The simplex algorithm and the Hirsch conjecture: **Lecture 1**

Thomas Dueholm Hansen

MADALGO & CTIC Summer School

August 8, 2011

madalgo
CENTER FOR MASSIVE DATA ALGORITHMICS

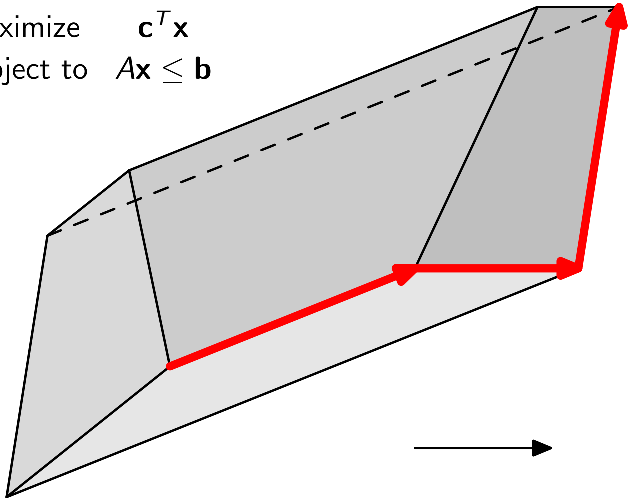
CTIC
交互计算

Overview

- **Lecture 1:**
  - Introduction to linear programming and the simplex algorithm.
  - Pivoting rules.
  - The RandomFacet pivoting rule.
- **Lecture 2:**
  - The Hirsch conjecture.
  - Introduction to Markov decision processes (MDPs).
  - Upper bound for the LargestCoefficient pivoting rule for MDPs.
- **Lecture 3:**
  - Lower bounds for pivoting rules utilizing MDPs. Example: Bland's rule.
  - Lower bound for the RandomEdge pivoting rule.
  - Abstractions and related problems.

The simplex algorithm, Dantzig (1947)

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \end{array}$$

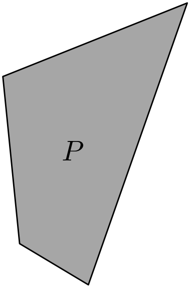


Convex polytopes

- A **convex polytope** (or **polyhedron**) P in dimension d is a set of points

$$P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$$

where A is an $n \times d$ matrix and b is a vector in $\mathbb{R}^n$.

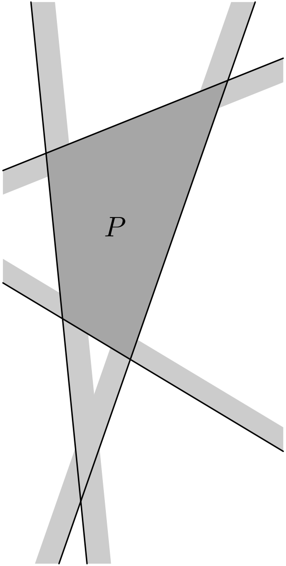


Convex polytopes

- A **convex polytope** (or **polyhedron**) P in dimension d is a set of points

$$P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$$

  where A is an $n \times d$ matrix and b is a vector in $\mathbb{R}^n$.
- I.e., P is the intersection of n halfspaces $a_i^T x \leq b_i$, where a_i^T is the i'th row of A.

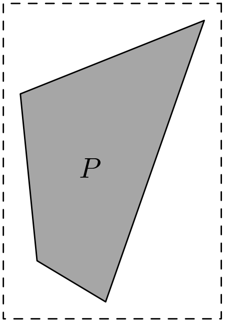


Convex polytopes

- A **convex polytope** (or **polyhedron**) P in dimension d is a set of points

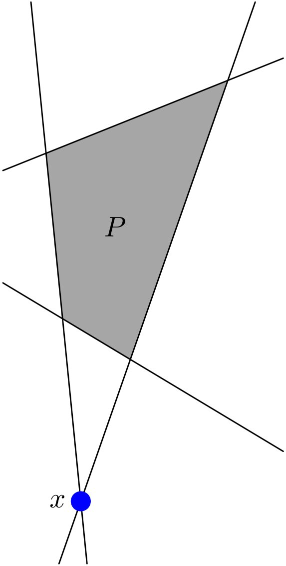
$$P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$$

  where A is an $n \times d$ matrix and b is a vector in $\mathbb{R}^n$.
- I.e., P is the intersection of n halfspaces $a_i^T x \leq b_i$, where a_i^T is the i'th row of A.
- P is **bounded** if there exists a constant K such that for all $x \in P$, the absolute value of every component x_i is at most K. Otherwise P is **unbounded**.



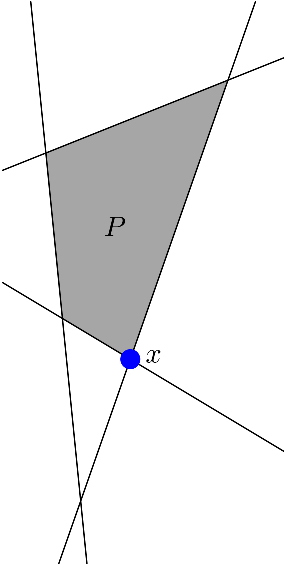
Convex polytopes

- A point $x \in \mathbb{R}^d$ is a **basic solution** if it satisfies d linearly independent constraints, $a_i^T x \leq b_i$, with equality.



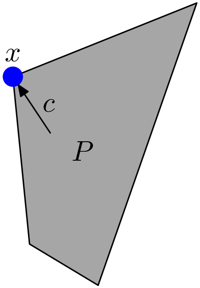
Convex polytopes

- A point $x \in \mathbb{R}^d$ is a **basic solution** if it satisfies d linearly independent constraints, $a_i^T x \leq b_i$, with equality.
- $x \in \mathbb{R}^d$ is a **basic feasible solution** if $x \in P$ and x is a basic solution.



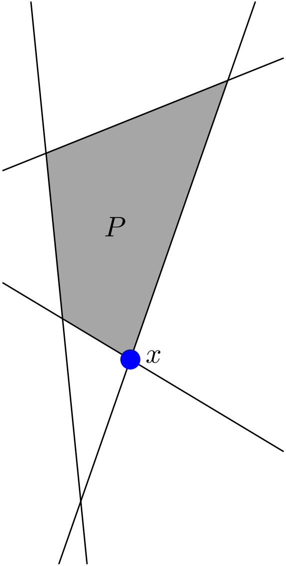
Convex polytopes

- A point $x \in \mathbb{R}^d$ is a **basic solution** if it satisfies d linearly independent constraints, $a_i^T x \leq b_i$, with equality.
- $x \in \mathbb{R}^d$ is a **basic feasible solution** if $x \in P$ and x is a basic solution.
- $x \in P$ is a **vertex** (or corner) if there exists a vector $c \in \mathbb{R}^d$ such that for all $y \in P$, if $y \neq x$ then $c^T x > c^T y$.



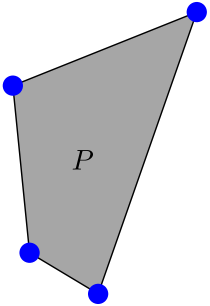
Convex polytopes

- A point $x \in \mathbb{R}^d$ is a **basic solution** if it satisfies d linearly independent constraints, $a_i^T x \leq b_i$, with equality.
- $x \in \mathbb{R}^d$ is a **basic feasible solution** if $x \in P$ and x is a basic solution.
- $x \in P$ is a **vertex** (or corner) if there exists a vector $c \in \mathbb{R}^d$ such that for all $y \in P$, if $y \neq x$ then $c^T x > c^T y$.
- Every basic feasible solution x is a vertex of P.



