CS 473: Algorithms

Ruta Mehta

University of Illinois, Urbana-Champaign

Spring 2018

Simplex and LP Duality

Lecture 19

March 29, 2018

Some of the slides are courtesy Prof. Chekuri

Outline

Simplex: Intuition and Implementation Details

- Computing starting vertex: equivalent to solving an LP!

Infeasibility, Unboundedness, and Degeneracy.

Duality: Bounding the objective value through *weak-duality*

Strong Duality, Cone view.

Part I

Recall

Feasible Region and Convexity

Canonical Form

Given $A \in R^{n \times d}$, $b \in R^{n \times 1}$ and $c \in R^{1 \times d}$, find $x \in R^{d \times 1}$

$$\begin{aligned} \textbf{max}: \quad & c \cdot x \\ s.t. \quad & Ax \leq b \end{aligned}$$

Feasible Region and Convexity

Canonical Form

Given $A \in R^{n \times d}$, $b \in R^{n \times 1}$ and $c \in R^{1 \times d}$, find $x \in R^{d \times 1}$

$$\begin{array}{ll} \max: & c \cdot x \\ s.t. & Ax \leq b \end{array}$$

1. Each linear constraint defines a **halfspace**, a convex set.
2. Feasible region, which is an intersection of halfspaces, is a convex **polyhedron**.
3. Optimal value attained at a vertex of the polyhedron.

Part II

Simplex

Simplex Algorithm

Simplex: Vertex hoping algorithm

Moves from a vertex to its neighboring vertex

Simplex Algorithm

Simplex: Vertex hoping algorithm

Moves from a vertex to its neighboring vertex

Questions

- Which neighbor to move to?
- When to stop?
- How much time does it take?

Observations

For Simplex

Suppose we are at a non-optimal vertex $\hat{x}$ and optimal is x^*, then $c \cdot x^* > c \cdot \hat{x}$.

Observations

For Simplex

Suppose we are at a non-optimal vertex $\hat{x}$ and optimal is x^*, then $c \cdot x^* > c \cdot \hat{x}$.

How does $(c \cdot x)$ change as we move from $\hat{x}$ to x^* on the line joining the two?

Observations

For Simplex

Suppose we are at a non-optimal vertex $\hat{x}$ and optimal is x^*, then $c \cdot x^* > c \cdot \hat{x}$.

How does $(c \cdot x)$ change as we move from $\hat{x}$ to x^* on the line joining the two?

Strictly increases!

Observations

For Simplex

Suppose we are at a non-optimal vertex $\hat{x}$ and optimal is x^*, then $c \cdot x^* > c \cdot \hat{x}$.

How does $(c \cdot x)$ change as we move from $\hat{x}$ to x^* on the line joining the two?

Strictly increases!

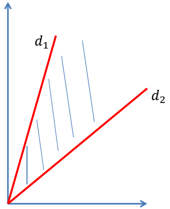
- $d = x^* - \hat{x}$ is the direction from $\hat{x}$ to x^*.
- $(c \cdot d) = (c \cdot x^*) - (c \cdot \hat{x}) > 0$.
- In $x = \hat{x} + \delta d$, as δ goes from 0 to 1, we move from $\hat{x}$ to x^*.
- $c \cdot x = c \cdot \hat{x} + \delta(c \cdot d)$. Strictly increasing with δ!
- Due to convexity, all of these are feasible points.

