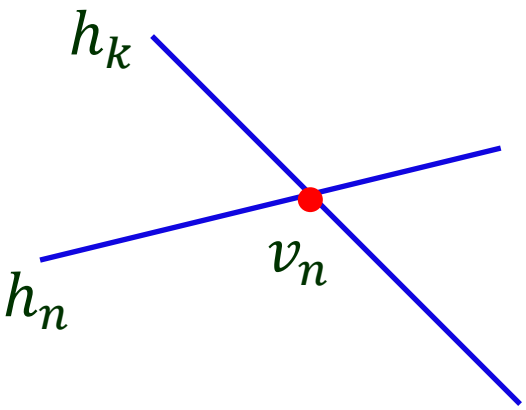
h_n a random element of $\{h_1, h_2, \ldots, h_n\}$

Backward Analysis

Look at the algorithm *backwards*.

Assume it has finished with the optimal vertex v_n found.

- v_n is a vertex of $R_n = \bigcap_{i=1}^{n} h_i$
- R_{n-1} is from R_n after removing h_n.
- Suppose that the optimal point changes at stage n.



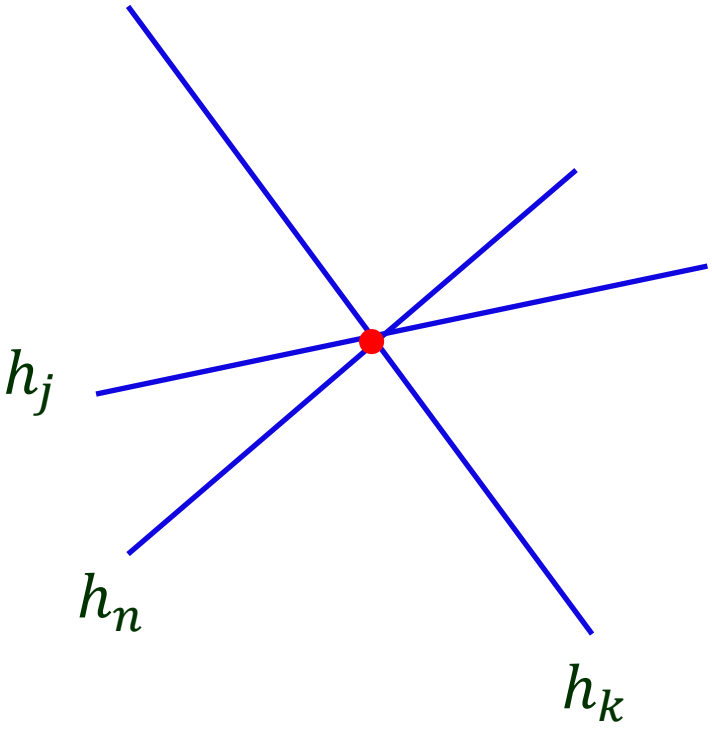
$\Longrightarrow$ h_n defines v_n

h_n a random element of $\{h_1, h_2, \ldots, h_n\}$

$\Longrightarrow$ Pr{h_n is one of v_n's only two defining half-planes}

$$\leq \frac{2}{n}$$

Why $\leq \frac{2}{n}$?



> 2 half-planes may have boundary lines through v_n.

- v_n already existed before addition of h_n.
  ($v_{n-1} \neq v_n$ and $v_{n-1} \notin h_n$)

- removal of h_n does not affect the vertex.

Stage i

To bound $E(X_i)$, we fix the set R_i of first i half-planes.

Think one step backward.

Stage i

To bound $E(X_i)$, we fix the set R_i of first i half-planes.

Think one step backward.

Pr(computing a new optimal vertex when adding h_i)

Stage i

To bound $E(X_i)$, we fix the set R_i of first i half-planes.

Think one step backward.

Pr(computing a new optimal vertex when adding h_i)

$=$ Pr(the optimal vertex changes when removing a half-plane from R_i)

Stage i

To bound $E(X_i)$, we fix the set R_i of first i half-planes.

Think one step backward.

Pr(computing a new optimal vertex when adding h_i)

$= \Pr$(the optimal vertex changes when removing a half-plane from R_i)

$\leq \frac{2}{i}$

Stage i

To bound $E(X_i)$, we fix the set R_i of first i half-planes.

Think one step backward.

Pr(computing a new optimal vertex when adding h_i)

$= \Pr$(the optimal vertex changes when removing a half-plane from R_i)

$\leq \frac{2}{i}$

Thus,

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

Expected Running Time

for solving an LP is

$$\sum_{i=3}^{n} O(i) \cdot \frac{2}{i} = O\left(\sum_{i=3}^{n} \frac{2i}{i}\right)$$

$$= O(n)$$

Expected Running Time

for solving an LP is

$$\sum_{i=3}^{n} O(i) \cdot \frac{2}{i} = O\left(\sum_{i=3}^{n} \frac{2i}{i}\right)$$

$$= O(n)$$

Theorem 2D linear programming with n constraints is solvable in $O(n)$ expected time and $O(n)$ storage.

III. Unbounded Linear Program

UNBOUNDEDLP$(H, \boldsymbol{c})$ yields

- a ray $\in \bigcap_{h \in H} h = R$ if LP is unbounded
- $h_1, h_2 \in H$ such that $(\{h_1, h_2\}, \boldsymbol{c})$ is bounded, otherwise.

III. Unbounded Linear Program

UNBOUNDEDLP$(H, \boldsymbol{c})$ yields

- a ray $\in \bigcap_{h \in H} h = R$ if LP is unbounded
- $h_1, h_2 \in H$ such that $(\{h_1, h_2\}, \boldsymbol{c})$ is bounded, otherwise.

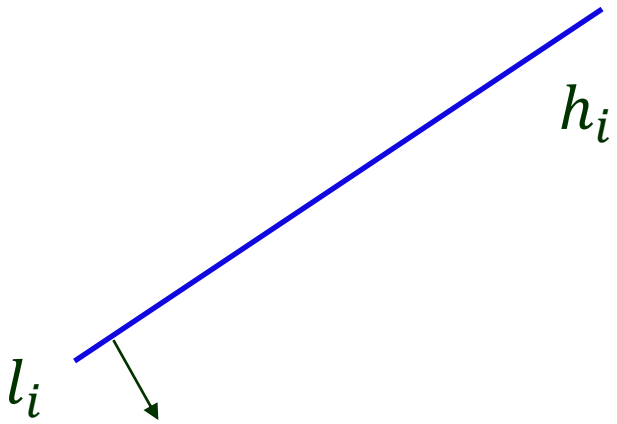
certificate half-plane

III. Unbounded Linear Program

UNBOUNDEDLP$(H, \boldsymbol{c})$ yields

- a ray $\in \bigcap_{h \in H} h = R$ if LP is unbounded
- $h_1, h_2 \in H$ such that $(\{h_1, h_2\}, \boldsymbol{c})$ is bounded, otherwise.

certificate half-plane

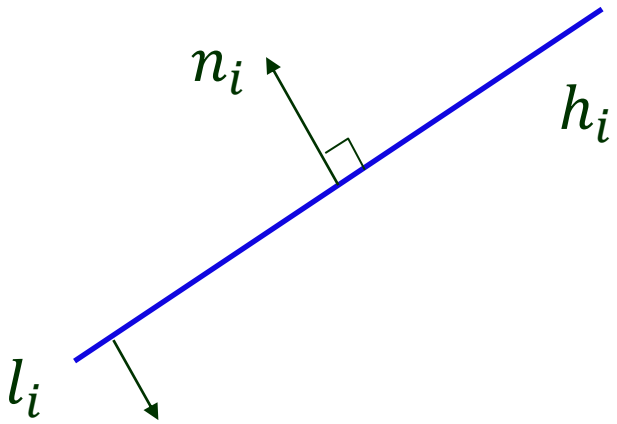


III. Unbounded Linear Program

UNBOUNDEDLP$(H, \boldsymbol{c})$ yields

- a ray $\in \bigcap_{h \in H} h = R$ if LP is unbounded
- $h_1, h_2 \in H$ such that $(\{h_1, h_2\}, \boldsymbol{c})$ is bounded, otherwise.

certificate half-plane



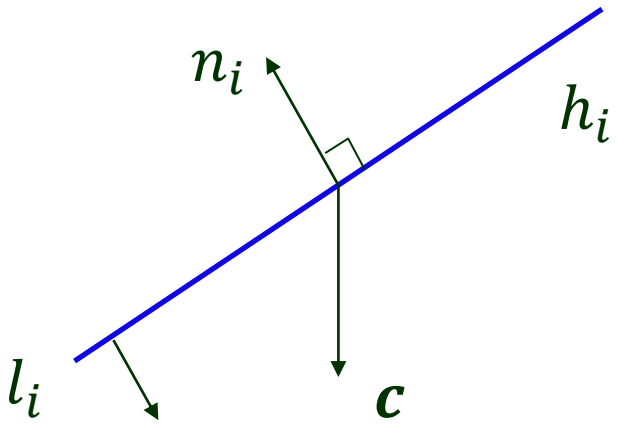
n_i: outward normal

III. Unbounded Linear Program

UNBOUNDEDLP$(H, \boldsymbol{c})$ yields

- a ray $\in \bigcap_{h \in H} h = R$ if LP is unbounded
- $h_1, h_2 \in H$ such that $(\{h_1, h_2\}, \boldsymbol{c})$ is bounded, otherwise.