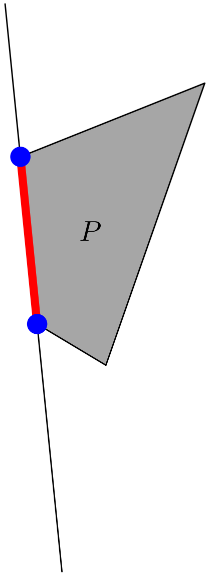
Convex polytopes

- If P is bounded, P can be equivalently defined as the convex hull of its vertices.



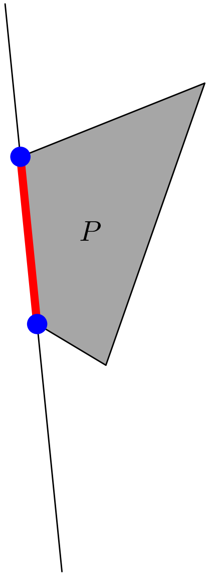
Convex polytopes

- If P is bounded, P can be equivalently defined as the convex hull of its vertices.
- A k-**face** is a k dimensional polytope defined by a set of vertices that satisfy the same $d - k$ constraints with equality.
  - A 0-face is a vertex.
  - A 1-face is an **edge**.
  - A $(d-1)$-face is a **facet**.



Convex polytopes

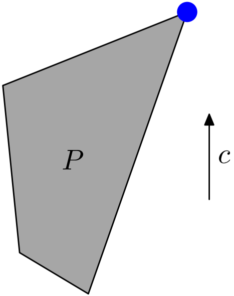
- If P is bounded, P can be equivalently defined as the convex hull of its vertices.
- A k-**face** is a k dimensional polytope defined by a set of vertices that satisfy the same $d - k$ constraints with equality.
  - A 0-face is a vertex.
  - A 1-face is an **edge**.
  - A $(d - 1)$-face is a **facet**.
- Alternatively, a k-face is the polytope obtained by eliminating $d - k$ variables using the $d - k$ constraints that are satisfied with equality.



Linear programming

- A **linear program** (LP) is the optimization problem:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & \leq & b \end{array}$$



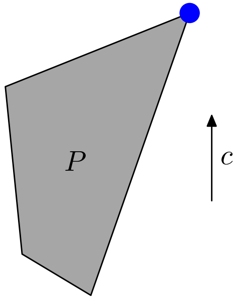
- A **linear program** (LP) is the optimization problem:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & \leq & b \end{array}$$

- For simplicity, we generally assume that a linear program is in **canonical form**:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & \leq & b \\ & x & \geq & 0 \end{array}$$

- Every linear program has an equivalent canonical form.



Linear programming

- A constraint $a_i^T x \leq b_i$ can be expressed equivalently as $(a_i^T x) + s_i = b_i$, where $s_i \geq 0$ is a non-negative **slack variable**.

Linear programming

- A constraint $a_i^T x \leq b_i$ can be expressed equivalently as $(a_i^T x) + s_i = b_i$, where $s_i \geq 0$ is a non-negative **slack variable**.
- A canonical form linear program can be transformed to **equational form** (or **standard form**) by introducing n slack variables:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & \leq & b \\ & x & \geq & 0 \end{array} \qquad \begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax + Is & = & b \\ & x, s & \geq & 0 \end{array}$$

The resulting linear program has $m = d + n$ non-negative variables.

- A constraint $a_i^T x \leq b_i$ can be expressed equivalently as $(a_i^T x) + s_i = b_i$, where $s_i \geq 0$ is a non-negative **slack variable**.
- A canonical form linear program can be transformed to **equational form** (or **standard form**) by introducing n slack variables:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & \leq & b \\ & x & \geq & 0 \end{array} \qquad \begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax + Is & = & b \\ & x, s & \geq & 0 \end{array}$$

  The resulting linear program has $m = d + n$ non-negative variables.

- Note that an inequality from the original linear program is satisfied with equality if the corresponding (slack) variable is zero in the equational form.

Linear programming

- Consider a linear program in equational form, defined by an $n \times m$ matrix A, with $m = d + n$:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

Linear programming

- Consider a linear program in equational form, defined by an $n \times m$ matrix A, with $m = d + n$:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

- A **basis** is a subset $B \subseteq \{1, \ldots, m\}$ of n linearly independent columns of A.

Linear programming

- Consider a linear program in equational form, defined by an $n \times m$ matrix A, with $m = d + n$:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

- A **basis** is a subset $B \subseteq \{1, \ldots, m\}$ of n linearly independent columns of A.
- Every basis B defines a basic solution $x_B \in \mathbb{R}^m$, as the unique solution to:

$$Ax = b \quad \text{and} \quad \forall i \notin B : \; x_i = 0$$

Linear programming

- Consider a linear program in equational form, defined by an $n \times m$ matrix A, with $m = d + n$:

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

- A **basis** is a subset $B \subseteq \{1, \ldots, m\}$ of n linearly independent columns of A.
- Every basis B defines a basic solution $x_B \in \mathbb{R}^m$, as the unique solution to:

$$Ax = b \quad \text{and} \quad \forall i \notin B: \; x_i = 0$$

- Every basic solution $x \in \mathbb{R}^m$ is defined by at least one basis. If x is defined by more than one basis, x is a **degenerate** basic solution.

Linear programming

- A basic solution x_B is feasible if $x_B \geq 0$.

- A basic solution x_B is feasible if $x_B \geq 0$.
- For some basis B, the variables x_i, for $i \in B$, are called **basic**, and the remaining variables are called **non-basic**.

- A basic solution x_B is feasible if $x_B \geq 0$.
- For some basis B, the variables x_i, for $i \in B$, are called **basic**, and the remaining variables are called **non-basic**.
  - If x_B is a basic feasible solution, then the non-basic variables correspond to facets defining the vertex.

- A basic solution x_B is feasible if $x_B \geq 0$.
- For some basis B, the variables x_i, for $i \in B$, are called **basic**, and the remaining variables are called **non-basic**.
  - If x_B is a basic feasible solution, then the non-basic variables correspond to facets defining the vertex.
- The operation of exchanging a single basic variable in B with a non-basic variable, producing a new basis B', is called **pivoting**.

Linear programming

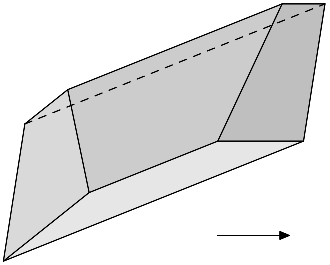
- A basic solution x_B is feasible if $x_B \geq 0$.
- For some basis B, the variables x_i, for $i \in B$, are called **basic**, and the remaining variables are called **non-basic**.
  - If x_B is a basic feasible solution, then the non-basic variables correspond to facets defining the vertex.
- The operation of exchanging a single basic variable in B with a non-basic variable, producing a new basis B', is called **pivoting**.
  - Geometrically, if x_B and $x_{B'}$ are different basic feasible solutions, pivoting corresponds to moving from x_B to $x_{B'}$ along an edge of the polytope.

The simplex algorithm, Dantzig (1947):

- Start with some basis B corresponding to a basic feasible solution x_B.
- Repeatedly perform pivots leading to new bases B' corresponding to basic feasible solutions $x_{B'}$ with better values, $c^T x_{B'} \geq c^T x_B$.
- Stop when no pivot can increase the value further.

Example: The tableau method

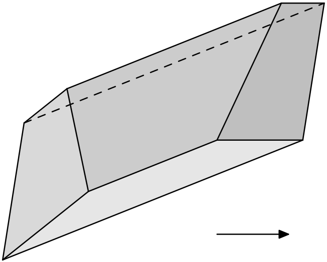
$$\begin{array}{lrcl}
\max & 2x_1 - 2x_2 - x_3 & & \\
\text{s.t.} & \frac{1}{3}x_1 - \frac{2}{3}x_2 - \frac{2}{3}x_3 & \leq & 1 \\
& x_2 + x_3 & \leq & 2 \\
& x_3 & \leq & 1 \\
& x_1, x_2, x_3 & \geq & 0
\end{array}$$



- Transform a linear program in canonical form to equational form by introducing slack variables.