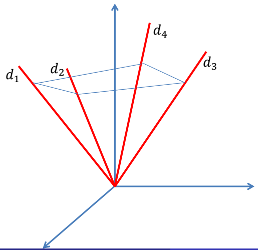
Cone

Definition

Given a set of vectors $D = \{d_1, \ldots, d_k\}$, the cone spanned by them is just their positive linear combinations, i.e.,

$$cone(D) = \{d \mid d = \sum_{i=1}^{k} \lambda_i d_i, \text{ where } \lambda_i \geq 0, \forall i\}$$



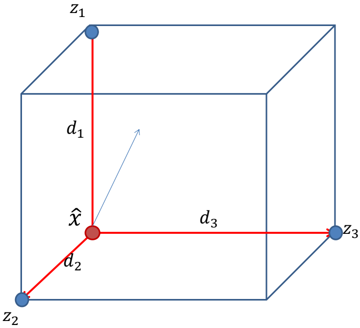


Cone at a Vertex

Let $z_1, \ldots, z_k$ be the neighboring vertices of $\hat{x}$. And let $d_i = z_i - \hat{x}$ be the direction from $\hat{x}$ to z_i.

Lemma

Any feasible direction of movement d from $\hat{x}$ is in the $cone(\{d_1, \ldots, d_k\})$.



Improving Direction Implies Improving Neighbor

Lemma

If $d \in cone(\{d_1, \ldots, d_k\})$ and $(c \cdot d) > 0$, then there exists d_i such that $(c \cdot d_i) > 0$.

Improving Direction Implies Improving Neighbor

Lemma

If $d \in cone(\{d_1, \ldots, d_k\})$ *and* $(c \cdot d) > 0$, *then there exists* d_i *such that* $(c \cdot d_i) > 0$.

Proof.

To the contrary suppose $(c \cdot d_i) \leq 0, \; \forall i \leq k$.
Since d is a positive linear combination of d_i's,

$$\begin{aligned} (c \cdot d) &= (c \cdot \textstyle\sum_{i=1}^k \lambda_i d_i) \\ &= \textstyle\sum_{i=1}^k \lambda_i (c \cdot d_i) \\ &\leq 0 \quad \text{A contradiction!} \end{aligned}$$

$\square$

Improving Direction Implies Improving Neighbor

Lemma

If $d \in cone(\{d_1, \ldots, d_k\})$ *and* $(c \cdot d) > 0$, *then there exists* d_i *such that* $(c \cdot d_i) > 0$.

Proof.

To the contrary suppose $(c \cdot d_i) \leq 0,\ \forall i \leq k$.
Since d is a positive linear combination of d_i's,

$$\begin{aligned} (c \cdot d) &= (c \cdot \textstyle\sum_{i=1}^{k} \lambda_i d_i) \\ &= \textstyle\sum_{i=1}^{k} \lambda_i (c \cdot d_i) \\ &\leq 0 \quad \text{A contradiction!} \end{aligned}$$

□

Theorem

If vertex $\hat{x}$ *is not optimal then it has a neighbor where cost improves.*

How Many Neighbors a Vertex Has?

Geometric view...

$A \in R^{n \times d}$ $(n > d)$, $b \in R^n$, the constraints are: $Ax \leq b$

Geometry of faces

- r linearly independent hyperplanes forms $(d - r)$ dimensional face.

How Many Neighbors a Vertex Has?

Geometric view...

$A \in R^{n \times d}$ $(n > d)$, $b \in R^n$, the constraints are: $Ax \leq b$

Geometry of faces

- r linearly independent hyperplanes forms $(d - r)$ dimensional face.
- Vertex: $\mathbf{0}$-D face. formed by d L.I. hyperplanes.

How Many Neighbors a Vertex Has?

Geometric view...

$A \in R^{n \times d}$ $(n > d)$, $b \in R^n$, the constraints are: $Ax \leq b$

Geometry of faces

- r linearly independent hyperplanes forms $(d - r)$ dimensional face.
- Vertex: 0-D face. formed by d L.I. hyperplanes.
- Edge: 1-D face. formed by $(d - 1)$ L.I. hyperlanes.

How Many Neighbors a Vertex Has?

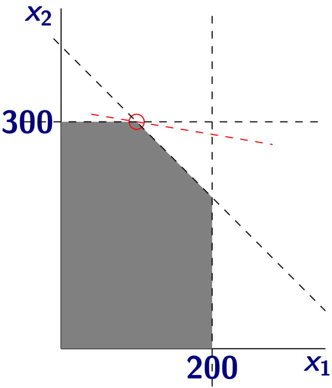
Geometric view...