certificate half-plane



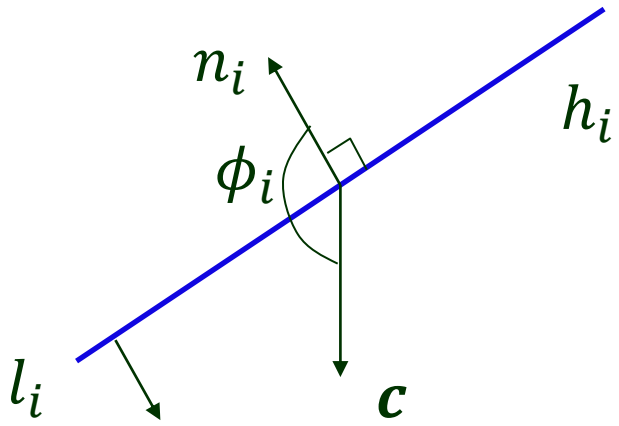
n_i: outward normal

III. Unbounded Linear Program

UNBOUNDEDLP$(H, \boldsymbol{c})$ yields

- a ray $\in \bigcap_{h \in H} h = R$ if LP is unbounded
- $h_1, h_2 \in H$ such that $(\{h_1, h_2\}, \boldsymbol{c})$ is bounded, otherwise.

certificate half-plane



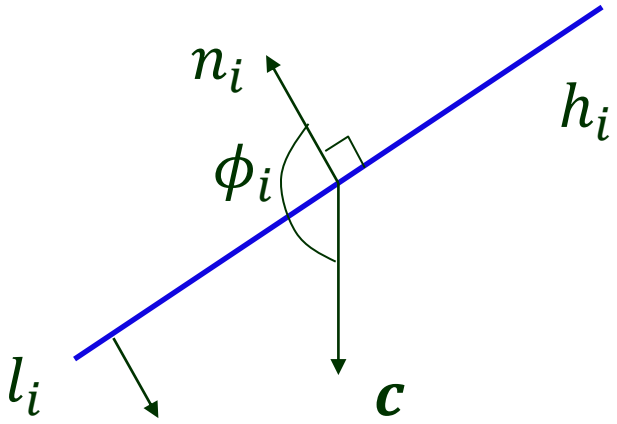
n_i: outward normal

III. Unbounded Linear Program

UNBOUNDEDLP$(H, \boldsymbol{c})$ yields

- a ray $\in \bigcap_{h \in H} h = R$ if LP is unbounded
- $h_1, h_2 \in H$ such that $(\{h_1, h_2\}, \boldsymbol{c})$ is bounded, otherwise.