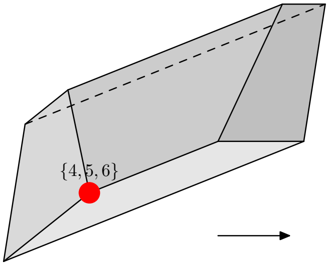
Example: The tableau method

$$
\begin{array}{lrcl}
\max & 2x_1 - 2x_2 - x_3 & & \\
\text{s.t.} & \frac{1}{3}x_1 - \frac{2}{3}x_2 - \frac{2}{3}x_3 + x_4 & = & 1 \\
& x_2 + x_3 + x_5 & = & 2 \\
& x_3 + x_6 & = & 1 \\
& x_1, x_2, x_3, x_4, x_5, x_6 & \geq & 0
\end{array}
$$



- Transform a linear program in canonical form to equational form by introducing slack variables.

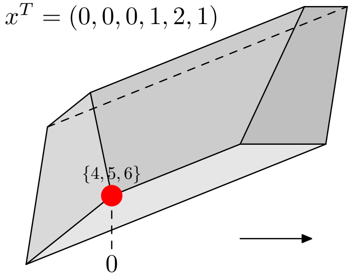
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{2x_1 - 2x_2 - x_3} \\ \text{s.t.} & x_4 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_2 + \frac{2}{3}x_3 \\ & x_5 & = & 2 - x_2 - x_3 \\ & x_6 & = & 1 - x_3 \\ & \multicolumn{2}{r}{x_1, x_2, x_3, x_4, x_5, x_6 \geq} & 0 \end{array}$$



- Transform a linear program in canonical form to equational form by introducing slack variables.
- Pick a basis, in this case $\{4, 5, 6\}$, and express the basic variables and the objective function in terms of non-basic variables. This representation is called a **tableau**.

Example: The tableau method

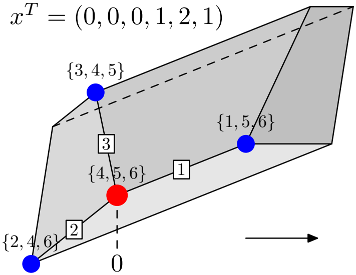
$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{2x_1 - 2x_2 - x_3} \\
\text{s.t.} & x_4 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_2 + \frac{2}{3}x_3 \\
 & x_5 & = & 2 - x_2 - x_3 \\
 & x_6 & = & 1 - x_3 \\
 & x_1, x_2, x_3, x_4, x_5, x_6 & \geq & 0
\end{array}
$$



- Transform a linear program in canonical form to equational form by introducing slack variables.
- Pick a basis, in this case $\{4,5,6\}$, and express the basic variables and the objective function in terms of non-basic variables. This representation is called a **tableau**.
- The corresponding basic solution and its value can be read by setting the non-basic variables to zero.

Example: The tableau method

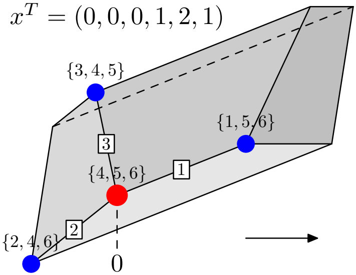
$$
\begin{array}{ll}
\max & 2x_1 - 2x_2 - x_3 \\
\text{s.t.} & x_4 \;=\; 1 - \frac{1}{3}x_1 + \frac{2}{3}x_2 + \frac{2}{3}x_3 \\
& x_5 \;=\; 2 - x_2 - x_3 \\
& x_6 \;=\; 1 - x_3 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \;\geq\; 0
\end{array}
$$



- If the coefficient of a non-basic variable x_i in the objective function is positive, increasing x_i will improve the value.

Example: The tableau method

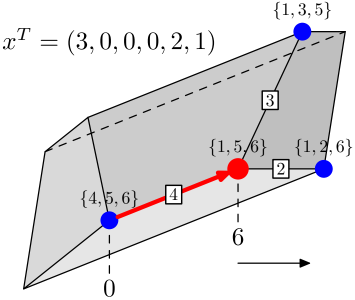
$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{2x_1 - 2x_2 - x_3} \\
\text{s.t.} & x_4 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_2 + \frac{2}{3}x_3 \\
& x_5 & = & 2 - x_2 - x_3 \\
& x_6 & = & 1 - x_3 \\
& \multicolumn{2}{r}{x_1, x_2, x_3, x_4, x_5, x_6 \;\geq} & 0
\end{array}
$$



- If the coefficient of a non-basic variable x_i in the objective function is positive, increasing x_i will improve the value.
- x_i can be increased until another basic variable x_j becomes zero, which completes the pivot. The basis is then updated by exchanging i and j. If no variable becomes zero the value is **unbounded**.

Example: The tableau method

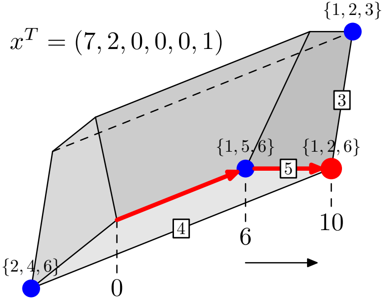
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{6 + 2x_2 + 3x_3 - 6x_4} \\ \text{s.t.} & x_1 & = & 3 + 2x_2 + 2x_3 - 3x_4 \\ & x_5 & = & 2 - x_2 - x_3 \\ & x_6 & = & 1 - x_3 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \ \geq \ 0} \end{array}$$



- If the coefficient of a non-basic variable x_i in the objective function is positive, increasing x_i will improve the value.
- x_i can be increased until another basic variable x_j becomes zero, which completes the pivot. The basis is then updated by exchanging i and j. If no variable becomes zero the value is **unbounded**.

Example: The tableau method

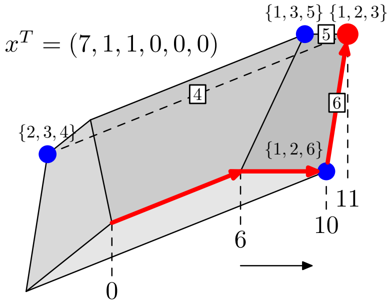
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{10 + x_3 - 6x_4 - 2x_5} \\ \text{s.t.} & x_1 & = & 7 - 3x_4 - 2x_5 \\ & x_2 & = & 2 - x_3 - x_5 \\ & x_6 & = & 1 - x_3 \\ & x_1, x_2, x_3, x_4, x_5, x_6 & \geq & 0 \end{array}$$



- If the coefficient of a non-basic variable x_i in the objective function is positive, increasing x_i will improve the value.
- x_i can be increased until another basic variable x_j becomes zero, which completes the pivot. The basis is then updated by exchanging i and j. If no variable becomes zero the value is **unbounded**.

Example: The tableau method

$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{11 - 6x_4 - 2x_5 - x_6} \\
\text{s.t.} & x_1 & = & 7 - 3x_4 - 2x_5 \\
& x_2 & = & 1 - x_5 + x_6 \\
& x_3 & = & 1 - x_6 \\
& \multicolumn{2}{r}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq 0
\end{array}
$$



- If the coefficient of a non-basic variable x_i in the objective function is positive, increasing x_i will improve the value.
- x_i can be increased until another basic variable x_j becomes zero, which completes the pivot. The basis is then updated by exchanging i and j. If no variable becomes zero the value is **unbounded**.
- When all coefficients are negative, the solution is optimal.