$A \in R^{n \times d}$ $(n > d)$, $b \in R^n$, the constraints are: $Ax \leq b$

Geometry of faces

- r linearly independent hyperplanes forms $(d - r)$ dimensional face.
- Vertex: $\mathbf{0}$-D face. formed by d L.I. hyperplanes.
- Edge: $\mathbf{1}$-D face. formed by $(d - 1)$ L.I. hyperlanes.

In 2-dimension $(d = 2)$



How Many Neighbors a Vertex Has?

Geometric view...

In 3-dimension ($d = 3$)

$A \in R^{n \times d}$ ($n > d$), $b \in R^n$, the constraints are: $Ax \leq b$

Geometry of faces

- r linearly independent hyperplanes forms $(d - r)$ dimensional face.
- Vertex: 0-dimensional face. formed by d L.I. hyperplanes.
- Edge: 1-D face. formed by $(d - 1)$ L.I. hyperlanes.

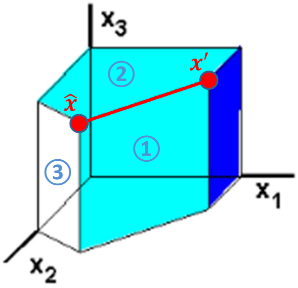


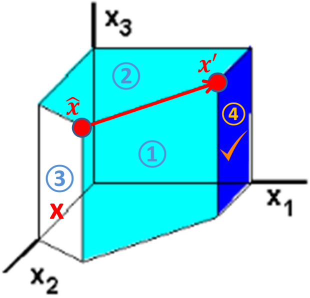
image source: webpage of Prof. Forbes W. Lewis

How Many Neighbors a Vertex Has?

Geometry view...

One neighbor per tight hyperplane. Therefore typically d.

- Suppose x' is a neighbor of $\hat{x}$, then on the edge joining the two $d-1$ constraints are tight.
- These $d-1$ are also tight at both $\hat{x}$ and x'.
- One more constraints, say i, is tight at $\hat{x}$. "Relaxing" i at $\hat{x}$ leads to x'.



Simplex Algorithm

Simplex: Vertex hoping algorithm

Moves from a vertex to its neighboring vertex

Questions + Answers

- Which neighbor to move to? One where objective value increases.

Simplex Algorithm

Simplex: Vertex hoping algorithm

Moves from a vertex to its neighboring vertex

Questions + Answers

- Which neighbor to move to? One where objective value increases.
- When to stop? When no neighbor with better objective value.

Simplex Algorithm

Simplex: Vertex hoping algorithm

Moves from a vertex to its neighboring vertex

Questions + Answers

- Which neighbor to move to? One where objective value increases.
- When to stop? When no neighbor with better objective value.
- How much time does it take? At most d neighbors to consider in each step.

Simplex in Higher Dimensions

Simplex Algorithm

1. Start at a vertex of the polytope.
2. Compare value of objective function at each of the d "neighbors".
3. Move to neighbor that improves objective function, and repeat step 2.
4. If no improving neighbor, then stop.

Simplex is a greedy local-improvement algorithm! Works because a local optimum is also a global optimum — convexity of polyhedra.

Part III

Implementation of the Pivoting Step (Moving to an improving neighbor)

Moving to a Neighbor

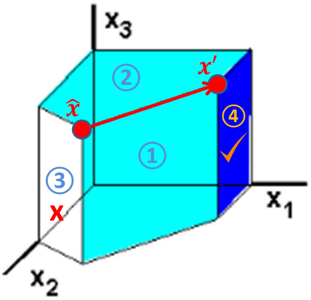
Fix a vertex $\hat{x}$. Let the d hyperplanes/constraints tight at $\hat{x}$ be,

$$\sum_{j=1}^{d} a_{ij} x_j = b_i, \quad 1 \le i \le d \quad \text{Equivalently, } \hat{A}x = \hat{b}$$

A neighbor vertex x' is connected to $\hat{x}$ by an edge.