certificate half-plane

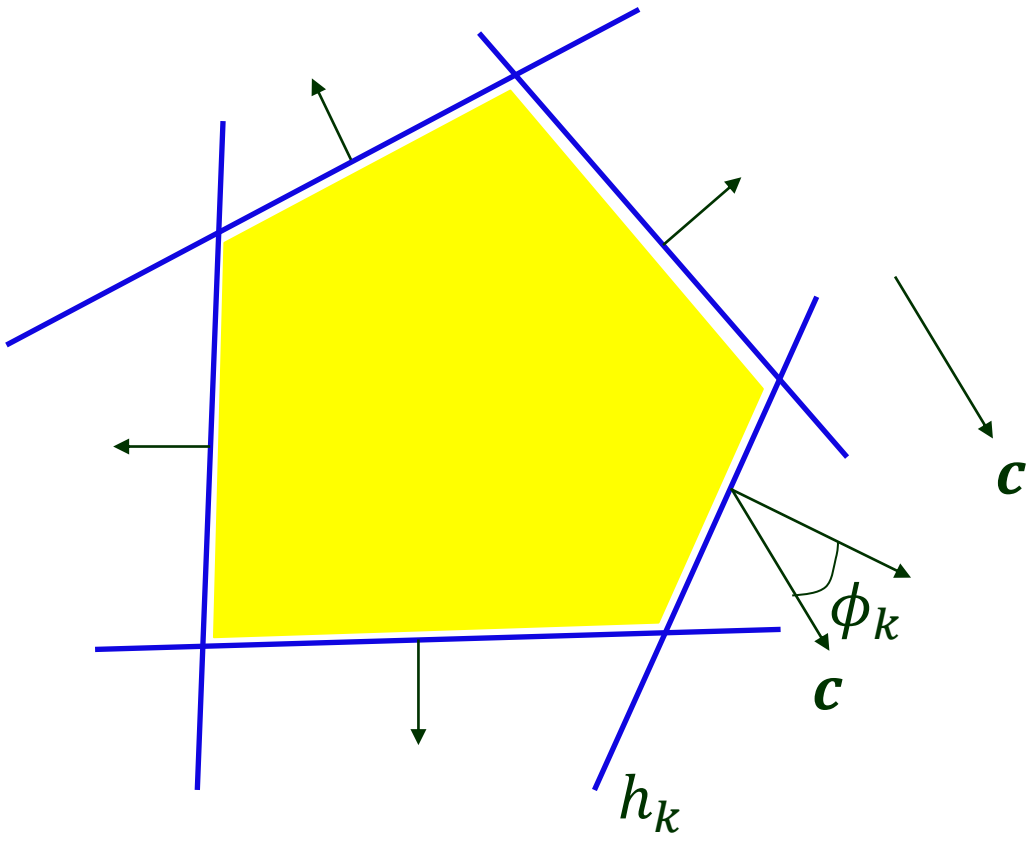


n_i: outward normal

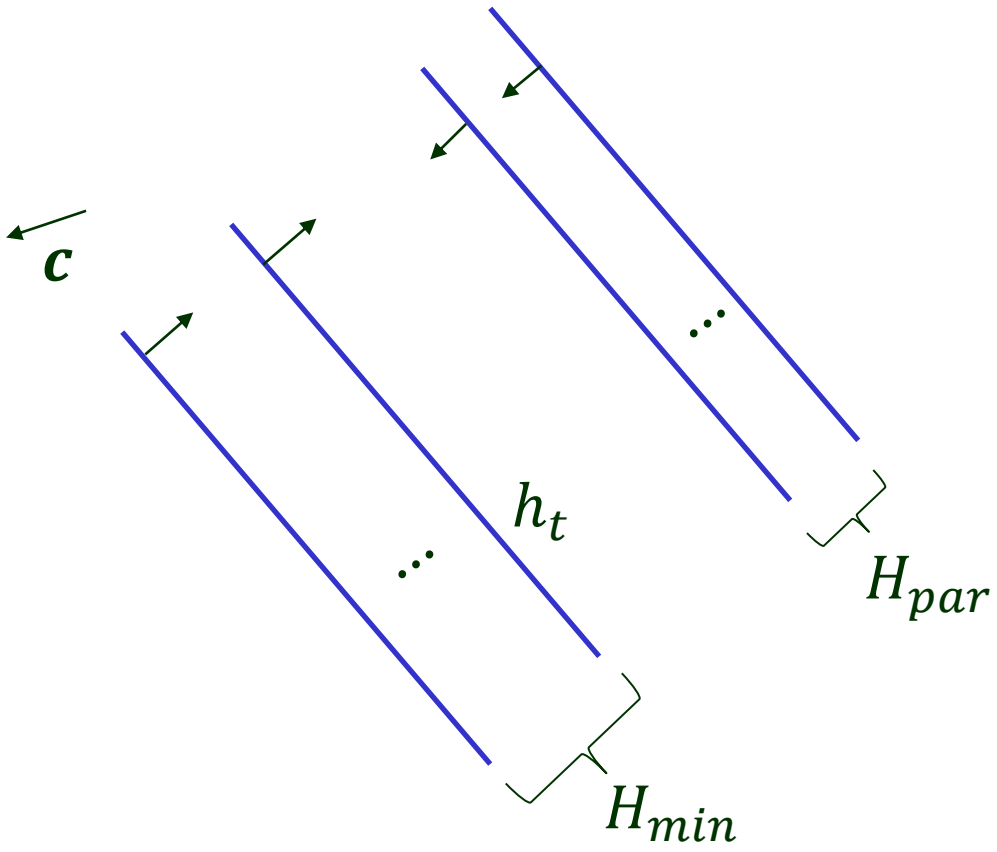
$\phi_i \in [0, \pi]$

Minimum Angle

$$\phi_k = \min_{1 \le i \le n} \phi_i$$



Initial Slab

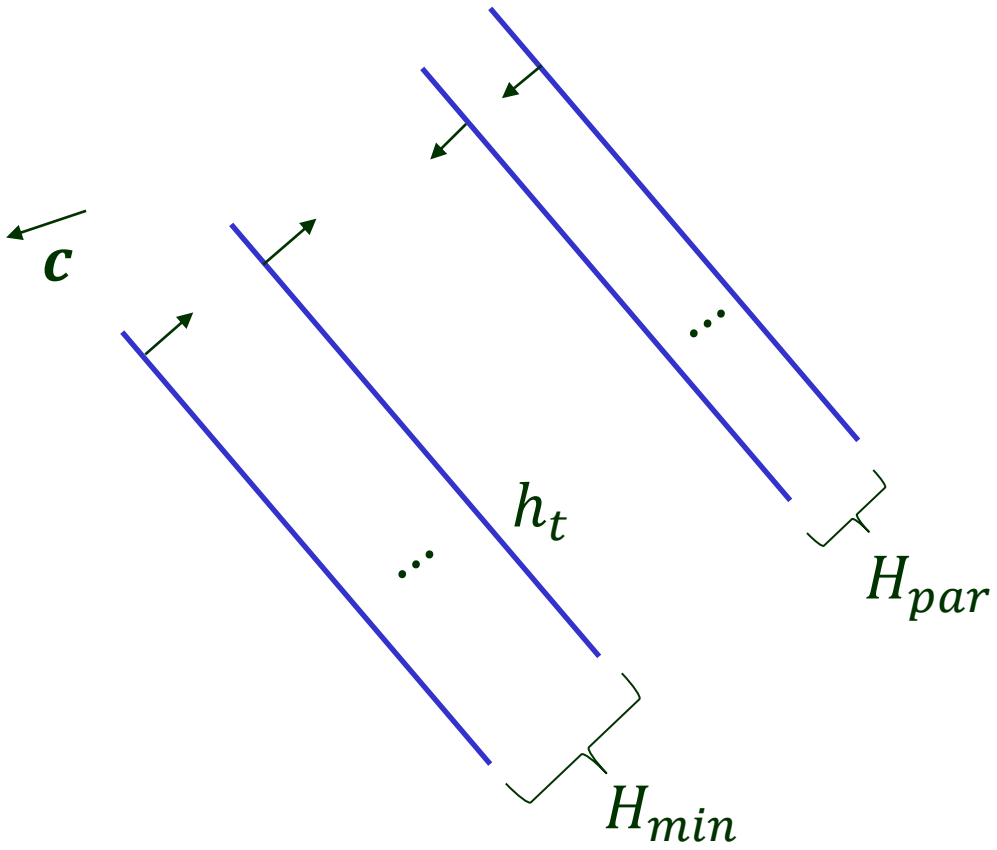


$$H_{min} = \{h_j \in H \mid n_j = n_k\}$$

$$H_{par} = \{h_j \in H \mid n_j = -n_k\}$$

Parallel boundary lines.

Initial Slab



$$H_{min} = \{h_j \in H \mid n_j = n_k\}$$

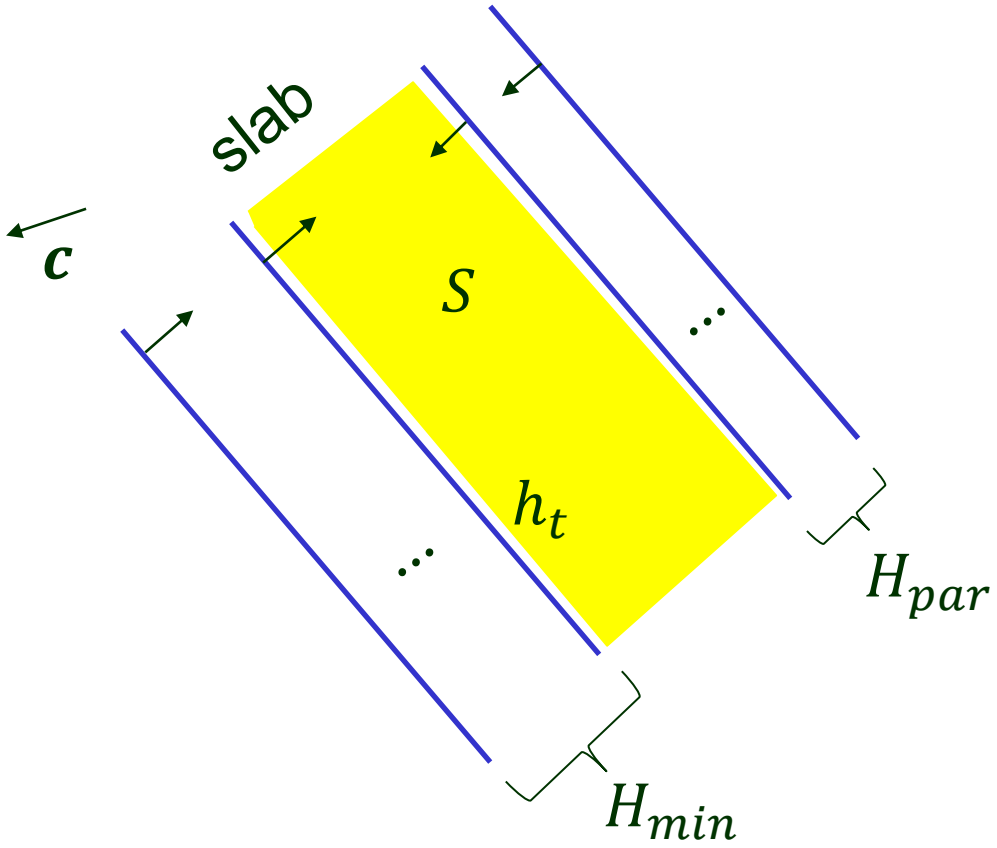
$$H_{par} = \{h_j \in H \mid n_j = -n_k\}$$

Parallel boundary lines.

Compute

$$S = \bigcap (H_{min} \cup H_{par})$$

Initial Slab



$$H_{min} = \{h_j \in H \mid n_j = n_k\}$$

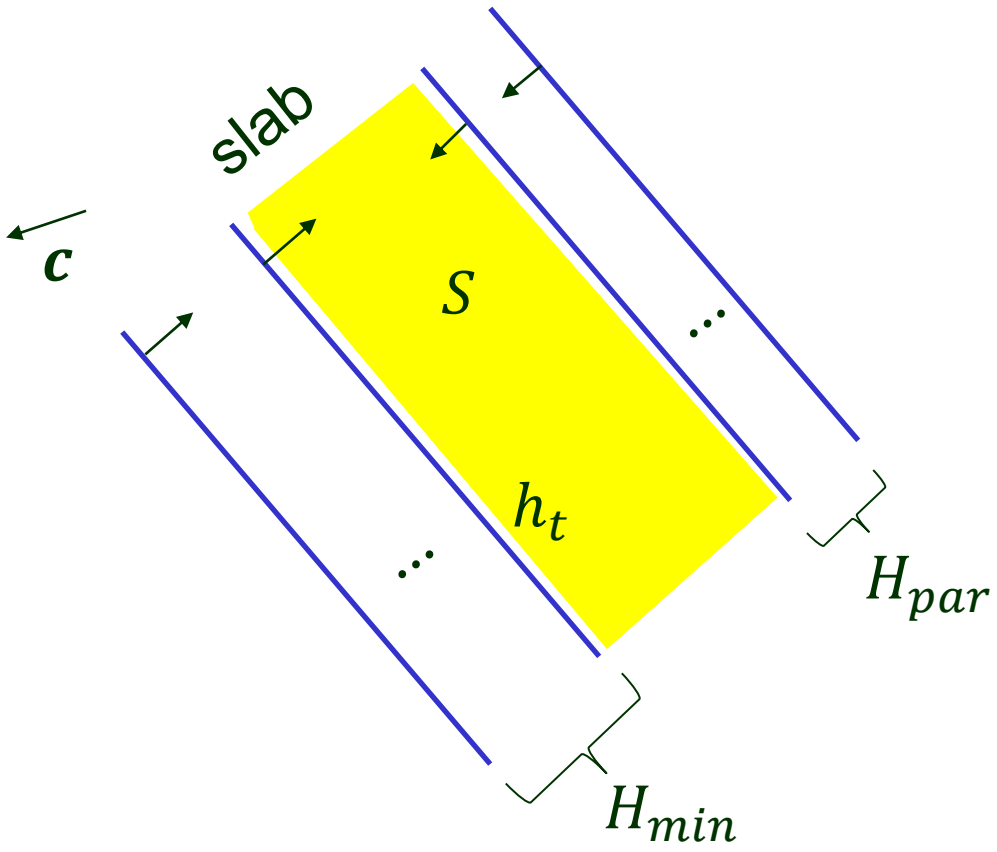
$$H_{par} = \{h_j \in H \mid n_j = -n_k\}$$

Parallel boundary lines.

Compute

$$S = \bigcap (H_{min} \cup H_{par})$$

Initial Slab



$$H_{min} = \{h_j \in H \mid n_j = n_k\}$$

$$H_{par} = \{h_j \in H \mid n_j = -n_k\}$$

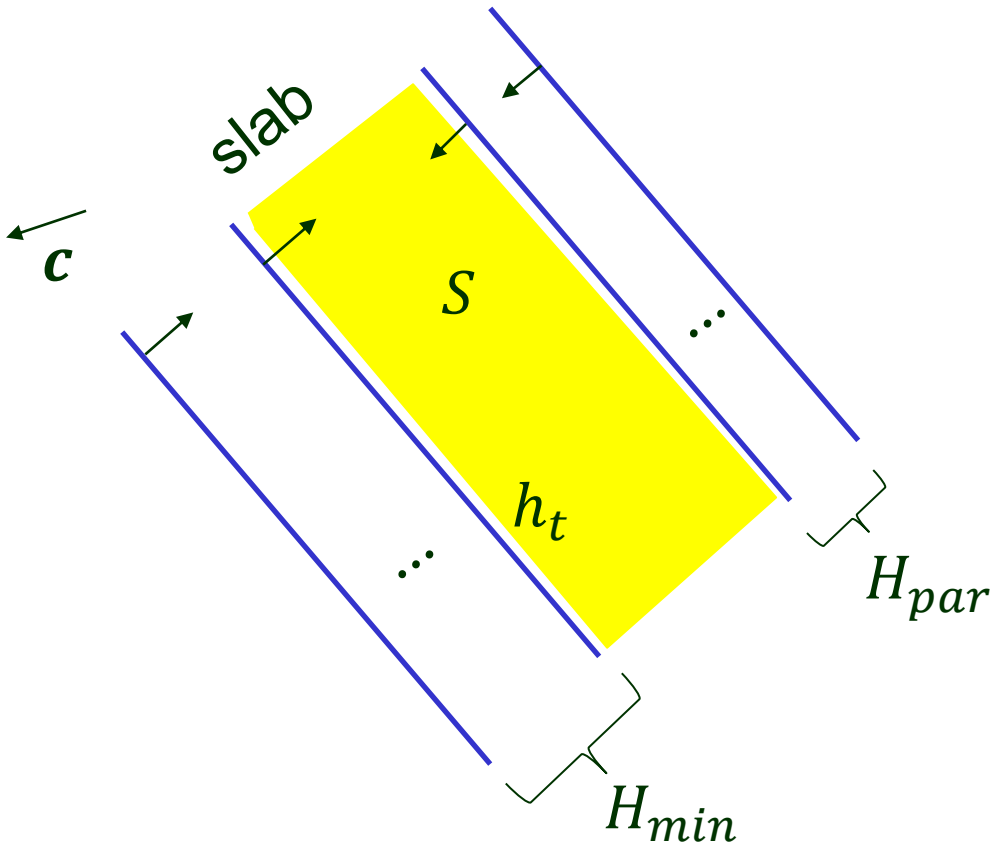
Parallel boundary lines.