The tableau method, formally

- Let B be a basis, and let $A = [A_B \mid A_{\bar{B}}]$ and $x^T = [x_B^T \mid x_{\bar{B}}^T]$. I.e., A_B is the matrix of basic columns and $A_{\bar{B}}$ is the matrix of non-basic columns, and similar for x.
- The tableau method rewrites the linear program:

$$\begin{array}{lrcl} \max & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array} \qquad\qquad \begin{array}{lrcl} \max & c_B^T A_B^{-1} b \; + \; \bar{c}^T x & & \\ s.t. & x_B & = & A_B^{-1} b - A_B^{-1} A_{\bar{B}} x_{\bar{B}} \\ & x & \geq & 0 \end{array}$$

where $\bar{c} \in \mathbb{R}^m$ is the vector of **reduced costs**:

$$\bar{c} = c - (A_B^{-1} A)^T c_B$$

Pivoting rules

- Two choices must be made when pivoting:
  1. Which non-basic variable with positive coefficient enters the basis?
  2. Which basic variable leaves the basis, in case of a tie?

  These choices are specified by a **pivoting rule**.

Pivoting rules

- Two choices must be made when pivoting:
  1. Which non-basic variable with positive coefficient enters the basis?
  2. Which basic variable leaves the basis, in case of a tie?

  These choices are specified by a **pivoting rule**.

- LargestCoefficient, Dantzig (1947)
  - The non-basic variable with largest coefficient enters the basis.

Pivoting rules

- Two choices must be made when pivoting:
  1. Which non-basic variable with positive coefficient enters the basis?
  2. Which basic variable leaves the basis, in case of a tie?

  These choices are specified by a **pivoting rule**.

- LargestCoefficient, Dantzig (1947)
  - The non-basic variable with largest coefficient enters the basis.
- LargestIncrease
  - The non-basic variable that gives the largest increase enters the basis.

Pivoting rules

- Two choices must be made when pivoting:
  1. Which non-basic variable with positive coefficient enters the basis?
  2. Which basic variable leaves the basis, in case of a tie?

  These choices are specified by a **pivoting rule**.

- LargestCoefficient, Dantzig (1947)
  - The non-basic variable with largest coefficient enters the basis.
- LargestIncrease
  - The non-basic variable that gives the largest increase enters the basis.
- SteepestEdge
  - The non-basic variable whose pivot corresponds to the edge with direction closest to c enters the basis.

Pivoting rules

- If the current basic feasible solution is degenerate, it is possible that the value does not increase when pivoting.
- Such situations may lead to cycling. The following two pivoting rules do not cycle, however.

Pivoting rules

- If the current basic feasible solution is degenerate, it is possible that the value does not increase when pivoting.
- Such situations may lead to cycling. The following two pivoting rules do not cycle, however.

- BLAND'S RULE, Bland (1977)
  - Always pick the available variable with the smallest index, both for entering and leaving the basis.

Pivoting rules

- If the current basic feasible solution is degenerate, it is possible that the value does not increase when pivoting.
- Such situations may lead to cycling. The following two pivoting rules do not cycle, however.

- Bland's rule, Bland (1977)
  - Always pick the available variable with the smallest index, both for entering and leaving the basis.
- Lexicographic rule, Dantzig, Orden and Wolfe (1955)
  - Pick any variable x_i with positive coefficient for entering the basis.
  - Pick x_j for leaving the basis such that the right-hand-side coefficients in the tableau are lexicographically smallest when divided by the coefficient of x_i in that row.
  - This corresponds to a small pertubation of the b vector.

Pivoting rules

- ShadowVertex
  - Let x_0 be some initial basic feasible solution, and let c_0 be a vector for which $c_0^T x_0$ is optimal. Define:

$$\begin{array}{lrcl} \max & (1-\lambda)c_0^T x + \lambda c^T x \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

Pivoting rules

- ShadowVertex
  - Let x_0 be some initial basic feasible solution, and let c_0 be a vector for which $c_0^T x_0$ is optimal. Define:

$$\begin{array}{lrcl} \max & (1-\lambda)c_0^T x + \lambda c^T x \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

  - Maintain an optimal solution for λ going from 0 to 1.

Pivoting rules

- SHADOWVERTEX
  - Let x_0 be some initial basic feasible solution, and let c_0 be a vector for which $c_0^T x_0$ is optimal. Define:

$$\begin{array}{lrcl} \max & (1-\lambda)c_0^T x + \lambda c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

  - Maintain an optimal solution for λ going from 0 to 1.
  - This corresponds to moving along edges of a 2-dimensional projection of the polytope (a "shadow").

Pivoting rules

- ShadowVertex
  - Let x_0 be some initial basic feasible solution, and let c_0 be a vector for which $c_0^T x_0$ is optimal. Define:

$$\begin{array}{ll} \max & (1-\lambda)c_0^T x + \lambda c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

  - Maintain an optimal solution for λ going from 0 to 1.
  - This corresponds to moving along edges of a 2-dimensional projection of the polytope (a "shadow").
  - Vertices and edges of the projection correspond to vertices and edges of the original polytope.

Pivoting rules

- ShadowVertex
  - Let x_0 be some initial basic feasible solution, and let c_0 be a vector for which $c_0^T x_0$ is optimal. Define:

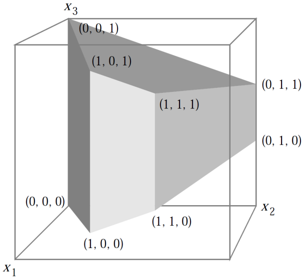
$$\begin{array}{lrcl} \max & (1-\lambda)c_0^T x + \lambda c^T x \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

  - Maintain an optimal solution for λ going from 0 to 1.
  - This corresponds to moving along edges of a 2-dimensional projection of the polytope (a "shadow").
  - Vertices and edges of the projection correspond to vertices and edges of the original polytope.
  - Spielman and Teng (2004) gave a *smoothed analysis* of the ShadowVertex pivoting rule, showing that it is polynomial under certain pertubations of the linear program.

Lower bounds

All the previous pivoting rules are known to require exponentially many steps in the worst case:

- LargestCoefficient: Klee and Minty (1972), the Klee-Minty cube[1].
- LargestIncrease: Jeroslow (1973).
- SteepestEdge: Goldfarb and Sit (1979).
- Bland's rule: Avis and Chvátal (1978).
- ShadowVertex: Murty (1980), Goldfarb (1983).
- Amenta and Ziegler (1996) gave a unified view of all these lower bounds.



[1] Picture from Gärtner, Henk and Ziegler (1998)

More pivoting rules

- RandomEdge
  - Let a uniformly random non-basic variable with positive coefficient enter the basis.

More pivoting rules

- RandomEdge
  - Let a uniformly random non-basic variable with positive coefficient enter the basis.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a uniformly random facet that contains the current vertex, and recursively find an optimal solution within that facet. If possible, make an improving pivot leaving the facet and repeat.

More pivoting rules

- RandomEdge
  - Let a uniformly random non-basic variable with positive coefficient enter the basis.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a uniformly random facet that contains the current vertex, and recursively find an optimal solution within that facet. If possible, make an improving pivot leaving the facet and repeat.
  - This randomized pivoting rule finds an optimal solution in an expected subexponential, $2^{O(\sqrt{(n-d)\log n})}$, number of steps.

More pivoting rules

- RandomEdge
  - Let a uniformly random non-basic variable with positive coefficient enter the basis.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a uniformly random facet that contains the current vertex, and recursively find an optimal solution within that facet. If possible, make an improving pivot leaving the facet and repeat.
  - This randomized pivoting rule finds an optimal solution in an expected subexponential, $2^{O(\sqrt{(n-d)\log n})}$, number of steps.
- Randomized Bland's rule
  - Reorder the indices of the variables according to a random permutation and use Bland's rule.