$d - 1$ hyperplanes tight on this edge.

To reach x', one hyperplane has to be relaxed, while maintaining other $d - 1$ tight.



Moving to a Neighbor (Contd.)

$$-\hat{A}^{-1} = \begin{bmatrix} \vdots & & \vdots \\ d_1 & \ldots & d_d \\ \vdots & & \vdots \end{bmatrix}$$

Lemma

Moving in direction d_i from $\hat{x}$, all except constraint i remain tight.

Proof.

For a small $\epsilon > 0$, let $y = \hat{x} + \epsilon(d_i)$, then

$$\hat{A}y = \hat{A}(\hat{x} + \epsilon d_i) \;=\; \hat{A}\hat{x} + \epsilon\hat{A}(-\hat{A}^{-1})_{(.,i)}$$

Moving to a Neighbor (Contd.)

$$-\hat{A}^{-1} = \begin{bmatrix} \vdots & & \vdots \\ d_1 & \dots & d_d \\ \vdots & & \vdots \end{bmatrix}$$

Lemma

Moving in direction d_i from $\hat{x}$, all except constraint i remain tight.

Proof.

For a small $\epsilon > 0$, let $y = \hat{x} + \epsilon(d_i)$, then

$$\begin{aligned} \hat{A}y = \hat{A}(\hat{x} + \epsilon d_i) &= \hat{A}\hat{x} + \epsilon\hat{A}(-\hat{A}^{-1})_{(.,i)} \\ &= \hat{b} + \epsilon[0, \dots, -1, \dots, 0]^T \end{aligned}$$

Clearly, $\sum_j a_{kj} y_j = b_k, \forall k \neq i$, and $\sum_j a_{ij} y_j = b_i - \epsilon < b_i$. □

Computing the Neighbor

Move in d_i direction from $\hat{x}$, i.e., $\hat{x} + \epsilon d_i$, and STOP when hit a new hyperplane!

Need to ensure feasibility. Above lemma implies inequalities 1 through d will be satisfied. For any $k > d$, where A_k is k^{th} row of A,

$$\begin{aligned} A_k \cdot (\hat{x} + \epsilon d_i) \leq b_k &\Rightarrow (A_k \cdot \hat{x}) + \epsilon (A_k \cdot d_i) \leq b_k \\ &\Rightarrow \epsilon (A_k \cdot d_i) \leq b_k - (A_k \cdot \hat{x}) \end{aligned}$$

Computing the Neighbor

Move in d_i direction from $\hat{x}$, i.e., $\hat{x} + \epsilon d_i$, and STOP when hit a new hyperplane!

Need to ensure feasibility. Above lemma implies inequalities 1 through d will be satisfied. For any $k > d$, where A_k is k^{th} row of A,

$$\begin{array}{rcl} A_k \cdot (\hat{x} + \epsilon d_i) \leq b_k & \Rightarrow & (A_k \cdot \hat{x}) + \epsilon(A_k \cdot d_i) \leq b_k \\ & \Rightarrow & \epsilon(A_k \cdot d_i) \leq b_k - (A_k \cdot \hat{x}) \\ (\text{If } (A_k \cdot d_i) > 0) & \Rightarrow & \epsilon \leq \frac{b_k - (A_k \cdot \hat{X})}{A_k \cdot d_i} \end{array}$$

Computing the Neighbor

Move in d_i direction from $\hat{x}$, i.e., $\hat{x} + \epsilon d_i$, and STOP when hit a new hyperplane!