Compute

$$S = \bigcap (H_{min} \cup H_{par})$$

$O(n)$ time

Initial Slab



$$H_{min} = \{h_j \in H \mid n_j = n_k\}$$

$$H_{par} = \{h_j \in H \mid n_j = -n_k\}$$

Parallel boundary lines.

Compute

$$S = \bigcap (H_{min} \cup H_{par})$$

$O(n)$ time

$S = \emptyset \implies$ LP infeasible!

Non-Empty Slab ($S \neq \emptyset$)

S bounded by some line l_t with $h_t \in H_{min}$.

Assumption At least one half-plane not in H_{min} or H_{par}.

Non-Empty Slab ($S \neq \emptyset$)

S bounded by some line l_t with $h_t \in H_{min}$.

Assumption At least one half-plane not in H_{min} or H_{par}.

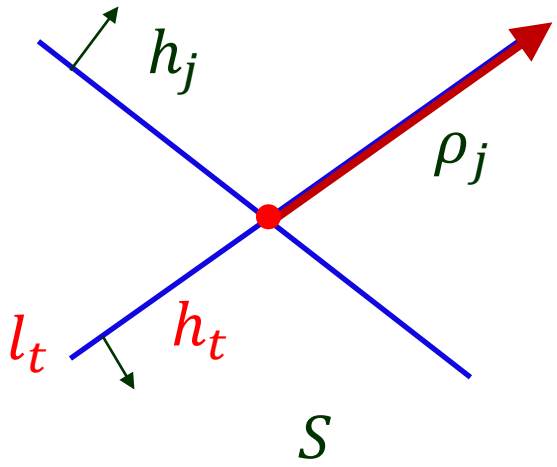
Ray $\rho_j = l_t \cap h_j$ where $1 \leq j \leq n, h_j \notin H_{min} \cup H_{par}$

Non-Empty Slab ($S \neq \emptyset$)

S bounded by some line l_t with $h_t \in H_{min}$.

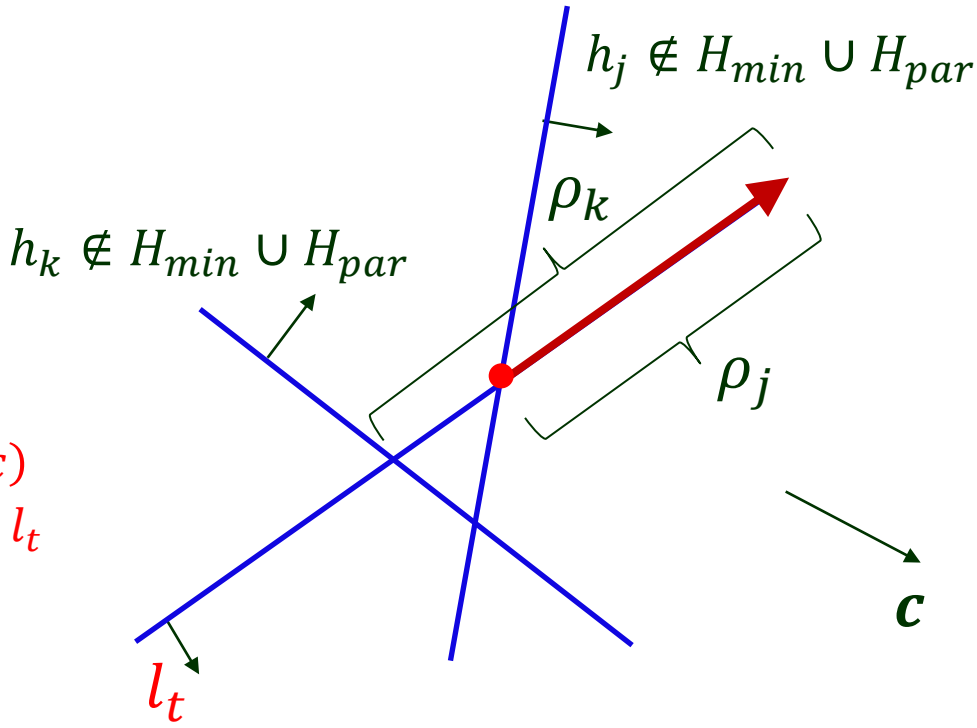
Assumption At least one half-plane not in H_{min} or H_{par}.

Ray $\rho_j = l_t \cap h_j$ where $1 \leq j \leq n, h_j \notin H_{min} \cup H_{par}$



Case 1: Unbounded LP

$\rho_j = l_t \cap h_j$ unbounded in $\boldsymbol{c}$ for all $1 \leq j \leq n, h_j \notin H_{min} \cup H_{par}$.

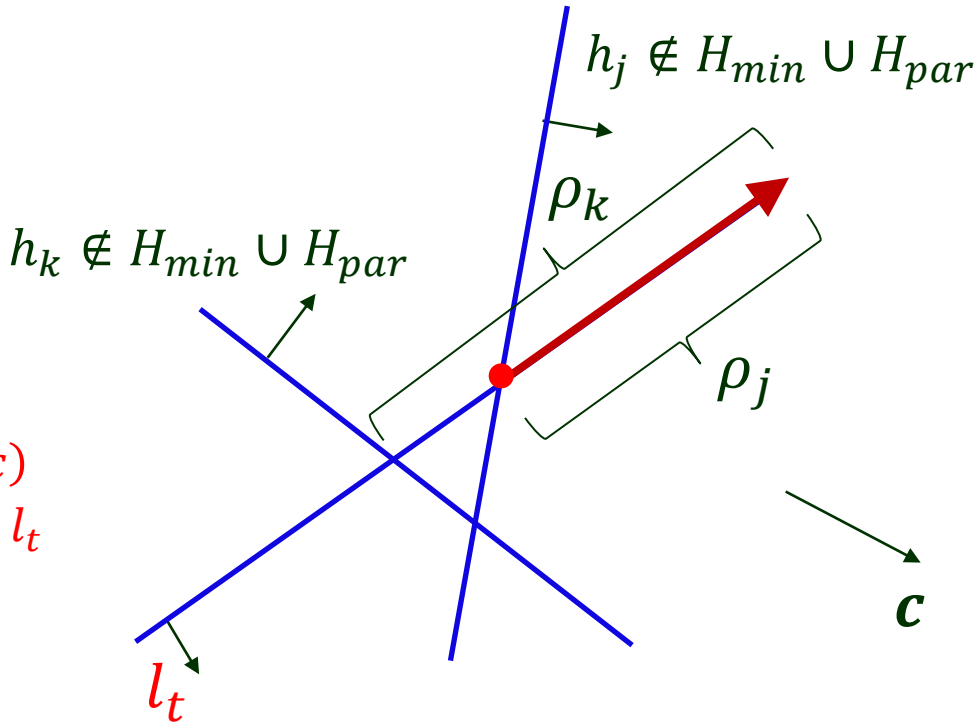


Lemma 1 If $\rho_j = l_t \cap h_j$ is unbounded in $\boldsymbol{c}$ for every $h_j \notin H_{min} \cup H_{par}$, then (H, c) is unbounded along a ray contained in l_t which can be computed in $O(n)$ time.

Case 1: Unbounded LP

$\rho_j = l_t \cap h_j$ unbounded in $\boldsymbol{c}$ for all $1 \leq j \leq n, h_j \notin H_{min} \cup H_{par}$.

All ρ_j lie on l_t.