More pivoting rules

- RandomEdge
  - Let a uniformly random non-basic variable with positive coefficient enter the basis.
- RandomFacet, Kalai (1992) and Matoušek, Sharir and Welzl (1992)
  - Pick a uniformly random facet that contains the current vertex, and recursively find an optimal solution within that facet. If possible, make an improving pivot leaving the facet and repeat.
  - This randomized pivoting rule finds an optimal solution in an expected subexponential, $2^{O(\sqrt{(n-d)\log n})}$, number of steps.
- Randomized Bland's rule
  - Reorder the indices of the variables according to a random permutation and use Bland's rule.
- LeastEntered, Zadeh (1980)
  - Pick the non-basic variable that has previously entered the basis the fewest number of times.

New lower bounds

- Friedmann, Hansen and Zwick (2011) proved lower bounds of subexponential form ($2^{\Omega(d^\alpha)}$, for $\alpha < 1$) for the worst-case expected number of steps of the pivoting rules:
  - RandomEdge
  - RandomFacet
  - Randomized Bland's rule
- Friedmann (2011) proved a subexponential lower bound for the worst-case number of steps required for the LeastEntered pivoting rule.
- These lower bounds are based on a tight connection between **Markov decision processes** and linear programs.

Solving linear programs

- Linear programs can be solved in polynomial time:
  - Khachiyan (1979): The ellipsoid method
  - Karmarkar (1984): The interior point method
  - Best complexity results - Renegar (1988), Gonzaga (1989), Roos and Vial (1990): $O(n^3 L)$ arithmetic operations, where L is the bit complexity. Based on the interior point method.

Solving linear programs

- Linear programs can be solved in polynomial time:
  - Khachiyan (1979): The ellipsoid method
  - Karmarkar (1984): The interior point method
  - Best complexity results - Renegar (1988), Gonzaga (1989), Roos and Vial (1990): $O(n^3 L)$ arithmetic operations, where L is the bit complexity. Based on the interior point method.
- Let $T(d, n)$ and $T_R(d, n)$ be the maximum (expected) number of arithmetic operations required for deterministic and randomized algorithms, respectively, for solving any linear program in d-space with n constraints.

Solving linear programs

- Linear programs can be solved in polynomial time:
  - Khachiyan (1979): The ellipsoid method
  - Karmarkar (1984): The interior point method
  - Best complexity results - Renegar (1988), Gonzaga (1989), Roos and Vial (1990): $O(n^3L)$ arithmetic operations, where L is the bit complexity. Based on the interior point method.
- Let $T(d,n)$ and $T_R(d,n)$ be the maximum (expected) number of arithmetic operations required for deterministic and randomized algorithms, respectively, for solving any linear program in d-space with n constraints.
- The best bound for $T_R(d,n)$ is subexponential in d. No subexponential bound is known for $T(d,n)$.

Solving linear programs

- Linear programs can be solved in polynomial time:
  - Khachiyan (1979): The ellipsoid method
  - Karmarkar (1984): The interior point method
  - Best complexity results - Renegar (1988), Gonzaga (1989), Roos and Vial (1990): $O(n^3 L)$ arithmetic operations, where L is the bit complexity. Based on the interior point method.
- Let $T(d,n)$ and $T_R(d,n)$ be the maximum (expected) number of arithmetic operations required for deterministic and randomized algorithms, respectively, for solving any linear program in d-space with n constraints.
- The best bound for $T_R(d,n)$ is subexponential in d. No subexponential bound is known for $T(d,n)$.
- Finding a (strongly) polynomial bound for $T(d,n)$ and $T_R(d,n)$ is a major open problem in linear programming.

Solving linear programs

- Linear programs can be solved in polynomial time:
  - Khachiyan (1979): The ellipsoid method
  - Karmarkar (1984): The interior point method
  - Best complexity results - Renegar (1988), Gonzaga (1989), Roos and Vial (1990): $O(n^3 L)$ arithmetic operations, where L is the bit complexity. Based on the interior point method.
- Let $T(d,n)$ and $T_R(d,n)$ be the maximum (expected) number of arithmetic operations required for deterministic and randomized algorithms, respectively, for solving any linear program in d-space with n constraints.
- The best bound for $T_R(d,n)$ is subexponential in d. No subexponential bound is known for $T(d,n)$.
- Finding a (strongly) polynomial bound for $T(d,n)$ and $T_R(d,n)$ is a major open problem in linear programming.
  - A polynomially bounded pivoting rule that performs each step in polynomial time would give such a bound.

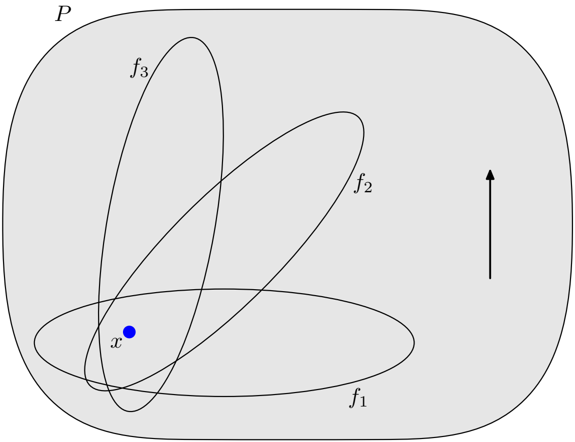
- RandomFacet, Kalai (1992):
  1. Pick a uniformly random facet f that contains the current basic feasible solution x.
  2. Recursively find the optimal solution x' within the picked facet f.
  3. If possible, make an improving pivot from x', leaving the facet f, and repeat from (1). Otherwise return x'.

The RandomFacet pivoting rule

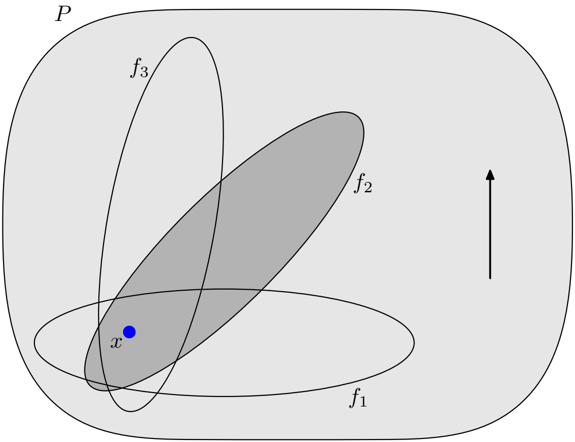
- RandomFacet, Kalai (1992):
  1. Pick a uniformly random facet f that contains the current basic feasible solution x.
  2. Recursively find the optimal solution x' within the picked facet f.
  3. If possible, make an improving pivot from x', leaving the facet f, and repeat from (1). Otherwise return x'.

- A **dual** variant of the RandomFacet pivoting rule was discovered independently by Matoušek, Sharir and Welzl (1992).

The RANDOMFACET pivoting rule

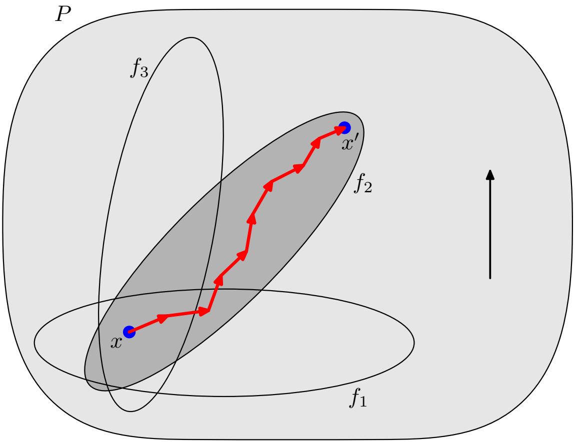


- Pick a uniformly random facet f_i that contains the current basic feasible solution x.