Need to ensure feasibility. Above lemma implies inequalities 1 through d will be satisfied. For any $k > d$, where A_k is k^{th} row of A,

$$\begin{array}{rcl} A_k \cdot (\hat{x} + \epsilon d_i) \leq b_k & \Rightarrow & (A_k \cdot \hat{x}) + \epsilon(A_k \cdot d_i) \leq b_k \\ & \Rightarrow & \epsilon(A_k \cdot d_i) \leq b_k - (A_k \cdot \hat{x}) \\ \text{(If } (A_k \cdot d_i) > 0\text{)} & \Rightarrow & \epsilon \leq \frac{b_k - (A_k \cdot \hat{X})}{A_k \cdot d_i} \quad \text{(positive)} \end{array}$$

If moving towards hyperplane k

Computing the Neighbor

Move in d_i direction from $\hat{x}$, i.e., $\hat{x} + \epsilon d_i$, and STOP when hit a new hyperplane!

Need to ensure feasibility. Above lemma implies inequalities 1 through d will be satisfied. For any $k > d$, where A_k is k^{th} row of A,

$$
\begin{aligned}
A_k \cdot (\hat{x} + \epsilon d_i) \leq b_k \quad & \Rightarrow \quad (A_k \cdot \hat{x}) + \epsilon(A_k \cdot d_i) \leq b_k \\
& \Rightarrow \quad \epsilon(A_k \cdot d_i) \leq b_k - (A_k \cdot \hat{x}) \\
\text{(If } (A_k \cdot d_i) > 0\text{)} \quad & \Rightarrow \quad \epsilon \leq \frac{b_k - (A_k \cdot \hat{X})}{A_k \cdot d_i} \quad \text{(positive)}
\end{aligned}
$$

If moving towards hyperplane k

$$
\text{(If } (A_k \cdot d_i) < 0\text{)} \quad \Rightarrow \quad \epsilon \geq \frac{b_k - (A_k \cdot \hat{X})}{A_k \cdot d_i} \quad \text{(negative)}
$$

If moving away from hyperplane k.

No upper bound, and -ve lower bound!

Computing the Neighbor

Algorithm

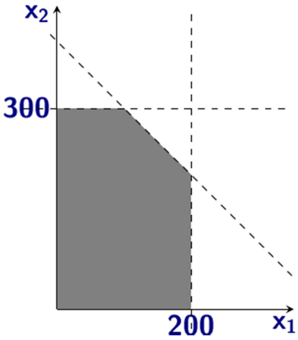
NextVertex($\hat{x}, d_i$)
Set $\epsilon \leftarrow \infty$.
For $k = d + 1 \dots n$
  $\epsilon' \leftarrow \frac{b_k - (A_k \cdot \hat{x})}{A_k \cdot d_i}$
  If $\epsilon' > 0$ and $\epsilon' < \epsilon$ then
    set $\epsilon \leftarrow \epsilon'$
If $\epsilon < \infty$ then return $\hat{x} + \epsilon d_i$.
Else return *null*.

If $(c \cdot d_i) > 0$ then the algorithm returns an *improving* neighbor.

Factory Example

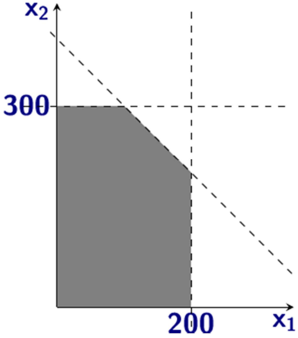
$$\hat{x} = (0, 0)$$

$$\begin{aligned} \textbf{max}: \quad & x_1 + 6x_2 \\ s.t. \quad & 0 \leq x_1 \leq 200 \\ & 0 \leq x_2 \leq 300 \\ & x_1 + x_2 \leq 400 \end{aligned}$$



Factory Example

$$\begin{aligned} \textbf{max}: \quad & x_1 + 6x_2 \\ s.t. \quad & 0 \le x_1 \le 200 \\ & 0 \le x_2 \le 300 \\ & x_1 + x_2 \le 400 \end{aligned}$$