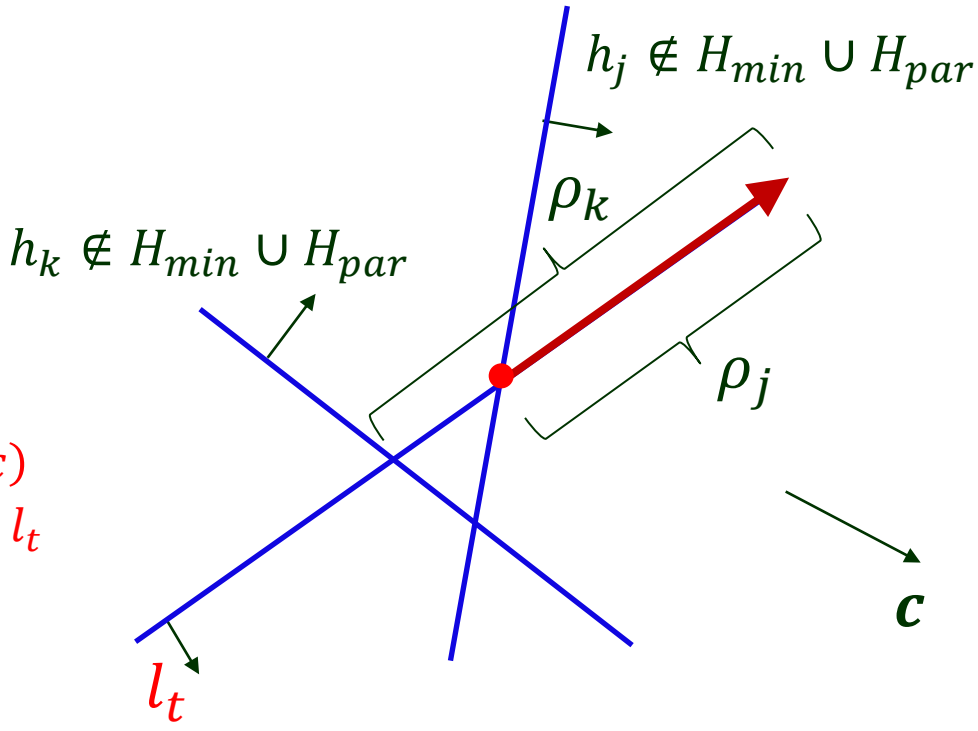
Lemma 1 If $\rho_j = l_t \cap h_j$ is unbounded in $\boldsymbol{c}$ for every $h_j \notin H_{min} \cup H_{par}$, then (H, c) is unbounded along a ray contained in l_t which can be computed in $O(n)$ time.

Case 1: Unbounded LP

$\rho_j = l_t \cap h_j$ unbounded in $\boldsymbol{c}$ for all $1 \le j \le n, h_j \notin H_{min} \cup H_{par}$.

All ρ_j lie on l_t.

$$\bigcap_{h_j \notin H_{min} \cup H_{par}} \rho_j$$



Lemma 1 If $\rho_j = l_t \cap h_j$ is unbounded in $\boldsymbol{c}$ for every $h_j \notin H_{min} \cup H_{par}$, then (H, c) is unbounded along a ray contained in l_t which can be computed in $O(n)$ time.

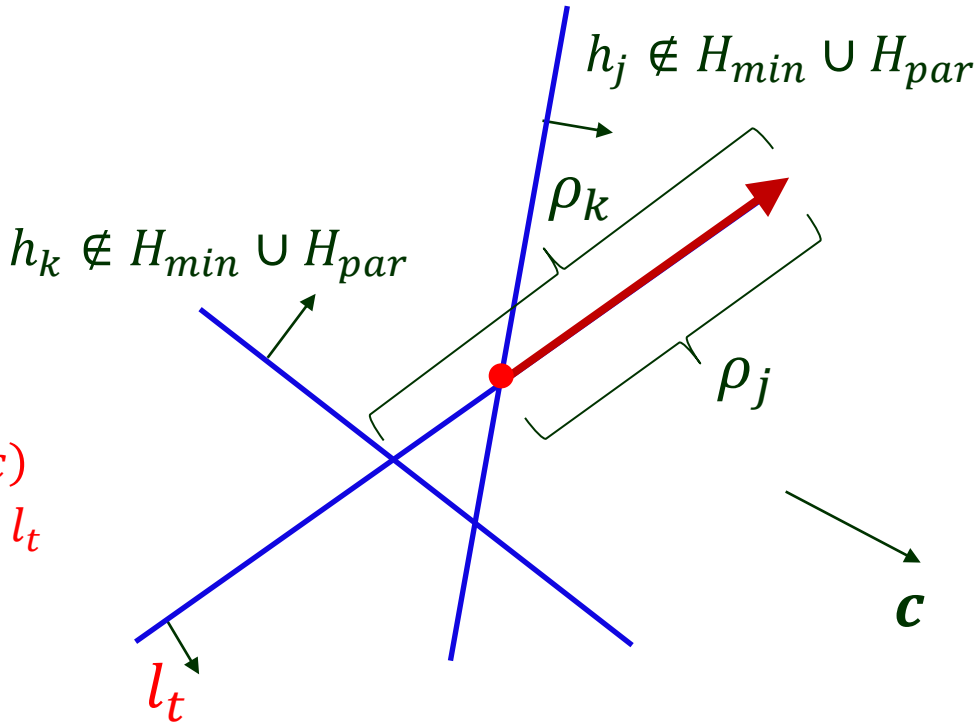
Case 1: Unbounded LP

$\rho_j = l_t \cap h_j$ unbounded in $\boldsymbol{c}$ for all $1 \le j \le n, h_j \notin H_{min} \cup H_{par}$.

All ρ_j lie on l_t.

$\Downarrow$

$\bigcap_{h_j \notin H_{min} \cup H_{par}} \rho_j$ is a ray unbounded in $\boldsymbol{c}$.



Lemma 1 If $\rho_j = l_t \cap h_j$ is unbounded in $\boldsymbol{c}$ for every $h_j \notin H_{min} \cup H_{par}$, then (H, c) is unbounded along a ray contained in l_t which can be computed in $O(n)$ time.

Case 1: Unbounded LP

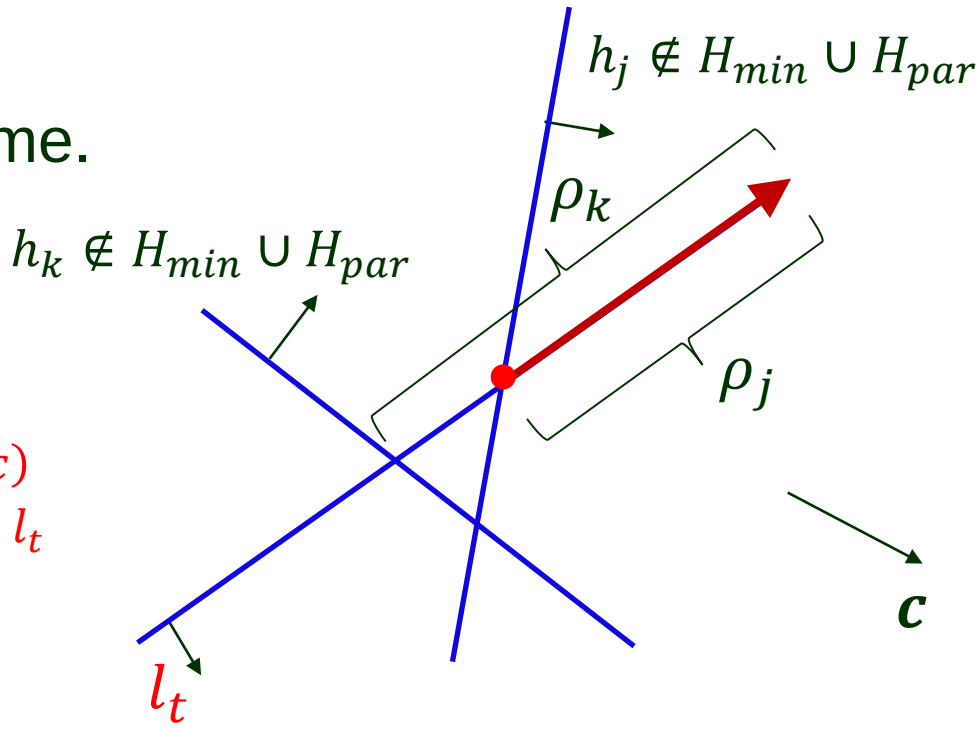
$\rho_j = l_t \cap h_j$ unbounded in $\boldsymbol{c}$ for all $1 \leq j \leq n, h_j \notin H_{min} \cup H_{par}$.

All ρ_j lie on l_t.

$\Downarrow$

$\bigcap_{h_j \notin H_{min} \cup H_{par}} \rho_j$ is a ray unbounded in $\boldsymbol{c}$.

Ray intersection in $O(n)$ time.



Lemma 1 If $\rho_j = l_t \cap h_j$ is unbounded in $\boldsymbol{c}$ for every $h_j \notin H_{min} \cup H_{par}$, then (H, c) is unbounded along a ray contained in l_t which can be computed in $O(n)$ time.

Case 2: Bounded LP

Boundary line l_t with $h_t \in H_{min}$ bounds S.

Lemma 2 If $\rho_j = l_t \cap h_j$ is bounded in $\boldsymbol{c}$ for some $h_j \notin H_{min} \cup H_{par}$, $1 \leq j \leq n$, then LP$\left(\{h_t, h_j\}, c\right)$ is bounded.

Case 2: Bounded LP

Boundary line l_t with $h_t \in H_{min}$ bounds S.

Lemma 2 If $\rho_j = l_t \cap h_j$ is bounded in $\boldsymbol{c}$ for some $h_j \notin H_{min} \cup H_{par}$, $1 \leq j \leq n$, then LP$\left(\{h_t, h_j\}, c\right)$ is bounded.