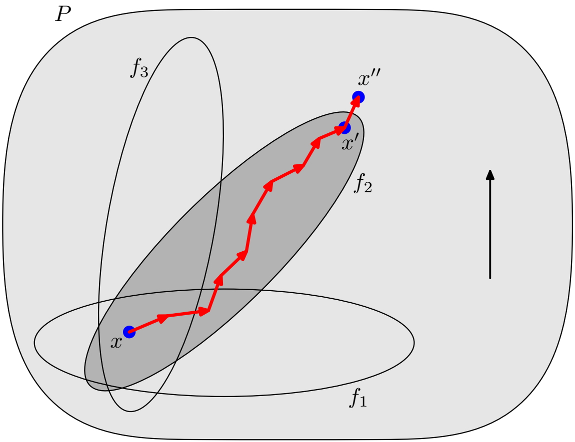


- Pick a uniformly random facet f_i that contains the current basic feasible solution x.

The RandomFacet pivoting rule

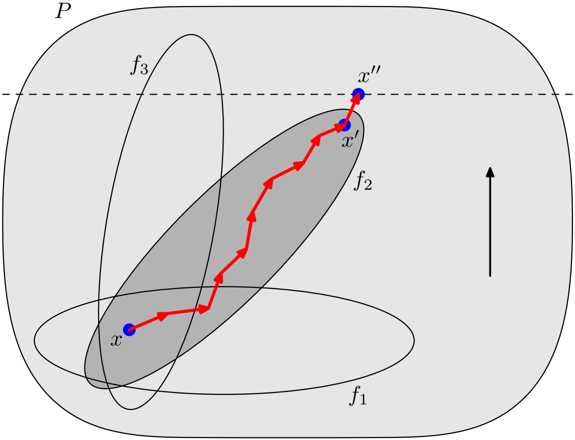


- Recursively find the optimal solution x' within the picked facet f_i.



- If possible, make an improving pivot from x', leaving the facet f_i, and repeat from the beginning. Otherwise return x'.

The RandomFacet pivoting rule



- Note that if the facets $f_1, \ldots, f_d$ containing x are ordered according to their optimal value, then from x'' we never visit $f_1, \ldots, f_i$ again.

The RandomFacet pivoting rule

- The number of pivoting steps for a linear program with d variables and n constraints, including non-negativity constraints, is at most:

$$f(d,n) \leq f(d-1,n-1) + 1 + \frac{1}{d}\sum_{i=1}^{d} f(d,n-i)$$

with $f(d,n) = 0$ for $n \leq d$.

The RandomFacet pivoting rule

- The number of pivoting steps for a linear program with d variables and n constraints, including non-negativity constraints, is at most:

$$f(d,n) \leq f(d-1,n-1) + 1 + \frac{1}{d}\sum_{i=1}^{d} f(d,n-i)$$

  with $f(d,n) = 0$ for $n \leq d$.

- Solving the corresponding recurrence gives:

$$f(d,n) \leq 2^{O(\sqrt{(n-d)\log n})}$$

The RandomFacet pivoting rule

- The RandomFacet pivoting rule can also be applied to the dual LP, which has $n - d$ free variables and n inequality constraints. It follows that:

$$T_R(d, n) = \min\left\{2^{O(\sqrt{(n-d)\log n})}, 2^{O(\sqrt{d\log n})}\right\}$$

The RandomFacet pivoting rule

- The RandomFacet pivoting rule can also be applied to the dual LP, which has $n-d$ free variables and n inequality constraints. It follows that:

$$T_R(d,n) = \min\left\{2^{O(\sqrt{(n-d)\log n})}, 2^{O(\sqrt{d\log n})}\right\}$$

- Clarkson (1988) showed that:

$$T_R(d,n) = O(d^2 n + d^4\sqrt{n}\log n + T_R(d,9d^2)d^2\log n)$$

The RandomFacet pivoting rule

- The RandomFacet pivoting rule can also be applied to the dual LP, which has $n-d$ free variables and n inequality constraints. It follows that:

$$T_R(d,n) = \min\left\{2^{O(\sqrt{(n-d)\log n})}, 2^{O(\sqrt{d\log n})}\right\}$$

- Clarkson (1988) showed that:

$$T_R(d,n) = O(d^2 n + d^4\sqrt{n}\log n + T_R(d,9d^2)d^2\log n)$$

- By combining Clarkson's algorithm and the RandomFacet algorithm applied to the dual we get:

$$T_R(d,n) = O(d^2 n + 2^{O(\sqrt{d\log d})})$$

The RandomFacet pivoting rule

- The RandomFacet pivoting rule can also be applied to the dual LP, which has $n - d$ free variables and n inequality constraints. It follows that:

$$T_R(d,n) = \min\left\{2^{O(\sqrt{(n-d)\log n})}, 2^{O(\sqrt{d\log n})}\right\}$$

- Clarkson (1988) showed that:

$$T_R(d,n) = O(d^2 n + d^4\sqrt{n}\log n + T_R(d, 9d^2)d^2\log n)$$

- By combining Clarkson's algorithm and the RandomFacet algorithm applied to the dual we get:

$$T_R(d,n) = O(d^2 n + 2^{O(\sqrt{d\log d})})$$

- This is the best known bound for $T_R(d,n)$. I.e., the best bound independent of the bit complexity.

RandomFacet: Non-recursive version

- The recursion of the RandomFacet pivoting rule can be unrolled, and the algorithm can be equivalently stated as:
  1. Start with a random permutation $x_1, \ldots, x_d$ of the non-basic variables.
  2. Let x_i be the first variable with positive coefficient according to the permutation. Make a pivot exchanging x_i with some other variable x in the basis.
  3. Replace x_i by x in the list of non-basic variables and randomly permute the first i variables. Repeat from (2).

RandomFacet: Non-recursive version

- The recursion of the RandomFacet pivoting rule can be unrolled, and the algorithm can be equivalently stated as:
  1. Start with a random permutation $x_1, \ldots, x_d$ of the non-basic variables.
  2. Let x_i be the first variable with positive coefficient according to the permutation. Make a pivot exchanging x_i with some other variable x in the basis.
  3. Replace x_i by x in the list of non-basic variables and randomly permute the first i variables. Repeat from (2).

- The procedure resembles the Randomized Bland's rule, but the expected number of steps is different.

RandomFacet: Non-recursive version

- The recursion of the RandomFacet pivoting rule can be unrolled, and the algorithm can be equivalently stated as:
  1. Start with a random permutation $x_1, \ldots, x_d$ of the non-basic variables.
  2. Let x_i be the first variable with positive coefficient according to the permutation. Make a pivot exchanging x_i with some other variable x in the basis.
  3. Replace x_i by x in the list of non-basic variables and randomly permute the first i variables. Repeat from (2).
- The procedure resembles the Randomized Bland's rule, but the expected number of steps is different.
- **Open problem:** Is there a subexponential upper bound on the expected number of pivoting steps performed by the Randomized Bland's rule?

Duality

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

- Basic feasible solutions give immediate lower bounds on the optimal value z^*. Is there a simple way to get upper bounds?

Duality

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

- Basic feasible solutions give immediate lower bounds on the optimal value z^*. Is there a simple way to get upper bounds?
- The optimal solution must satisfy any linear combination $y \in \mathbb{R}^n$ of the equality constraints.

Duality

$$\begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array}$$

- Basic feasible solutions give immediate lower bounds on the optimal value z^*. Is there a simple way to get upper bounds?
- The optimal solution must satisfy any linear combination $y \in \mathbb{R}^n$ of the equality constraints.
- If we can construct a linear combination of the equality constraints $y^T(Ax) = y^T b$, for $y \in \mathbb{R}^n$, such that $c^T x \leq y^T(Ax)$, then $y^T(Ax) = y^T b$ is an upper bound on z^*.