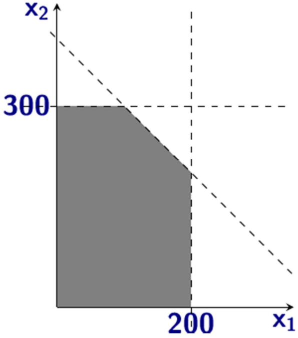
$$\hat{x} = (0, 0)$$

$$\hat{A} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$-\hat{A}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = [d_1 \; d_2]$$

Factory Example

$$\begin{aligned} \textbf{max}: \quad & x_1 + 6x_2 \\ s.t. \quad & 0 \le x_1 \le 200 \\ & 0 \le x_2 \le 300 \\ & x_1 + x_2 \le 400 \end{aligned}$$



$$\hat{x} = (0, 0)$$

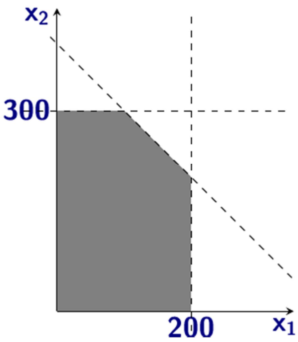
$$\hat{A} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$-\hat{A}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = [d_1 \; d_2]$$

Moving in direction d_1 gives $(200, 0)$

Factory Example

$$\begin{aligned} \max: \quad & x_1 + 6x_2 \\ s.t. \quad & 0 \le x_1 \le 200 \\ & 0 \le x_2 \le 300 \\ & x_1 + x_2 \le 400 \end{aligned}$$



$\hat{x} = (0, 0)$

$$\hat{A} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

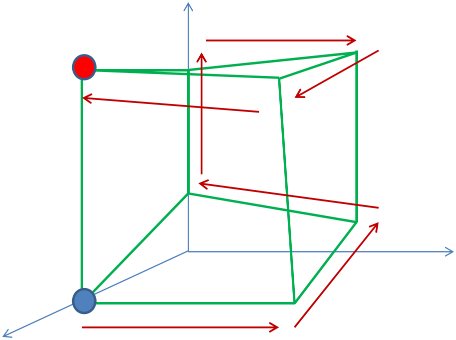
$$-\hat{A}^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = [d_1 \ d_2]$$

Moving in direction d_1 gives $(200, 0)$

Moving in direction d_2 gives $(0, 300)$.

Solving Linear Programming in Practice

1. Naïve implementation of Simplex algorithm can be very inefficient – Exponential number of steps!



Solving Linear Programming in Practice

1. Naïve implementation of Simplex algorithm can be very inefficient
   1. Choosing which neighbor to move to can significantly affect running time
   2. Very efficient Simplex-based algorithms exist
   3. Simplex algorithm takes exponential time in the worst case but works extremely well in practice with many improvements over the years
2. Non Simplex based methods like interior point methods work well for large problems.

Issues

1. Starting vertex
2. The linear program could be infeasible: No point satisfy the constraints.
3. The linear program could be unbounded: Polygon unbounded in the direction of the objective function.
4. More than d hyperplanes could be tight at a vertex, forming more than d neighbors.

Computing the Starting Vertex

Equivalent to solving another LP!

Find an x such that $Ax \leq b$.
If $b \geq \mathbf{0}$ then trivial!

Computing the Starting Vertex

Equivalent to solving another LP!

Find an x such that $Ax \leq b$.
If $b \geq \mathbf{0}$ then trivial! $x = \mathbf{0}$. Otherwise.

Computing the Starting Vertex

Equivalent to solving another LP!

Find an x such that $Ax \leq b$.
If $b \geq \mathbf{0}$ then trivial! $x = \mathbf{0}$. Otherwise.

$$\begin{array}{ll} \textbf{min}: & s \\ s.t. & \sum_j a_{ij}x_j - s \leq b_i, \quad \forall i \\ & s \geq \mathbf{0} \end{array}$$

Trivial feasible solution:

Computing the Starting Vertex

Equivalent to solving another LP!