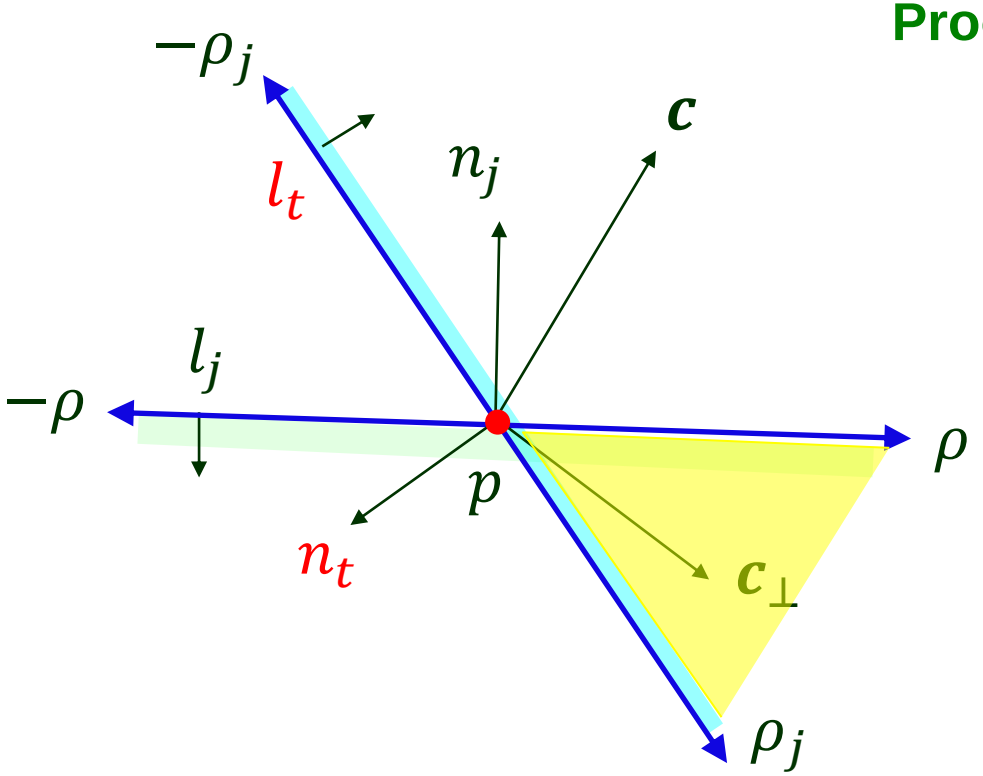
Proof By contradiction. Assume that LP$\left(\{h_t, h_j\}, c\right)$ is unbounded.

ρ_j bounded in $\boldsymbol{c}$

Case 2: Bounded LP

Boundary line l_t with $h_t \in H_{min}$ bounds S.

Lemma 2 If $\rho_j = l_t \cap h_j$ is bounded in $\boldsymbol{c}$ for some $h_j \notin H_{min} \cup H_{par}$, $1 \le j \le n$, then LP$(\{h_t, h_j\}, c)$ is bounded.



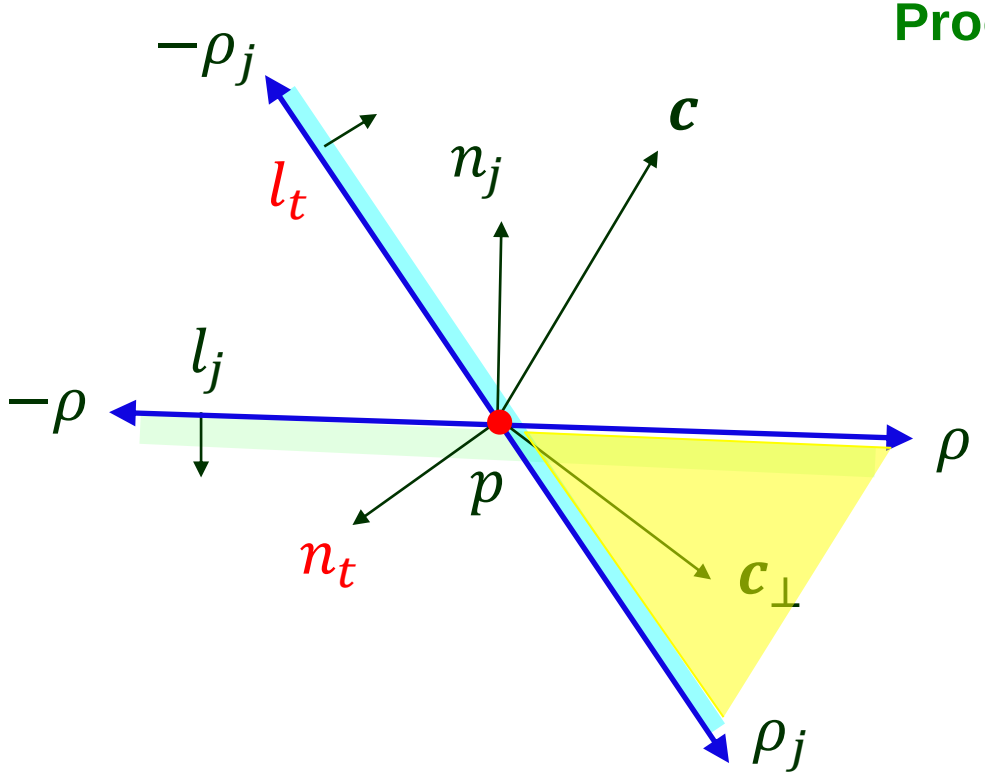
Proof By contradiction. Assume that LP$(\{h_t, h_j\}, c)$ is unbounded.

ρ_j bounded in $\boldsymbol{c}$

Case 2: Bounded LP

Boundary line l_t with $h_t \in H_{min}$ bounds S.

Lemma 2 If $\rho_j = l_t \cap h_j$ is bounded in $\boldsymbol{c}$ for some $h_j \notin H_{min} \cup H_{par}$, $1 \le j \le n$, then LP$(\{h_t, h_j\}, c)$ is bounded.



Proof By contradiction. Assume that LP$(\{h_t, h_j\}, c)$ is unbounded.

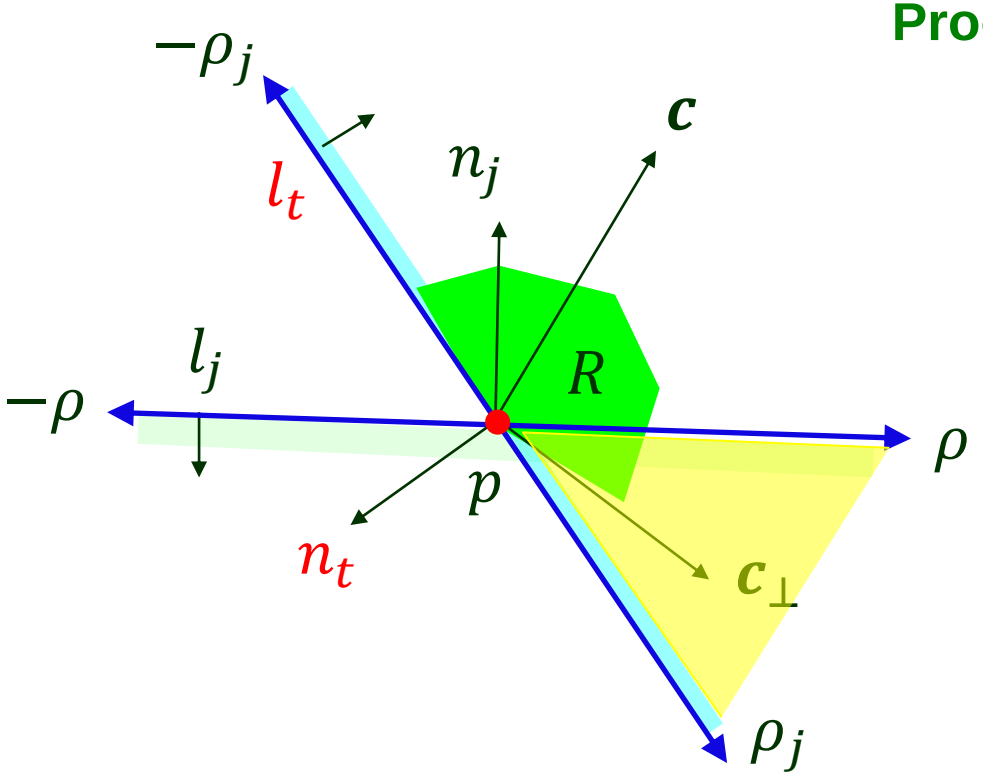
⇓ ρ_j bounded in $\boldsymbol{c}$

$\rho = l_j \cap h_t$ must be unbounded.

Case 2: Bounded LP

Boundary line l_t with $h_t \in H_{min}$ bounds S.

Lemma 2 If $\rho_j = l_t \cap h_j$ is bounded in $\boldsymbol{c}$ for some $h_j \notin H_{min} \cup H_{par}$, $1 \le j \le n$, then LP$(\{h_t, h_j\}, c)$ is bounded.



Proof By contradiction. Assume that LP$(\{h_t, h_j\}, c)$ is unbounded.