Duality

$$(P)\quad \begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array} \qquad\qquad (D)\quad \begin{array}{lrcl} \text{minimize} & b^T y & & \\ s.t. & A^T y & \geq & c \end{array}$$

- Basic feasible solutions give immediate lower bounds on the optimal value z^*. Is there a simple way to get upper bounds?
- The optimal solution must satisfy any linear combination $y \in \mathbb{R}^n$ of the equality constraints.
- If we can construct a linear combination of the equality constraints $y^T(Ax) = y^T b$, for $y \in \mathbb{R}^n$, such that $c^T x \leq y^T(Ax)$, then $y^T(Ax) = y^T b$ is an upper bound on z^*.
- The problem of finding the best such upper bound can be formulated as a **dual** linear program (D). The original linear program (P) is referred to as **primal**.

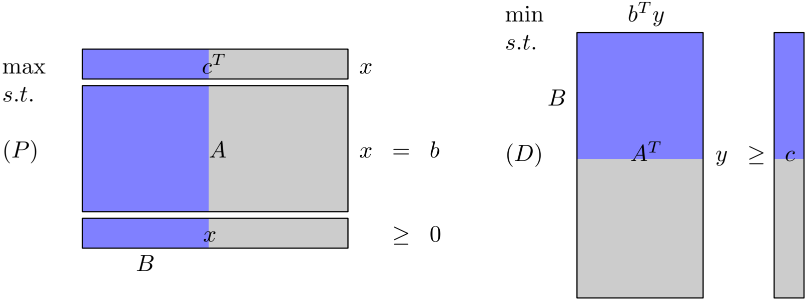
Duality

$$(P) \quad \begin{array}{lrcl} \text{maximize} & c^T x & & \\ s.t. & Ax & = & b \\ & x & \geq & 0 \end{array} \qquad (D) \quad \begin{array}{lrcl} \text{minimize} & b^T y & & \\ s.t. & A^T y & \geq & c \end{array}$$

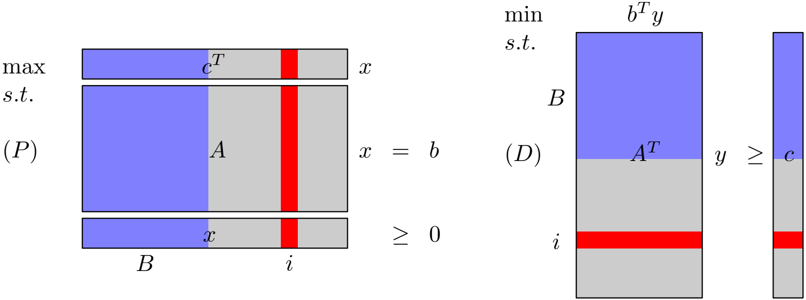
- Basic feasible solutions give immediate lower bounds on the optimal value z^*. Is there a simple way to get upper bounds?
- The optimal solution must satisfy any linear combination $y \in \mathbb{R}^n$ of the equality constraints.
- If we can construct a linear combination of the equality constraints $y^T(Ax) = y^T b$, for $y \in \mathbb{R}^n$, such that $c^T x \leq y^T(Ax)$, then $y^T(Ax) = y^T b$ is an upper bound on z^*.
- The problem of finding the best such upper bound can be formulated as a **dual** linear program (D). The original linear program (P) is referred to as **primal**.
- By the **strong duality** theorem, (P) and (D) have the same value, assuming that (P) is feasible and has a maximal value.

$$
\begin{array}{llcl}
\max & c^T x & & \\
s.t. & & & \\
(P) & A x & = & b \\
 & x & \geq & 0
\end{array}
\qquad\qquad
\begin{array}{llcl}
\min & b^T y & & \\
s.t. & & & \\
(D) & A^T y & \geq & c
\end{array}
$$

- Consider a primal linear program (P) and its dual (D).

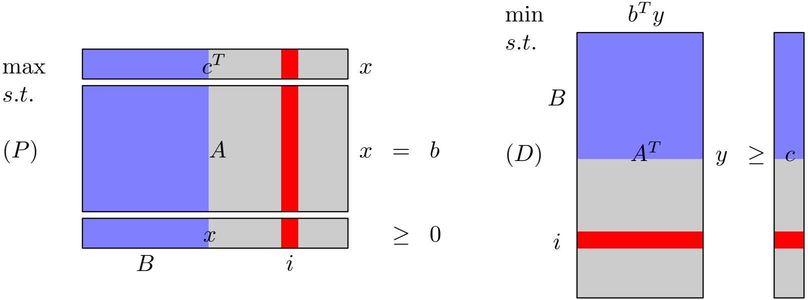


- Consider a primal linear program (P) and its dual (D).
- Recall that the non-basic variables for some basis B correspond to facets that contain the basic feasible solution x_B.



- Consider a primal linear program (P) and its dual (D).
- Recall that the non-basic variables for some basis B correspond to facets that contain the basic feasible solution x_B.
- Staying within a facet f_i means that the variable x_i stays non-basic.

RandomFacet: Dual view



- Consider a primal linear program (P) and its dual (D).
- Recall that the non-basic variables for some basis B correspond to facets that contain the basic feasible solution x_B.
- Staying within a facet f_i means that the variable x_i stays non-basic.
- I.e., x_i is fixed to 0, which is like removing the i'th column of A, or for the dual like removing the i'th constraint.

RandomFacet: Dual view

- Let H be the set of constraints (halfspaces) of the dual.
- Every subset of constraints $G \subseteq H$ defines a linear program.

RandomFacet: Dual view

- Let H be the set of constraints (halfspaces) of the dual.
- Every subset of constraints $G \subseteq H$ defines a linear program.
- Let z_G be the optimal value of the linear program defined by G.

RandomFacet: Dual view

- Let H be the set of constraints (halfspaces) of the dual.
- Every subset of constraints $G \subseteq H$ defines a linear program.
- Let z_G be the optimal value of the linear program defined by G.
  - If z_G is finite, then the corresponding primal LP has the same optimal solution.

- Let H be the set of constraints (halfspaces) of the dual.
- Every subset of constraints $G \subseteq H$ defines a linear program.
- Let z_G be the optimal value of the linear program defined by G.
  - If z_G is finite, then the corresponding primal LP has the same optimal solution.
- A basis for the dual refers to n linearly independent constraints. I.e., a basic solution.
- A basis $B \subseteq G \subseteq H$ is **optimal** for G if $z_B = z_G$.

RandomFacet: Dual view

- Let H be the set of constraints (halfspaces) of the dual.
- Every subset of constraints $G \subseteq H$ defines a linear program.
- Let z_G be the optimal value of the linear program defined by G.
  - If z_G is finite, then the corresponding primal LP has the same optimal solution.
- A basis for the dual refers to n linearly independent constraints. I.e., a basic solution.
- A basis $B \subseteq G \subseteq H$ is **optimal** for G if $z_B = z_G$.
- A constraint h is **violated** by B if $z_B < z_{B \cup \{h\}}$.

- Let H be the set of constraints (halfspaces) of the dual.
- Every subset of constraints $G \subseteq H$ defines a linear program.
- Let z_G be the optimal value of the linear program defined by G.
    - If z_G is finite, then the corresponding primal LP has the same optimal solution.
- A basis for the dual refers to n linearly independent constraints. I.e., a basic solution.
- A basis $B \subseteq G \subseteq H$ is **optimal** for G if $z_B = z_G$.
- A constraint h is **violated** by B if $z_B < z_{B \cup \{h\}}$.
    - If h is violated by a basis B, then z_B is not optimal for the corresponding primal LP, and adding h to the basis must be an improving pivot.