Find an x such that $Ax \le b$.
If $b \ge \mathbf{0}$ then trivial! $x = \mathbf{0}$. Otherwise.

$$
\begin{array}{ll}
\textbf{min :} & s \\
s.t. & \sum_j a_{ij}x_j - s \le b_i, \quad \forall i \\
 & s \ge \mathbf{0}
\end{array}
$$

Trivial feasible solution: $x = \mathbf{0}$, $s = |\min_i b_i|$.

Computing the Starting Vertex

Equivalent to solving another LP!

Find an x such that $Ax \leq b$.
If $b \geq \mathbf{0}$ then trivial! $x = \mathbf{0}$. Otherwise.

$$\begin{array}{ll} \textbf{min}: & s \\ s.t. & \sum_j a_{ij} x_j - s \leq b_i, \quad \forall i \\ & s \geq \mathbf{0} \end{array}$$

Trivial feasible solution: $x = \mathbf{0}$, $s = |\,\mathbf{min}_i\, b_i|$.

If $Ax \leq b$ feasible then optimal value of the above LP is $s = \mathbf{0}$.

Computing the Starting Vertex

Equivalent to solving another LP!

Find an x such that $Ax \leq b$.
If $b \geq \mathbf{0}$ then trivial! $x = \mathbf{0}$. Otherwise.

$$\begin{aligned} \textbf{min}: \quad & s \\ s.t. \quad & \textstyle\sum_j a_{ij}x_j - s \leq b_i, \quad \forall i \\ & s \geq \mathbf{0} \end{aligned}$$

Trivial feasible solution: $x = \mathbf{0}$, $s = |\min_i b_i|$.

If $Ax \leq b$ feasible then optimal value of the above LP is $s = \mathbf{0}$.

Checks Feasibility!

Unboundedness: Example

$$\begin{aligned} \text{maximize} \quad & x_2 \\ x_1 + x_2 &\geq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Unboundedness depends on both constraints and the objective function.

Unboundedness: Example

$$\begin{aligned} \text{maximize} \quad x_2 & \\ x_1 + x_2 &\geq 2 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Unboundedness depends on both constraints and the objective function.

If unbounded in the direction of objective function, then the pivoting step in the simplex will detect it.

Degeneracy and Cycling

More than d constraints are tight at vertex $\hat{x}$. Say $d + 1$.

Suppose, we pick first d to form $\hat{A}$ such that $\hat{A}\hat{x} = \hat{b}$, and compute directions $d_1, \dots, d_d$.

Degeneracy and Cycling

More than d constraints are tight at vertex $\hat{x}$. Say $d + 1$.

Suppose, we pick first d to form $\hat{A}$ such that $\hat{A}\hat{x} = \hat{b}$, and compute directions $d_1, \ldots, d_d$.

Then NextVertex($\hat{x}, d_i$) will encounter $(d + 1)^{th}$ constraint tight at $\hat{x}$ and return the same vertex. Hence we are back to $\hat{x}$!

Degeneracy and Cycling

More than d constraints are tight at vertex $\hat{x}$. Say $d + 1$.

Suppose, we pick first d to form $\hat{A}$ such that $\hat{A}\hat{x} = \hat{b}$, and compute directions $d_1, \ldots, d_d$.

Then NextVertex($\hat{x}, d_i$) will encounter $(d + 1)^{th}$ constraint tight at $\hat{x}$ and return the same vertex. Hence we are back to $\hat{x}$!

Same phenomenon will repeat!

Degeneracy and Cycling

More than d constraints are tight at vertex $\hat{x}$. Say $d+1$.

Suppose, we pick first d to form $\hat{A}$ such that $\hat{A}\hat{x} = \hat{b}$, and compute directions $d_1, \ldots, d_d$.

Then NextVertex($\hat{x}, d_i$) will encounter $(d+1)^{th}$ constraint tight at $\hat{x}$ and return the same vertex. Hence we are back to $\hat{x}$!

Same phenomenon will repeat!

This can be avoided by adding small random perturbation to b_is.