⇩ ρ_j bounded in $\boldsymbol{c}$

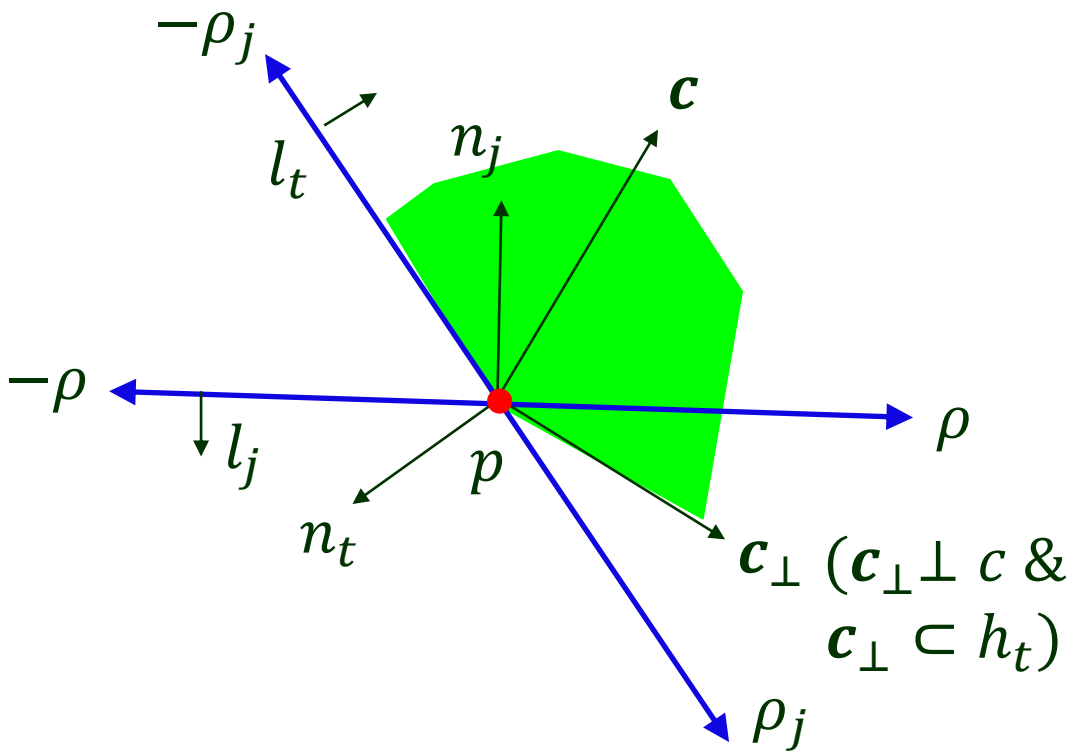
$\rho = l_j \cap h_t$ must be unbounded.

⇩

ρ must be in the region R between $-\rho_j$ and $\boldsymbol{c}_\perp$ ($\boldsymbol{c}_\perp \perp \boldsymbol{c}$ & $\boldsymbol{c}_\perp \subset h_t$) that is contained in h_t.

Contradiction to Choice of h_t

ρ in the region between $\boldsymbol{c}_\perp$ and $-\rho_j$.

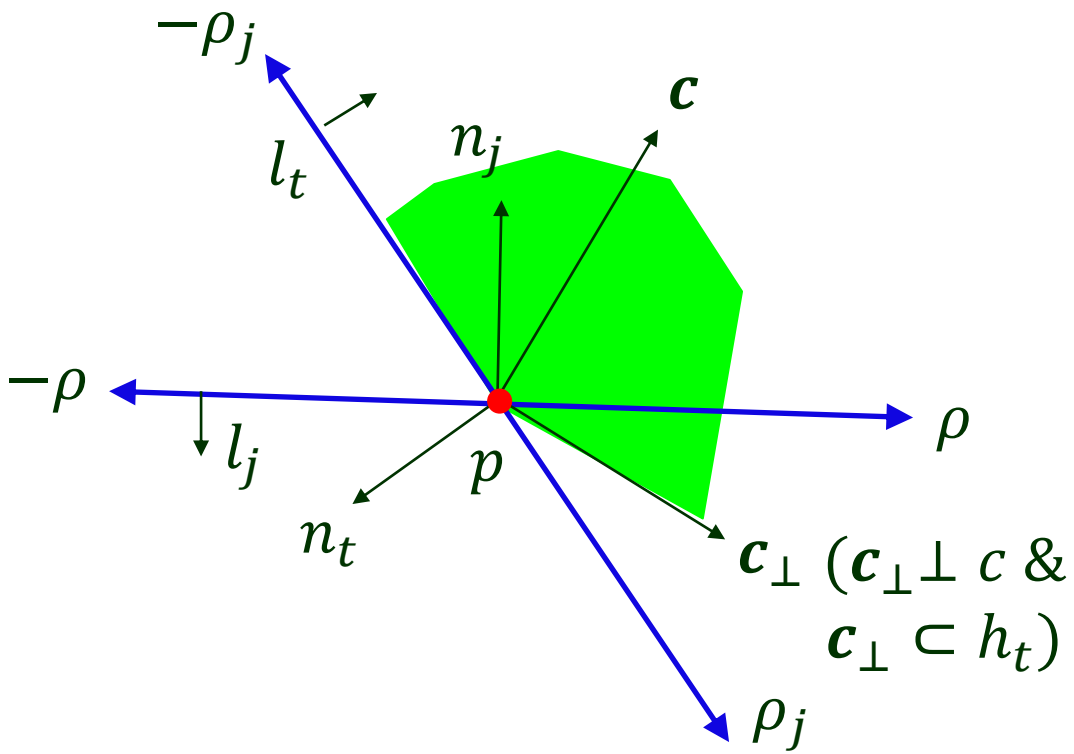


Contradiction to Choice of h_t

ρ in the region between $\boldsymbol{c}_\perp$ and $-\rho_j$.

rotate each vector counterclockwise by $\frac{\pi}{2}$.

ρ, $\boldsymbol{c}_\perp$, $-\rho_j$ becomes n_j, $\boldsymbol{c}$, n_t, respectively.



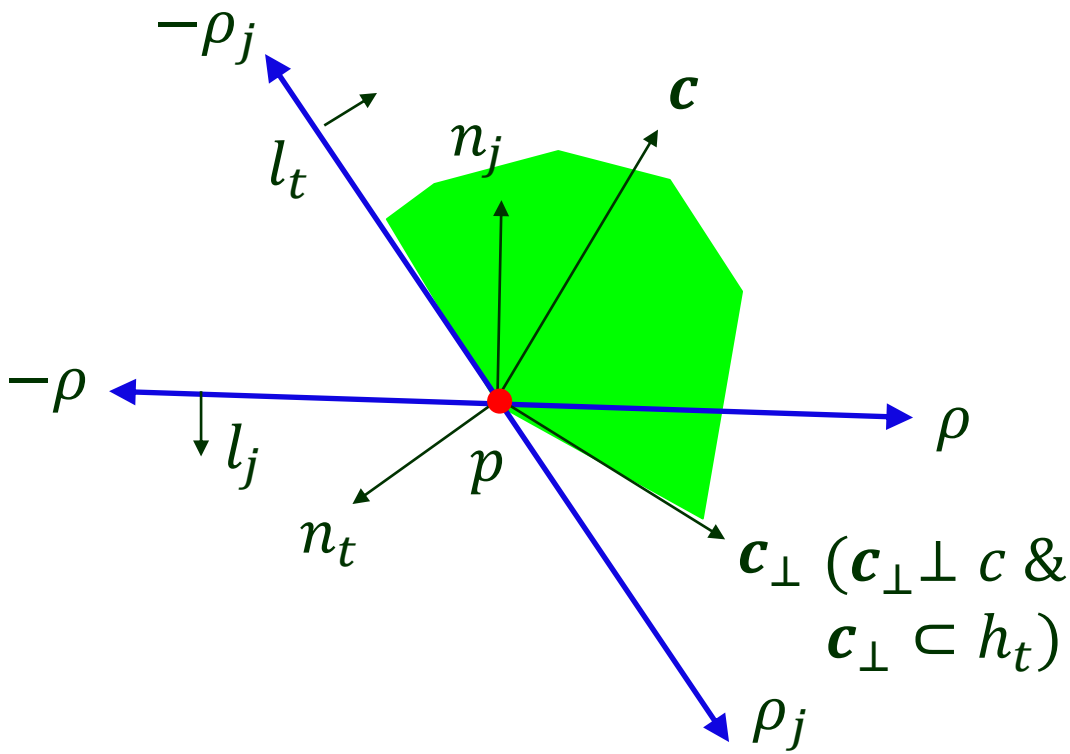
Contradiction to Choice of h_t

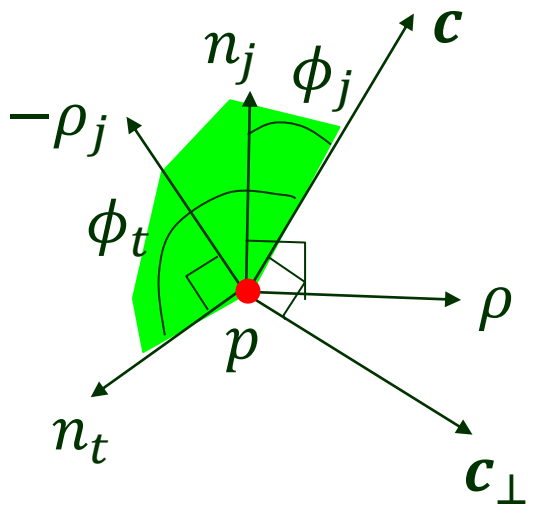
rotate each vector counterclockwise by $\frac{\pi}{2}$.
ρ, $\boldsymbol{c}_\perp$, $-\rho_j$ becomes n_j, $\boldsymbol{c}$, n_t, respectively.

ρ in the region between $\boldsymbol{c}_\perp$ and $-\rho_j$.