RandomFacet: Dual view

- The dual RandomFacet algorithm starts with a basis B such that $z_B > -\infty$.

RandomFacet: Dual view

- The dual RandomFacet algorithm starts with a basis B such that $z_B > -\infty$.
- Any basic feasible solution of the primal LP gives such a basis, but the algorithm works even if the corresponding basic solution in the primal is not feasible.

RandomFacet: Dual view

- The dual RandomFacet algorithm starts with a basis B such that $z_B > -\infty$.
- Any basic feasible solution of the primal LP gives such a basis, but the algorithm works even if the corresponding basic solution in the primal is not feasible.

- RandomFacet, dual view, Matoušek, Sharir and Welzl (1992):
  1. Remove a uniformly random constraint $h \in H$ that is not in the current basis B.
  2. Recursively find an optimal basis B' for $H \setminus \{h\}$.
  3. If B' violates h repeat from the beginning, starting with the optimal basis B'' for $B' \cup \{h\}$. Otherwise return B'.

Repeating the analysis

- Order constraints $h \in H \setminus B$ such that:

$$z_{H\setminus\{h_1\}} \;\leq\; z_{H\setminus\{h_2\}} \;\leq\; \dots \;\leq\; z_{H\setminus\{h_i\}} \;\leq\; \dots \;\leq\; z_{H\setminus\{h_{m-n}\}}$$

Repeating the analysis

- Order constraints $h \in H \setminus B$ such that:

$$z_{H\setminus\{h_1\}} \;\le\; z_{H\setminus\{h_2\}} \;\le\; \dots \;\le\; z_{H\setminus\{h_i\}} \;\le\; \dots \;\le\; z_{H\setminus\{h_{m-n}\}}$$

- If $z_{B''} > z_{H\setminus\{h_i\}}$, then after reaching B'' the constraints $h_1, \dots, h_i$ must remain in the basis until termination.

Repeating the analysis

- Order constraints $h \in H \setminus B$ such that:

$$z_{H\setminus\{h_1\}} \;\leq\; z_{H\setminus\{h_2\}} \;\leq\; \dots \;\leq\; z_{H\setminus\{h_i\}} \;\leq\; \dots \;\leq\; z_{H\setminus\{h_{m-n}\}}$$

- If $z_{B''} > z_{H\setminus\{h_i\}}$, then after reaching B'' the constraints $h_1, \dots, h_i$ must remain in the basis until termination.
  - This corresponds to the facets never being visited again.

Repeating the analysis

- Order constraints $h \in H \setminus B$ such that:

$$z_{H\setminus\{h_1\}} \;\leq\; z_{H\setminus\{h_2\}} \;\leq\; \dots \;\leq\; z_{H\setminus\{h_i\}} \;\leq\; \dots \;\leq\; z_{H\setminus\{h_{m-n}\}}$$

- If $z_{B''} > z_{H\setminus\{h_i\}}$, then after reaching B'' the constraints $h_1, \dots, h_i$ must remain in the basis until termination.
  - This corresponds to the facets never being visited again.
- The number of steps is bounded by:

$$f_D(k,m) = f_D(k,m-1) + 1 + \frac{1}{m-n}\sum_{i=1}^{m-n} f_D(k-i,m)$$

  where k is the number of unfixed constraints (the "hidden dimension"), and $f_D(m,k) = 0$ for $m \leq k$ or $k \leq 0$.
- Again, $f_D(k,m) \leq 2^{O(\sqrt{k \log m})}$.

LP-type problems

An **LP-type problem**, (H, ω), is defined as follows:

- $H = \{1, \ldots, m\}$ is a finite set.
- $\omega : 2^H \to \mathcal{W}$ is a function that maps subsets of H to a linearly ordered set $(\mathcal{W}, \leq)$ with minimal value $-\infty$, such that:
  1. *Monotonicity*: For all $F \subseteq G \subseteq H$, $\omega(F) \leq \omega(G)$.
  2. *Locality:* For all $F \subseteq G \subseteq H$ with $-\infty < \omega(F) = \omega(G)$, and any $h \in H$:

$$\omega(G) < \omega(G \cup \{h\}) \quad \Rightarrow \quad \omega(F) < \omega(F \cup \{h\})$$

LP-type problems

An **LP-type problem**, (H, ω), is defined as follows:

- $H = \{1, \ldots, m\}$ is a finite set.
- $\omega : 2^H \to \mathcal{W}$ is a function that maps subsets of H to a linearly ordered set $(\mathcal{W}, \leq)$ with minimal value $-\infty$, such that:
  1. *Monotonicity*: For all $F \subseteq G \subseteq H$, $\omega(F) \leq \omega(G)$.
  2. *Locality:* For all $F \subseteq G \subseteq H$ with $-\infty < \omega(F) = \omega(G)$, and any $h \in H$:

$$\omega(G) < \omega(G \cup \{h\}) \quad \Rightarrow \quad \omega(F) < \omega(F \cup \{h\})$$

- $B \subseteq H$ is an optimal basis if $\omega(B) = \omega(H)$, and for every proper subset $B' \subsetneq B$, $\omega(B') < \omega(B)$.

LP-type problems

An **LP-type problem**, (H, ω), is defined as follows:

- $H = \{1, \ldots, m\}$ is a finite set.
- $\omega : 2^H \to \mathcal{W}$ is a function that maps subsets of H to a linearly ordered set $(\mathcal{W}, \leq)$ with minimal value $-\infty$, such that:
  1. *Monotonicity*: For all $F \subseteq G \subseteq H$, $\omega(F) \leq \omega(G)$.
  2. *Locality:* For all $F \subseteq G \subseteq H$ with $-\infty < \omega(F) = \omega(G)$, and any $h \in H$:

$$\omega(G) < \omega(G \cup \{h\}) \quad \Rightarrow \quad \omega(F) < \omega(F \cup \{h\})$$

- $B \subseteq H$ is an optimal basis if $\omega(B) = \omega(H)$, and for every proper subset $B' \subsetneq B$, $\omega(B') < \omega(B)$.
- Goal: Find an optimal basis for H.

LP-type problems

An **LP-type problem**, (H, ω), is defined as follows:

- $H = \{1, \ldots, m\}$ is a finite set.
- $\omega : 2^H \to \mathcal{W}$ is a function that maps subsets of H to a linearly ordered set $(\mathcal{W}, \leq)$ with minimal value $-\infty$, such that:
  1. *Monotonicity*: For all $F \subseteq G \subseteq H$, $\omega(F) \leq \omega(G)$.
  2. *Locality:* For all $F \subseteq G \subseteq H$ with $-\infty < \omega(F) = \omega(G)$, and any $h \in H$:

$$\omega(G) < \omega(G \cup \{h\}) \quad \Rightarrow \quad \omega(F) < \omega(F \cup \{h\})$$

- $B \subseteq H$ is an optimal basis if $\omega(B) = \omega(H)$, and for every proper subset $B' \subsetneq B$, $\omega(B') < \omega(B)$.
- Goal: Find an optimal basis for H.

The dual RandomFacet algorithm can be applied to any LP-type problem.

Overview

- **Lecture 1:**
  - Introduction to linear programming and the simplex algorithm.
  - Pivoting rules.
  - The RandomFacet pivoting rule.
- **Lecture 2:**
  - The Hirsch conjecture.
  - Introduction to Markov decision processes (MDPs).
  - Upper bound for the LargestCoefficient pivoting rule for MDPs.
- **Lecture 3:**
  - Lower bounds for pivoting rules utilizing MDPs. Example: Bland's rule.
  - Lower bound for the RandomEdge pivoting rule.
  - Abstractions and related problems.