$\Downarrow$

n_j must be in the region between $\boldsymbol{c}$ and n_t (note $n_j \neq n_t$).





Contradiction to Choice of h_t

rotate each vector counterclockwise by $\frac{\pi}{2}$.
ρ, $\boldsymbol{c}_\perp$, $-\rho_j$ becomes n_j, $\boldsymbol{c}$, n_t, respectively.

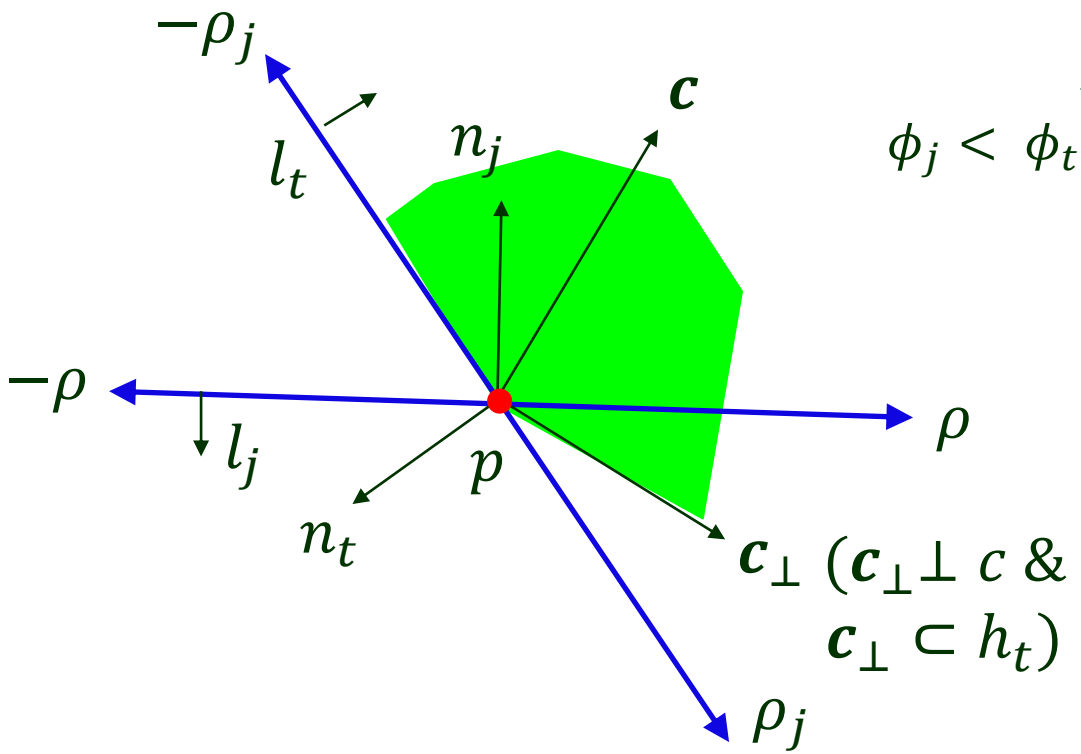
ρ in the region between $\boldsymbol{c}_\perp$ and $-\rho_j$.

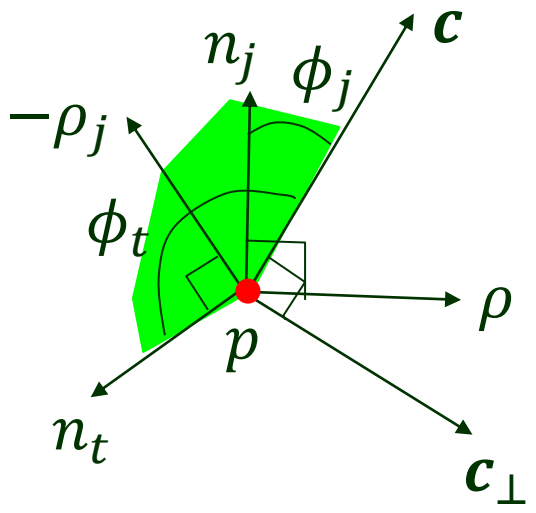
$\Downarrow$

n_j must be in the region between $\boldsymbol{c}$ and n_t (note $n_j \neq n_t$).

$\Downarrow$

$$\phi_j < \phi_t = \min_{1 \le i \le n} \phi_i$$





Contradiction to Choice of h_t

rotate each vector counterclockwise by $\frac{\pi}{2}$.
ρ, $\boldsymbol{c}_\perp$, $-\rho_j$ becomes n_j, $\boldsymbol{c}$, n_t, respectively.

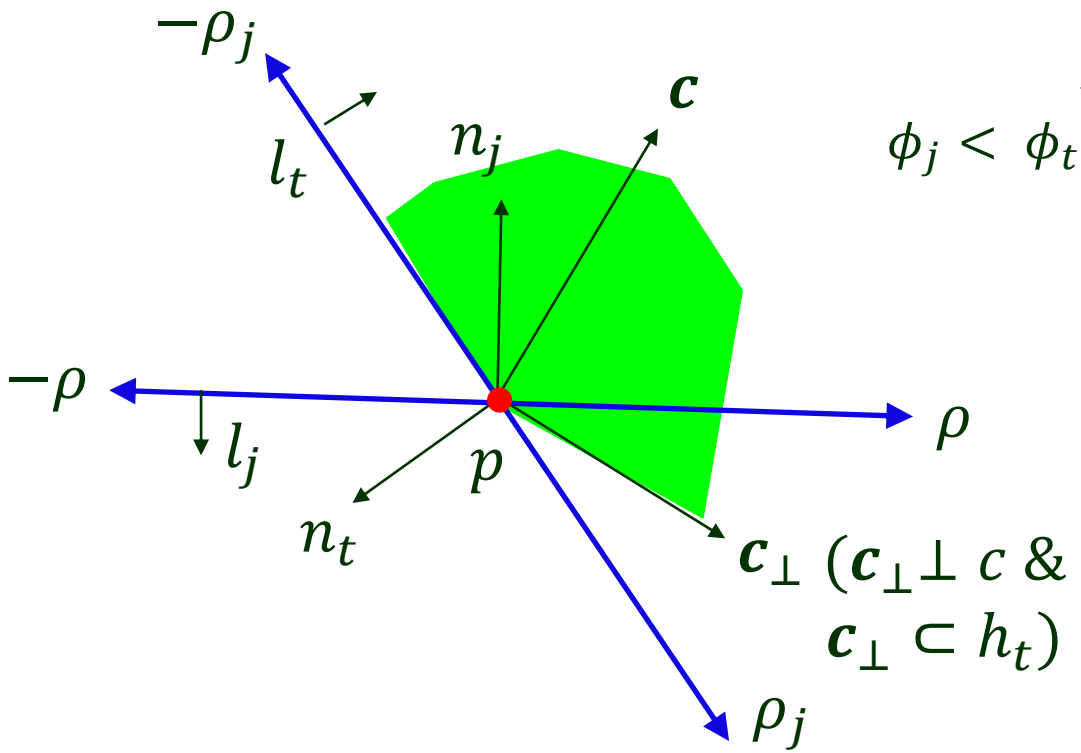
ρ in the region between $\boldsymbol{c}_\perp$ and $-\rho_j$.

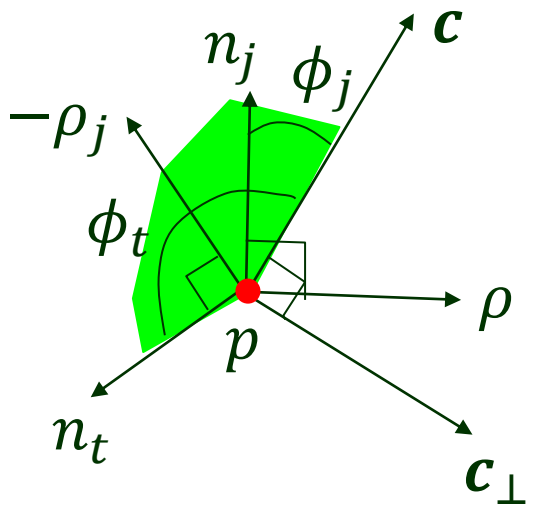
⇩

n_j must be in the region between $\boldsymbol{c}$ and n_t (note $n_j \neq n_t$).

⇩

$\phi_j < \phi_t = \min_{1 \le i \le n} \phi_i$ Contradiction!





UNBOUNDEDLP(H, $\boldsymbol{c}$)

Compute ϕ_i for all $1 \le i \le n$

Choose k such that $\phi_k = \min_{1 \le i \le n} \phi_i$

$H_{min} \leftarrow \{h_j \in H \mid n_j = n_k\}$

$H_{par} \leftarrow \{h_j \in H \mid n_j = -n_k\}$

$\widehat{H} \leftarrow H \setminus (H_{min} \cup H_{par})$

$S \leftarrow \cap\, (H_{min} \cup H_{par})$

```
if S = ∅
    then LP(H,c) is infeasible
    else h_t ∈ H_min that bounds S
        if ρ_j = l_t ∩ h_j bounded for some h_j ∈ Ĥ
            then LP(H,c) is bounded // it may still be infeasible
            else LP(H,c) is unbounded along the ray l_t ∩ (∩ Ĥ)
```

Correctness and Running Time

Lemma 3 UNBOUNDEDLP(H, $\boldsymbol{c}$) decides whether a 2D LP with n constrains is unbounded in $O(n)$ time, and if so, returns a ray in the feasible region unbounded in $\boldsymbol{c}$.